

Web Technologies-III

DCAP312



L OVELY
P ROFESSIONAL
U NIVERSITY



WEB TECHNOLOGIES - II

Copyright © 2013, Kanika Garg
All rights reserved

Produced & Printed by
EXCEL BOOKS PRIVATE LIMITED
A-45, Naraina, Phase-I,
New Delhi-110028
for
Lovely Professional University
Phagwara

CONTENTS

Unit 1:	Making Sense of .NET and Anatomy of an ASP.NET Page	1
Unit 2:	Introduction to C#	29
Unit 3:	Server Controls Basic	49
Unit 4:	Advanced Server Controls	65
Unit 5:	Database Access	82
Unit 6:	Error Handling	103
Unit 7:	Advanced ASP.NET	123
Unit 8:	Creating More Advanced ASP.NET	144
Unit 9:	The Database Model	177
Unit 10:	Web Services	203
Unit 11:	Web Services in Visual Studio .NET	225
Unit 12:	Security and Membership	239
Unit 13:	Adding E-commerce Essentials	255
Unit 14:	Debugging and Optimization	277

SYLLABUS

Web Technologies - II

Objectives:

- To impart the skills needed for web programming.
- To impart the skills needed for web administration.
- To enable the student to learn web site development.
- To enable the student to learn various server controls.
- To enable the student to understand web services.
- To enable the student to understand various security issues related to web technologies
- To enable the student to understand data Processing on web pages using ADO.NET
- To enable the student to learn debugging and optimizing techniques of web applications.

S.No.	Description
1.	Making Sense of .NET & Anatomy of an ASP.NET Page: The Microsoft .NET Vision, ASP in NET, Introduction to C#, A Simple Web Page, Adding a Web Control, Introduction to In-Line Script, The Page Class.
2.	Server Controls: Postback, Data Binding, Web Server Controls.
3.	Server Controls: HTML Server Controls, Validation Controls.
4.	Database Access: Error Handling, Database Access Using ADO.NET, Connection, Command, DataAdapter, and DataSet, DataReader, Connection Pooling.
5.	Creating More Advanced ASP.NET Pages: Communicating with the Browser, Web.Config.
6.	Creating More Advanced ASP.NET Pages: Page Sub-classing, User Controls, More Advanced Data Binding.
7.	Applying What We've Learned So Far: The Database Model, Creating a Basic Object Model, Creating the User Interface.
8.	Web Services: XML Web Services, Uses for Web Services, Web Services in Visual Studio .NET, Creating Web Services, Expanding Web Application with Web Services.
9.	Security and Membership: IIS Security, ASP.NET Authentication. Adding E-Commerce Essentials: XML Tools, Freight Calculations, Email.
10.	Debugging and Optimization: Debugging in an ASP.NET Application, Optimization, Optimizing Using Caching, Optimizing via Performance Profiling.

Unit 1: Making Sense of .NET and Anatomy of an ASP.NET Page

CONTENTS

Objectives

Introduction

- 1.1 Features of .NET
- 1.2 The Challenges of Building Modern Applications
 - 1.2.1 Limitations in Win32 Clients
 - 1.2.2 Limitations in DNA Based Internet Development or Browser Based Clients
- 1.3 .NET Philosophy
 - 1.3.1 Microsoft.NET – The Solution
 - 1.3.2 Other Benefits of Using .NET Architecture
 - 1.3.3 N-tier Architecture with .NET
- 1.4 Understanding the .NET Platform and its Layers
 - 1.4.1 Constituents of .NET Platform
 - 1.4.2 The Common Language Runtime (CLR)
 - 1.4.3 The .NET Class Framework
 - 1.4.4 User Interface
 - 1.4.5 Languages
 - 1.4.6 .NET Products
 - 1.4.7 .NET Services
 - 1.4.8 .NET Runtime
- 1.5 Understanding the Various Components of the .NET Platform
 - 1.5.1 Common Language Runtime
 - 1.5.2 Automatic Memory Management
 - 1.5.3 Common Type System
 - 1.5.4 Common Type System Architecture
- 1.6 ASP in .NET
 - 1.6.1 Many Faces of ASP.NET
- 1.7 Problems with Traditional ASP
- 1.8 Advantages of ASP.NET
- 1.9 ASP.NET Architecture
- 1.10 Summary
- 1.11 Keywords
- 1.12 Review Questions
- 1.13 Further Readings

Notes

Objectives

After studying this unit, you will be able to:

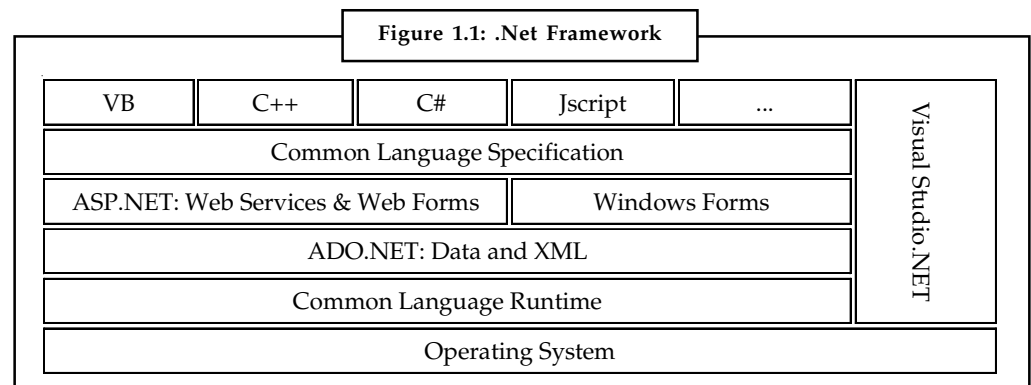
- Discuss the features of .NET
- Define challenges of building modern applications
- Discuss the .NET philosophy
- Understand the .NET platform and its layers
- Define the various components of the .NET platform
- Describe ASP in .NET
- Discuss the problems with traditional ASP
- Define the advantages of ASP.NET
- Discuss the ASP.NET architecture

Introduction

Microsoft .Net Framework is a programming infrastructure formed by Microsoft for construction, deploying, and process applications and services that use .NET technologies, such as desktop applications and Web services. The Microsoft .NET Framework is a application software framework that can be installed on PC’s running Windows operating systems. It includes a huge library of coded solutions to general programming problems and a virtual system that manages the execution of codes written specifically for the framework.

The objective in application creation is always the same: Build the best feasible software in the least amount of time. Yet the bar is constantly raised, as customer demands keeps increasing. To fulfill these demands, the developers build on and the widget they use must get better and better they must evolve.

Consider the illustration in Figure 1.1.



1.1 Features of .NET

The following are features of .NET:

- *Rich Functionality Out of the Box*

The .NET framework gives colorful of functionality out of the box. It contains hundreds of modules that present collection of functionality that you can use in your applications. As a

programmer you need not go into details of many functions such as file IO, network communication, etc.

- ***Easy Development of Web-Based Applications***

The ASP.NET is a technology available on .NET platform for creating dynamic and data driven web-based applications. ASP.NET has an event driven programming model/webforms with intricate UI. ASP.NET server controls provide sophisticated UI elements (like DetailsView and GridView) that save lot of coding effort.

- ***OOPs Support***

.NET is a fully object oriented environment. According to .NET philosophy "Object is mother of all." Languages like Visual Basic.NET now support many of the Object Oriented characteristics that were missing traditionally. Even integer and characters can also be treated as objects something not available even in C++.

- ***Multi-Language Support***

Generally enterprises have shifting skill sets. For example, People working in a company might posses different programming skills in Visual Basic, C++, and Java etc. Since .NET support multiple languages it makes the existing system more robust and reduces the training and development cost incurred by an organization to keep pace with advancement in programming languages. This means that if you have skills in Java, you just mould them to suit .NET environment. As of now four languages are available right out of the box namely Visual Basic .NET, C# (C-sharp), Jscript.NET and Managed C++ (a dialect of Visual C++). Many developers are working on creating language compilers for other languages (20+ language compilers are already available). Best part of multi-language support lies in the fact that even though the syntax of each programming language is different, the basic functionalities of each language remain at par with each other.

- ***Multi-Device Support***

Current lifestyle is increasingly embracing mobile and wireless devices such as SamrtPhones, Tablets and handheld PCs. .NET provides platform for programming such devices. .NET Compact Framework and Mobile Internet Toolkit are step ahead in this direction.

- ***Automatic memory management***

Earlier developer was expected to always keep an eye on system resources like memory as memory leaks has proven to be the major reasons for failure of application. .NET helps developer by automatic memory handling in which the garbage collector keeps freeing unused objects at appropriate intervals.

- ***Compatibility with COM and COM+***

Earlier COM was the de-facto standard for componentized software engineering. Organizations have invested a lot of time, effort and money in creating COM components and controls. The best part is one can still use COM components and ActiveX controls under .NET. Enabling you to use your existing investment in .NET applications. .NET still relies on COM+ for features like transaction management and object pooling. In fact it provides enhanced declarative support for configuring COM+ application right from your code. Your COM+ knowledge still remains as a valuable asset.

- ***No more DLL Hell***

There are a number of problems commonly encountered with DLLs – especially after numerous applications have been installed and uninstalled on a system. The difficulties include conflicts between DLL versions, difficulty in obtaining required DLLs, and having many unnecessary

Notes

DLL copies. DLL Hell can manifest itself in many different ways; typically applications do not launch or work correctly. Applications on modern versions of Windows suffer less from this issue following the introduction of the .NET Framework, registry-free COM and features in the operating system that prevent system files from being overwritten.

- **Strong XML support**

Now days programmers are well versed with XML. XML has gained strong industry support that almost all the developers have released upgrades/patches for their existing software so that it becomes “XML compatible”. Currently, .NET being the only platform that has built with XML right into the core framework. .NET not only provides support for manipulating and transforming XML documents, .NET also offers XML web services that are based on standards like HTTP, XML and SOAP.

- **Ease of Deployment and Configuration**

Installing windows applications particularly that used COM components has always been a deadly task. Since .NET does not require any registration, deployment has been simplified. Which makes XCOPY deployment viable. Configuration is one more area where .NET—particularly ASP.NET – stands out over traditional languages. The configuration is done via specific files having special XML vocabulary and most of the configuration is done via configuration files, it has overcome the need to physically sit in front of the machine and manually configure the application. This is more important for web applications; simply FTP new configuration file makes necessary changes.

- **Security**

Windows has always been criticized for poor security system. .NET platform is a result of Microsoft great efforts which makes it safe and secure for enterprise applications. Features such as safety, code access security and role based authentication make overall application more robust and secure.



Did u know? Development on the .NET Framework is started in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.



Task Search more about the history of programming languages.

Self Assessment

Multiple Choice Questions:

1. Which of the following statements correctly define .NET Framework?
 - a. It is an environment for developing, building, deploying and executing Desktop Applications, Web Applications and Web Services.
 - b. It is an environment for developing, building, deploying and executing only Web Applications.
 - c. It is an environment for developing, building, deploying and executing Web Services.
 - d. It is an environment for development and execution of Windows applications.

2. Which of the following constitutes the .NET Framework? Notes
- ASP.NET Applications
 - CLR
 - WinForm Applications
 - Windows Services
3. Which of the following is part of the .NET Framework?
- The Framework Class Libraries (FCL)
 - Microsoft Published Web Services
 - Applications deployed on IIS
 - Mobile Applications
4. Which of the following jobs are done by Common Language Runtime?
- It provides core services such as memory management, thread management, and remoting.
 - It provides Code Access Security.
 - It provides Garbage Collection Services.
 - All of the above.
5. Which of the following statements is correct about Managed Code?
- Managed code is the code that is compiled by the JIT compilers.
 - Managed code is the code where resources are Garbage Collected.
 - Managed code is the code that runs on top of Windows.
 - Managed code is the code that is written to target the services of the CLR.

1.2 The Challenges of Building Modern Applications

Creating a modern application is not a simple task — the requirements are substantial. Traditional concerns such as creating effective business logic and allowing access via a Web browser are still important, but they're no longer enough. Modern applications present a range of new challenges, including the following:

- Users increasingly expect Web browser interfaces to act like installed Windows applications. Loading a new page whenever something has changed is no longer sufficient – it's just too slow. What's needed is better *support for responsive browser applications*.
- Data remains central to most applications. How that data can be represented, however, has expanded considerably. Relational data is still important, as is mapping between objects and relations. Yet the amount of data represented using XML continues to increase, a trend that's not likely to change. And even though it's not always viewed in this way, a running program's objects also contain data. A technology that allowed *consistent access to diverse data* would help developers create applications in less time and with fewer errors.
- Applications commonly communicate with other applications, both inside and outside the organization. Modern applications also must often fit into a service-oriented architecture (SOA), exposing some of their functionality as interoperable services accessible by other software. Achieving these goals requires *support for service-oriented applications*.

Notes

- Organizations are increasingly taking a process-oriented view of what they do. Since most applications automate some part of a business process, it can be useful to make the steps in this process explicit in the code. An effective way to do this is by using workflow technology, an approach that requires *support for workflow-based applications*.
- The requirements for a modern user interface have grown significantly. Providing real business value can commonly require working with various kinds of documents, using two- and three-dimensional graphics, displaying video, and more. Meeting these needs requires a *unified approach to diverse user interfaces*.
- The people who use an application commonly need a way to convey information about who they are. Many different technologies for defining and using a digital identity are in use, and problems such as phishing are common. Given this, a modern application and the people who use it can benefit from *consistent user control of digital identities*.

Given that today's applications commonly need to tackle some or all of these challenges, the platform those applications are built on should also address all of them. The goal of the .NET Framework is to do this for Windows software.

1.2.1 Limitations in Win32 Clients

If we talked about Client-Server Model then to create highly interactive applications we use some tools like visual basic. But these client softwares are difficult to deploy and maintain as they need to install on every client machine and whenever any upgradation carried out it further need to change on every client. Another Problem is DLL conflict as every client variate in the means of software and operating system. Visual Basic is the most common one. This requires high level of expertise in COM. Since these middle-tire components are implemented using Microsoft Transaction Server on Windows NT or COM+services on Windows 2000.

These components use stateless designs, which can appear very different from the stateful designs often utilized in client-based components. COM components, in the middle tier must work together, versioning all the components properly so that these people recognize each other's interfaces can be challenging. This requires a extremely sophisticated skill level and a well-controlled deployment procedure. COM works well upon Microsoft platforms. But it suffers from lack of interoperability with other systems. One of the most important ways functionality can end up being reused is for a software component to inherit another component. However COM does not assistance inheritance.

Visual basic is a proprietary programming language written by Microsoft, so programs written in Visual basic cannot, easily, be transferred to other operating systems. Visual Basic is utilized in two major roles forms based VB Clients and COM components. This VB6 language has its own limitations it does not have access to the capability of multi-threading, lack of OOPS concepts, Inadequate error handling ability and poor integration with other 'languages'. Hence it makes it undesirable for development of object based frameworks. These days's applications need to make use of the Win32 API for any variety of purposes like monitor widows messages, change controls, reading and contacting INI files and socket programming etc. But these widows API are hard to program for variety associated with reasons, like it is not object oriented and complicated calls to the capabilities with long lists of arguments, since Win32 API is written within C++ language, getting calling conventions right on data types is messy.

1.2.2 Limitations in DNA Based Internet Development or Browser Based Clients

Visual Basic 6 is easily the most popular language for developing applications with the DNA model. It is used in two major roles: forms-based VB clients and COM components (either on the

client or the server). There are other options, of course, including C++, J++, and various third-party languages such as Delphi and Perl, but the number of VB developers outnumbers them all put together. However, although it's popular, VB6 isn't without its limitations, which include:

No capability for multi-threading – Which implies, for example, that VB6 can't be used to write an NT-type service. There are also situations in which the apartment threading used by components created in VB6 limits performance.

A lack of implementation inheritance and other object-oriented features – This makes VB6 unsuitable for the development of object-based frameworks.

Poor error-handling ability – VB6's archaic error handling becomes especially annoying in a multi-tier environment. It's difficult in VB6 to track and pass errors through a stack of component interfaces.

Poor integration with other languages such as C++ – VB6's implementation of COM, although easy to use, causes problems with such integration. Class parameters (object interfaces) in VB6 are "variant compliant", forcing C++ developers who want to integrate with VB to convert parameters to less appropriate types. These varying data structures and interface conventions must be resolved before components in VB can be integrated into a multiple language project. Besides requiring extra code, these conversions may also result in a performance hit.

No effective user interface for Internet-based applications – Perhaps the biggest drawback to using VB6 became apparent when developing for the Internet. While VB6 forms for a Win32 client were state-of-the-art, for applications with a browser interface VB6 was mostly relegated to use in components.

Microsoft tried to address this last problem in VB6 with **WebClasses** and **DHTML Pages** but neither caught on:

WebClasses offered an obscure programming model, and limited control over visual layout.

DHTML Pages in VB6 had to send a (usually large) DLL to the client, and so needed a high-bandwidth connection to be practical. This limited their use mostly to intranet applications. DHTML Pages were also restricted to Internet Explorer.

1.3 .NET Philosophy

Microsoft .NET is the first software development platform to be designed from the ground up with the Internet in mind – although .NET is not exclusively for Internet development; rather it provides a consistent programming model that can be used for many types of applications. However, when an application needs Internet capabilities, access to those capabilities is almost transparent, unlike tools currently used for Internet-enabled applications.

1.3.1 Microsoft.NET – The Solution

Microsoft's .NET initiative is broad-based and very ambitious. It includes the .NET Framework, which encompasses the languages and execution platform, plus extensive class libraries providing rich built-in functionality. Besides the core .NET Framework, the .NET initiative includes protocols (such as the Simple Object Access Protocol, commonly known as SOAP) to provide a new level of integration of software over the Internet, and a set of pre-built web-based services called .NET My Services (formerly codenamed Hailstorm). Microsoft also released several products early in 2001, which were described as being part of the .NET Enterprise Server family: SQL Server 2000, Commerce Server 2000, BizTalk Server, Exchange 2000, Host Integration Server (the successor to SNA Server), and Internet Security and Administration (ISA) Server (the successor to Proxy Server). Some of the marketing literature for these products emphasizes that they are part of

Notes

Microsoft's .NET strategy. However, it is important to understand the difference between these products and the .NET Framework. The .NET Enterprise Servers are not based on the .NET Framework. Most of them are successors to previous server-based products, and they use the same COM/COM+ technologies as their predecessors. These .NET Enterprise Servers still have a major role to play in future software development projects. When actual .NET Framework projects are developed, most will depend on the technologies in the .NET Enterprise Servers for functions like data storage and messaging.

1.3.2 Other Benefits of Using .NET Architecture

In addition to the advantages conferred by meeting the goals we discussed previously, .NET offers a number of additional benefits. These include:

- Faster development – we have less to do as the system handles more
- More reuse because of inheritance
- Greater scalability – many capabilities to help applications scale are built into .NET
- Easier to build sophisticated development tools – debuggers and profilers can target the Common Language Runtime, and thus become accessible to all .NET-enabled languages
- Fewer bugs – whole classes of bugs should be unknown in .NET; for example, with the CLR handling memory management, memory leaks should be a thing of the past
- Potentially better performance – Microsoft's heavy investment in system level code for memory management, garbage collection, and the like have yielded an architecture that should meet or exceed performance of typical COM-based applications today

1.3.3 N-tier Architecture with .NET

N-Tier architecture is an industry-proved software **architecture** model, suitable to support enterprise-level client/server applications by resolving issues like scalability, security, fault tolerance and etc. .NET has many tools and features, but .NET doesn't have predefined ways to guard how to implement **N-Tier architecture**. Therefore, in order to achieve good design and implementation of **N-Tier architecture** in .NET, understanding fully its concepts is very important. However, many of us may hear, read or use **N-Tier architecture** for many years but still misunderstand its concepts more or less.

Diagram for the **N-Tier architecture** model in .NET is given in Figure 1.2.

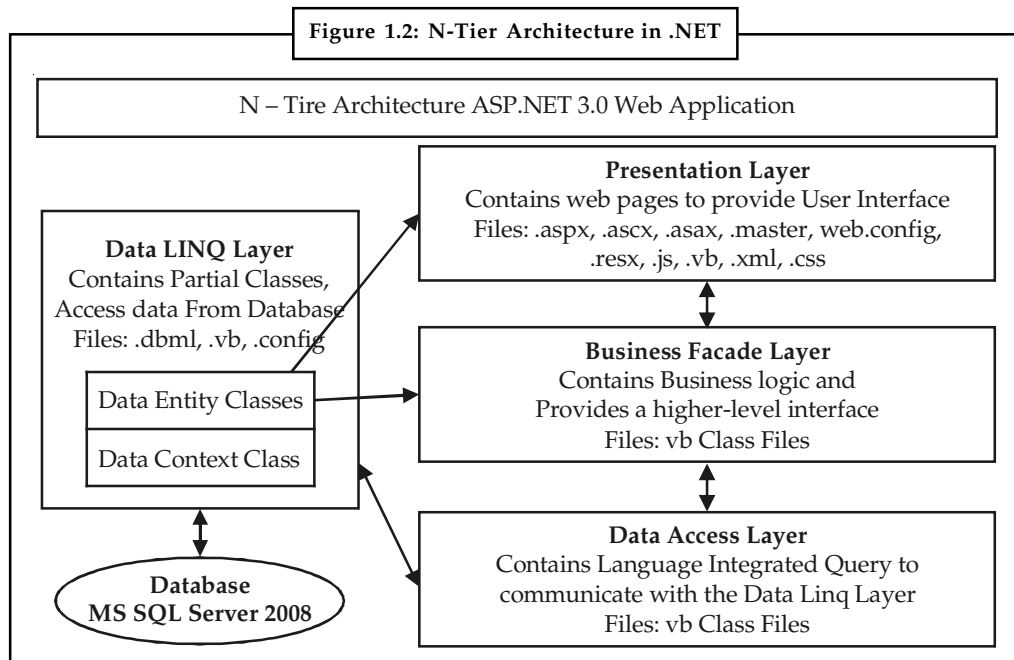
The oval components in each layer may co-exist or exist individually only. We summary all layers as below:

Client layer: This layer is involved with users directly. There may be several different types of clients coexisting, such as WPF, Window Form, HTML web page and etc.

Client presenter layer: Contains the presentation logic needed by clients, such as ASP .NET MVC in IIS web server. Also it adapters different clients to the business layer. There are three components drawn in this layer: common lib, ASP .NET, WPF client lib. Common lib holds reusable common code for all types of clients. ASP .NET lib and WPF client lib are libraries individually for web client and WPF client. If there is a new type of client added, we may need to add additional lib to support the new type client only, such as Windows Form client.


Business layer: Handles and encapsulates all of business domains and logic; also called domain layer. There are two components in this layer: WCF lib and WCF host application; all business logic and implementation are in WCF lib; WCF host application only deal with the WCF

deployment. In a 2-tier **architecture** configuration, client presenter layer will call the WCF lib directly; whereas, in a complete **N-Tier architecture** configuration, client presenter layer will call the WCF host application directly. The business interface in this layer is usually a business oriented facade layer which exposes the business operations to the clients at the degree just to meet the client's need. Therefore, it also acts as a layer of proxy to the client side to protect the business domain and logic. Client sides usually see the business layer as a black box: send a request to the business layer, then the request gets done and responded; the client sides usually don't know how their requests are fulfilled.



Persistence layer: Handles the read/write of the business data to the data layer, also called data access layer (DAL). There are three components in this layer: Entity Framework lib, persistence lib and optional WCF data service. Persistence lib is a generic library to facilitate and simplify the usage of Entity Framework or other persistence technique in business layer; it also decouples the business layer from Entity Framework. Business layer will call directly the persistence library other than Entity framework. Depending on the application configuration, the persistence lib will then either call Entity Framework lib directly or call directly the WCF data service which is created on the top of the Entity Framework lib. If the database is serving as a data center for a variety of different types of applications, WCF data service will be a good choice; otherwise, we may need to avoid WCF data service to gain performance.

Data layer: The external data source, such as database server, CRM system, ERP system, mainframe or other legacy systems and etc. Data layer is used to store the application data. Database server is the most popular nowadays. We list three modern databases in Figure 1.2 for selection.



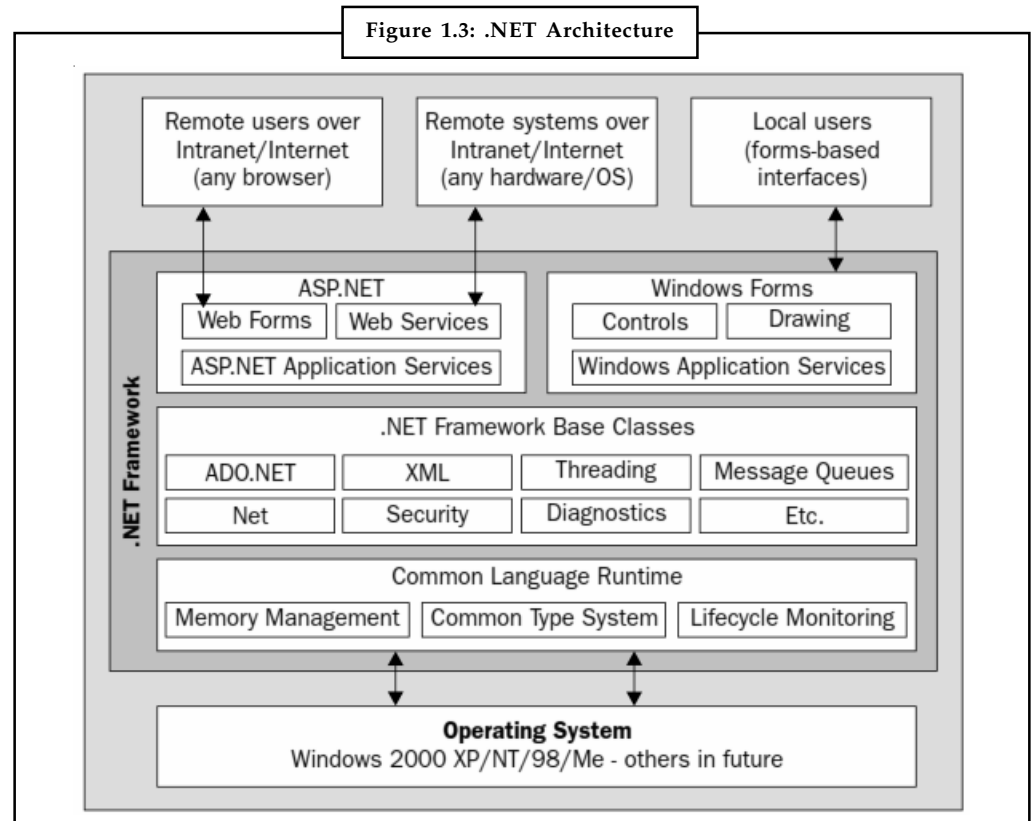
Task Search more about the history of programming languages.

1.4 Understanding the .NET Platform and its Layers

These are ambitious goals. To understand how they are accomplished, we need to understand the general structure of Microsoft .NET. One way to look at .NET is to see how it fits into the rest

Notes

of the computing world. Here is a diagram of the major layers of .NET, showing how they sit on top of an operating system, and provide various ways to interface to the outside world. Note how the entire architecture has been created to make it as easy to develop Internet applications, as it is to develop for the desktop:



The first point of this Figure 1.3 is that .NET is a framework that covers all the layers of software development above the operating system. It provides the richest level of integration among presentation technologies, component technologies, and data technologies ever seen on a Microsoft, or perhaps any, platform.

The .NET Framework wraps the operating system, insulating software developed with .NET from most operating system specifics such as file handling and memory allocation.

The .NET Framework itself starts with the execution engine, memory management, and component loading, and goes all the way up to multiple ways of rendering user and program interfaces. In between, there are layers that provide just about any system-level capability that a developer would need.

1.4.1 Constituents of .NET Platform

The .NET consists of the following three main parts:

1. *.NET Framework* – A completely re-engineered development environment.
2. *.NET Products* – Applications from MS based on the .NET platform, including Office and Visual Studio.
3. *.NET Services* – Facilitates web services enable to quickly integrate applications across multiple platforms, systems and even across businesses.

1.4.2 The Common Language Runtime (CLR)

The CLR is at the heart of the .NET Framework. The core of the CLR is an execution engine that loads, executes, and manages code that has been compiled into an intermediate byte-code format called Microsoft Intermediate Language (MSIL and often referred to as just IL). This code is not interpreted – it is compiled to native binary code before execution by just-in-time compilers built into the CLR. That means there are two levels of compilers in .NET. The language compiler takes the source code and creates MSIL. This MSIL byte code is portable to any .NET platform. At execution time, this code is then compiled by the just-in-time compilers into the native binary code of the machine the code is executed on.

1.4.3 The .NET Class Framework

The next layer up in the framework provides the services and object-models for data, input/output, security, and so forth. It is called the .NET Framework class library, sometimes referred to as the .NET base classes. .NET includes functionality that is, in many cases, a duplication of existing class libraries. There are several reasons for this:

- The .NET Framework class library is implemented in the .NET Framework, which makes them easier to integrate with .NET-developed programs.
- The .NET Framework class library brings together most of the system class libraries into one location, which increases consistency and convenience.
- The class libraries in the .NET Framework class library are much easier to extend than older class libraries.
- Having the libraries as part of the .NET Framework simplifies deployment of .NET applications. Once the .NET Framework is installed on a system, individual applications don't need to install base class libraries for common functions like data access.

1.4.4 User Interface

In a sense, the top layer of the .NET Framework is an extension of the .NET Framework Base Classes layer immediately underneath it. It comprises highly innovative user and program interfaces that allow .NET to work with the outside world. These interfacing technologies are all highly innovative:

- Windows Forms is a language-independent forms engine that brings the drag-and-drop design features of Visual Basic to all .NET-enabled languages, and also enables developers to develop forms-based interfaces with little or no access to the Win32 API.
- Web Forms brings drag-and-drop design and an event-driven architecture to Web-based interfaces, implementing a programming model that is much like standard VB6 forms-based development. User interfaces created with Web Forms also have built-in browser independence and state management.
- Web Services allow remote components, possibly running on completely different operating systems, to be invoked and used. This capability for communications and interoperability with remote components over the Internet serves as the mechanism by which highly-distributed applications can be built, going far beyond what is feasible with existing technologies like DCOM.

Notes

1.4.5 Languages

The CLR executes binary code in MSIL, and that code looks the same regardless of the original source language. All .NET-enabled languages use the same data types and the same interfacing conventions. This makes it possible for all .NET languages to interoperate transparently. One language can call another easily, and languages can even inherit classes written in another language and extend them. No other platform has anywhere near this level of language interoperability.

This makes choosing a language mostly a matter of taste. .NET-enabled languages will typically have the same performance characteristics, the same overall functionality, and will interoperate with other languages the same.

One of the most important aspects of meeting this goal is that Visual Basic becomes a first-class language. It has almost exactly the same capabilities as C#. It has inheritance, structured error handling, and other advanced features. With the large number of developers who already know Visual Basic, VB.NET should set the stage for Visual Basic to continue to be the most popular programming language in the world.

1.4.6 .NET Products

Microsoft Visual Studio .NET

Microsoft Visual Studio .NET represents the very best development environment for the .NET platform.

Integrations is the important thing in the new VS.NET IDE, thus a single IDE can be used to program in a variety of managed languages through VB.NET to Visual C++ with managed extensions.

1.4.7 .NET Services

XML Web Services

The XML is turning the way we create and make use of software. The Web revolutionized how users talk to applications. XML is transformed how applications interacts with other applications or more broadly, how systems interacts with other system by providing a universal data format that lets data be effortlessly adapted or transformed:

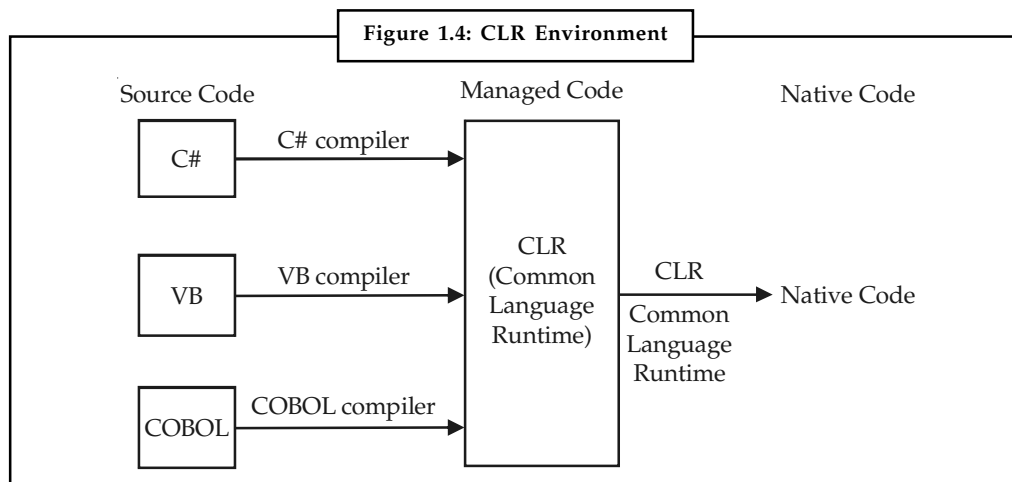
- XML Web services allow different applications to share data among themselves.
- XML Web services are distinct units of code; each made to handle a limited set of responsibilities.
- The universal language of IDE is based on XML, and can be executed across platforms and operating systems, regardless of programming language used.
- Microsoft .NET is a set of software technologies used for connecting your world of information, people, systems, and devices through the use of XML Web services.

1.4.8 .NET Runtime

Notes

The CLR is an Execution Environment. It works as a layer between Operating Systems and the applications written in .Net languages that conforms to the Common Language Specification (CLS). The main function of Common Language Runtime (CLR) is to convert the Managed Code into native code and then execute the Program. The Managed Code compiled only when it needed, that is it converts the appropriate instructions when each function is called. The CLR's Just-In-Time (JIT) compilation converts Intermediate Language (MSIL) to native code on demand at application run time.

During the execution of the program, the CLR manages memory, Thread execution, Garbage Collection (GC), Exception Handling, Common Type System (CTS), Code safety verifications, and other system services. The CLR defines the Common Type System (CTS), which is a standard type system used by all .Net languages. That means all .NET programming languages uses the same representation for common Data Types, so CLR is a language-independent runtime environment. The CLR environment is also referred to as a managed environment, because during the execution of a program it also controls the interaction with the Operating System. In the coming section you can see what are the main functions of CLR.



Functions of CLI

The Common Language Runtime (CLR) is an Execution Environment. Common Language Runtime (CLR)'s main tasks are to convert the .NET Managed Code to native code, manage running code like a Virtual Machine and also controls the interaction with the Operating System.

Common Language Runtime (CLR) manages Thread executions, Memory Management that is allocation of Objects and Buffers, Garbage Collection (GC) – Clean up the unused Objects and buffers, Exception Handling, Common Type System (CTS) that is all .NET language that conforms to the Common Language Specification (CLS) have the same primitive Data Types, Code safety verifications – code can be verified to ensure type safety, Language integration that is Common Language Runtime (CLR) follow a set of specification called Common Language Specification (CLS), this will ensure the interoperability between languages, Integrated security and other system services.

To enable the runtime to provide services to managed code, language compilers must emit metadata, which the runtime uses to locate and load classes, lay out instances in memory.

1.5 Understanding the Various Components of the .NET Platform

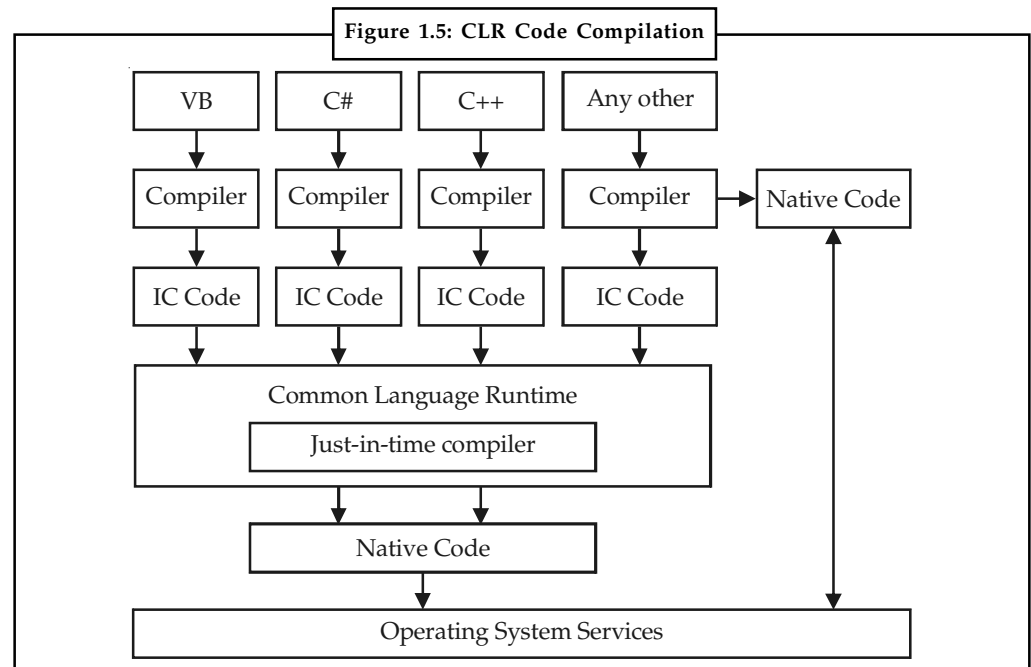
The .NET framework has following components:

1.5.1 Common Language Runtime

.Net Framework provides runtime environment called **Common Language Runtime (CLR)**. It provides an environment to run all the .Net Programs. The code which runs under the CLR is called as **Managed Code**. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management.

Programmatically, when our program needs memory, CLR allocates the memory for scope and de-allocates the memory if the scope is completed.

Language Compilers (e.g. C#, VB.Net, J#) will convert the Code/Program to **Microsoft Intermediate Language (MSIL)** intern this will be converted to **Native Code** by CLR. See the Figure 1.5 below:



There are currently over 15 language compilers being built by Microsoft and other companies also producing the code that will execute under CLR.

Now we discuss some of the more significant features provided to .NET applications by the CLR. These include:

- Automatic Memory Management
- Common Type System

1.5.2 Automatic Memory Management

Automatic memory management involves two major steps:

1. **Allocation of memory:** When a new process is initialized, runtime reserves address place in memory for the process. This particular address space inside the memory is known as

heap. At the really initial stage, the pointer is at the base address of managed heap. When the application creates an object, the garbage collector allocates memory in the address space immediately following the first object. As long as address space is available, the GC keeps allocating space to the objects one over another. Runtime allocates memory for an object by adding something to a pointer directed to previously allocated item. This makes it faster.

2. **Releasing memory:** Garbage collector figures the best time to carry out collection. This is carried out on the basis of how memory was allocated. GC releases all the actual idle objects. To discover which ones are not in use, GC examines the software's beginnings. All the application roots are assigned to the objects or are set to null. A graph is formed on the foundation of this and the actual objects which are not part of this graph are unveiled from the memory.

1.5.3 Common Type System

The common type system defines how types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for cross-language integration. The common type system performs the following functions:

- Establishes a framework that helps enable cross-language integration, type safety, and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other.

Common Language Specification (CLS)

Common Language Specification (CLS) is a set of basic language features that .Net Languages needed to develop Applications and Services , which are compatible with the .Net Framework. When there is a situation to communicate Objects written in different .Net Complaint languages, those objects must expose the features that are common to all the languages . Common Language Specification (CLS) ensures complete interoperability among applications, regardless of the language used to create the application.

Common Language Specification Rules

It describes the minimal and complete set of features to produce code that can be hosted by CLR. It ensures that products of compilers will work properly in .NET environment.

The most important rules, and which apply to public and protected members are:

- All types appearing in a method prototype must be CLS-compliant.
- Array elements must have a CLS-compliant element type. Arrays must also be 0-indexed.
- A CLS compliant class must inherit from a CLS-compliant class.
- System.Object is CLS compliant.
- Although method names in CLS-compliant classes are not case-sensitive, no two method names can be different only in the case of the letters in their name.
- Enumerations must be of type Int16, Int32, or Int64. Enumerations of other types are not compliant.
- Also there are several naming guidelines, but they are not mandatory.

1.5.4 Common Type System Architecture

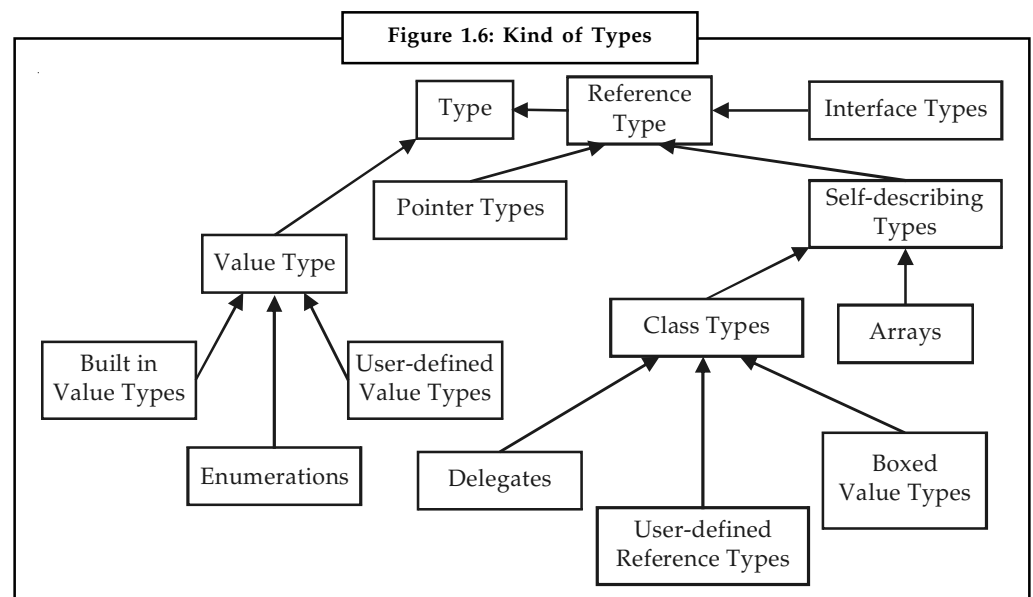
The common type system supports two general categories of types:

Value types: Value types are data types whose objects are represented by the object’s actual value. If an instance of a value type is assigned to a variable, that variable is given a fresh copy of the value.

They include:

- Boolean
- Byte
- Char
- Date Time
- Decimal
- Double
- Guid
- Int16
- Int32
- Int64
- SByte
- Single
- Timespan

Reference types: Reference types are data types whose objects are represented by a reference (similar to a pointer) to the object’s actual value. If a reference type is assigned to a variable, that variable references (points to) the original value. No copy is made.



1.6 ASP in .NET

Notes

ASP.NET is the unified Web development model that includes the services essential for you to construct enterprise-class Web (apps with minimal coding). ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you get access to classes in the .NET Framework. You can code your applications in any kind of language compatible with the most popular language runtime (CLR), including Microsoft Visual Basic as well as C#. These languages allow you to develop ASP.NET programs that benefit from the common language runtime, type security, inheritance, and so on.

1.6.1 Many Faces of ASP.NET

With ASP.NET 3.5, Microsoft aims to continue its achievement through calming and attractive ASP.NET. The good news is that Microsoft has not removed features, replaced functionality, or reversed direction. Rather, almost all the modifications add higher-level functions that can make your programming more productive.

All in all, there have been four major releases of ASP.NET:

- **ASP.NET 1.0:** The initial release of ASP.NET provides a huge range of essential features.
 - ❖ Object-oriented Web application development supporting inheritance, polymorphism and other standard OOP features
 - ❖ Developers are no longer forced to use `Server.CreateObject(...)`, so early-binding and type safety are possible.
 - ❖ Based on Windows programming; the developer can make use of DLL class libraries and other features of the Web server to build more robust applications that do more than simply rendering HTML (e.g. exception handling)
- **ASP.NET 1.1:** The second release of framework does not consist any new feature but added performance tune ups and bug fixes:
 - ❖ Mobile controls
 - ❖ Automatic input validation
- **ASP.NET 2.0:** This third release piled on a huge set of new features, all of which were built on top of the existing ASP.NET plumbing.
 - ❖ New data controls (GridView, FormView, DetailsView)
 - ❖ New technique for declarative data access (SqlDataSource, ObjectDataSource, XmlDataSource controls)
 - ❖ Navigation controls
 - ❖ Master pages
 - ❖ Login controls
 - ❖ Themes
 - ❖ Skins
 - ❖ Web parts
 - ❖ Personalization services
 - ❖ Full pre-compilation

Notes

- ❖ New localization technique
- ❖ Support for 64-bit processors
- ❖ Provider class model
- **ASP.NET 3.5:** This fourth release keeps the same basic engine as ASP.NET 2.0, but adds a few frills and two more dramatic changes. The most significant enhancements were as follows:
 - ❖ New data controls (ListView, DataPager)
 - ❖ ASP.NET AJAX included as part of the framework
 - ❖ Support for HTTP pipelining and syndication feeds.
 - ❖ WCF support for RSS, JSON, POX and Partial Trust
 - ❖ All the .NET Framework 3.5 changes, like LINQ etc.



Did u know? ASP.NET 1.0 was released on January 5, 2002 as part of version 1.0 of the .NET Framework.

1.7 Problems with Traditional ASP

In today's scenario of Web Application Development ASP consists many problems:

1. **Interpreted and Loosely Typed Code:** Jscript or VBscript is generally used for scripting code in ASP. ASP execution engine interprets code line by line, every time the page is called. In addition, although it supports variables but they are all loosely typed as variants and bound to particular types only when the code is executed. Both these factors hamper performance, and late binding of types makes error catching harder when you are writing code.
2. **Mixes Layout (HTML) and Logic (Scripting Code):** Files in ASP are generally a combination of script code with HTML resulting in a ASP script which is lengthy, difficult to read, and switch frequently between code and HTML. The combination of HTML with ASP is problematic for larger applications, where content is required to be kept separate from business logic.
3. **Limited Development and Debugging Tools:** Microsoft Visual InterDev, Macromedia Visual UltraDev, along with other tools have attempted to increase the productivity of ASP programmers by providing graphical development environments. However, these power tools never achieved the simplicity of use or the level of acceptance achieved by Microsoft Windows application development resources, such as Visual Basic or Microsoft Access. ASP programmers still rely seriously or entirely on Note pad.

Debugging is an unavoidable part of any software program development process, and the actual debugging tools for ASP have been minimal. Most ASP programmers resort to embedding temporary Response. Write statements in their code to trace the progress of its execution.

4. **Stateless Connection:** Client-server applications, once logged in, maintain a persistent connection between the user's program, the database and any other resources required by the program. Web programming in general and ASP in particular follow a "stateless" concept in which the server treats every request as if none preceded it. Though ASP has session variables, cookies and other ways to keep track of the information in a user's login

session, the stateless model complicates the programmer's job. For example, every time a user requests a new Web page or refreshes an existing one, the page program must log in to the database with a user ID and password. The stateless model prevents the database from keeping the user logged in throughout the session.

5. **Vulnerabilities:** Web programs in general and ASP in particular are vulnerable to a variety of exploits. For example, ASP programs must carefully check user input for database commands embedded in user logins and other text input fields; otherwise, a malicious user could use this vulnerability to probe the database for confidential information or possibly delete the database itself.



Caution If your Web application makes use of components, copying new files to your application should only be done when the Web server is stopped. Otherwise it is like pulling the rug from under your application's feet, because the components may be in use and locked.

1.8 Advantages of ASP.NET

This server technology is an important contribution to the development of the network. Amazon.com, eBay.com, and many other popular sites use ASP.NET as the basis for your site, would not be possible without ASP.NET.

ASP.NET provides important advantages compared with other models of web application development:

1. ASP.NET dramatically reduces the amount of code needed to build large applications.
2. With built-in Windows authentication and application settings, and applications safe and secure.
3. It provides higher performance by using early binding, just-in-time compilation, native optimization, and caching services right out of the box.
4. ASP.NET framework is complemented by a rich and designer tools in Visual Studio integrated development environment. WYSIWYG-editing, drag and drop controls, firewall and automatic deployment are just some of the features of this powerful tool provides.
5. ASP.NET provides a simple and easy to perform common tasks, from simple form submission and client authentication configuration and deployment site.
6. The source code and HTML together, so that ASP.NET pages are easy to maintain and write. In addition, the source code is executed on the server. It provides greater power and flexibility to Web pages.
7. All processes are carefully controlled and managed by ASP.NET, so that if the process is dead, the new process can be created in its place, which helps to keep your application constantly available to handle requests.
8. This is purely server-side technologies, so that the ASP.NET code runs on the server before being sent to the browser.
9. Be independent of the language, it allows you to select the language that best applies to your application or partition applications in many languages.
10. ASP.NET makes for easy deployment. No need to register components because the configuration information is embedded.

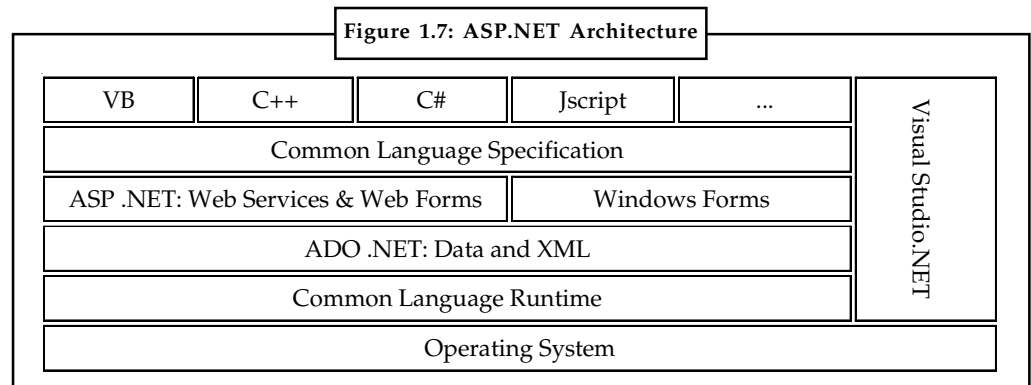
Notes

11. The Web server continuously monitors the pages, components and applications running on it. If he notices any memory leaks, infinite loops, other illegal activities, which immediately destroys the activity and restarts.
12. It is easy to work with ADO.NET using data binding and formatting of the page. This is an application that works faster and counters large volumes of users without performance problems.

In short ASP.NET, the next generation version of Microsoft’s ASP, is a programming framework used to generate enterprise-class sites, web applications, and technologies. ASP.NET developed applications are available on a worldwide basis leading to efficient knowledge management. Whether you are building a small business web-site or a huge corporate web application distributed across multiple networks, ASP.NET will provide you all the features you could possibly require and at an affordable cost, i.e., FREE!

1.9 ASP.NET Architecture

ASP.NET is based on the .NET Framework architecture. Visual studio Environment provides a uniform way to combine number of features in this Architecture.



Architecture is explained from under side to top in the below paragraphs:

1. At the foundation of the Architecture is Common Language Runtime. .NET Framework common language runtime resides on top of the operating system services and acts as an execution engine for .NET applications and serves as the interface between .NET applications and the operating system. The CLR provides many services such as:
 - ❖ Loads and executes code
 - ❖ Converts intermediate language to native machine code
 - ❖ Separates processes and memory
 - ❖ Manages memory and objects
 - ❖ Enforces code and access security
 - ❖ Handles exceptions
 - ❖ Interfaces between managed code, COM objects, and DLLs
 - ❖ Provides type-checking
 - ❖ Provides code metadata (Reflection)
 - ❖ Provides profiling, debugging, etc.

2. Framework class library sits on top of the common language runtime and is really a collection of over 7000 classes and data types that enable .NET applications to read and create files, access databases, process XML, display a graphical user interface, draw graphics, use Web services, etc. The FCL wraps much of the massive, complex Win32 API into more simple .NET objects that may be used by C# and other .NET programming languages.
3. Much of the underlying integration of .NET is accomplished with XML:
 - ❖ Web Services depend completely on XML for interfacing with remote objects.
 - ❖ The information about execution modules, called assemblies, can be exported as XML.
 - ❖ ADO.NET, the successor to ADO, is heavily dependent on XML for remote representation of data. Essentially, when ADO.NET creates what it calls a DataSet (a more complex successor to a recordset), the data is converted to XML for manipulation by ADO.NET. Then the changes to that XML are posted back to the data store by ADO.NET when remote manipulation is finished.

4. The 4th layer of the framework consists of the Windows forms and, in parallel, the ASP.NET. ASP. NET-includes Web Forms and Web Services.

Web Forms brings drag-and-drop design and an event-driven architecture to Web-based interfaces, implementing a programming model that is much like standard of VB6 forms-dependent development. User interfaces created with Web Forms also possess built-in browser self-reliance and state management.

Internet Services allow remote elements, possibly running on completely different operating systems, to end up being invoked and used. This capability for communications as well as interoperability with remote elements over the Internet serves as the mechanism by which highly-distributed applications could be built, going far beyond what is feasible along with existing technologies like DCOM.

5. .NET is amalgamation and interoperability between different programming languages. In order to achieve this there is a need for certain rules that must be laid and all the languages follow them. The CLS is a common platform that integrates code and components from multiple .NET programming languages. In other words, a .NET application can be written in multiple programming languages with no extra work by the developer (though converting code between languages can be tricky).

.NET includes new object-oriented programming languages such as C#, Visual Basic .NET, J# (a Java clone) and Managed C++. These languages, plus other experimental languages like F#, all compile to the Common Language Specification and can work together in the same application.

File Name Extensions

Applications coded in ASP.NET will comprise of many files with different file name extensions. The most common are listed below:

.aspx : Global.aspx, used for application-level logic

.ascx : Web UserControls: custom controls to be placed onto web pages.

.ashx : custom HTTP handlers.

.asmx : web service pages. From version 2.0 a Code behind page of an asmx file is placed into the app_code folder.

Notes

.axd : when enabled in web.config requesting trace.axd outputs application-level tracing. Also used for the special webresource.axd handler which allows control/component developers to package a component/control complete with images, script, css etc. for deployment in a single file (an 'assembly').

.config : web.config is the only file in a specific Web application to use this extension by default (machine.config similarly affects the entire Web server and all applications on it), however ASP.NET provides facilities to create and consume other config files. These are stored in XML format.

.cs/vb : Code files (cs indicates C#, vb indicates Visual Basic). Code behind files (see above) predominantly have the extension ".aspx.cs" or ".aspx.vb" for the two most common languages. Other code files (often containing common "library" classes) can also exist in the web folders with the cs/vb extension. In ASP.NET 2 these should be placed inside the App_Code folder where they are dynamically compiled and available to the whole application.

.dbml : LINQ to SQL data classes file.

.master : 2.0 master page file.

.resx : resource files for internationalization and localization. Resource files can be global (e.g. messages) or "local" which means specific for a single aspx or ascx file.

.sitemap : sitemap configuration files. Default file name is web.sitemap.

.skin : theme skin files.

.svc : Windows Communication Foundation service file.

Directives

A directive is special instructions on how ASP.NET should process the page. The most common directive is <%@ Page %> which can specify many attributes used by the ASP.NET page parser and compiler. For each directive you can set different attributes.

Code Declaration Blocks

Code-declaration blocks define sections of server code that are embedded in ASP.NET application files within <script> blocks marked with a **runat="server"** attribute, which tells ASP.NET that these controls can be accessed on the server and on the client. Optionally you can specify the language for the block.

Code Render Blocks

Code render blocks define inline code or inline expressions that execute when the page is rendered. There are two styles of code render blocks: inline code and inline expressions. Use inline code to define self-contained lines or blocks of code. Use inline expressions as a shortcut for calling the Write method.

HTML Control Syntax

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a **runat="server"** attribute to the HTML element. This attribute indicates that the element should be treated as a server control.

- All HTML server controls must be within a <form> tag with the **runat="server"** attribute!
- ASP.NET requires that all HTML elements must be properly closed and properly nested.

Custom Control Syntax

Notes

Custom controls are deployed as individual assemblies. They are compiled into a dynamic link library (DLL) and used as any other ASP.Net server control. They could be created in either of the following way:

- By deriving a custom control from an existing control
- By composing a new custom control combining two or more existing controls
- By deriving from the base control class

Data Binding Expression

Data-binding expressions are contained within `<%#` and `%>` delimiters and use the Eval and Bind functions. The Eval function is used to define one-way (read-only) binding. The Bind function is used for two-way (updatable) binding. In addition to calling Eval and Bind methods to perform data binding in a data-binding expression, you can call any publicly scoped code within the `<%#` and `%>` delimiters to execute that code and return a value during page processing.

Data-binding expressions are resolved when the DataBind method of a control or of the Page class is called. For controls such as the GridView, DetailsView, and FormView controls, data-binding expressions are resolved automatically during the control's PreRender event and you are not required to call the DataBind method explicitly.

Server-side Object Tags

Declares and creates COM and .NET objects in a Web Forms page. When the ASP.NET page parser encounters a server-side object tag in an .aspx file, it generates a read-only property on the page, using the id attribute of the tag as the property name. The read property is then configured to create an instance of the object on first use. The resulting instance is not added as an object within the page's hierarchical server control tree; it is instead treated as a non-UI variable declaration.

The classid, progid, and class attributes are mutually exclusive. You cannot include more than one of these attributes in a single server-side object tag. You can, however, include multiple server-side object tags on a Web Forms page and use these attributes in different tags.

Server-side Include Directives

Inserts the contents of a specified file anywhere within an ASP.NET page:

- The file name value for the included file must be enclosed in quotation marks (" ").
- The included file is processed before any dynamic code is executed.
- The `#include` tag must be enclosed within HTML/XML comment delimiters to avoid being interpreted as literal text.

Server-side Comments

Allows you to include code comments in the body of an .aspx file. Any content between opening and closing tags of server-side comment elements, whether ASP.NET code or literal text, will not be processed on the server or rendered to the resulting page.

Notes



Task Prepare a list to show all the languages used in .NET environment.



Case Study

Switching from Java to .NET

CoreLab Partners is a multinational company providing science-driven services and cutting-edge technology to support clinical trials for drug development using image-based analysis and tools.

As a pioneer in this niche market, CoreLab found that no real “off-the-shelf” solutions met their needs. CoreLab’s IT team, therefore, is responsible for creating entirely new applications via custom coding and the integration of other systems. As a result, they have a large stable of custom applications and a comprehensive set of development, validation, and staging environments. The team is continuously innovating to stay on the forefront of the industry.

After an unsuccessful attempt to develop a mission-critical imaging application (including an interface to a system provided by Siemens Corporate Research) in Java, CoreLab decided to move the development of this application to the Microsoft .NET Framework, using the Visual Studio IDE and Visual Studio Team Foundation Server. These tools have since become integral to CoreLab’s IT strategy, and CoreLab is currently evaluating expanding the platform to include Microsoft SharePoint to build upon the innovation capabilities offered by Visual Studio and .NET.

Situation

CoreLab Partners, Inc., was branded in February 2010 when RadPharm, Inc., merged with Medifacts International. The combined company provides comprehensive medical image assessment and cardiac safety services to the pharmaceutical, biotechnology, and medical device industries. As such, CoreLab Partners’ IT team drives a large portfolio of custom development projects for their core imaging and clinical trial systems, other line of business applications for finance, and even marketing.

One of the most critical recent custom development projects was the development of an application that would enable integration of CoreLab’s image repository and Clinical Trial Management Systems (CTMS) with the platform of one of CoreLab’s key partners – Siemens Corporate Research. CoreLab initially decided to develop the interface (called WorkList Server) on Apache Axis2 and Java as most of their internal development skills were on the Java platform.

The development started in March 2009 and the intention was to finish the interface within six months. With each iteration and review meeting, CoreLab found itself facing new obstacles and missing deadlines. Three months into the project, it became apparent that the Java development environment was not providing the functionalities and the support needed to build this interface. As Mr. Guindi explained, “We discovered that with Java it was quite challenging to find the right documentation and support in order to successfully develop this application.”

Questions:

1. Discuss the initial problem of working with JAVA.
2. What are the benefits of switching the project to ASP.NET?

Self Assessment

Notes

Multiple Choice Questions:

6. Which of the following components of the .NET framework provide an extensible set of classes that can be used by any .NET compliant programming language?
 - a. .NET class libraries
 - b. Common Language Runtime
 - c. Common Language Infrastructure
 - d. Component Object Model
 - e. Common Type System
7. Code that targets the Common Language Runtime is known as
 - a. Unmanaged
 - b. Distributed
 - c. Legacy
 - d. Managed Code
 - e. Native Code
8. Common language specification (CLS)
 - a. is an execution engine for all .net application
 - b. is similar to JVM as in Java
 - c. defines standard rules for defining .Net compliant languages
 - d. is a compiler
9. Which of the following is not a .NET compatible language?
 - a. C#
 - b. J#
 - c. VB.NET
 - d. Java
10. Which statement is not correct?
 - a. CLR is an execution engine of .NET
 - b. Assembly is a logical unit of deployment
 - c. CLR executes managed code
 - d. .Net provides cross language interoperability using code access security
11. CLR component which allows .net application exchange data with COM application?
 - a. Framework Class library
 - b. COM marshaller
 - c. Class loader
 - d. Thread support
12. Features provided by Automatic Memory management in .NET
 - a. Allocating memory
 - b. Releasing memory
 - c. Implementing finalizers
 - d. All of the above
13. The .Net class library:
 - a. contains over 25,000 classes.
 - b. uses namespaces to manage all of the classes.
 - c. has the System.Form namespace for classes used in Windows-based application.
 - d. Both a and b.

Notes

14. Which is the file extension used for an ASP.NET file?
- a. asn
 - b. asp
 - c. aspn
 - d. aspx
 - e. asx
15. It is best to use a web instead of a windows application when the application:
- a. has a thin front end (client).
 - b. needs to be available to the public.
 - c. must be platform-independent.
 - d. Both a and b.
 - e. All of the above.

1.10 Summary

- Microsoft.Net Framework is a programming infrastructure created by Microsoft for building, deploying, and running applications and services that use .NET technologies, such as desktop applications and Webservice.
- Features of .Net includes Easy Development, OOPs support, Multi-language support and Compatibility of COM and COM+.
- Benefits of Using .NET Architecture – Faster Development, greater scalability, fewer bugs and potentially better performance.
- .NET is a framework that covers all the layers of software development above the Operating System. It provides the richest level of integration among presentation technologies, component technologies, and data technologies ever seen on Microsoft, or perhaps any, platform.
- ASP.NET supports N-Tier Architecture.
- .Net consists – .Net Framework, .Net products and .Net services.
- The Common Language Runtime (CLR) is an Execution Environment for .Net.
- Most significant features provided to .NET applications by the CLR are Memory Management and Common Type system.
- ASP.NET comes with built-in WebForms controls, which are responsible for generating the user interface.
- CTS support two types category value types and reference types.
- There are currently over 15 language compilers being built by Microsoft.

1.11 Keywords

CLR (Common Language Runtime): It manages the execution of code and provides services that make the development process easier.

CLS (Common Language Specification): CLS is a set of basic language features that .Net Languages needed to develop Applications and Services, which are compatible with the .Net Framework.

CTS (Common Type System): It defines how data types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for the Cross Language Integration.

DLLHell: This refers to the set of problems caused when multiple applications attempt to share a common component like a dynamic link library (DLL) or a Component Object Model (COM) class.

FCL (Framework Class Library): It sits on top of the common language runtime and is really a collection of over 7000 classes and data types that enable .NET applications to read and create files, access databases, process XML, display a graphical user interface, draw graphics, use Web services, etc.

GC (Garbage Collection): Clean up the unused Objects and buffers, Exception Handling, Common Type System (CTS) that is all .NET language that conforms to the Common Language Specification (CLS) have the same primitive Data, Types, Code.

JIT (Just In Time): Compilation converts Intermediate Language (MSIL) to native code on demand at application run time.



Lab Exercise

1. Search about the CLR architecture.
2. Prepare a flow chart for debugging a program by CLR.

1.12 Review Questions

1. Define .NET.
2. Give the reasons behind the genesis of .NET.
3. What are the limitations/drawbacks of C, C++, Java, MFC and Windows DNA?
4. Discuss the Core Features of .NET.
5. Discuss the role of CLR, CTS, CLS and Base Class Libraries with the help of block diagram.
6. Explain the role of Garbage Collector in Automatic Memory management.
7. What are the "Primitive Types" in .NET?
8. What is DLL Hell problem and How .Net resolve it?
9. What is the significance of CIL or IL for the execution of any .NET aware language?
10. How is NET framework different from other programming environments like COM, C++, VB6, Java, etc.?

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (a) | 2. (b) | 3. (a) | 4. (d) |
| 5. (a) | 6. (a) | 7. (d) | 8. (c) |
| 9. (d) | 10. (d) | 11. (b) | 12. (d) |
| 13. (d) | 14. (d) | 15. (e) | |

Notes

1.13 Further Readings



Books

ASP.NET, by Yashwanth Kanethkar

Practical ASP. NET 3.5 Projects for Beginners, by B. M. Harwani



Online link

<http://www.w3schools.com/aspnet/default.asp>

Unit 2: Introduction to C#

Notes

CONTENTS

Objectives

Introduction

- 2.1 A Simple Web Page
 - 2.1.1 Types of .NET frameworks: Microsoft's and Mono's
 - 2.1.2 Syntax
 - 2.1.3 Variables
 - 2.1.4 Types
- 2.2 Web Control
 - 2.2.1 Adding a Web User Control
 - 2.2.2 Advantages of a Web User Control
 - 2.2.3 Drawbacks/Disadvantages
- 2.3 Introduction to In-line Script
- 2.4 The Page Class
 - 2.4.1 Objects
- 2.5 Summary
- 2.6 Keywords
- 2.7 Review Questions
- 2.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Create a simple Web page
- Discuss the Web controls
- Explain the in-line script
- Define the page class

Introduction

C# (pronounced "C sharp") is a simple, modern, object-oriented, and type-safe programming language. It will immediately be familiar to C and C++ programmers. C# combines the high productivity of Rapid Application Development (RAD) languages and the raw power of C++.

Visual C# .NET is Microsoft's C# development tool. It includes an interactive development environment, visual designers for building Windows and Web applications, a compiler, and a debugger. Visual C# .NET is part of a suite of products, called Visual Studio .NET, that also includes Visual Basic .NET, Visual C++ .NET, and the JScript scripting language. All of these languages provide access to the Microsoft .NET Framework, which includes a common execution

Notes

engine and a rich class library. The .NET Framework defines a “Common Language Specification” (CLS), a sort of lingua franca that ensures seamless interoperability between CLS-compliant languages and class libraries. For C# developers, this means that even though C# is a new language, it has complete access to the same rich class libraries that are used by seasoned tools such as Visual Basic .NET and Visual C++ .NET. C# itself does not include a class library.

2.1 A Simple Web Page

To compile your first C# function, one need to install .NET Framework SDK on to his/her computer.

2.1.1 Types of .NET frameworks: Microsoft’s and Mono’s

Microsoft

Windows user needs to download .Net Framework SDK from Microsoft’s .NET Framework Developer Center. If the default Windows directory (the directory where Windows or WinNT is installed) is C:\WINDOWS, the .Net Framework SDK installation places the Visual C# .NET Compiler (csc) in the C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705 directory for version 1.0, the C:\WINDOWS\Microsoft.NET\Framework\v1.1.4322 directory for version 1.1, or the C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727 directory for version 2.0.

Mono

An installer can be downloaded for mono website for Linux and other operating systems.

For Linux, CSCC which is a good compiler and can be downloaded for free from DotGNU portable .Net project page. The compiled programs can then be run with ilrun.

If you are working on Windows then you should add the path to the folders that contain cs.exe or mcs.exe to the Path environment variable by doing his you need not type the full path each time you want to compile.

C#.NET programs can be written with a simple text editor, but in that case you have to compile the code by yourself. Microsoft offers variety of code editing programs under the Visual Studio line that offer syntax highlighting as well as compiling and debugging capabilities. As of now C#.NET can be compiled in Visual Studio 2002 and 2003 (only supports the .NET Framework version 1.0 and 1.1) and Visual Studio 2005 (supports the .NET Framework 2.0 and earlier versions with some tweaking).

The code below will demonstrate a C# program written in a simple text editor. Start by saving the following code to a text file called HelloCSharp.cs:

```
classHelloCSharp
{
staticvoidMain(string[]args)
{
System.Console.WriteLine("Hello from C#");
}
}
```

Having written the C# source code the next step is to compile it down to the Common Intermediate Language (CIL) format. If you have installed the .NET Framework and configured your PATH so that the C# compiler can be found then you are ready to compile. If, on the other hand, you installed Visual Studio you can launch a Command Tool ready compile by selecting *Start->All Programs->Microsoft Visual Studio->Visual Studio Tools->Visual Studio Command Prompt*.

Once a command prompt is displayed, simply enter the following command to compile the sample program:

```
cscHelloCsharp.cs
```

The compiler will display output similar to the following and compile the code to create a *HelloCsharp.exe* executable:

```
C:\Documents and Settings\Neil>cscHelloCsharp.cs
Microsoft (R) Visual C# 2008 Compiler version 3.5.21022.8
for Microsoft (R) .NET Framework version 3.5
Copyright (C) Microsoft Corporation. All rights reserved.
```

If the compiler displays any syntax errors go back to the source and double check you haven't made any typing errors. Remember that C# is case sensitive. A common mistake made by C programmers, for example, is to write *Main* as *main* which will cause the compiler to display a syntax error.

Assuming there are no syntax errors, the next step is to execute the program.

Executing a Compiled C# Program

Assuming the absence of syntax errors, the C# compiler will have created an executable file called *HelloCsharp.exe*. Running this program executable is simply a matter of entering the name at the command prompt:

```
C:\Demo>HelloCsharp
Hello from C#.
```

If you see the "Hello from C#" message then you have successfully written, compiled and executed your first ever C# console program. In the next chapter we will learn how to create a simple GUI based application using Visual Studio.

2.1.2 Syntax

C# syntax and Java syntax looks quite related to each other because both accede too much of their syntax from C and C++. The object-oriented natural of C# requires the high level structure of a C# program to be defined in terms of classes, whose detailed behaviours are defined by their statements.

Statements

Statements are the basic unit which carries out the C# program. A statement can declare a variable, define an expression, perform a simple action by calling a method, control the flow of execution of other statements, create an object, or initialize the variable, property, or field.

- Statements generally end with a semicolon.
- Statements can be grouped into comma-separated statement lists or brace-enclosed statement blocks.



Example:

```
intVar_Number//declaringavariabale Var_Number=5;//assigningavalue
MyClassmyobj=newMyClas();//constructing anewobject
myobj.SampleInstanceMethod();//callinganinstancemethod
MyClass.SampleStaticMethod();//callingastaticmethod
//executinga"for"loopwithanembedded"if"statement
```

Notes

```
for(int i=0;i<upperLimit;i++)
{
if(MyClass.SampleStaticMethodReturningBoolean(i))
{
sum+=myobj.SampleInstanceMethodReturningInteger(i);
}
}
```

Statement Blocks

A block of code is a series of statements surrounded by curly braces. In addition, code blocks helps to limit the scope of variables/objects defined within them. Nested Code blocks often appear as the bodies of methods, the protected statements of a try block, and the code within a related catch block.

Let's say I have defined a method which might cause an exception. The method would look like this (in rather pseudo-code):

```
public int Calculate(int x,int y)
{
try
{
doSomeCalc();
}
catch(SomeException ex)
{
doExceptionHandling();
}
return result;
}
```

Now if another part of the application wants to use this method should it use another try-catch block?

```
public MyMainApp() {
try
{
Calculate(1,2);
}
catch(SomeException ex)
{
doExceptionHandling();
}
}
```

Comments

Comments can be used to document what the program does and what specific blocks or lines of code do. Since the C# compiler ignores comments, you can include them anywhere in a program without affecting your code. The C# compiler ignores explanation.

A comment is preceded by an double forward slash. The code of line after the double forward slash is simply ignored by the compiler.

These are following types of comments in C#

Notes

Single line comments

```
// for single line comments
```

Multiple line comments

```
/* for multi line comments */
```

XML tags comments

```
/// XML tags displayed in a code comment
```

Case Sensitivity

C# is case-sensitive, including its variable and method names.

The variables `myMethod` and `MyMethod` below are distinct because C# is case-sensitive:

```
int myMethod = 3;
```

```
int MyMethod = 5;
```

Compiler error will be generated by the following code.(unless a custom class or variable named

```
consolehasamethodnamedwriteline()):
    //Compilererror!
    Console.WriteLine("Hello"); As in writeline "w " is generating
Compiler Error.
CorrectSyntax is
    Console.WriteLine("Hello");
```

2.1.3 Variables

Variables represent storage locations. Every variable has a type that determines what values can be stored in the variable. Local variables are variables that are declared in methods, properties, or indexers. A local variable is defined by specifying a type name and a declaratory that specifies the variable name and an optional initial value, as in:

```
int a;
int b = 1;
```

A variable must be assigned before its value can be obtained. The example

```
class Test
{
    static void Main() {
        int a;
        int b = 1;
        int c = a + b; // error, a not yet assigned
        ...
    }
}
```

results in a compile-time error because it attempts to use the variable `a` before it is assigned a value.

Notes

Fields

A field is a variable of any type that is declared directly in a class or struct. Fields are members of their containing type.

A class or struct may have instance fields or static fields or both. Instance fields are specific to an instance of a type. If you have a class T, with an instance field F, you can create two objects of type T, and modify the value of F in each object without affecting the value in the other object. By contrast, a static field belongs to the class itself, and is shared among all instances of that class. Changes made from instance A will be visibly immediately to instances B and C if they access the field.

Generally, you should use fields only for variables that have private or protected accessibility. Data that your class exposes to client code should be provided through methods, properties and indexers. By using these constructs for indirect access to internal fields, you can guard against invalid input values. A private field that stores the data exposed by a public property is called a backing store or backing field.

Each field has a visibility of public, protected, internal, and protected internal, or private (from most visible to least visible).

Local Variables

A variable refers to a storage location by a name that the program can later assign and modify. Local indicates that the programmer declared the variable within a method.

To declare a variable is to define it, which you do by

1. Specifying the type of data which the variable will contain
2. Assigning it an identifier (name)

Parameters

Formal parameter declarations also define variables. There are four kinds of parameters: value parameters, reference parameters, output parameters, and parameter arrays.

A value parameter is used for “in” parameter passing, in which the value of an argument is passed into a method, and modifications of the parameter do not impact the original argument. A value parameter refers to its own variable, one that is distinct from the corresponding argument. This variable is initialized by copying the value of the corresponding argument.

A reference parameter is used for “by reference” parameter passing, in which the parameter acts as an alias for a caller-provided argument. A reference parameter does not itself define a variable, but rather refers to the variable of the corresponding argument. Modifications of a reference parameter impact the corresponding argument. A reference parameter is declared with a ref modifier.

An output parameter is similar to a reference parameter, except that the initial value of the caller-provided argument is unimportant. An output parameter is declared with an out modifier.

A parameter array is declared with a parameter modifier. There can be only one parameter array for a given method, and it must always be the last parameter specified. The type of a parameter array is always a single dimensional array type. A caller can either pass a single argument of this array type, or any number of arguments of the element type of this array type.

2.1.4 Types

C# supports two kinds of types: value types and reference types. Value types include simple types (e.g., `char`, `int`, and `float`), enum types, and struct types. Reference types include class types, interface types, delegate types, and array types.

Value types differ from reference types in that variables of the value types directly contain their data, whereas variables of the reference types store references to objects. With reference types, it is possible for two variables to reference the same object, and thus possible for operations on one variable to affect the object referenced by the other variable. With value types, the variables each have their own copy of the data, and it is not possible for operations on one to affect the other.

The predefined reference types are `object` and `string`. The type `object` is the ultimate base type of all other types. The type `string` is used to represent Unicode string values. Values of type `string` are immutable.

The predefined value types include signed and unsigned integral types, floating point types, and the types `bool`, `char`, and `decimal`. The signed integral types are `sbyte`, `short`, `int`, and `long`; the unsigned integral types are `byte`, `ushort`, `uint`, and `ulong`; and the floating point types are `float` and `double`.

Integral Types

In C#, an *integral* is a category of types. For any one confused because the word *Integral* sounds like a mathematical term, from the perspective of C# programming, these are actually defined as Integral types in the C# programming language specification. They are whole numbers, either signed or unsigned, and the `char` type. The `char` type is a Unicode character, as defined by the Unicode Standard. Table 2.1 shows the integral types, their size, and range.

Table 2.1: The Size and Range of C# Integral Types

Type	Size (in bits)	Range
<code>sbyte</code>	8	-128 to 127
<code>byte</code>	8	0 to 255
<code>short</code>	16	-32768 to 32767
<code>ushort</code>	16	0 to 65535
<code>int</code>	32	-2147483648 to 2147483647
<code>uint</code>	32	0 to 4294967295
<code>long</code>	64	-9223372036854775808 to 9223372036854775807
<code>ulong</code>	64	0 to 18446744073709551615
<code>char</code>	16	0 to 65535

Integral types are well suited for those operations involving whole number calculations. The `char` type is the exception, representing a single Unicode character. As you can see from the table above, you have a wide range of options to choose from, depending on your requirements.

Floating Point and Decimal Types

A C# floating point type is either a `float` or `double`. They are used any time you need to represent a real number, as defined by IEEE 754. For more information on IEEE 754, visit the IEEE Web

Notes

Site. Decimal types should be used when representing financial or money values. Table 2.2 shows the floating point and decimal types, their size, precision, and range.

Table 2.2: The Floating Point and Decimal Types with Size, Precision, and Range

Type	Size (in bits)	Precision	Range
float	32	7 digits	1.5×10^{-45} to 3.4×10^{38}
double	64	15–16 digits	5.0×10^{-324} to 1.7×10^{308}
decimal	128	28–29 decimal places	1.0×10^{-28} to 7.9×10^{28}

Floating point types are used when you need to perform operations requiring fractional representations. However, for financial calculations, the *decimal* type is the best choice because you can avoid rounding errors.

The string Type

A string is a sequence of text characters. You typically create a string with a string literal, enclosed in quotes: "This is an example of a string." You've seen strings being used in Unit 1, where we used the *Console.WriteLine* method to send output to the console.

Some characters aren't printable, but you still need to use them in strings. Therefore, C# has a special syntax where characters can be escaped to represent non-printable characters. For example, it is common to use newlines in text, which is represented by the '\n' char. The backslash, '\', represents the escape. When preceded by the escape character, the 'n' is no longer interpreted as an alphabetical character, but now represents a newline.

You may be now wondering how you could represent a backslash character in your code. We have to escape that too by typing two backslashes, as in '\\'. Table 2.3 shows a list of common escape sequences.

Table 2.3: C# Character Escape Sequences

Escape Sequence	Meaning
\'	Single Quote
\"	Double Quote
\\	Backslash
\0	Null, not the same as the C# <i>null</i> value
\a	Bell
\b	Backspace
\f	form Feed
\n	Newline
\r	Carriage Return
\t	Horizontal Tab
\v	Vertical Tab

Another useful feature of C# strings is the verbatim literal, which is a string with a @ symbol prefix, as in @"Some string". Verbatim literals make escape sequences translate as normal characters to enhance readability. To appreciate the value of verbatim literals, consider a path statement

such as `"c:\\topdir\\subdir\\subdir\\myapp.exe"`. As you can see, the backslashes are escaped, causing the string to be less readable. You can improve the string with a verbatim literal, like this: `@"c:\\topdir\\subdir\\subdir\\myapp.exe"`.

That is fine, but now you have the problem where quoting text is not as easy. In that case, you would specify double quotes. For example, the string `"copy \\c:\\source file name with spaces.txt\\c:\\newfilename.txt"` would be written as the verbatim literal `@"copy ""c:\\source file name with spaces.txt"" c:\\newfilename.txt"`.

The following illustrates the cross-language compatibility of types by comparing C# code with the equivalent Visual Basic .NET code:

```
// C#
public void UsingCSharpTypeAlias()
{
    int i = 42;
}

public void EquivalentCodeWithoutAlias()
{
    System.Int32 i = 42;
}

' Visual Basic .NET
Public Sub UsingVisualBasicTypeAlias()
    Dim i As Integer = 42
End Sub

Public Sub EquivalentCodeWithoutAlias()
    Dim i As System.Int32 = 42
End Sub
```

Using the language-specific type aliases is often considered more readable than using the fully-qualified .NET Framework type names.

The fact that each C# type corresponds to a type in the unified type system gives each *value type* a consistent size across platforms and compilers. That consistency is an important distinction from other languages such as C, where, e.g. a **long** is only guaranteed to be *at least as large as an int*, and is implemented with different sizes by different compilers. As *reference types*, variables of types derived from **object** (i.e. any **class**) are exempt from the consistent size requirement. That is, the size of *reference types* like `System.IntPtr`, as opposed to *value types* like `System.Int32`, may vary by platform. Fortunately, there is rarely a need to know the actual size of a *reference type*.

There are two predefined *reference types*: **object**, an alias for the `System.Object` class, from which all other reference types derive; and **string**, an alias for the `System.String` class. C# likewise has several integral value types, each an alias to a corresponding value type in the `System` namespace of the .NET Framework. The predefined C# type aliases expose the methods of the underlying .NET Framework types. For example, since the .NET Framework's `System.Int32` type implements a `ToString()` method to convert the value of an integer to its string representation, C#'s **int** type exposes that method:

```
int i = 97;
string s = i.ToString(); // The value of s is now the string "97".
Likewise, the System.Int32 type implements the Parse() method, which can
therefore be accessed via C#'s int type:
string s = "97";
int i = int.Parse(s); // The value of i is now the integer 97.
```

Notes

The unified type system is enhanced by the ability to convert value types to reference types (*boxing*) and likewise to convert certain reference types to their corresponding value types (*unboxing*). This is also known as *casting*.

```
objectboxedInteger=97;
intunboxedInteger=(int)boxedInteger;
```

Boxing and casting are, however, not type-safe: the compiler won't generate an error if the programmer mixes up the types. In the following short example the mistake is quite obvious, but in complex programs it may be very difficult to spot. Avoid boxing, if possible.

```
objectgetInteger="97";
intanInteger=(int)getInteger;
```



Did u know? The name "C sharp" was inspired by musical notation where a sharp indicates that the written note should be made a semitone higher in pitch. This is similar to the language name of C++, where "++" indicates that a variable should be incremented by 1.



Task Write a program to show the example of boxing and unboxing.



Caution Using unboxing can be cause of decreasing or loss the value of data.

2.2 Web Control

Web User control or the .ascx file in asp.net is a replacement for the include file feature in ASP. It actually is a positive evolution from the asp model. Classic asp had used .inc file and included it wherever we needed it. Though this model is also similar, Web User Controls have a lot of other features added along with them.

Web user controls are derived from System.Web.UI.UserControl namespace. These controls once created, can be added to the aspx page either at design time or programmatically during run time. But they lack the design time support of setting the properties created along with the control. They cannot run on their own and they need to stand on another platform like an aspx page. Even if we try to load it on a web browser, IIS will not serve the files of the type .ascx.

2.2.1 Adding a Web User Control

- Create new web application.
- Right click on the project in the Project Explorer and click Add – Add New Item and select a Web User Control item. This adds a file with the extension .ascx to the project. This is the file that the user control will use to expose its interface. An .ASCX file cannot be viewed directly in the browser. It will need to be placed within a container (such as another Web form) to be viewed. The following line is added in the .ascx file:

```
<%@ Control Language="C#" AutoEventWireup="true"
CodeFile="ItemSummary.ascx.cs" Inherits="Controls_ItemSummary"
ClassName="Controls_ItemSummary" %>
```

Add the following code to .ascx file after the above line:

Notes

```
<table cellpadding="0" cellspacing="0" runat="server"
  width="100%" id="tblUC">
  <tr>
  <td width="15%">
  <asp:Image ID="imgItem" runat="server"
  AlternateText="No ImageAvaliable"/>
  </td>
  <td>
  <asp:Label ID="lblSummary" runat="server"/>
  </td>
  </tr>
</table>
```

Now open the .ascx.cs file and add the properties for the control.

```
protected override void OnPreRender(EventArgs e)
{
  imgItem.ImageUrl = itemImageUrl; // sets the Image
  lblSummary.Text = itemSummary; //sets text for the summary
}

// to get item summary

private string itemSummary;
public string ItemSummary
{
  get{ return itemSummary; }
  set{ itemSummary = value; }
}

// to get set item image URL

private string itemImageUrl;
public string ItemImageUrl
{
  get{return itemImageUrl;}
  set{itemImageUrl = value; }
}
```

Open your solution explorer and drag the user Control to a web page in designer view. This will add the following code in the aspx file, which actually registers the control on the aspx page.

```
<%@ Register Src="Controls/ItemSummary.ascx" TagName="ItemSummary"
  TagPrefix="ucl" %>
```

Once the control is registered on the page, you can freely add many instances of the controls to the page.

- **TagPrefix:** Specifies the prefix to use for the control(s). It is sort of like a namespace in which several controls might share a single prefix value. This is why all ASP.NET server controls specify the asp prefix. The only difference is that the ASP.NET server control directive is implied and not explicitly declared.

Notes

- **TagName:** Specifies the name of the control.
- **Src:** Location of the control.
- **runat='server':** To manipulate the user control programmatically. Otherwise, only the raw HTML is sent back to the browser.

The following code is added within the form tags of the User control container:

```
<uc1:ItemSummary ID="ItemSummary1" runat="server"
    ItemSummary="My Summary goes here"
    ItemImageUrl="~/App_Themes/Images/monitor.gif" />
```

2.2.2 Advantages of a Web User Control

The biggest advantage of the Web User controls is that they can be created as a site template and used throughout the site. For example they can be made to contain the Menu/Link structure of a site and can be used at all the other aspx pages.

- Another advantage of ASP.NET User Controls is that it allows you to save a part of the web form and reuse it many other web forms. This drastically reduces the developers' time.
- ASP.NET User Controls are self-contained entities that are saved in an independent file. You can relate a user control with a "black box". ASP.NET User Controls are very helpful when you want to use functionality on multiple pages of a web-application.
- As ASP.NET User Controls are self-contained the developers working on the same project need not worry about transgressing on other's code. ASP.NET User Controls offer a great deal of functionality than Server-side Includes (SSIs). Using SSIs in encapsulating the site functionality is a quite tedious task when compared to ASP.NET User Controls. You can create a user control by using the Register directive. This process is called registering the user control. For instance, you can use this code to register a user control:

```
<%@ Register TagPrefix="sampNamespace" TagName="Myname" Src="mypage.ascx"
%>
```

- The TagPrefix is the exclusive namespace provided to the user control so that multiple ASP.NET User Controls with similar name can be discriminated from each other. The TagName is the name of the user control and the Src is the virtual path to the user control. After registering the user control you can place the tag of the user control in an appropriate location on the web form. You have to include the runat="server" attribute with the tag.

2.2.3 Drawbacks/Disadvantages

Though the User controls offer a flexibility of having site wide modifications, if the whole structure of the site changes, the HTML/aspx code of all the pages should be modified. But as long as the site maintains a same layout, then Web User controls is the number one choice for maintaining the generic layout of the site. But it has some disadvantages:

- Limited support for consumers who use a visual design tool.
- A separate copy of the control is required in each application.
- Cannot be added to the Toolbox in Visual Studio.
- Good for static layout only not for dynamic layout.

Self Assessment**Notes**

Multiple Choice Questions:

1. Which of the following is not value types?
 - a. Integer
 - b. Single
 - c. String
 - d. Long
2. Which of the following statements is correct?
 - a. Information is never lost during narrowing conversions.
 - b. The CInteger() function can be used to convert a Single to an Integer.
 - c. Widening conversions take place automatically.
 - d. Assigning an Integer to an Object type is known as Unboxing.
3. Which of the following statement correctly assigns a value 33 to a variable c?
byte a = 11, b = 22, c;
 - a. c = (byte) (a + b);
 - b. c = (byte) a + (byte) b;
 - c. c = (int) a + (int) b;
 - d. c = (int)(a + b);
4. Which of the following is the correct default value of a Boolean type?
 - a. 0
 - b. 1
 - c. True
 - d. False
 - e. -1
5. Which of the following does not store a sign?
 - a. Short
 - b. Integer
 - c. Long
 - d. Byte
 - e. Single

2.3 Introduction to In-line Script

The default model for adding code to an ASP.NET Web page is to either create a code-behind class file (a code-behind page) or to write the page's code in a script block with the attribute `runat="server"` (a single-file page). The difference between them is that in in-line code the presentation logic and business logic is placed within the same page is called in-line code. The business logic embedded with in `<script>` tags. Whereas in code-behind The presentation logic and business logic are separated with two file that is called code behind. Like presentation logic means .aspx file and business logic means .cs file.

Using this syntax, you can use inline code in ASP.NET (.aspx) pages. The server-side code will be automatically compiled by the .NET framework the first time the page is requested on the server. The compiled .dll file is stored in the "Temporary ASP.NET Files" system folder. Changing the code in .aspx files will trigger a new compilation, generating new .dll files. The old .dll files are phased out by the framework and eventually deleted.

```
<%@ Page Language="C#" %>
<script runat=server>
protected String GetTime()
{
```


Notes

```
        return DateTime.Now.ToString("t");
    }
</script>
<html>
<body>
    <form id="form1" runat="server">
        Current server time is <% =GetTime()%>.
    </form>
</body>
```

The following code example shows an embedded code block that displays the value of a public `GetTime()` function inside a `span` element. In embedded code blocks, the syntax `<% = expression %>` is used to resolve an expression and return its value into the block.

Embedded code blocks must be written in the page's default language. For example, if the page's `@ Page` directive contains the attribute `language="VB"`, the page will use the Visual Basic compiler to compile code in any script block marked with `runat="server"` and any in-line code in `<% %>` delimiters.



Task Write an inline code for a "Welcome!" page.

2.4 The Page Class

When an ASP.NET page is requested and renders markup to a browser, ASP.NET creates an instance of a class that represents your page. That class is composed not only of the code that you wrote for the page, but also code that is generated by ASP.NET. This topic provides an overview of the code that is generated by ASP.NET.

Web forms are actually instances of the ASP.NET page class, which is defined in the `System.Web.UI` namespace. The `Page` class is inherited from the `Template Control` class, which in turn is inherited from the `Control` class therefore `Page` class provides useful properties and methods that can be used while writing the code.

2.4.1 Objects

Let us now discuss the various objects of the `Page` class.

Session

The `Session` object provides a reference to an object of the `HttpSessionState` class. The object enables ASP.NET applications to keep the status of the client. The `Session` object also provides access to the session wide cache, which can be used to store information pertaining to the client. The session starts and ends when the client connects and disconnects to a Web site. The session is also terminated if the client remains inactive for a specific period. The default timeout period is 20 minutes.

Application

The `Application` object provides a reference to an object of the `HttpApplicationState` class. The `Application` object is used to access information that is defined for the entire Web application. For example, the connection string used to connect to the database sever can be stored in the `Application` object.

Cache

Notes

Cache object is an instance of the `Caching.Cache` class and stores global information. It is also a name/value collection of objects, but we can set customized expiration policies and dependencies for each item therefore items are automatically deleted when other resources, such as files or database tables, are modified.

Request

The Request object provides a reference to an object of the `HttpRequest` class. The object enables ASP.NET applications to access information sent by the client during a Web request.

You can use the following code snippet to display the client information such as the IP address, browser name, and browser version on the page:

```
void Page_Load()
{
    //Using the Browser property of the Request object to obtain an
    instance //of HttpBrowserCapabilities which contain the details
    about the client //browser
    HttpBrowserCapabilities bc = Request.Browser;
    ///Using the UserHostAddress property of the Request object to
    obtain // //the IP Address of requesting client
    lbl1.Text = "IP Address = " + Request.UserHostAddress;
    //Using instance of HttpBrowserCapabilities for displaying the
    client browser details.
    lbl2.Text = "Browser Name = " + bc.Browser;
    lbl3.Text = " Browser Version = " + bc.Version;
}
```

HttpRequest Properties

- **ApplicationPath:** Gets the ASP.NET application's virtual application root path on the server.
- **PhysicalPath:** Gets the physical file system path corresponding to the requested URL.
- **AnonymousID:** Gets the anonymous identifier for the user, if present.
- **Browser:** Gets or sets information about the requesting client's browser capabilities.
- **ClientCertificate:** Gets the current request's client security certificate.
- **Cookies:** Gets a collection of cookies sent by the client.
- **FilePath:** Gets the virtual path of the current request.
- **Current Execution File Path:** Gets the virtual path of the current request.
- **Form:** Gets a collection of form variables.
- **ServerVariables:** Gets a collection of Web server variables.
- **IsAuthenticated:** Gets a value indicating whether the request has been authenticated.
- **IsSecureConnection:** Gets a value indicating whether the HTTP connection uses secure sockets (that is, HTTPS).
- **IsLocal:** Gets a value indicating whether the request is from the local computer.

Notes

- **QueryString:** Gets the collection of HTTP query string variables.
- **URLReferrer:** Gets information about the URL of the client's previous request that linked to the current URL.
- **UserAgent:** Gets the raw user agent string of the client browser.
- **UserHostAddress:** Gets the IP host address of the remote client.
- **UserHostName:** Gets the DNS name of the remote client.
- **UserLanguages:** Gets a sorted string array of client language preferences.

Response

The Response object provides a reference to an object of the `HttpResponse` class. The object enables ASP.NET applications to send information to the client.

HttpResponse Members

- **BufferOutput:** Gets or sets a value indicating whether to buffer output and send it after the entire page is finished processing.
- **Cache:** Gets the caching policy (expiration time, privacy, vary clauses) of a Web page.
- **ContentType:** Gets or sets the HTTP MIME type of the output stream.
- **Cookies:** Gets the response cookie collection.
- **Expires:** Gets or sets the number of minutes before a page cached on a browser expires. If the user returns to the same page before it expires, the cached version is displayed. `Expires` is provided for compatibility with previous versions of ASP.
- **ExpiresAbsolute:** Gets or sets the absolute date and time at which to remove cached information from the cache. `ExpiresAbsolute` is provided for compatibility with previous versions of ASP.
- **IsClientConnected:** Gets a value indicating whether the client is still connected to the server.
- **OutputStream:** Enables binary output to the outgoing HTTP content body.
- **BinaryWrite:** Writes a string of binary characters to the HTTP output stream.
- **WriteFile:** Overloaded. Writes the specified file directly to an HTTP content output stream.
- **Write:** Overloaded. Writes information to an HTTP output content stream.
- **Redirect:** Overloaded. Redirects a client to a new URL.

Server

The Server object provides a reference to an object of the `HttpServerUtility` class. The object provides methods that can be used to access the methods and properties of the Web server.

MachineName: Gets the server's computer name.

CreateObject(): Creates a server instance of a COM object identified by the object's programmatic identifier (ProgID).

GetLastError: Returns the previous exception.

This retrieves the exemption object for the most just encountered error. This is most commonly used in an application event handler that checks for error conditions.



Did u know? During the development of the .NET Framework, the class libraries were originally written using a managed code compiler system called Simple Managed C (SMC).



Case Study

New Securities – Trading Platform Delivers Rich User Experience and Greater Flexibility

UNX LLC (UNX), which provides trading tools and other services for institutional investors and brokers, wanted to develop a next-generation execution management system (EMS). The company used the Microsoft Visual Studio 2008 development system and the Microsoft .NET Framework 3.5 to build Catalyst, a desktop application for the Windows operating system that UNX's customers can easily adapt to meet their needs. Based on the Windows Presentation Foundation, Catalyst uses the Managed AddIn Framework to run all functionality as loadable add-ins, and it supports scripts in languages such as Ruby and Python through the use of the Dynamic Language Runtime. Through Catalyst, UNX is delivering the performance, flexibility, and reliability that its customers require along with the ability to rapidly extend and customize solutions – in turn helping UNX better meet customer needs and win new business.

Situation

Founded in 1999, UNX provides sophisticated electronic-trading tools and agency brokerage services for institutional investors, hedge funds, brokerage firms, and third-party vendors. The company's solutions streamline trading, helping traders and buy-side investors find the best execution opportunities, minimize market impact, reduce transaction costs, and manage today's fragmented liquidity to their advantage.

From its inception, UNX has employed its extensive knowledge of market structure and advanced programming techniques to develop sophisticated front-end trading tools and back-end services that work together to help institutional traders maximize their performance. Several of the company's core technology solutions are already in their fifth generation, and UNX constantly evaluates new technologies to determine how they can be used to better meet client needs.

UNX set out to develop a next-generation execution management system (EMS) that would provide state-of-the-art execution management and active trading capabilities across a large network of broker-dealers and liquidity venues. "Our clients demand uncompromised performance and reliability, but they also need a very high degree of flexibility so that they can adapt to changing market conditions and structures quickly," says Alex Shindich, Vice President of Product Management at UNX. "To meet those needs, we set out to build a dynamic application framework that uses open standards to achieve a broker-neutral, fully extensible, and highly customizable desktop trading environment."

Questions:

1. Why UNX LLC go for Visual Studio 2008 and not for other technology?
2. Will .NetT application provide Increased Competitive Advantage?

Notes

Self Assessment

Multiple Choice Questions:

6. Where is the compiled .dll file stored ?
 - a. Temp folder
 - b. CON folder
 - c. Temporary asp.net files system folder
 - d. My documents
7. From which class is page class inherited?
 - a. File class
 - b. Session class
 - c. Template control class
 - d. Register class
8. Which object provides a reference to an object of HttpSessionState clas?
 - a. Session
 - b. Application
 - c. Cache
 - d. Request
9. Which of the following property of HttpRequest gets the virtual path of the current request?
 - a. Form
 - b. FilePath
 - c. HttpPath
 - d. Cookies_path
10. Which object is used to provide reference to an object of HttpResponse class?
 - a. Response
 - b. Cache
 - c. Content
 - d. Write

2.5 Summary

- C# is the native language for the .NET Common Language Runtime.
- The object-oriented nature of C# requires the high level structure of a C# program to be defined in terms of classes, whose detailed behaviours are defined by their statements.
- A block of code is a series of statements surrounded by curly braces.
- Variables represent storage locations. Every variable has a type that determines what values can be stored in the variable.
- A field is a variable of any type that is declared directly in a class or struct.
- There are four kinds of parameters: value parameters, reference parameters, output parameters, and parameter arrays.
- C# supports two kinds of types: value types and reference types.
- Value types include simple types (e.g., char, int, and float), enum types, and struct types.
- Reference types include class types, interface types, delegate types, and array types.
- Web User control or the .ascx file in asp .net is a replacement for the include file feature in ASP.

- The biggest advantage of the Web User controls is that they can be created as a site template and used throughout the site.
- A large part of the power of C# comes with the common .NET Framework API, which provides a large set of classes, including ones for encryption, TCP/IP socket programming, and graphics.
- Web user controls are derived from System.Web.UI.UserControl namespace.

2.6 Keywords

Application: It stores data in the form of name/value pairs.

C#: (pronounced "C sharp") is a simple, modern, object-oriented, and type-safe programming language.

Cache: It is also a name/value collection of objects.

Fields: Fields are members of their containing type.

Parameter: Parameters are variables associated with a method.

Request: It represents the values and properties of the HTTP request that caused our page to be loaded.

Response: It represents the web server's response to a client request.

Session: It acts as a global repository to store any type of user specific data that needs to persist between web-page requests.

Statement: The basic unit of execution in a C# program is the statement.

Web User Control: It is a replacement for the include file feature in ASP, once created, can be added to the aspx page either at design time or programmatically during run time.



- Lab Exercise*
1. Write a C# program to show the message.
 2. Create a Web page using HTML.

2.7 Review Questions

1. Which class is at the top of .NET class hierarchy?
2. Is string a value type or a reference type?
3. Explain the difference between Boxing and Unboxing.
4. What's the difference between // comments, /* */ comments and /// comments?
5. What are the characteristics of C#?
6. Define namespace.
7. What is the difference between Object and Instance?
8. What is the use of using statement in C#?
9. What is the difference between Custom Control and User Control?
10. What are the different types of variables in C#?

Notes

Answers: Self Assessment

- | | | | |
|--------|---------|--------|--------|
| 1. (c) | 2. (c) | 3. (a) | 4. (d) |
| 5. (d) | 6. (c) | 7. (c) | 8. (a) |
| 9. (b) | 10. (a) | | |

2.8 Further Readings



Books

Let Us C# (BPB Publications), by Yashawant Kanetkar

Quest - C#.NET Programming, by Asang Dani



Online links

<http://www.homeandlearn.co.uk/csharp/csharp.html>

Unit 3: Server Controls Basic

Notes

CONTENTS

Objectives

Introduction

3.1 PostBack

3.1.1 AutoPostBack Property in ASP.NET

3.1.2 Life Cycle of a Web Page

3.2 Data Binding

3.2.1 Properties of Data Binding

3.3 Web Server Controls

3.3.1 Generalities of ASP.NET Server Controls

3.3.2 Identifying a Server Control

3.3.3 Naming Containers

3.3.4 Themeable Controls

3.3.5 Control State

3.4 Summary

3.5 Keywords

3.6 Review Questions

3.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Define postback
- Explain data binding
- Understand web server controls

Introduction

The HTML server controls are Hypertext Markup Language (HTML) elements that include a `runat=server` attribute. The HTML server controls have the same HTML output and the same properties as their corresponding HTML tags. In addition, HTML server controls provide automatic state management and server-side events. HTML server controls offer the following advantages:

- The HTML server controls map one to one with their corresponding HTML tags.
- When the ASP.NET application is compiled, the HTML server controls with the `runat=server` attribute are compiled into the assembly.
- Most controls include an `OnServerEvent` for the most commonly used event for the control. For example, the `<input type=button>` control has an `OnServerClick` event.

Notes

- The HTML tags that are not implemented as specific HTML server controls can still be used on the server side; however, they are added to the assembly as `HtmlGenericControl`.
- When the ASP.NET page is reposted, the HTML server controls keep their values.

The `System.Web.UI.HtmlControls.HtmlControl` base class contains all of the common properties. HTML server controls derive from this class.

3.1 PostBack

A postback originates from the client browser. Usually one of the controls on the page will be manipulated by the user (a button clicked or dropdown changed, etc.), and this control will initiate a postback. The state of this control, plus all other controls on the page, (known as the View State) is Posted Back to the web server.

Most commonly the postback causes the web server to create an instance of the code behind class of the page that initiated the postback. This page object is then executed within the normal page lifecycle with a slight difference. If you do not redirect the user specifically to another page somewhere during the page lifecycle, the final result of the postback will be the same page displayed to the user again, and then another postback could happen, and so on.

- The state of the controls on the posting back page are available within the context. This will allow you to manipulate the page controls or redirect to another page based on the information there.
- Controls on a web form have events, and therefore event handlers, just like any other controls. The initialisation part of the page lifecycle will execute before the event handler of the control that caused the post back. Therefore the code in the page's `Init` and `Load` event handler will execute before the code in the event handler for the button that the user clicked.
- The value of the `"Page.IsPostBack"` property will be set to `"true"` when the page is executing after a postback, and `"false"` otherwise.
- Technologies like Ajax and MVC have changed the way postbacks work.

3.1.1 AutoPostBack Property in ASP.NET

Autopostback is the mechanism, by which the page will be posted back to the server automatically based on some events in the web controls. In some of the web controls, the property called auto post back, which if set to true, will send the request to the server when an event happens in the control.

Whenever we set autopostback attribute to true in any of the controls, the .net framework will automatically insert few code in to the HTML generated to implement this functionality.

- A Java script method with name `__doPostBack` (eventtarget, eventargument)
- Two Hidden variables with name `__EVENTTARGET` and `__EVENTARGUMENT`
- `OnChange` JavaScript event to the control

We will discuss one by one of these new entries.

a. `__EVENTTARGET` and `__EVENTARGUMENT`

These two controls are added to the HTML source, when ever any autopostback attribute is set to true for any of the web control.

The `__EVENTTARGET` hidden variable will tell the server, which control actually does the server side event firing so that the framework can fire the server side event for that control.

The `__EVENTARGUMENT` variable is used to provide additional event information if needed by the application, which can be accessed in the server.

b. **`__doPostBack (eventtarget, eventargument)`**

Why the framework is inserting this method to the HTML source, and what it really does. This method is inserted to the HTML source to implement the autopostback functionality. This method will submit the form, when ever called. The two parameters in this method i.e. `eventtarget` and `eventargument` do the actual work of selecting the control to fire the event.

`Eventtarget` will contain the name of the control which initiates the post back event, and event arguments will contain the additional parameters needed for the event. (Refer to the previous block).

This method will set the value of the `__EVENTTARGET` hidden variable with the `eventtarget` parameter and `__EVENTARGUMENT` value with the `eventargument` parameter.

The next activity is to submit the form, so that in the server side, the framework will check for the name of the control in the `__EVENTTARGET` hidden variable and will fire the appropriate event.

c. **`OnChange event`**

This event is added by the framework to any of the control where autopostback is set to true, this method will fire the client side `OnChange` event and calls the `__doPostBack` event with the name of the control where the `OnChange` event is happened.

For example, if we set `autopostback = true` to a textfield with id `myTextField`, then the HTML for the test field will look like this.

```
name="myTextField" OnChange="__doPostBack(' myTextField ',' )" id="myTextField"
```

So whenever the `OnChange` event in client occurs, the `doPostBack` method will be called with the name of the Textfield as the first parameter.

This name will be set to the `__EVENTARGUMENT` hidden variable by the `__doPostBack` JavaScript method and the form will be submitted. In the server side the `__EVENTARGUMENT` hidden variable will be checked and will take the textfield name and the server side event for the same will be fired.

`IsPostBack` – When we discuss about autopostback, we should have an understanding of the `IsPostBack` property of Page class. `IsPostBack` property is used by the Page to determine whether the page is posted back from the client. If `IsPostBack` property is false, then the page is loading for the first time, and if true, then the request is because of some event generated by web controls.

`IsPostBack` is used when we want to load some information when the page loads, for example, if we want to load some information from the database and show in the data grid in a page for the first time, then we can load and bind the grid in the `page_load` when `IsPostBack` property is false.

3.1.2 Life Cycle of a Web Page

ASP.NET is a powerful platform for building Web applications. With any platform, it is important to understand what is going on behind the scenes to build robust applications. The ASP.NET

Notes

page life cycle is a good example to explore so you know how and when page elements are loaded and corresponding events are fired.

Ready, aim, fire!

The requesting of an ASP.NET page triggers a sequence of events that encompass the page life cycle. The Web browser sends a post request to the Web server. The Web server recognizes the ASP.NET file extension for the requested page and sends the request to the HTTP Page Handler class. The following list is a sampling of these events, numbered in the order in which they are triggered.

- **PreInt:** This is the entry point of the ASP.NET page life cycle – it is the pre-initialization, so you have access to the page before it is initialized. Controls can be created within this event. Also, master pages and themes can be accessed. You can check the `IsPostBack` property here to determine if it is the first time a page has been loaded.
- **Init:** This event fires when all controls on the page have been initialized and skin settings have been applied. You can use this event to work with control properties. The `Init` event of the page is not fired until all control `Init` events have triggered – this occurs from the bottom up.
- **InitComplete:** This event fires once all page and control initializations complete. This is the last event fired where `ViewState` is not set, so `ViewState` can be manipulated in this event.
- **PreLoad:** This event is triggered when all `ViewState` and `PostBack` data have been loaded for the page and all of its controls – `ViewState` loads first, followed by `PostBack` data.
- **Load:** This is the first event in the page life cycle where everything is loaded and has been set to its previous state (in the case of a `PostBack`). The page `Load` event occurs first followed by the `Load` event for all controls (recursively). This is where most coding is done, so you want to check the `IsPostBack` property to avoid unnecessary work.
- **LoadComplete:** This event is fired when the page is completely loaded. Place code here that requires everything on the page to be loaded.
- **PreRender:** This is the final stop in the page load cycle where you can make changes to page contents or controls. It is fired after all `PostBack` events and before `ViewState` has been saved. Also, this is where control databinding occurs.
- **PreRenderComplete:** This event is fired when `PreRender` is complete. Each control raises this event after databinding (when a control has its `DataSourceID` set).
- **SaveStateComplete:** This is triggered when view and control state have been saved for the page and all controls within it. At this point, you can make changes in the rendering of the page, but those changes will not be reflected on the next page `PostBack` since view state is already saved.
- **Unload:** This event fires for each control and then the page itself. It is fired when the HTML for the page is fully rendered. This is where you can take care of cleanup tasks, such as properly closing and disposing database connections.



Did u know? Microsoft started development on the .NET Framework in the late 1990s originally under the name of Next Generation Windows Services (NGWS). By late 2000 the first beta versions of .NET 1.0 were released.



Caution Do not set the AutoPostBack property to True. A popup control cannot work properly if it causes postbacks.



Task Create a flow chart for Web page life cycle.

3.2 Data Binding

The ASP.NET data binding model underwent a significant refresh with version 2.0, mostly due to the introduction of data source controls and data binding expressions. These two families of components provide ASP.NET with a declarative model for data fetching and data manipulation. Data source components force layering and promote a neater separation between presentation and data access, even in relatively simple projects. Data source components offer an alternative to the classic ASP.NET binding model based on a programmatic and explicit binding between enumerable data and the DataSource property of data-bound controls.

Starting with ASP.NET 2.0, all data-bound controls support data source components for fetching, but only a few of them are designed to use data source components for other operations such as insertion, deletion, and update. For example, the GridView control is just a smarter version of the DataGrid control that, among other little things, fully supports the new data binding model. Designed to render tabular data, the GridView is partnered by DetailsView and FormView that render out the content of a single record at a time.

In ASP.NET 3.5, the family of “view” data-bound controls has a new addition – the ListView control. In actual fact, the ListView control fills a gap. There’s no control, in previous versions of ASP.NET, for iterating data templates for all the records in a data source. ASP.NET provides Repeater and DataList controls, but they only support data source components for fetching. The ListView control can be considered as an enhanced version of the DataList control that fully supports data source components. In this article, I’ll provide a developer’s perspective of the control, and show in action both its obvious and less obvious capabilities and features.

Here is the example of simple data binding

```
<asp:ListView ID="ListView1"
    runat="server"
    DataSourceID="ObjectDataSource1"
    ItemPlaceholderID="ListViewContent">
    <LayoutTemplate>
        <div>
            <h1>Customer List</h1>
        </div>
        <div>
            <table id="customerTable">
                <thead>
                    <tr>
                        <th>Company</th>
                        <th>Country</th>
                    </tr>
                </thead>
                <tbody>
```

Notes

```

        <asp:place_holder
        runat="server"
        id="ListViewContent" />
    </tbody>
</table>
</div>
</LayoutTemplate>
<ItemTemplate>
    <tr id="mainCustomerLabel">
        <td>
            <asp:Label runat="server"
            ID="lblCompany"
            Text='<%# Eval("CompanyName") %>' />
        </td>
        <td>
            <asp:Label runat="server"
            ID="lblCountry"
            Text='<%# Eval("Country") %>' />
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <i>To contact this
            customer, please call <b>
            <%# Eval("Phone") %></b></i>
        </td>
    </tr>
</ItemTemplate>
<ItemSeparatorTemplate>
    <tr id="customerSep">
        <td></td>
    </tr>
</ItemSeparatorTemplate>
</asp:ListView>

```

As you can see, you are not limited in the use of richer table tags such as thead and tbody. If you need to style table elements you can use CSS styles both through the class HTML attribute or a CSS style explicitly associated with a control ID.

In the listing above, the ItemPlaceholderID property of the ListView control is assigned the "ListViewContent" string. This string has to match the ID of a placeholder control anywhere in the layout template. It is essential to note that the dynamically generated markup tree is not parented into the placeholder control, but fully replaces it instead. For example, consider the following markup:

```

<table>
    <thead>
    :
    </thead>
    <tbody runat="server"
        id="ListViewContent">
    </tbody>
</table>

```

The ListView processes the markup by inserting the markup for item templates in lieu of the tbody tag. As a result, the final markup served to the browser won't contain a tbody tag. To preserve the tbody tag, you should define a child tag – for example a div.

```
<table>
  <thead>
    :
  </thead>
  <tbody>
    <div runat="server"
      id="ListViewContent" />
  </tbody>
</table>
```

Data Pager

The DataPager never gets in touch with the underlying data source. All that it does is communicate to the paged control what is the next set of records to select and display. The DataPager is only responsible for displaying the user interface for pagination and for returning the interval of records to be retrieved by the paged control to be consistent with the pager's user interface.

```
<asp:DataPager ID="DataPager1"
  runat="server"
  PagedControlID="ListView1"
  PageSize="4">
  <Fields>
    <asp:NextPreviousPagerField />
  </Fields>
</asp:DataPager>
```

The user interface for paging is not part of the control, but can be placed anywhere in the page and even driven through the query string of the container page. The user interface of the data pager control is largely customizable. You do that through a collection of DataPagerField objects. You have a pager field for next/previous pagination, numeric pagination, and a custom, template-based pager bar. Pager fields use several visual properties to set the style of buttons, companion text, and images to use instead of text.

In-Place

As a developer, you are expressly required to define an edit and/or insert template which will be used to edit the contents of the selected item. Edit and insert templates are separate from one another and from the item template. If you bind the ListView control to a data source control then you can take advantage of the control's full support for two-way data binding. You still use data binding expressions <%# ... %> to bind to data but opt for the Eval method for read-only operations and the Bind method for full I/O operations. To turn the ListView user interface into edit mode, you need an ad hoc button control with a command name of Edit. A command name is a plain string that can be assigned to the CommandName property of a button control.

```
<asp:Button ID="Button1" runat="server"
  Text="Edit" CommandName="edit" />
```

When such a button is clicked, the ItemEditing event is raised. By handling this event, you can verify whether the operation is legitimate. If something comes up to invalidate the call, you set the Cancel property of the event data structure to abort the operation.

Notes

```
void ListView1_ItemEditing(object sender,
    ListViewEditEventArgs e)
{
    // Just deny the edit operation
    e.Cancel = true;
}
```

An edit item template wouldn't be much help without at least a couple of predefined buttons to save and discard changes. You can define buttons using any kind of custom control that implements the `IButtonControl` interface; typically, a `Button`, `LinkButton` or perhaps an `ImageButton`. Any button clicking done within the context of a `ListView` control originates the `ItemCommand` server event. In the event handler, you use the `CommandName` property on the event data structure to check the requested command.

```
void ListView1_ItemCommand(object sender,
    ListViewCommandEventArgs e)
{
    :
}
```

In addition, any clicking on buttons associated with predefined command buttons results in subsequent, and more specific, events. For example, `ItemUpdating` and `ItemUpdated` are fired before and after a record is updated. You can use these events to make last-minute checks on the typed data before the data is sent down to the database. Note that before the update is made, the `ListView` checks the validity of any data typed by calling the `IsValid` method on the `Page` class. If any validator is defined in the template it is evaluated at this time. Any update operation is conducted through the connected data source control. For update and delete operations, you need to identify the record uniquely. This is where the `DataKeyNames` property of the `ListView` control gets into the game. You use this property to define a collection of fields that form the primary key on the data source.

```
<asp:ListView ID="ListView1"
    runat="server"
    DataSourceID="ObjectDataSource1"
    DataKeyNames="id">
    :
</asp:ListView>
```

In this case, the `DataKeyNames` tells the underlying data source control that the ID field on the bound record has to be used as the key.

3.2.1 Properties of Data Binding

The `Control.DataBindings` collection holds `Binding` objects, each of which has a `DataSource` property of type `Object`.

- The `DataSource` property of `ListBox`, `DataGridView` etc. is of type `Object`.
- The `BindingSource` class also has a `DataSource` of type `Object`.
- In the real world, you will probably use a `BindingSource` object as the `DataSource` property of list controls and of `Bindings`. If you use databases, the `DataSource` property of your `BindingSource` is usually a `DataSet`; otherwise, it may be an object of a class in your own application.

Let's start with the typical case: you assign a `BindingSource` object as the `DataSource` of a control. You can think of `BindingSource` as a "2-in-1" data source. It has:

Notes

- A single object named the Current object. A property of a Control can be bound to a property of Current.
- A List object that implements IList. The list should contain objects that have the same type as the Current object. List is a read-only property that either returns a BindingSource's "internal list" (if the DataMember string is not set), or an "external list" used if DataMember is set. Current is always a member of List (or null). When you set DataSource to a single (non-list) object, your List only contains that one item.

The way data binding works differs among different kinds of controls.

- ComboBox and ListBox bind to the List through their DataSource and DisplayMember properties. You normally set the DataSource to a BindingSource, and you set DisplayMember to the name of one of the attributes of the Current object.
- DataGrid and DataGridView bind to the List through their DataSource property. DataGrid and DataGridView do not have a DisplayMember property because they can display several properties from the data source (one in each column), not just one. DataGridView has an additional property called DataMember, which seems to be analogous to the DataMember property of BindingSource. You would not normally set your grid's DataMember to anything unless its DataSource is not a BindingSource (if you use a BindingSource, you would set BindingSource.DataMember instead.)
- "Simple" controls such as TextBox, Button, CheckBox, etc., bind to individual properties of the Current object through the Control.DataBindings collection. Actually, even list controls have a DataBindings collection, but it is not used as much. The DataBindings list can be changed in the designer under the "(Advanced)" line under the "(Data Bindings)" node.



Task Write a program for data binding.

Self Assessment

Multiple Choice Questions:

1. You need to customize the display format of the DataList control
 - a. Set the DisplayFormat property of the DataList control to Custom
 - b. Set the CustomFormat property of the DataList control to True
 - c. ItemTemplate will allow you to format the appearance of each DataList item
 - d. The display format of the DataList control is predefined and cannot be customized
2. If you need to enable paging for a DataGrid control
 - a. You can use a DataReader
 - b. You can use a DataTable
 - c. You can use either the DataReader or DataTable
 - d. A DataGrid does not support paging
3. Which property allows ASP.NET controls to maintain their values when a page is posted to itself.

a. EnableViewState	b. MaintainValue
c. SaveValue	d. AutoPostBack

Notes

4. is the DataType return in IsPostBack property.
 - a. Boolean
 - b. Int
 - c. Object
 - d. string
5. Choose the form in which Postback occur?
 - a. HTMLForms
 - b. Webforms
 - c. Winforms

3.3 Web Server Controls

ASP.NET Web Server controls are objects on ASP.NET Web pages that run when the page is requested and render markup to a browser. Many Web server controls are similar to familiar HTML elements, such as buttons and text boxes. Other controls encompass complex behavior, such as calendar controls, and controls that manage data connections.

When you create ASP.NET Web pages, you can use these types of controls:

HTML Server Controls: They are the HTML elements exposed to the server so you can program them. HTML server controls expose an object model that maps very closely to the HTML elements that they render.

Web Server Controls: They are the Controls with more built-in features than HTML server controls. Web server controls include not only form controls such as buttons and text boxes, but also special-purpose controls such as a calendar, menus, and a tree view control. Web server controls are more abstract than HTML server controls in that their object model does not necessarily reflect HTML syntax.

Validation Controls: They are the Controls that incorporate logic to enable you to what users enter for input controls such as the Textbox control. Validation controls enable you to check for a required field, to test against a specific value or pattern of characters, to verify that a value lies within a range, and so on.

User Controls: They are the Controls that you create as ASP.NET Web pages. You can embed ASP.NET user controls in other ASP.NET Web pages, which is an easy way to create toolbars and other reusable elements.

Web server controls are a second set of controls designed with a different emphasis. They do not necessarily map one-to-one to HTML server controls. Instead, they are defined as abstract controls in which the actual markup rendered by the control can be quite different from the model that you program against. For example, a RadioButtonList Web server control might be rendered in a table or as inline text with other markup. Web server controls include traditional form controls such as buttons and text boxes as well as complex controls such as tables. They also include controls that provide commonly used form functionality such as displaying data in a grid, choosing dates, displaying menus, and so on. The controls use syntax such as the following:

```
<asp: button attributes runat="server" id="Button1" />
```

In the following example we declare a Button server control in an .aspx file. Then we create an event handler for the Click event which changes the font and style of the button:

```
<script runat="server">
Sub submit(Source As Object, e As EventArgs)
button1.Style("background-color")="#0000ff"
button1.Style("color")="#ffffff"
button1.Style("width")="200px"
```

```

button1.Style("cursor")="pointer"
button1.Style("font-family")="verdana"
button1.Style("font-weight")="bold"
button1.Style("font-size")="14pt"
    button1.Text="You clicked me!"
End Sub
</script>

<!DOCTYPE html>
<html>
<body>

<form runat="server">
<asp:Button id="button1" Text="Click me!" runat="server" OnClick="submit"
/>
</form>

</body>
</html>

```

3.3.1 Generalities of ASP.NET Server Controls

All server controls using in ASP.NET are inherited from control class which is defined in System.Web.UI namespace includes web control, HTML controls or custom control.

The Control class is declared as follows:

- **Public class Control:** IComponent, IDisposable, IParserAccessor,
- IUrlResolutionService, IDataBindingsAccessor,
- IControlBuilderAccessor, IControlDesignerAccessor,
- IExpressionsAccessor

The IComponent interface defines the way in which the control interacts with the other components running in the common language runtime (CLR), whereas IDisposable implements the common pattern for releasing managed objects deterministically.

3.3.2 Identifying a Server Control

Every control that is included in an ASP.NET Web page must contain a unique identifier (ID). If you do not explicitly assign an ID to a control when you add the control to a Web page, ASP.NET creates a default ID for you. When a control is included inside a repeating control, such as the GridView, ListView, or Repeater control, a single control defined in markup often creates multiple instances of the control in the rendered HTML. The ASP.NET page framework provides a mechanism to make sure that each control in the rendered HTML for a page has a unique ID.

ID Properties

ASP.NET Web server controls include the following properties that act as identifiers:

- ID – this is the ID you specify in markup or by setting the control's ID property.
- UniqueID – this is an ID that is generated by ASP.NET for use by code that runs on the server.

Notes

- ClientID – this is an ID that is generated by ASP.NET for use by client code (it is rendered as the value of the id attribute in HTML).

3.3.3 Naming Containers

A naming container is primarily a control that acts as a container for other controls. The naming container generates a sort of namespace so that the actual ID of contained controls is rooted in the ID of the parent. The role of naming containers is particularly evident if you think about iterative controls such as Repeater or DataList. Each item these controls display is made of the same set of constituent controls. Typically, you specify constituent controls by populating a template property like ItemTemplate. The contents of the template are then iterated for each item displayed by the Repeater or DataList. Taken individually, each constituent control is given the same ID and is repeated as many times as there are items to show. Does this mean that the final ASP.NET page will have duplicated IDs? Of course not. The Repeater and DataList control act as naming containers and guarantee that the ID of each item-specific control is unique to the page.

The following example shows the NamingContainer of the button control:

```
<script runat="server">
Sub Button1_Click(sender As Object, e As EventArgs)
Response.Write("The naming container is: ")
Response.Write(button1.NamingContainer)
End Sub
</script>

<form runat="server" >
<asp:Button ID="button1" OnClick="Button1_Click"
Text="Get NamingContainer" runat="server" />
</form>
```

3.3.4 Themeable Controls

Themes in ASP.NET 2.0 is another cool new feature added to the plethora of additions and improvements made into the next version of the programming model. Using themes, you can easily customize your server controls with the predefined looks bundled with the .NET Framework or can make your own themes according to the look and feel of your website. Designing a website was never this easy before. It counter the troubles faced by developers to define the layout of server controls and make them look coherent to the overall application, with minimum possible efforts. Default or Global themes are contained in a special folder inside the framework and can be declared in the source as well as class files. However, custom made themes can also be saved inside the predefined "App_Themes" folder inside ASP.NET applications, making them easier to manage and interpolate according to your needs. The essential part of themes are skin files with .skin extension. Besides skin files, a theme can be composed of styles sheets .css files as well as images for added support for the layout of the website.

3.3.5 Control State

Control state is a new construct within ASP.NET 2.0, and it is really nothing more than view state; however, it is view state with a significant advantage; that advantage is that other developers using your control cannot disable control state as they can view state. Control state survives

even if the developer using the custom control disables view state. The advantage is pretty obvious; in prior versions of Visual Studio, the consumer of a custom control could disable view state; if the control relied on view state to maintain state information, the control would cease to function or at least misbehave. Creating control state removed the ability of a control consumer to disable view state based state management within the control; naturally, the control designer may still opt to not use either view state or control state based state management.

Each control needs to populate the signal to the page that it requires control state. Next, there is no exclusive container to store data, such as View State; but the data can be recovered from any object we want – arrays, collections, or a slew of instance variables. Each control endures and loads its control state using a pair of overridable methods, as shown below:

Protected override object SaveControlState ()

Protected override void LoadControlState (object state)

Control state works similarly to view state and is saved and loaded at the same stage of the Pipeline that view state is processed. Ultimately, control state is persisted in the same hidden field as the view state.



Caution There are potential security risks when using and developing custom ASP.NET controls.



Case Study

Solution Provider Cuts Development Time for User Interfaces from Days to Minutes

For more than two decades, ComponentOne, a Microsoft Gold Certified Partner, has offered tools and controls for the Microsoft Visual Studio development system. The company's goal is to simplify development for its customers, who need help designing user interfaces for a variety of applications. ComponentOne recently updated its product line to support Microsoft Visual Studio 2010 and the Microsoft .NET Framework 4. The company implemented features that make it easier to create Microsoft Silverlight applications and data-driven Web sites and conduct automated user interface testing. By using the Microsoft tools and ComponentOne controls, customers can complete design and testing tasks in a streamlined environment. ComponentOne states that customers can cut development time from days to minutes while building more compelling user interfaces.

Situation

ComponentOne wanted to help organizations of any size work more efficiently and design better applications. It realized that many development teams need help designing user interfaces.

Questions:

1. How developers able to cut Development Time of Project using Microsoft Visual Studio?
2. What are the other benefits Customer will get through this Solution?

Notes

Self Assessment

Multiple Choice Questions:

6. In an ASP.NET page, which is the best way to find if the contents of a TextBox has changed after the form containing the TextBox has been posted back to the server
 - a. By comparing the old value with the new value
 - b. By checking the NewValue property of the TextBox control
 - c. By handling the TextBox control's TextChanged event
 - d. By checking the HasChanged property of the TextBox
7. Select the control for which by default post back is true:
 - a. Button
 - b. TextBox
 - c. Label
 - d. Checkbox
8. Which is the parent class of the Web server control?
 - a. The System.Web.UI.Control
 - b. The System.Web.Forms
 - c. The System.Object
 - d. The System.Web.Collections
9. Displays one record at a time in a tabular layout and enables us to edit, delete, and insert records. We can also page through multiple records.
 - a. DataList
 - b. ListBox
 - c. DetailsView
 - d. DataGrid
10. is a process of connecting a Web control to a data source element.
 - a. Data Rendering
 - b. DataBinding
 - c. DataBonding
 - d. DataSource
11. A control bind to a data source control and automatically fetch data at the appropriate time in the page request lifecycle.
 - a. Data source
 - b. Data reader
 - c. Data manager
 - d. Data bound
12. The entire ASP.NET engine was completely built in and all extensibility is provided via?
 - a. Active server page and Active server extensions
 - b. managed code and managed code extensions
 - c. visual studio and visual 2008
 - d. None of the above
13. In which event of page life cycle are the controls fully loaded?
 - a. Page Load
 - b. Page Init
 - c. Page Render
14. Which property determines whether a control is displayed to the user?
 - a. Hide
 - b. Show
 - c. Visible
 - d. Enabled
 - e. Cursor

15. Which is an example of a web document?
- | | |
|-------------------|------------------|
| a. Server script | b. Web page |
| c. Client browser | d. Both a and b. |

Notes

3.4 Summary

- The HTML server controls are Hypertext Markup Language (HTML) elements that include a `runat=server` attribute.
- Post Back is the name given to the process of submitting an ASP.NET page to the server for processing. Post Back is done if certain credentials of the page are to be checked against a database.
- Autopostback is the mechanism, by which the page will be posted back to the server automatically based on some events in the web controls.
- ASP.NET pages are made of code, mark-up tags, literal text, and server controls. Based on the request, the server controls generate the right mark-up language.
- The ASP.NET runtime combines the output of all controls and serves the client a page to display in a browser.
- Validation controls enable you to check for a required field, to test against a specific value or pattern of characters, to verify that a value lies within a range, and so on.
- **User Controls:** They are the Controls that you create as ASP.NET Web pages. You can embed ASP.NET user controls in other ASP.NET Web pages, which is an easy way to create toolbars and other reusable elements.
- A naming container is primarily a control that acts as a container for other controls.
- The .NET Framework allows to make your own themes according to the look and feel of your website.
- Control state is a collection of critical view state data that controls need to function. Because of its critical role, control state data is contained in separate member variables from normal view state and is not affected when view state is disabled.

3.5 Keywords

App_Themes: Folder inside ASP.NET applications, making user easier to manage and interpolate custom themes.

ASP.NET: It is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services.

Data binding: It is a general technique that binds two data/information sources together and maintains synchronization of data.

Data Pager: The Data Pager is only responsible for displaying the user interface for pagination and for returning the interval of records to be retrieved by the paged control to be consistent with the pager's user interface.

Hyper Text Mark-up Language (HTML): It is the main mark-up language for displaying web pages and other information that can be displayed in a web browser.

Notes

Namespace: A namespace (sometimes also called a name scope) is an abstract container or environment created to hold a logical grouping of unique identifiers or symbols (i.e., names).

PostBack: It is the name given to the process of submitting an ASP.NET page to the server for processing.



- Lab Exercise*
1. Write a program to show the use of post back property.
 2. Write a program to create a home page.

3.6 Review Questions

1. How can we identify that the Page is Post Back?
2. What is AutoPostBack?
3. What is the difference between the Page_Init and Page_Load events?
4. What exactly happens when ASPX page is requested from a browser?
5. How do you bind a Content Page to a Master Page?
6. Explain Data Binding with an example.
7. What is the difference between Web User Controls and Custom Controls?
8. Explain the difference between methods and events in the ASP.NET page life cycle.
9. What is the @Register directive used for? What is the purpose of @Register directive in ASP.NET?
10. What is a server control in ASP.NET?

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (c) | 2. (b) | 3. (a) | 4. (a) |
| 5. (b) | 6. (c) | 7. (a) | 8. (a) |
| 9. (c) | 10. (b) | 11. (d) | 12. (b) |
| 13. (a) | 14. (c) | 15. (d) | |

3.7 Further Readings



Books

Pro ASP.Net 3.5 Server Controls and Ajax Components by Rob Cameron, Dale Michalk



Online links

CGAQ6AEwBg#v=onepage&q=Server%20Controls%20Basic&f=false

http://books.google.co.in/books?id=5RfcxwmvRXwC&pg=PA1&dq=Server+Controls+Basic&hl=en&sa=X&ei=oJ_6T63NO4L3rQfu7uDABg&sqi=2&ved=0

Unit 4: Advanced Server Controls

Notes

CONTENTS

Objectives

Introduction

4.1 HTML Server Controls

4.1.1 Advantages of Using HTML Server Controls

4.1.2 Difference between Web Server Control and HTML Server Control

4.1.3 HTML Server Controls Advantage

4.1.4 HTML Server Controls Disadvantage

4.2 Validation Controls

4.2.1 Understanding the Difference between Server-Side and Client-Side Validation

4.2.2 .NET to the Rescue

4.3 Summary

4.4 Keywords

4.5 Review Questions

4.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Define HTML server controls
- Explain validation controls

Introduction

Server joystick is an intrinsic aspect of each ASP.NET function we create. They encapsulate browser look and server functionality inside a reusable entity. They can be used across multiple pages inside a single ASP.NET web application as well as throughout multiple ASP.NET web applications. ASP.NET comes with many different prebuilt server controls. We now have simple controls such as the label and we possess complex controls such as the Grid View. We also have the ability to produce our own server regulators to meet a need not met by one of the existing controls by inheriting from the appropriate class and overriding its methods as needed.

This model of utilizing server controls to encapsulate browser appearance and server functionality has served our needs well since the actual inception of ASP.NET 1.0, but our server control needs are changing. A new server control need that has recently surfaced is the capability to incorporate Ajax functionality directly into the server control.

This requirement arose because the web applications need to become more receptive and visually interactive compared to conventional ASP.NET repaint-the-entire-screen model and so the conventional server control supplies. This particular requirement has emerged because users have started using web sites for example Gmail, Hotmail, and others that do not repaint the

Notes

screen every time they click a button or even need to receive fresh information. Rather, they rely on Ajax to fetch fresh information and then update or add to a portion of the display based upon that data. Because these web sites are heavily utilized and users really enjoy their own experience while using these websites they expect other web sites to perform with the exact same elegance as they do. Whenever a web site does not carry out with the same elegance the end user will often move onto competitor web site that does. Those popular applications have raised the actual bar for what is a good acceptably user-friendly web site.

Stepping back for a second, unlike other technologies we may have read books on, ASP.NET AJAX server controls do not really provide we with anything that people could not already do. We have always been able to add Ajax-functionality into server controls it was just a time consuming and painful activity. There were a couple of different methods we could make use of to include the JavaScript with this server control such as embedding it as a resource, but we eventually ended-up having to do the exact same three tasks. To make our server control have some good client capabilities we always had to concatenate strings together to produce JavaScript statements and functions, create browser sniffing statements to ensure that the JavaScript was cross-browser compatible, and add attributes or even render out HTML that connected the JavaScript functionality to the user versions of our server controls. It was not impossible, however it was error-prone and there was always this mingling of machine code and JavaScript that had been hard to maintain and even harder to read.

4.1 HTML Server Controls

The HTML server controls are basically the original HTML controls but enhanced to enable server side processing. The HTML controls like the header tags, anchor tags and input elements are not processed by the server but sent to the browser for display.

They are specifically converted to a server control by adding the attribute `runat="server"` and adding an id attribute to make them available for server-side processing. For example, consider the HTML input control:

```
<input type= "text" size= "30">
```

It could be converted to a server control, by adding the `runat` and `id` attribute:

```
<input type= "text" id= "testing" size= "30" runat= "server">
```

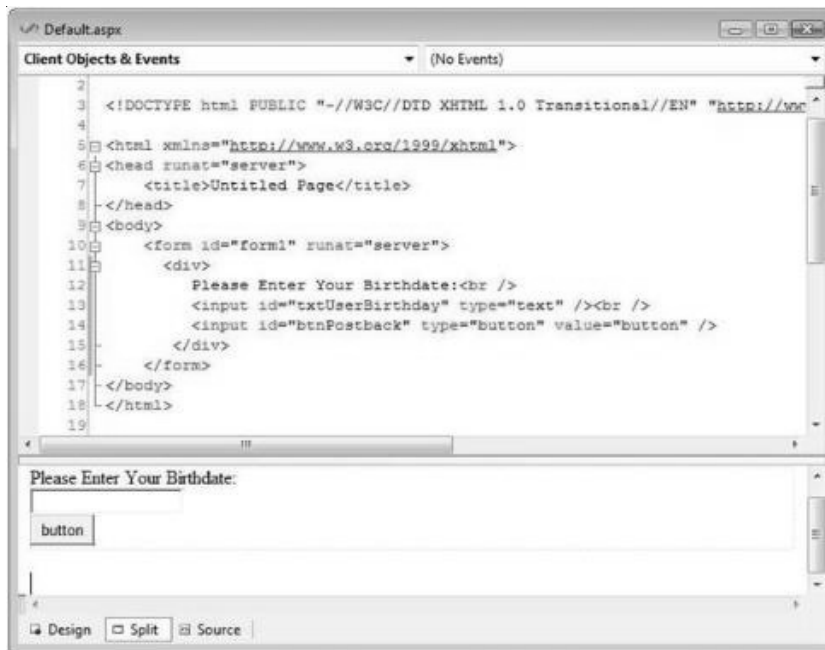
4.1.1 Advantages of Using HTML Server Controls

HTML server controls have the advantage of retaining status (viewstate) – that means, the server knows if the checkbox is checked or not and if an option is selected or not. This is particularly helpful if you refresh a pages or go back to it. A regular HTML control would be reset to the state in the HTML code. An HTML server control on the other hand checks with the server and the server will tell the page, that the user had selected option B and that the name entered in the text field was Susanne. With an HTML server control, the code for the control and the code behind is executed on the server but it can interact with client-side code.

HTML server controls seem to be the in-between step between old-fashioned HTML controls which don't do much and are purely client-side and fancier Web controls or ASP.NET controls. They resemble HTML controls and make the transition from a static HTML page to a server driven ASP page easier. ASP.NET controls are not based on HTML anymore and are basically a new programming language that you have to learn.

The following example illustrates how to convert HTML controls to HTML Server controls.

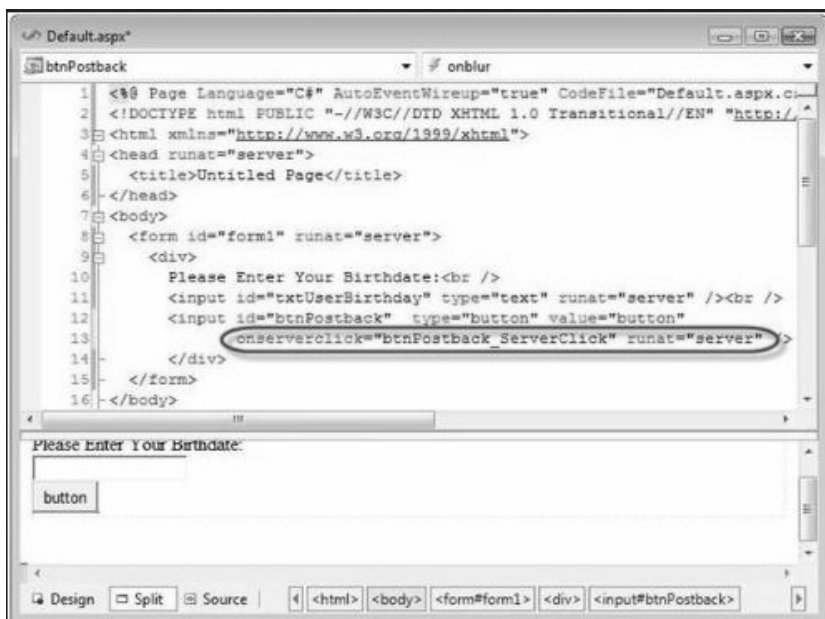
Notes



Source: <http://www.asp-training-guide.com/sitebuilder/images/pic-488 × 383.jpg>

By default, when HTML controls are placed onto the Design pane, they are *not* configured to function as server-side controls. The HTML controls above, `txtUserBirthday` and `btnPostback`, must interact with .NET code on the Web server, so they must be converted into HTML Server controls. Specifically, the button's click event will be handled on the Web server to obtain the value within the `txtUserBirthday` input area.

To do so, just add the `runat="server"` attribute to each control. If it is C#, also add the `onserverclick` attribute.



Source: <http://www.asp-training-guide.com/sitebuilder/images/picture-495 × 371.jpg>

Notes

The code-behind file holds the button click event handler. C# developers must type in the full event handler while VB'ers can use the two drop down menus at the top of the code-behind file to create the event handler.

```
public partial class _Default : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{ }
protected void btnPostBack_ServerClick(object sender, EventArgs e)
{
// Get Birthday value.
stringsBirthday = txtUserBirthday.Value;
// Process data...
}
}
```

The VB code is more or less identical, except that the Handles keyword assigns the server-side method to the Click event.

```
Partial Class Default2
Inherits System.Web.UI.Page

Protected Sub btnPostBack_ServerClick(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles btnPostBack.ServerClick
' Get birthday value.
Dim userBDay As String = Me.txtUserBirthday.Value

' Process data...
End Sub
End Class
```

Ultimately, when the Web page is loaded into the Web browser, the call to the server-side event is handled via a client-side JavaScript function. The input button now has an OnClick attribute which maps to the generated client side JavaScript function, “__doPostBack”. A pair of hidden form fields is used to represent the incoming parameters to the delegate target. Thus, when the user clicks the button, the Web browser calls this function, which results in a post back to the server-side event handler.

Here is the crux of the generated client source code. The generated source for any Web page can be viewed by right- clicking on the page within the browser and selecting ‘View Source’.

```
<html xmlns="http://www.w3.org/1999/xhtml" >
<head><title> Untitled Page</title></head>
<body>
<form name="form1" method="post" action="Default2.aspx" id="form1">
<div>
...
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
/>
</div>

<script type="text/javascript">
<!--
vartheForm = document.forms['form1'];
```

```

if (!theForm) {
theForm = document.form1;
}
function __doPostBack(eventTarget, eventArgument) {
if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
    theForm.__EVENTTARGET.value = eventTarget;
    theForm.__EVENTARGUMENT.value = eventArgument;
    theForm.submit();
}
}
// ->
</script>
<div>
Please Enter Your Birthdate:<br />
<input name="txtUserBirthday" type="text" id="txtUserBirthday" /><br />
<input language="javascript" onclick="__doPostBack('btnPostback','') "
    name="btnPostback" type="button" id="btnPostback" value="button" />
...
</html>

```

Text is not HTML encoded before it is displayed in the LinkButton control. This makes it possible to embed script within HTML tags in the text. If the values for the control come from user input, be sure to validate the values to help prevent security vulnerabilities.

4.1.2 Difference between Web Server Control and HTML Server Control

Web server controls are a second set of controls designed with a different emphasis. They do not necessarily map one-to-one to HTML server controls. Instead, they are defined as abstract controls in which the actual markup rendered by the control can be quite different from the model that you program against. For example, a RadioButtonList Web server control might be rendered in a table or as inline text with other markup. Web server controls include traditional form controls such as buttons and text boxes as well as complex controls such as tables. They also include controls that provide commonly used form functionality such as displaying data in a grid, choosing dates, displaying menus, and so on. The controls use syntax such as the following:

```
<asp: button attributes runat="server" id="Button1" />
```

The HTML server controls are basically the original HTML controls but enhanced to enable server side processing. The HTML controls like the header tags, anchor tags and input elements are not processed by the server but sent to the browser for display.

They are specifically converted to a server control by adding the attribute `runat="server"` and adding an `id` attribute to make them available for server-side processing. For example, consider the HTML input control:

Features of HTML Server Controls:

- Unlike HTML controls, the HTML Server controls are programmable on the web server.
- Using HTML server control with asp.net provide a better migration path for existing web forms.
- They are able to handle events in client script.
- Data binding to one or more properties of the control.

Notes

- You can add any attributes you need to an HTML server control and the page framework will read them and render them without any change in functionality. This allows you to add browser-specific attributes to your controls.
- When the ASP.NET page is re-posted, the HTML server controls keep their values.
- An object model that you can program against on the server. Each server control exposes properties that allow you to manipulate the HTML attributes programmatically in server code.
- A set of events for which you can write event handlers in much the same way that you would in a client-based form. However, the event is handled in server-side code.
- HTML controls give you a slightly better performance and consume less memory on the server when compared to the server controls.

4.1.3 HTML Server Controls Advantage

- The HTML Server Controls follow the HTML-centric object model. Model similar to HTML
- Here the controls can be made to interact with Client side scripting. Processing would be done at client as well as server depending on your code.
- Migration of the ASP project thought not very easy can be done by giving each intrinsic HTML control a runat-server to make it HTML Server side control.
- The HTML Server Controls have no mechanism of identifying the capabilities of the client browser accessing the current page.
- A HTML Server Control has similar abstraction with its corresponding HTML tag and offers no abstraction.

4.1.4 HTML Server Controls Disadvantage

1. A HTML Server Control has similar abstraction with its corresponding HTML tag and offers no abstraction.
2. You would need to code for the browser compatibility.



Did u know? Version 3.0 of the .NET Framework is included with Windows Server 2008 and Windows Vista. Version 3.5 is included with Windows 7, and can also be installed on Windows XP and the Windows Server 2003 family of operating systems on 12 April 2010, .NET Framework 4 was released alongside Visual Studio 2010.



Task Create a Web page using HTML controls.

Self Assessment

Multiple Choice Questions:

1.can detect browser automatically and adapt display of control accordingly.
 - a. HTML controls
 - b. HTML Server Controls
 - c. Server Controls
 - d. User Controls

2. When.....is being performed, the user cannot post the page to the server if there are errors on the page thus reducing round-trips. Notes
- a. Server Side Validation b. Client Side Validation
3. The are Hypertext Markup Language (HTML) elements that include a runat=server attribute.
- a. HTML controls b. HTML Server Controls
- c. Server Controls d. User Controls
4. can maintain data across requests using view state whereas have no such mechanism to store data between requests.
- a. HTML Controls, Server Controls b. Data Controls, Server Controls
- c. Server Controls, HTML Controls d. HTML Controls, Data Controls
5.allow to connect to the database, execute command and retrieve data from database.
- a. HTML Controls b. Data Controls
- c. Server Controls

4.2 Validation Controls

An important aspect of creating ASP.NET Web pages for user input is to be able to check that the information users enter is valid. ASP.NET provides a set of validation controls that provide an easy-to-use but powerful way to check for errors and, if necessary, display messages to the user.

This set of controls provide Rapid Application Development (RAD) features for automatically checking the specified validity of user inputs. These controls are available in the System.Web.UI.WebControls namespace.

One of the most tiresome tasks when building interactive web applications is the requirement for validating values that the user enters into the input controls. This is particularly the case if we need to perform client-side as well as server side validation. Mostly, we use JavaScript for client side coding. Help is at hand with the range of validation controls that are included in ASP.NET.

They cover almost all the validation scenarios. A validation control enables us to validate an input and display an error message if necessary. It is very much like other server-side controls, with certain additional methods and properties. First, the server treats it as an invisible control. After the user has entered erroneous data, it becomes visible. It is a powerful, rapid application development feature; however, a developer needs to understand its behavior and the methods thoroughly before he or she can appreciate it. All the validation controls inherit from the base class BaseValidator, which is part of the class library namespace. System.Web.UI.WebControls.BaseValidator exposes a series of properties and methods that are common to all the validation controls.

RequiredFieldValidator:

```
<asp:RequiredFieldValidator>
```



Caution When creating a client-side validation function, be sure to also include the functionality of the server-side validation function. If you create a client-side validation function without a corresponding server-side function, it is possible for malicious code to bypass validation.

4.2.1 Understanding the Difference between Server-Side and Client-Side Validation

Many persons new to ASP.NET do not know that it distinguish between client-side and server-side validation. ASP.NET has two different ways of validating the data user's input into a Web form. After the user enters data into a Web form and clicks on Submit button it sends the data to the server and then server-side validation on the data can be performed. If the data send is invalid, a response stating incorrectness of the data is send back. However, when a scripting language that is a part of HTML page is initiated as soon as the submit button is clicked and check the validity of the data before sending it to the server is called client-side validation.

It was a lot easier to understand the difference between these forms of validation when you coded Active Server Pages 3.0 because, as the programmer, you personally performed almost all data validation. You yourself either programmed it to be client-side or server-side.

When you used server-side validation with ASP 3.0, if something the user entered was wrong, you could re-post the form and ask the user to correct the information in that particular field of the form. Sometimes, you carried the correct input from the other fields back to the form page, and populated the fields for the users so they didn't have to re-enter the same information again. Some sites on the Internet don't carry this inputted information back to the form page, and the user is then required to enter all the information into the form a second time. Obviously, this may cause people to leave your site for another.

The bad thing about server-side validation is that it requires trips back and forth to the server. This takes a lot of resources and makes for a slower-paced form for the user. Nothing is more annoying to a user who is on a dial-up connection than clicking the Submit button on the form and then waiting for 20 seconds to find out that they didn't enter their password correctly.

JavaScript can be used to validate data in HTML forms before sending off the content to a server. Form data that typically are checked by a JavaScript could be:

- has the user left required fields empty?
- has the user entered a valid e-mail address?
- has the user entered a valid date?
- has the user entered text in a numeric field?

The function below checks if a field has been left empty. If the field is blank, an alert box alerts a message, the function returns false, and the form will not be submitted:

```
Client-SideJavaScriptforformValidation
<!DOCTYPE html>
<html>
<head>

<script>

Function validateForm()
{
var x=document.forms["myForm"]["fname"].value;
if (x==null || x=="")
{
alert("First name must be filled out");
return false;
}
```

```

}
}
</script>
</head>

<body>
<form name="myForm" action="demo_form.asp" onsubmit="return validateForm()"
method="post">
First name: <input type="text" name="fname">
<input type="submit" value="Submit">
</form>
</body>

</html>

```

4.2.2 .NET to the Rescue

Earlier developer use to spend considerable amount of time in setting various validation rules in ASP 3.0 but things have changed considerably with ASP.NET where we can easily validate the user input by using ASP.NET validation controls. It provides an easy to use mechanism for all common types of standard validation. This validation controls can be used with HTML and Web Server controls. Six validation controls are available with ASP.NET as of now, they are as follows.

- **The RequiredFieldValidator Control** – Ensures that the user does not skip a mandatory entry field.

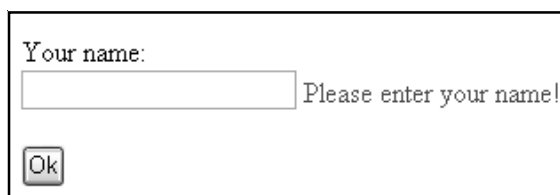
The **RequiredFieldValidator** is actually very simple, and yet very useful. You can use it to make sure that the user has entered something in a TextBox control. Let's give it a try, and add a RequiredFieldValidator to our page. We will also add a TextBox to validate, as well as a button to submit the form with.

```

<form id="form1" runat="server">
  Your name:<br />
<asp:TextBoxrunat="server" id="txtName" />
<asp:RequiredFieldValidatorrunat="server" id="reqName"
controltovalidate="txtName" errormessage="Please enter your name!" />
<br /><br />
<asp:Buttonrunat="server" id="btnSubmitForm" text="Ok" />
</form>

```

Actually, that's all we need to test the most basic part of the RequiredFieldValidator. I'm sure that all the attributes of the controls makes sense by now, so I won't go into details about them. Try running the website and click the button. You should see something like this:



The screenshot shows a web form with a text input field. The label 'Your name:' is positioned to the left of the input field. To the right of the input field, the text 'Please enter your name!' is displayed, indicating a validation error. Below the input field, there is a button labeled 'Ok'.

If your browser supports JavaScript, which most modern browsers do, then you will notice that the page is not being posted back to the server – the validation is performed

Notes

clientside! This is one of the really cool things about ASP.NET validators. Validation is only performed serverside if necessary! To see how it feels, you can add `enableclientscript="false"` to the `RequiredFieldValidator` and try again. Now you will see the browser posting back to the server, but the result will be the same – the validator still works!

Right now, the button does nothing, besides posting back if the page is valid. We will change this by adding an `onclick` event to it:

```
<asp:Button runat="server" id="btnSubmitForm" text="Ok"
onclick="btnSubmitForm_Click" />
```

In the `CodeBehind` file, we add the following event handler:

```
protected void btnSubmitForm_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        btnSubmitForm.Text = "My form is valid!";
    }
}
```

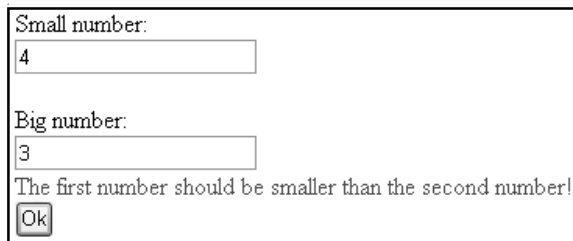
- **The CompareValidator Control** – Compares one controls value with another controls value, constants and data type using a comparison operator (equals, greater than, less than, and so on).

In the next example, I will show you a small example of how it can be used.

```
Small number:<br />
<asp:TextBox runat="server" id="txtSmallNumber" /><br /><br />
Big number:<br />
<asp:TextBox runat="server" id="txtBigNumber" /><br />
<asp:CompareValidator runat="server" id="cmpNumbers"
controltovalidate="txtSmallNumber" controltocompare="txtBigNumber"
operator="LessThan" type="Integer" errorMessage="The first number should
be smaller than the second number!" /><br />
```

As you see, we only use one validator to validate the two fields. It might seem a bit overwhelming, but it's actually quite simple. Like with the `RequiredFieldValidator`, we use the `controltovalidate` attribute to specify which control to validate. In addition to that, we specify a control to compare. The operator attribute specifies which method to use when comparing. In this case, we use the `LessThan` operator, because we wish for the first control to have the smallest value. We set the type to integer, because we want to compare integers. Dates, strings and other value types can be compared as well.

Now, try running the website, and test the two new fields. Here is what happens if you don't fill out the form correctly:



If you switch the numbers, you will see another cool thing about the ASP.NET validators and clientside scripting: Validation is also performed upon exiting the fields. If you don't want this, you can turn off clientside scripting with the `enableclientscript` attribute.

You can also compare a field to a static value. To do this, remove the `controltocompare` attribute, and add the value to compare attribute.

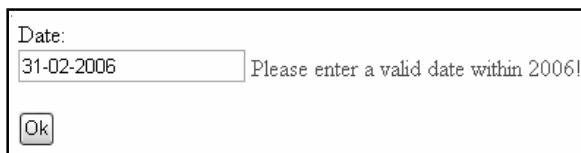
However, as you may have noticed, the content of the two textboxes is not checked before comparing it. For instance, it would be possible to enter nothing, or enter a piece of text instead of a number. You should always consider this when using the `CompareValidator`. You can protect against empty fields with the `RequiredFieldValidator`, and you can protect against wrong values like text with for instance a `RegularExpressionValidator`, which we will show you how to use a bit later.

- **The `RangeValidator` Control** – Checks the user's input is in a given range (e.g.: numbers or characters).

The **RangeValidator** does exactly what the name implies; it makes sure that the user input is within a specified range. You can use it to validate both numbers, strings and dates, which can make it useful in a bunch of cases. Since we validated numbers the last time, we will try with a date this time.

```
Date:<br />
<asp:TextBoxrnat="server" id="txtDate" />
<asp:RangeValidatorrunat="server" id="rngDate"
controltovalidate="txtDate" type="Date" minimumvalue="01-01-2006"
maximumvalue="31-12-2006" errormessage="Please enter a valid date within
2006!" />
<br /><br />
```

The date format might seem a bit weird to you if you're not from Europe, where we use dd-mm-yy. You can just change it if it doesn't fit the date format on the machine you're working on. Now, try running the website, and enter a date in our new `TextBox`. It's only valid if the date is within 2006, and a cool side effect is that the date is also checked for validity. Have a look at this screenshot, which shows us that the validator reacts to an impossible date as well:



And once again, if the clientside validation is not working, it will be caught in our `CodeBehind`, as shown with the `RequiredFieldValidator`.

- **The `RegularExpressionValidator` Control** – Checks that the user's entry matches a pattern defined by a regular expression.

The `RegularExpressionValidator` is a bit more difficult to write about, because it requires knowledge of Regular Expressions to really use it. However, `RegularExpressionValidator` is one of the most useful validators, because it can be used to check the validity of any kind of string. If you don't feel like learning Regular Expressions, but still feel like using this validator, have a look around the Internet. You can find many pre-made expressions out there, which can be very useful to you. Here is an example where we require a 4 digit number – nothing else is acceptable:

```
4 digit number:<br />
<asp:TextBoxrnat="server" id="txtNumber" />
<asp:RegularExpressionValidatorrunat="server" id="rexNumber"
controltovalidate="txtNumber" validationexpression="^[0-9]{4}$"
errormessage="Please enter a 4 digit number!" />
<br /><br />
```

Notes

The only new attribute we use, is the validation expression. This attribute simply holds the Regular Expression which should be used to validate the field. Since Regular Expressions are beyond the scope of this tutorial, I won't try to explain it, other than it simply tells the validator that a value of 4 digits is acceptable, and nothing else. You can use the `RegularExpressionValidator` for almost everything, for instance validating an e-mail or an URL.

- **The CustomValidator Control** – Checks the user's entry using custom-coded validation logic.

If none of the other validators can help you, the **CustomValidator** usually can. It doesn't come with a predefined way of working; you write the code for validating your self. This is of course very powerful, since the possibilities are basically endless. A common way of using the `CustomValidator` is when you need to make a database lookup to see if the value is valid. Since this tutorial hasn't treated database access yet, we will do a simpler example, but hopefully you will see that you can do just about everything with the `CustomValidator`. The control allows you to validate both clientside and serverside, where the serverside approach is probably the most powerful. Of course, doing serverside validation requires a postback to validate, but in most cases, that's not really a problem.

In this example, we will simply check the length of the string in the `TextBox`. This is a very basic and that useful example, only made to show you how you may use the `CustomValidator`.

```
Custom text:<br />
<asp:TextBoxruntat="server" id="txtCustom" />
<asp:CustomValidatorruntat="server" id="cusCustom"
c o n t r o l t o v a l i d a t e = " t x t C u s t o m "
onservervalidate="cusCustom_ServerValidate" errormessage="The text must
be exactly 8 characters long!" />
<br /><br />
```

As you can see, it's pretty simple. The only unknown property is the `onservervalidate` event. It's used to reference a method from CodeBehind which will handle the validation. Switch to our CodeBehind file and add the following method:

```
protected void cusCustom_ServerValidate(object sender,
ServerValidateEventArgs e)
{
if(e.Value.Length == 8)
e.IsValid = true;
else
e.IsValid = false;
}
```

This is very simple. The validator basically works by setting the `e.IsValid` boolean value to either true or false. Here we check the `e.Value`, which is the string of the control being validated, for it's length. If it's exactly 8 characters long, we return true, otherwise we return false.

Although the serverside approach is probably the most powerful, you need to consider your site hosting, especially if your database receives plenty of queries.

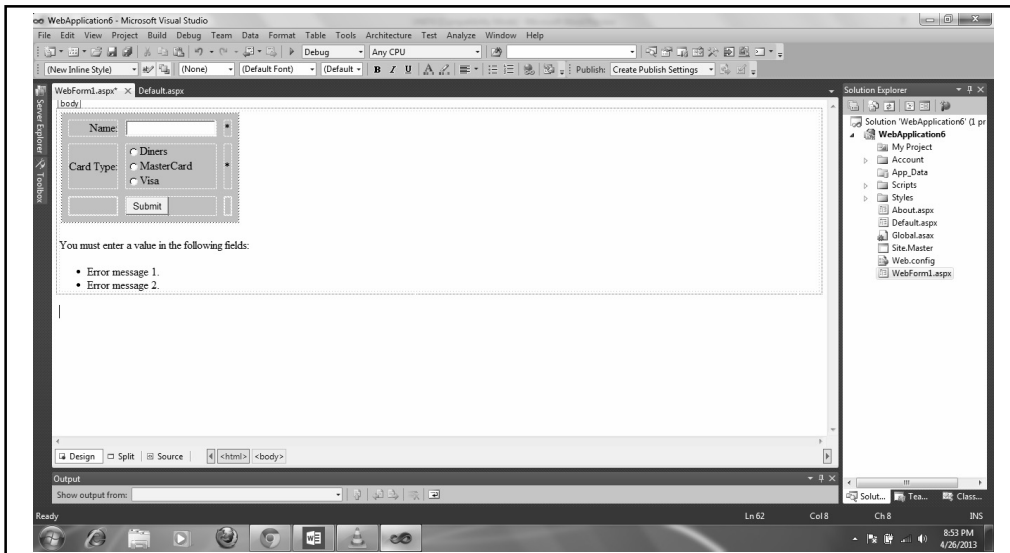
- **The ValidationSummary Control** – Displays a summary of all validation errors inline on a web page, in a message box, or both.

All validation controls except `ValidationSummary` are used to validate the user input, whereas the `ValidationSummary` control is just used to display the entire validation error messages in a particular area. This error list can be printed in the browser and or displayed

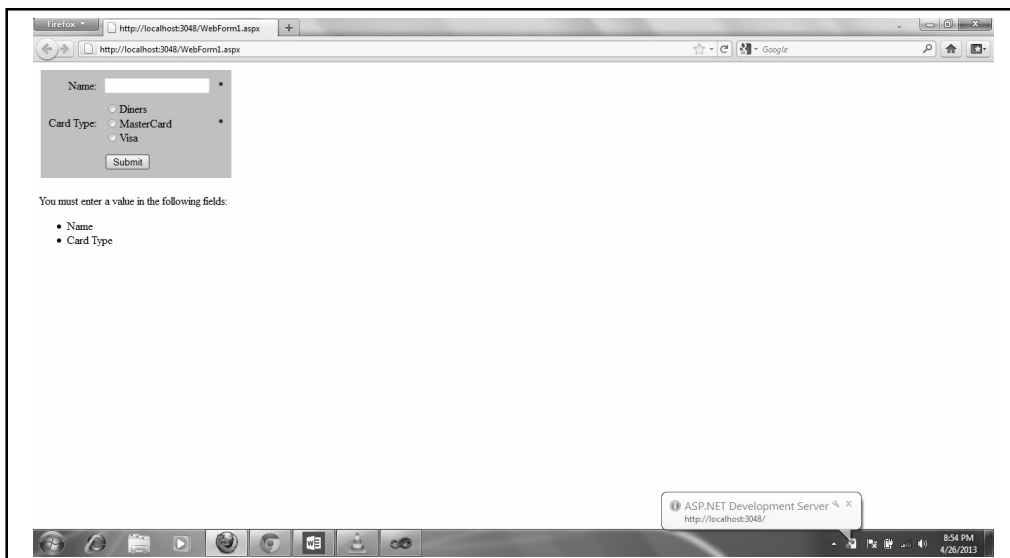
Notes

as a dialogue box. The CustomValidator allows you to implement your own validations. By using CustomValidator you can perform complex validations or you can validate the data from the database etc.

Use the ValidationSummary control to write a bulleted list of fields that are required but left empty by the user.



Press F5 or debug the application. If you do not provide the input then it will provide error message.



Did u know? The first versions of Microsoft Windows ran every program in a single address space. Every program was meant to co-operate by yielding the CPU to other programs so that the graphical user interface (GUI) could multi-task and be maximally responsive.



Task Create a Web page to show the use of compare validator.

Notes



Case Study

IT Services Provider Migrates from Mainframe, Boosts Agility and Innovation with Existing Code

Scandinavisk Data Center (SDC) helps small and medium-sized banks compete with larger ones by providing competitive IT services at an affordable price. To maintain its long-term viability, SDC is moving its core banking system from an IBM mainframe to the Microsoft Application Platform. By doing so, SDC can deliver 99.8 percent availability, increase agility, and provide high performance. In addition, the company can save more than DKK100 million (U.S.\$16 million) in yearly operational costs.

Mission-Critical Systems

The IT infrastructure at SDC supports the delivery and management of 5.8 million accounts at 146 Scandinavian banks. Any system outage jeopardizes the ability of SDC and its customers to conduct business.

Solution Overview

In June 2007, SDC decided to move its core banking system from a mainframe to a more modern platform. Although the company had previously used Java and software from BEA Systems, SDC chose to move its mission-critical system to the Microsoft Application Platform. “When we compared all of the possible solutions, we found that the Microsoft .NET Framework is more open in terms of the programming languages it supports,” explains Robert Elgaard, Chief Technology Officer at SDC. “And by using Fujitsu NetCOBOL for .NET, we could instantly compile 80 percent of our existing mainframe software.”

In March 2009, SDC began to design and test its solution, which will run in parallel with the mainframe until 2012 – when the implementation is complete. To help, SDC engaged Microsoft Services. Its consultants worked with SDC to configure the new infrastructure so that it provides for high availability and reliability. Thirty mirrored Fujitsu PRIMERGY RX300 S6 server computers support the solution, which includes the following key components:

- Microsoft SQL Server 2008 data management software. Will store all mission-critical data.
- The Windows Server 2008 R2 operating system. Provides the foundation software for the core banking system.
- The Microsoft .NET Framework 3.5. Supports a common development and runtime environment for the system.
- Microsoft Host Integration Server 2006. Works with an existing enterprise service bus—developed with the .NET Framework in 2004 – to help direct the flow of transactions between the new and existing systems.
- Microsoft System Center Operations Manager 2007 R2. Gives IT administrators a central console and tool set to manage and monitor server computers.
- Microsoft System Center Configuration Manager 2007 R2. Offers a centralized console to assess, deploy, and update software.
- The Microsoft Visual Studio Team System 2008 Team Suite development system. Provides an integrated environment with built-in tools for database development.

Contd....

- Fujitsu NetCOBOL for .NET Enterprise from Alchemy Solutions. Compiles existing software from the mainframe so that it can run on the .NET Framework.

Notes

Questions:

1. Discuss the Reasons why SDC choose Microsoft platform?
2. Discuss the Profits Gained by SDC at Technical and Business Front after migrating its core banking system to the Microsoft Application Platform.

Self Assessment

Multiple Choice Questions:

6. What data types do the asp:RangeValidator control support?
 - a. Currency
 - b. Date
 - c. Double Integer
 - d. String
 - e. All Above
7. What is used to validate complex string patterns?
 - a. Extended expressions
 - b. Basic expressions
 - c. Regular expressions
 - d. Irregular expressions
8. Which of the following control is used to validate that two fields are equal?
 - a. RegularExpressionValidator
 - b. CompareValidator
 - c. equals() method
 - d. RequiredFieldValidator
9.validation control is very helpful when checking email address, phone number, zip code etc.
 - a. Range Validator
 - b. Compare Validator
 - c. Regular Expression
 - d. Required Field validator
10. The only one unique property that offered by Regular Expression validator is
 - a. Validation Expression
 - b. Minimum Value
 - c. Maximum Value
 - d. Expression String
11. I have written code for comparing two emails, age needs to be an integer value and experience needs to be greater than 5 Years. The corresponding validationetc. properties are assigned accordingly.
 - a. controls Type
 - b. ValueToCompare
 - c. ControlToCompare
 - d. Operator
 - e. All of the above
12. Client-side function to validate all the Validation Controls with in the page.
 - a. Page_Validators
 - b. Page_Isvalid
 - c. Page_ClientValidate
 - d. Page_ClientValidate(Val Group)
13. Gets a value indicating whether page validation succeeded.
 - a. IsValid
 - b. IsPostBack
 - c. IsValidate
 - d. IsActive

Notes

14.provide automatic state management and server-side events and respond to the user events by executing event handler on the server.
 - a. HTML Controls
 - b. Data Controls
 - c. Server Controls
15.Validate the data entered using a client-side script or a server-side code.
 - a. Compare Validator
 - b. Custom Validator
 - c. Required Field Validator
 - d. Range Validator

4.3 Summary

- The HTML server controls are basically the original HTML controls but enhanced to enable server side processing. HTML server controls have the advantage of retaining state.
- ASP.NET Server Controls has higher level of abstraction. An output of an ASP.NET server control can be the result of many HTML tags that combine together to produce that control and its events.
- Web Server Controls are group of controls derived directly from the System.Web.UI.WebControls base class. They are reusable components that can perform function as the ordinary HTML controls; the real advantage of web controls is that they are programmable.
- A validation control enables us to validate an input and display an error message if necessary.
- ASP.Net uses both server side and client side validation.
- The RequiredFieldValidator server control makes sure that the user enters something into the field that it is associated with in the form. We need to tie the RequiredFieldValidator to each control that is a required field in the form.
- The CompareValidator server control compares the value entered into the form field to another field, a database value, or other value that we specify.
- The **RangeValidator** does exactly what the name implies; it makes sure that the user input is within a specified range.
- RegularExpressionValidator is used to check the validity of any kind of string like e-mail, mobile number etc.
- The CustomValidator Control – Checks the user’s entry using custom-coded validation logic.
- The ValidationSummary Control – Displays a summary of all validation errors inline on a web page, in a message box, or both.

4.4 Keywords

ASP.NET: It is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services.

Database: A database is an organized collection of data, today typically in digital form. The data are typically organized to model relevant aspects of reality.

Hypertext Mark-up Language (HTML): It is the main mark-up language for displaying web pages and other information that can be displayed in a web browser.

Notes



Lab Exercise

1. Write a program to create a login form using HTML controls.
2. Create a Web page using the required field validator.

4.5 Review Questions

1. What are the different types of server controls present in ASP.Net? When you will use them?
2. How many types of Validation controls are there in ASP.NET?
3. Define Validation Control in ASP.NET.
4. Differentiate between client-side and server-side validations in Web pages.
5. What are Custom User Controls in ASP.NET?
6. Explain the difference between Server control and HTML control.
7. Define the steps to set up validation control.
8. What is the role of Validation Summary?
9. Which Controls can be validated in ASP.Net?
10. Can we apply multiple validations on a single control? If yes then how?

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (c) | 2. (b) | 3. (c) | 4. (c) |
| 5. (b) | 6. (e) | 7. (c) | 8. (b) |
| 9. (c) | 10. (a) | 11. (e) | 12. (c) |
| 13. (a) | 14. (c) | 15. (b) | |

4.6 Further Readings



Books

Advanced Asp.Net Ajax Server Controls for .Net Framework 3.5, by Calderon Adam



Online links

<http://books.google.co.in/books?id=yp7gUxY4zn4C&printsec=frontcover&dq=Server+Controls+Advance+3.5&hl=en&sa=X&ei=0CD9T7nsOpDPPrQeJ8ujRBg&ved=0CDYQ6AEwAA#v=onepage&q&f=false>

Unit 5: Database Access

CONTENTS

Objectives

Introduction

5.1 Database Access Using ADO.NET

5.1.1 ADO.NET

5.1.2 The ADO.NET Object Model

5.1.3 Display Data in a DataGrid

5.1.4 DataBindings for Textboxes

5.2 Database Connection

5.3 Database Command

5.3.1 How to Use Parameters

5.4 DataAdapter

5.4.1 Choosing between the DataAdapter and the DataReader

5.4.2 Getting the Best Performance from the DataAdapter

5.4.3 The Role of the CommandBuilder

5.4.4 Fetching Data with the DataAdapter

5.4.5 The Single-Table Fetch Approach

5.4.6 The Multiple Resultset Approach

5.4.7 The JOIN Product Approach

5.4.8 The Composite Query Approach

5.5 ADO.NET DataSet

5.6 ADO.NET DataReader

5.7 Connection Pooling

5.7.1 Connection Pooling in ADO.NET

5.7.2 Testing Connection Pooling Behavior

5.8 Summary

5.9 Keywords

5.10 Review Questions

5.11 Further Readings

Objectives

After studying this unit, you will be able to:

- Define the database access using ADO.NET
- Explain the connection in ADO.NET

- Describe the command in ADO.NET
- Explain the ADO.NET DataAdapter
- Understand the ADO.NET DataSet
- Define ADO.NET DataReader
- Understand the connection pooling

Introduction

The concept behind concerning to the database has forever been confusing with regard to learner application developers. This unit focuses on the dissimilar methods accessible for connecting to databases in C#.

The software support provided by a database vendor to access its own creation is actually identified as a 'Native driver'. This provides the simplest process of connectivity since the data access takes place without any interference by other software layers. But, as the database changes, the information access mechanism also needs to be changed accordingly in this move towards.

So as to standardize data access, Microsoft released a set of general access drivers called OleDb, which stands for Object Linking and Embedding DataBase. It is said to support compound skill, which implies that it can either connect to a database using resident drivers or seek the help of other Data Access drivers like ODBC (Open DataBase Connectivity) to access the data store.

A set of APIs were made to communicate effectively with data in the data store. The common APIs available were:

- DAO – Data Access Object
- RDO – Remote Data Object
- ADO – ActiveX Data Object
- ADO.NET – ActiveX Data Object for .NET

The two distinct type of data access models defined are:

- Connection Oriented (Connected or Online mode)
- Connectionless (Disconnected or Offline mode)

In Connected approach, whenever a query is run, the results tend to be stored in an ActiveXDataSet in the server. The pointer to this ActiveX dataset is returned to the client. This pointer is known as Cursor. Therefore, to access the outcomes, a connection must be regularly open between the server and also the client.

In the Disconnected strategy, the results of a query and sent back to the customer, and stored in a customer side cursor. This in change means that continuous connectivity do not need to exist between the server and also the client.

A DataAdapter acts as a bridge between a data source and a data class which is disconnected from the data source. It can provide some methods to carry out operations like Select, Insert, Update and Delete. The DataSet is the equivalent for customer side cursor, which holds a portion of the database after execution of a query. It can be defined as a collection of DataRows and DataColumns. More popularly referred to as an in-memory database.

Notes

There are three commands of executing a query in C#.

- ExecuteReader() – which will return the address of the cursor.
- ExecuteNonQuery() – which will return the number of rows affected.
- ExecuteScalar() – which will return the first [row][column] of the result or a single value.

5.1 Database Access Using ADO.NET

A database is a collection of information in the form of row and column that is prearranged so that it can easily be accessed, managed, and updated.

Databases are sometimes classified by the organizational approach they adopt. The most common approach is the relational database. A distributed database is one that can be diffuse or simulated among different points in a network. An object-oriented programming database is one that is congruent with the data defined in object classes and subclasses.

ASP.Net hides the complex processes of data access and provides much higher level of classes and objects through which data is accessed easily. These classes hide all complex coding for connection, data retrieving, data querying and data manipulation.

ADO.Net is the technology that provides the bridge between various ASP.Net control objects and the backend data source. We will come to ADO.Net in due time.

5.1.1 ADO.NET

ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources exposed through OLE DB and XML. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.

ADO.NET cleanly factors data access from data manipulation into discrete components that can be used separately or in tandem. ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results. Those results are either processed directly, or placed in an ADO.NET DataSet object in order to be exposed to the user in an ad-hoc manner, combined with data from multiple sources, or remoted between tiers. The ADO.NET DataSet object can also be used independently of a .NET Framework data provider to manage data local to the application or sourced from XML.

The ADO.NET classes are found in System.Data.dll, and are integrated with the XML classes found in System.Xml.dll. When compiling code that uses the System.Data namespace, reference both System.Data.dll and System.Xml.dll. For an example of compiling an ADO.NET application using a command line compiler, see ADO.NET Sample Application.

You can use ADO.NET to access data by using the new .NET Framework data providers which are –

- Data Provider for SQL Server(System.Data.SqlClient).
- Data Provider for OLEDB(System.Data.OleDb).
- Data Provider for ODBC(System.Data.Odbc).
- Data Provider for Oracle(System.Data.OracleClient).

The ADO.NET is a set of classes that expose data access services to the .NET developer. The ADO.NET classes are found in System.Data.dll and integrated with XML classes in System.Xml.dll.

There are two central components of ADO.NET classes:

- the *DataSet*, and
- the .NET Framework *Data Provider*.

Data Provider is a set of components including the:

- Connection object (SqlConnection, OleDbConnection, OdbcConnection, OracleConnection),
- Command object (SqlCommand, OleDbCommand, OdbcCommand, OracleCommand),
- DataReader object (SqlDataReader, OleDbDataReader, OdbcDataReader, OracleDataReader), and DataAdapter object (SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter, OracleDataAdapter).

DataSet object represents a disconnected cache of data which is made up of Data Tables and Data Relations that represent the result of the command.

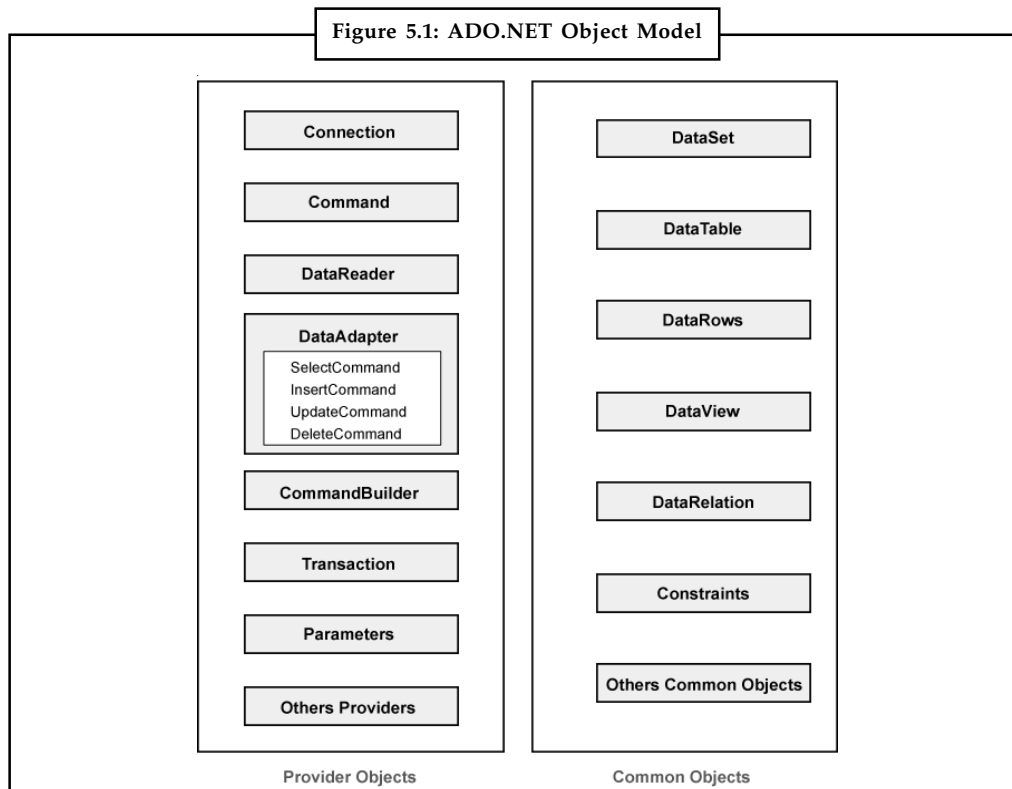
5.1.2 The ADO.NET Object Model

ADO.NET is designed to help developers work efficiently with multi tier databases, across intranet or Internet scenarios.

The ADO.NET object model consists of two key components as follows:

- Connected model (.NET Data Provider – a set of components including the Connection, Command, DataReader, and DataAdapter objects)
- Disconnected model (DataSet)

The ADO.NET object model is as follows (See Figure 5.1):



Notes

1. Connection to an ADO.NET Database

Connections are part of a Data Provider and the Connection object provides connectivity to a data source.

Connections can be opened in two ways:

1. Explicitly by calling the Open method on the connection;
2. Implicitly when using a DataAdapter.

The following examples demonstrate how to create and open connections to SQL Server (SqlClient) and OLE DB (OleDb) databases.

```
SqlConnection
SqlConnection myConn = new SqlConnection
("Data Source=localhost;Integrated Security=SSPI;" + "Initial
Catalog=northwind");
myConn.Open();
Connection String Format - SqlConnection
```

The SQL Server.NET Data Provider supports a connection string format that is similar to the OLE DB (ADO) connection string format. For valid string format names and values, see the SqlConnection.ConnectionString Property.

```
OleDb
OleDbConnection myConn = new OleDbConnection
("Provider=SQLOLEDB;Data Source=localhost;" + "Integrated
Security=SSPI;Initial Catalog=northwind");
myConn.Open();
Closing the Connection
```

You must always close the Connection when you are finished using it. This can be done using either the Close or Dispose methods of the Connection object. Connections are not implicitly released when the Connection object falls out of scope or is reclaimed by garbage collection.

2. DataSet

The DataSet object offers a disconnected data source architecture. The DataSet can work with the data it contain, without knowing the source of the data coming from. That is, the DataSet can work with a disconnected mode from its Data Source The DataSet component allows you to create multiple tables and create relationships among them. You can use the DataSet object if you want to manipulate or update the data or persists the data across multiple requests. The DataView is just a replica of the tables in the DataSet. The DataView has simple procedures to select and satisfy a particular condition. Thus, ADO.NET Object Model provides a flexible way to access and manipulate the data from a data source. The dataset is present in the dataset class in the system.data namespace.

The key components of dataset are:

- **DataTableCollection:** It contains all the tables retrieved from the Datasource.
- **DataRelationCollection:** It contains relationship and links between tables in a dataset.
- **DataTable:** It represents a table in the datatable collection of a dataset.
- **DataRowCollection:** It Contains all the rows in a database.
- **DataColumnCollection:** It Contains all the columns in a database.

There are two types of DataSets:

Typed DataSets use explicit names and DataTypes for their members.



Example: `NorthwindDataSet.Products.ProductNameColumn.Caption = "pnames";`

They have .xsd file (Xml Schema definition) file associated with them and do error checking regarding their schema at design time using the .xsd definitions.

UnTyped DataSets:

UnTyped DataSets use table and column collections for their members



Example: `ds.Tables["emp"].Columns["eno"].ReadOnly=true;`

They do not do error checking at the design time as they are filled at run time when the code executes.

5.1.3 Display Data in a DataGrid

DataGrid control represent data in a series of rows and columns with in windows form application it provides an interface to ADO.NET DataSets and allow update, delete and insert a record using that interface to the database. When DataGrid control is connected to a suitable database, the control will be automatically populated, creating table based on the data. DataGrid control can be used for representing either a single table or the hierarchical relationships between connected tables. DataGrid should be bound to a data source by using

- the DataSource and
- DataMember properties at design time
- or
- the SetDataBinding method at run time.

If you update the data in the bound DataSet through any mechanism, the DataGrid control reflects the changes. You can update the data in the DataSet through the DataGrid control, if the data grid and its table styles and column styles have the ReadOnly property set to false.

There are four most typical valid data sources for the DataGrid:

- DataTable class
- DataView class
- DataSet class
- DataViewManager class

5.1.4 DataBindings for Textboxes

Data Binding is binding controls to data from the database. With data binding we can bind a control to a particular column in a table from the database or we can bind the whole table to the data grid. Data binding provides simple, convenient, and powerful way to create a read/write link between the controls on a form and the data in their application. Windows Forms supports binding data to ADO.NET DataSet, Array, ArrayList, etc. A control can be bound to any collection that supports indexed access to the elements in that collection. Data binding provides a way for developers to create a read/write link between the controls on a form and the data in their

Notes

application (their data model). Classically, data binding was used within applications to take advantage of data stored in databases. Windows Forms data binding allows you to access data from databases as well as data in other structures, such as arrays and collections.

1. Simple Data Binding
2. Complex Data Binding

Simple Data Binding

ASP.NET introduces a new declarative syntax, `<%# %>`. This syntax is the basis for using data binding in an .aspx page. All data binding expressions must be contained within these characters. The following list includes examples of simple data binding from multiple sources:

- Simple property (syntax for a customer):
- `<%# custID %>`
- Collection (syntax for an order):
- `<asp:ListBox id="List1" datasource='<%# myArray %>' runat="server">`
- Expression (syntax for a contact):
- `<%# (customer.First Name + " " + customer.LastName) %>`
- Method result (syntax for the outstanding balance):
- `<%# GetBalance(custID) %>`

In the preceding examples, the inline `<%# %>` tags indicate where the information from a specific data source is to be placed in the .aspx page. The following data binding example uses a TextBox Web server control:

```
<asp:textbox id=txt text="<%# custID %>" runat=server />
```

Complex Data Binding

Complex data binding is the ability of a control to bind to more than one data element, typically more than one record in a database, or to more than one of any other type of bindable data element. DataGrid, ListBox and ErrorProvider controls support Complex data binding. Here is the method used in this project to bind all TextBoxes:

```
private void fnGetDataBindingForTextBoxes()
{
    this.textboxFirstname.DataBindings .Add( "Text", datc.dSet.Tables[
    "PersonTable"], "FirstName");
    this.textboxLastname.DataBindings . A d d ( " T e x t " , d a t c . d S e
    t . T a b l e s [ "PersonTable"], "LastName");
    this.textboxTitle.DataBindings.Add( "Text", datc.dSet.Tables[ "PersonTable"],
    "Title");
    this.textboxCity.DataBindings.Add( "Text", datc.dSet.Tables[ "PersonTable"],
    "City");
    t h i s . t e x t b o x C o u n t r y . D a t a B i n d i n g s . A d d ( " T e x t " , d a t c . d S e t .
    Tables["PersonTable"], "Country");
}
```



Did u know? The term Object Database first introduced in 1985.

5.2 Database Connection

An instance of the OleDbConnection class in .NET Framework is supported the OLEDB Data Provider. The OleDbConnection instance takes Connection String as argument and pass the value to the Constructor statement. When the connection is established, SQL Commands may be executed, with the help of the Connection Object, to retrieve or manipulate data in the database.

Once the Database activities over, Connection should be closed and release the resources. The Close() method in SqlConnection class is used to close the Database Connection. The Close method rolls back any pending transactions and releases the Connection from the Database connected by the OLEDB Data Provider.

```
Imports System.Data.OleDb
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim connectionString As String
        Dim cnn As OleDbConnection
        connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=yourdatabasename.mdb;"
        cnn = New OleDbConnection(connectionString)
        Try
            cnn.Open()
            MsgBox("Connection Open ! ")
            cnn.Close()
        Catch ex As Exception
            MsgBox("Can not open connection ! ")
        End Try
    End Sub
End Class
```

Run your program and you should see the two message boxes display. The form will display after you click OK on these.



Caution Do not calls Close or Dispose on a Connection, a DataReader, or any other managed object in the Finalize method of your class. In a finalizer, you should only release unmanaged resources that your class owns directly. If your class does not own any unmanaged resources, do not include a Finalize method in your class definition.

5.3 Database Command

The Command object enables access to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information. Commands contain information that is submitted to a database as a query, and, like connections, are represented by the provider-specific classes SqlCommand and OleDbCommand. Functionally, once the Connections are established and the Commands are executed the results are in the form of streams. These resultant streams can be accessed either by DataReader object, or passed into a DataSet object via a DataAdapter.

The SqlCommand class provides four different methods to execute a command. They are: ExecuteReader, ExecuteNonQuery, ExecuteScalar and, newest but not least, ExecuteXmlReader.

Notes

Essentially, such methods differentiate only for the type of input they expect, and consequently for the result they return. In general, once you know the operation to accomplish, determining the right method to use is rather straightforward.

Incidentally, an `OleDbCommand` object does not support `ExecuteXmlReader`.

`ExecuteReader` expects to run a query command or a stored procedure that selects records. It expects to have one or more resultsets to return.

```
cmd.Connection.Open();
SqlDataReader dr = cmd.ExecuteReader();
// process the resultset(s) here
cmd.Connection.Close();
```

You access the selected records using the `SqlDataReader` object and use the method `Read` to loop through them. You move to the next resultset using the `NextResults` method.

`ExecuteNonQuery` expects to run a command, or a stored procedure, that affects the state of the specified table. This means anything but a query command. You normally use this method to issue an `INSERT`, `UPDATE`, `DELETE`, `CREATE`, and `SET` statement.

`ExecuteNonQuery` returns only the number of rows affected by the command execution, or `-1` should this information be unavailable. It doesn't give you a chance to access any resultset generated by the statement or the stored procedure. Actually, there's really nothing to prevent you from using this method for a query command, but in this case you get neither the resultset nor the number of the affected rows.

```
cmd.Connection.Open();
nRecsAffected = cmd.ExecuteNonQuery();
cmd.Connection.Close();
// check the record(s) affected here
```

The number of affected rows is also made available through the `RecordsAffected` property of the `SqlCommand` object. This property equals `-1` in case of errors or if a query command is executed.

`ExecuteScalar` expects to run a query command, or more likely a stored procedure, that returns data. However, this method is different from `ExecuteReader` in that it just makes available, as a scalar value, the first column on the first row of the selected resultset.

```
cmd.Connection.Open();
Object o = cmd.ExecuteScalar(); cmd.Connection.Close();
// work on the scalar here
```

The method returns the value as a boxed object. It's then up to you to unbox or cast that value to the proper, expected type.

`ExecuteScalar` turns out to be particularly useful when you have statistical or aggregate operations to accomplish on a certain amount of data. In these and similar circumstances, there is just one value that you might want to return back to the caller. Because of its use cases, you normally use this method on more or less complex stored procedures rather than on single SQL statements.

`ExecuteXmlReader` builds up and returns an `XmlReader` object after a `SELECT` command that exploits XML features in SQL Server 2000 has been issued.

5.3.1 How to Use Parameters

Parameters should always be used when inserting values into SQL statements. Many people, generally new developers, don't follow this rule because they either don't realize that parameters exist or they don't understand the issues that their use helps to avoid.



Example:

```
Dim sql As String = "INSERT INTO User (FirstName, LastName, DateOfBirth,
ChildCount) " & _
    "VALUES (" & Me.firstNameField.Text & _
    "', '" & Me.lastNameField.Text & _
    "', '" & Me.dateOfBirthPicker.Value.ToString("yyyy-MM-dd") & _
    "', " & Me.childrenSpinner.Value & ")"
Dim myCommand As New SqlCommand(sql)
```



Task Write a program to insert the data in the data base.

Self Assessment

Multiple Choice Questions:

- Which database is the ADO.NET SqlConnection object designed for?
 - Access
 - Microsoft SQL Server
 - MySQL
 - Oracle
 - None of the above.
- Which of the following is not a member of ADODBCommand object?
 - ExecuteScalar
 - ExecuteStream
 - Open
 - ExecuteReader
- What is the best way to store the connection strings?
 - Config files
 - Database text file
 - session
- What method of the Command object would you use if you need to retrieve a single value (for example an aggregate value) from a database
 - ExecuteScalar()
 - ExecuteReader()
 - ExecuteSingle()
 - ExecuteNonQuery()
- Which object does the data-aware control bind to?
 - Dataset
 - DataAdapter
 - Connection
 - Both a and b.
 - All of the above.

5.4 DataAdapter

The DataAdapter provides a set of methods and properties to retrieve and save data between a DataSet and its source data store. It does the actual work of putting returned data from a database into a DataSet. It also manages reconciling how data should be updated against a database.

Connections and Commands whose properties are set early on in code are often passed into DataAdapters for use when their action methods are invoked.

Notes

The DataAdapter object encapsulates a set of data commands and a database connection, which are used to fill the DataSet and update the data source. The Fill method of the DataAdapter calls the SELECT command while Update method calls INSERT, UPDATE or DELETE command for each changed row.

One of the great features about the DataAdapter object is that these commands can be set explicitly in order to control the statements used at runtime to resolve changes, including the use of stored procedures.

The .NET documentation notes that the CommandBuilder object can be used to generate these commands at run-time based upon a select statement for “ad-hoc queries”. However, this requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will always result in better run-time performance.

The DataAdapter is the object that connects to the database to fill the DataSet. It also connects to the database in order to update the data, and this is based on the operations that take place while the DataSet holds the data.

However, there are cases where the DataAdapter and DataSet objects are bypassed. In these cases, the DataReader object is used.

5.4.1 Choosing between the DataAdapter and the DataReader

When you decide whether your application should use a DataReader or a DataAdapter, consider the type of functionality that your application requires. Use a DataAdapter to do the following:

- Cache data locally in your application so that you can manipulate it. If you only need to read the results of a query, the DataReader is the better choice.
- Remote data between tiers or from an XML Web service.
- Interact with data dynamically such as binding to a Windows Forms control or combining and relating data from multiple sources.
- Perform extensive processing on data without requiring an open connection to the data source, which frees the connection to be used by other clients.

If you do not require the functionality provided by the DataAdapter, you can improve the performance of your application by using the DataReader to return your data in a forward-only, read-only manner. Although the DataAdapter uses the DataReader to fill the contents of a DataSet, by using the DataReader, you can boost performance because you will save memory that would be consumed by the DataSet, and avoid the processing that is required to create and fill the contents of the DataSet.

5.4.2 Getting the Best Performance from the DataAdapter

The disconnected architecture implemented through ADO.NET assumes that your own code returns a set of rows to the client and posts changes to the in-memory copy of the data. These changes are not reflected in the database until you submit a “batch” of updates using the DataAdapter Update method. No, it is not a real batch in the traditional sense, but that is what Microsoft chose to call it. What really happens is ADO.NET walks the set of rows passed to the Update method and executes either the UpdateCommand, DeleteCommand or InsertCommand in line with the row’s RowState value. Every execution requires a round trip. It does not help that all too many of the examples you will encounter suggest that you use the DataAdapter to fetch all of the rows (and columns) from the database table and construct suitable action queries

to enable the actual Update method. While this approach is valid for “toy” databases, it flies in the face of the constraints of scalable, high-performance, real-world systems. If you’re working with home databases or those in your office along with only a few hundred rows, and you are not concerned with multi-user issues (and in no way plan to be), whether or not you fetch all of the rows from the table wouldn’t make much difference. However, in many business applications, especially those that have to scale up to support dozens to thousands of users and work with thousands to millions of rows, how you fetch and update the data is critical to a successful application.

5.4.3 The Role of the CommandBuilder

CommandBuilder generates insert/update/delete commands for Data adapter based on select command. Automatic creation of insert/update/delete commands hinders performance. In case one knows the contents of insert/update/delete, should create those explicitly. Better to create explicit stored procedures for insert/update/delete and assign those.

The CommandBuilder uses SelectCommand property of DataAdapter to determine values for other commands. If there is change in SelectCommand of DataAdapter, remember to call RefreshSchema to update the command properties.

CommandBuilder only generates a command for data adapter’s Command property if command property is null. By default command properties are null for data adapter. If you explicitly set a command property, the CommandBuilder does not overwrite it. You need to set the command property to null to allow CommandBuilder to generate a command for Command property.

5.4.4 Fetching Data with the DataAdapter

The SqlDataAdapter serves as a bridge between a DataSet and SQL Server for retrieving and saving data. The SqlDataAdapter provides this bridge by mapping Fill, which changes the data in the DataSet to match the data in the data source, and Update, which changes the data in the data source to match the data in the DataSet, using the appropriate Transact-SQL statements against the data source. The update is performed on a by-row basis. For every inserted, modified, and deleted row, the Update method determines the type of change that has been performed on it (Insert, Update, or Delete). Depending on the type of change, the Insert, Update, or Delete command template executes to propagate the modified row to the data source. When the SqlDataAdapter fills a DataSet, it creates the necessary tables and columns for the returned data if they do not already exist. However, primary key information is not included in the implicitly created schema unless the MissingSchemaAction property is set to AddWithKey. You may also have the SqlDataAdapter create the schema of the DataSet, including primary key information, before filling it with data using FillSchema. For more information, see Adding Existing Constraints to a DataSet (ADO.NET).

The following code example populates a list of customers from the **Northwind** database on Microsoft SQL Server, and a list of orders from the **Northwind** database stored in Microsoft Access 2000. The filled tables are related with a **DataRelation**, and the list of customers is then displayed with the orders for that customer.

```
// Assumes that customerConnection is a valid SqlConnection object.
// Assumes that orderConnection is a valid OleDbConnection object.
SqlDataAdapter custAdapter = new SqlDataAdapter(
    "SELECT * FROM dbo.Customers", customerConnection);
OleDbDataAdapter ordAdapter = new OleDbDataAdapter(
    "SELECT * FROM Orders", orderConnection);
```

Notes

```
DataSet customerOrders = new DataSet();

custAdapter.Fill(customerOrders, "Customers");
ordAdapter.Fill(customerOrders, "Orders");

DataRelation relation = customerOrders.Relations.Add("CustOrders",
    customerOrders.Tables["Customers"].Columns["CustomerID"],
    customerOrders.Tables["Orders"].Columns["CustomerID"]);

foreach (DataRow pRow in customerOrders.Tables["Customers"].Rows)
{
    Console.WriteLine(pRow["CustomerID"]);
    foreach (DataRow cRow in pRow.GetChildRows(relation))
        Console.WriteLine("\t" + cRow["OrderID"]);
}
```

5.4.5 The Single-Table Fetch Approach

Most of the examples illustrates the use of the DataAdapter by coding the SelectCommand.CommandText as "SELECT * FROM Writers". This simple query masks several important issues. First, it does not restrict the number of rows or columns returned from the data table. This approach is fine for home databases, but cripples an application dealing with serious amounts of data. This means you will need an example of fetching selected columns focused (and limited) by a WHERE clause for your production application which extracts rows from that 100,000 (or 1,000) row table (presuming you want to build a scalable application). This simplistic approach also assumes that you can return the info you want without benefit of information from other tables. Sure, there are numerous cases where a simple query is going to do. But SELECT * is evil. It blindly returns all columns from a product whether you need them or not and assumes that changes to the actual database table post-deployment would not affect your application. This is known as optimistic programming, or programming by wishful thinking. Ironically, this approach is the one most likely to work with the actual DataAdapter's CommandBuilder used to produce the action queries to update the fetched data.

5.4.6 The Multiple Resultset Approach

Assuming the single-table query approach would not work for your application, you should examine the outcome of the "multiple resultset" query.

This query returns two independent rowsets which might be logically associated. In this case, ADO.NET constructs two DataTables (or it will if the Fill method is executed) one containing selected Publisher names and also the other associated Titles. It is up to my own client-side code to relate these two tables if, allow ADO.NET manage and bind regulates to show these relationships. This method assumes that you do not be prepared to update the source tables, you expect to use the DataAdapter Update method in order to update only one of the source tables, or you plan to provide your own update routines and not use the Update method. In this case the CommandBuilder will be able to create the action commands for you – however only for the first data table specified by the initial SELECT.

5.4.7 The JOIN Product Approach

Of course, you can still use the actual server's ability to JOIN two or even more database tables or views together to produce a composite SQL product and the DataAdapter may execute this

query and construct a DataTable from the product's rowset. No, ADO.NET does not treat this rowset differently from any other – one DataTable is created for each unique rowset returned by the server. This implies that you can return a rowset from a JOIN product and another from a single-table query or another JOIN product and use these to generate multiple DataTables as necessary.

Unlike the previous approach, the CommandBuilder wouldn't be able to construct the action Commands so you will have to fall back on your personal ad hoc action commands or stored methods to make changes to the base tables. For many developers, this is what they have had to do for years anyway.

5.4.8 The Composite Query Approach

One of the more interesting (and powerful) approaches you can use is construction of "composite" queries that return rowsets from more than one data source. The trick here is to use more than one DataAdapter to specify different data sources sketching rowsets from the same or different servers, from different providers or from non-database data sources. Once the data is downloaded into DataTable objects, it is a simple matter to construct relationships between them. Updating can also be straightforward in this case, and if the actual SelectCommand is simple enough the CommandBuilder can be used to generate the action commands. Ideally, you can use stored procedures on each data source system to perform updates to the data table associated with each DataAdapter.

This approach will become one of the most popular as it deals with so many issues quite nicely. Regardless of the SQL product's source, you can define a suitable DataAdapter to focus the update operation on a specific fellow member. That is, you define a DataAdapter UpdateCommand, InsertCommand and DeleteCommand to change a specific row in a specific data table – you define a separate DataAdapter for each database table you wish to update. Just because a DataSet contains DataTables drawn from disparate data sources that is no reason you can't update any or all of the base database tables as needed.



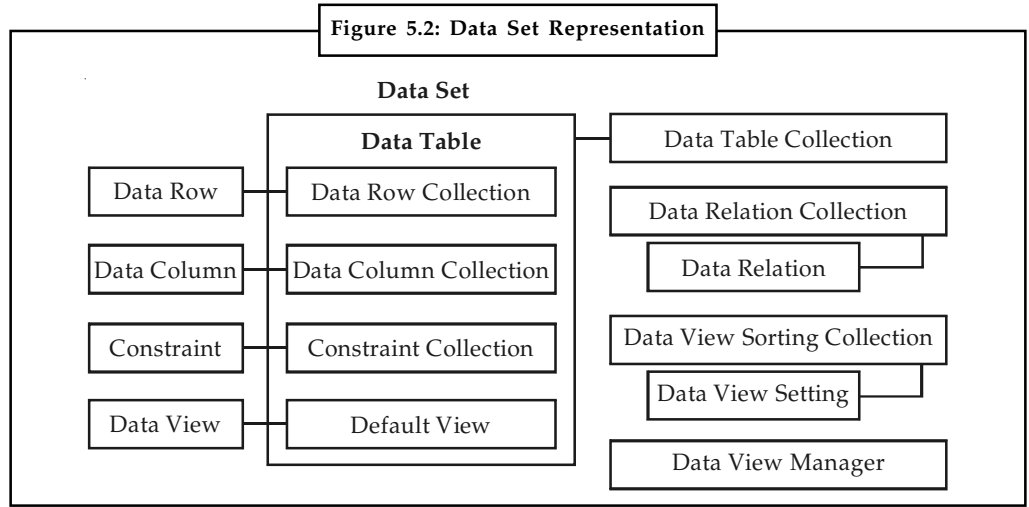
Task Write a program to INSERT, UPDATE and DELETE data from a table.

5.5 ADO.NET DataSet

The ADO.NET DataSet contains DataTableCollection and their DataRelationCollection. It represents a collection of information retrieve from the Data Source. We could use Dataset in combination with DataAdapter class. The DataSet object provides disconnected data source architecture. The Dataset can work with the information it contains, without knowing the source of the data coming from. That is, the Dataset can work with the disconnected mode from its DataSource. This gives a better advantage over DataReader, because the DataReader is working only with connection oriented Data Sources.

The Dataset contains the copy of the data we requested. The Dataset consists of more than one Table at a given point of time. We can set up Data Relations between these tables within the DataSet. The data set may comprise data for a number of members, corresponding to the number of rows. The DataAdapter object allows us to populate DataTables in a DataSet. We can use Fill method of the DataAdapter for populating data in the Dataset. The DataSet can be filled either from a data source or dynamically. A DataSet can be saved to an XML file and then loaded back into memory very effortlessly.

Notes

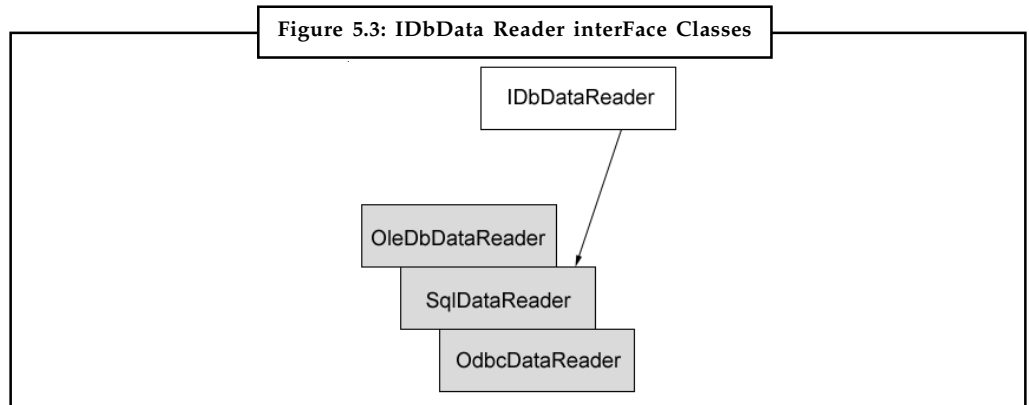


5.6 ADO.NET DataReader

A data reader provides an easy way for the programmer to read data from a database as if it were coming from a stream. The DataReader is the solution for forward streaming data through ADO.NET. The data reader is also called a firehose cursor or forward read-only cursor because it moves forward through the data. The data reader not only allows you to move forward through each record of database, but it also enables you to parse the data from each column. The DataReader class represents a data reader in ADO.NET.

Similar to other ADO.NET objects, each data provider has a data reader class for example; OleDbDataReader is the data reader class for OleDb data providers. Similarly, SqlDataReader and ODBC DataReader are data reader classes for Sql and ODBC data providers, respectively.

The IDataReader interface defines the functionality of a data reader and works as the base class for all data provider-specific data reader classes such as OleDbDataReader, SqlDataReader, and OdbcDataReader. Figure shows some of the classes that implement IDbDataReader.



Initializing DataReader

As you’ve seen in the previous examples, you call the ExecuteReader method of the Command object, which returns an instance of the DataReader. For example, use the following line of code:

```
SqlCommand cmd = new SqlCommand(SQL, conn);
// Call ExecuteReader to return a DataReader
SqlDataReader reader = cmd.ExecuteReader();
```

Once you're done with the data reader, call the Close method to close a data reader:

Notes

```
reader.Close();
```



Did u know? SQL was initially developed at IBM by Donald D. Chamberlin and Raymond F. Boyce in the early 1970s.

5.7 Connection Pooling

A connection is implicitly opened if it is not yet opened when DataAdapter's Fill or Update is called, and implicitly closed if it had been implicitly opened. This approach will produce unnecessary overhead if we need to call a batch of Fill and/or Update methods. In this case we can explicitly open the connection before the batch of calls, and explicitly close it after them.

In fact, a used connection is not destroyed. Instead it is by default pooled. When you open a connection with the same connection string as the used one before it times out, you are actually using the same pooled connection. To turn off the default connection pooling, add "OLE DB Services=-4" into the connection string of a OLE DB.NET data provider, or "Pooling=False" to the connection string of a SQL Client .NET data provider.

Because an existing connection in the pool is only reused if the new one has the same connection string, if you have a data access middle tier to talk to database, to enable that the same connection is reused when different users acquire the same data, you can not include the user's credentials in the connection string. You have to use the same connection string for different users. This means that instead of letting the database to validate the user, the middle tier should do it itself using network security measures such as SSL.

5.7.1 Connection Pooling in ADO.NET

Establishing a connection with a database server is a hefty and high resource consuming process. If any application needs to fire any query against any database server, we need to first establish a connection with the server and then execute the query against that database server.

When you are writing any stored proc or a query, the query returns the results with better response time than the response time, when you execute that same query from any of your client applications. I believe, one of the reasons for such behavior is the overheads involved in getting the desired results from the database server to the client application; and one of such overheads is establishing the connection between the ADO.

Web applications frequently establish the database connection and close them as soon as they are done. Also notice how most of us write the database driven client applications. Usually, we have a configuration file specific to our application and keep the static information like Connection String in it. That in turn means that most of the time we want to connect to the same database server, same database, and with the same user name and password, for every small and big data.

ADO.NET with IIS uses a technique called connection pooling, which is very helpful in applications with such designs. What it does is, on first request to database, it serves the database call. Once it is done and when the client application requests for closing the connection, ADO.NET does not destroy the complete connection, rather it creates a connection pool and puts the released connection object in the pool and holds the reference to it. And next time when the

Notes

request to execute any query/stored proc comes up, it bypasses the hefty process of establishing the connection and just picks up the connection from the connection pool and uses that for this database call. This way, it can return the results comparatively faster.

5.7.2 Testing Connection Pooling Behavior

All forms of connection pooling work by examining the connection string. A connection is reused only if the connection string matches exactly. Most ADO.NET providers use a case-sensitive full-text matching algorithm. This means that even if you have the same connection string parameters, but they are in a different order, the connections isn't reused. To ensure that your connections can be reused, store the connection string in a single location (such as a configuration file) and don't enter it directly in your code.

The following example demonstrates how connection pooling works with a single Connection object:

```
string conString1 = "Data Source=localhost;" + "Initial
Catalog=Northwind;Integrated Security=SSPI";
string conString2 = "Data Source=localhost;" + "Initial
Catalog=pubs;Integrated Security=SSPI";
SqlConnection con = new SqlConnection();
con.ConnectionString = conString1;
con.Open();
// The initial pool is created (we'll call it pool A).
con.Close()
// The connection is returned to pool A.
con.ConnectionString = conString2;
con.Open();
// A new pool is created (pool B), because the connection strings differ.
con.Close()
// This connection is returned to pool B.
con.ConnectionString = conString1;
con.Open();
// The open connection from pool A is reused. This saves time.
con.Close()
// The connection is returned to pool A.
```

You'll notice several important factors in this example:

- It doesn't matter whether you are using one Connection object or several Connection objects. When you call Close(), the underlying connection is placed in the pool. When you call Open(), the provider searches the pool for available connections.
- Connections are reused only if the parameters match. This makes sense in the previous example, you wouldn't reuse a connection to the Northwind database if the client thinks it is opening a connection to the pubs database. Similarly, a change in security information (for example, the user account and password) or the location of the database server would cause problems if connections were reused indiscriminately.
- If no pool exists, it is created the first time you call Open(). Depending on connection string settings, the pool may be initially populated with a set number of connections, or it may be limited to a certain maximum number.



Case Study

Insurance Firm Saves \$2.5 Million Annually and Supports New Business Growth

Irish Life, based in Dublin, Ireland, offers life insurance and pension products to individuals and businesses throughout the country. To respond to a difficult economy, the company sought a more cost-effective environment to support its critical insurance administration application. Irish Life migrated the application to a new solution based on Microsoft database technology. As a result, the company will save US\$2.5 million in annual operating costs and will be able to increase its business. The solution also lowers system runtimes by 75 percent, boosts application availability, and increases developer productivity.

Business Needs

Irish Life provides risk protection, pensions, savings, and investment products to customers across Ireland. The company's retail division, Irish Life Financial Services, sells and distributes these products to more than 1 million people.

Because the Irish economy has been in recession for the past four years, the market for retail life insurance and pensions has fallen by more than 65 percent. To help compensate for this market trend, Irish Life wanted to find a more cost-effective technology to support CLOAS, its life insurance administration system. "This is our core system, on which all policies are administered, and it represents a significant element of the company's total operating costs," says Brendan Healy, CIO, Irish Life. "We were paying a premium for the IBM mainframe environment it was based on."

To differentiate itself in a challenging business climate, Irish Life must also guarantee the highest availability for CLOAS, which is used internally to administer policies and externally by customers for self-service policy administration.

However, ensuring high availability had become difficult. For instance, the system's underlying batch system architecture was not flexible or efficient, and nightly batch processing routinely took 8 hours or more to complete. "That meant the system was sometimes inaccessible by our internal and external customers in the morning," Healy says. "This downtime was not acceptable to us, because our business goal is to provide market-leading service. Part of that is having our systems available at all times."

In early 2009, Irish Life began the process of searching for a more cost-effective solution for CLOAS. This work was led by Denis McLoughlin, Head of Finance and IT, Retail Business, Irish Life.

Questions:

1. How Irish Life has improved the application's performance and reduced the system batch-processing time by 75 percent?
2. Why Irish Life search for a more cost-effective solution for CLOAS?

Notes

Self Assessment

Multiple Choice Questions

6. Which one thing you need to take care of in your ASP.Net application to take advantage of connection pooling?

Notes

- a. You need to set the Pooling property of the Connection object to True
 - b. You need to use the same exact connection string whenever you open a database connection
 - c. You need to call the ReleaseToPool() method after the connection is closed
 - d. You does not need to explicitly close the connection with the Close() method
7. Which property of the database must the connection object contain?
- a. Location
 - b. Type
 - c. Both a and b
 - d. All of the above.
8. Which is not a tab on the DataLink Property window?
- a. Advanced
 - b. Adapter
 - c. Connection
 - d. Provider
9. The first step of configuring a DataAdapter is to select:
- a. an adapter object.
 - b. a connection object.
 - c. a database object.
 - d. None of the above.
10. Which is **not** an ADO.NET DataAdapter Object?
- a. SQLDataAdapter
 - b. QueryDataAdapter
 - c. Both a and b.
 - d. All of the above.
11. Which object contains the Position property of the current record in a dataset?
- a. BindingContext
 - b. BindingData
 - c. DataBound
 - d. DataContext
12. Which is specified by the DataMember Property?
- a. Connection object
 - b. Database field
 - c. Database table
 - d. Dataset object
13. The first record in a dataset has a position property of:
- a. zero.
 - b. one.
 - c. any value defined by the programmer.
 - d. All of the above.
14. Which is specified by the DataSource Property?
- a. Connection object
 - b. DataAdapter object
 - c. Database table
 - d. Dataset object
15. Which DataAdapter Query Type can be used with the Access database?
- a. Use SQL statements.
 - b. Create new stored procedure.
 - c. Both a and b.
 - d. All of the above.

5.8 Summary

Notes

- A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.
- The ADO.NET provides data access services in the Microsoft.NET Platform.
- The ADO.NET classes are found in System.Data.dll and integrated with XML classes in System.Xml.dll.
- There are two central components of ADO.NET classes-Data Set and Data Provider.
- Data Provider is a set of components including Connection object, Command object and Data Reader object.
- SqlCommand class provides four different methods to execute a command. They are: ExecuteReader, ExecuteNonQuery, ExecuteScalar and, newest but not least, ExecuteXmlReader.
- ADO.Net worked in two Modes – Connected and Disconnected.
- ADO.Net Connected Mode use Data Provider.
- ADO.Net Disconnected mode use DataSet.
- Data Sets are of two types – Typed DataSet and Untyped DataSet.
- Data Set is a copy of a extracted data being downloaded and cached in the client system.
- The Windows Forms Data Grid control displays data in a series of rows and columns. The Windows Forms Data Grid control provides a user interface to ADO.NET data sets.
- A Data Set can be saved to an XML file and then loaded back into memory very easily.
- Complex data binding is the ability of a control to bind to more than one data element, typically more than one record in a database, or to more than one of any other type of bindable data element.
- When you open a connection with the same connection string as the used one before it times out, you are actually using the same pooled connection.

5.9 Keywords

ADO.NET: The ADO.NET is the new database technology of the .NET (DotNet) platform, and it builds on ADO (ActiveX Data Objects).

DataAdapter: A DataAdapter acts as a bridge between a data source and a data class which is disconnected from the data source.

DataBinding: It is the ability to bind some elements of a data source with some graphical elements of an application.

Database Connection Pool: A database connection pool is a set of database connections that are held open with the database and are kept ready to serve.

DataSet: A DataSet is the equivalent for client side cursor, which holds a portion of the database after execution of a query.

Notes



Lab Exercise

1. Write a program to connect a database with the frontend.
2. Write a program to select and display the records from the table.

5.10 Review Questions

1. What are major differences between classic ADO and ADO.NET?
2. What are the two fundamental objects in ADO.NET?
3. Which namespaces are required to enable the use of databases in ASP.NET pages?
4. What is Connection Pooling? State the pre-requisites for connection pooling.
5. Which properties are used to bind a *DataGridView* control? Explain with an example.
6. Mention different types of data providers available in .NET Framework.
7. How can you identify whether or not any changes are made to the *DataSet* object since it was last loaded?
8. What is the use of Connection object and *CommandBuilder* class?
9. Describe the disconnected architecture of ADO.NET's data access model.
10. Out of Windows authentication and SQL Server authentication, which authentication technique is considered as a trusted authentication method.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (b) | 2. (c) | 3. (a) | 4. (a) |
| 5. (a) | 6. (b) | 7. (c) | 8. (b) |
| 9. (b) | 10. (b) | 11. (a) | 12. (c) |
| 13. (a) | 14. (d) | 15. (a) | |

5.11 Further Readings



Books

ASP.NET, by Yashwanth Kanethkar

Practical ASP. NET 3.5 Projects for Beginners, by B.M. Harwani



Online link

<http://www.w3schools.com/aspnet/default.asp>

Unit 6: Error Handling

Notes

CONTENTS

Objectives

Introduction

6.1 Logics of Error Handling

6.1.1 Redirecting the User to an Error Page

6.1.2 Custom Error Handling in ASP.NET

6.2 Aspects of Error Handling

6.3 Levels of Error Handling

6.3.1 Local Error Handling

6.3.2 Page Level

6.3.3 Application Level

6.3.4 HTTP Module Level

6.4 Web Application Error Handling in ASP.NET

6.4.1 The Problem

6.4.2 The Solution

6.4.3 Using the Page Error or on Error Sub

6.4.4 Using the Global.Asax File

6.4.5 Using the Web.Config File

6.5 Error Handling in ASP.NET Web API

6.5.1 HttpError

6.5.2 HttpResponseMessage

6.5.3 Error Detail

6.5.4 Applying Error Handling to Handle Invalid Model States

6.6 Summary

6.7 Keywords

6.8 Review Questions

6.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Define the logics of error handling
- Describe all aspects of error handling
- Explain the levels of error handling
- Define the web application error handling in ASP.NET
- Define error handling in ASP.NET web API

Introduction

ASP.NET applications must be able to handle errors that occur during execution in a consistent manner. ASP.NET uses the common language runtime (CLR), which provides a way of notifying applications of errors in a uniform way. When an error occurs, an exception is thrown. An exception is any error, condition, or unexpected behavior that an application encounters.

In the .NET Framework, an exception is an object that inherits from the System.Exception class. An exception is thrown from an area of code where a problem has occurred. The exception is passed up the call stack to a place where the application provides code to handle the exception. If the application does not handle the exception, the browser is forced to display the error details.

As a best practice, handle errors in at the code level in Try/Catch/Finally blocks within your code. Try to place these blocks so that the user can correct problems in the context in which they occur. If the error handling blocks are too far away from where the error occurred, it becomes more difficult to provide users with the information they need to fix the problem.

6.1 Logics of Error Handling

When an unhandled immunity propagates, the user may well be redirect to an error linen using different ASP.NET make-up settings. However, such a redirection may be prohibited to begin with by treatment the exceptions that will get thrown. Error handling in ASP.NET therefore, may be split into two separate logics:

1. Redirect the user to an error page when errors go unhandled.
2. Handle exceptions when they get thrown.

6.1.1 Redirecting the User to an Error Page

There are two different possibilities where it can be specified which page the user will be redirected to, when errors go unhandled:

1. Page level (applies to errors that happen within a single page).
2. Application level (applies to errors that happen anywhere in the application).

Page Level

A page-level handler returns the user to the page where the error occurred, but because instances of controls are not maintained, there will no longer be anything on the page. To provide the error details to the user of the application, you must specifically write the error details to the page.

You would typically use a page-level error handler to log unhandled errors or to take the user to a page that can display helpful information.

This code example shows a handler for the Error event in an ASP.NET Web page. This handler catches all exceptions that are not already handled within Try/Catch blocks in the page.

```
private void Page_Error(object sender, EventArgs e)
{
    Exception exc = Server.GetLastError();

    // Handle specific exception.
```

Notes

```

if (exc is HttpUnhandledException)
{
    ErrorMessageTextBox.Text = "An error occurred on this page. Please verify your
" +
    "information to resolve the issue."
}
// Clear the error from the server.
Server.ClearError();
}

```

After you handle an error, you must clear it by calling the `ClearError` method of the `Server` object (`HttpServerUtility` class), otherwise you will see an error that has previously occurred.

Application Level

You can handle default errors at the application level either by modifying your application's configuration or by adding an `Application_Error` handler in the `Global.asax` file of your application.

You can handle default errors and HTTP errors by adding a `customErrors` section to the `Web.config` file. The `customErrors` section allows you to specify a default page that users will be redirected to when an error occurs. It also allows you to specify individual pages for specific status code errors.

```

<configuration>
  <system.web>
    <customErrors mode="On"
defaultRedirect="ErrorPage.aspx?handler=customErrors%20section%20-
%20Web.config">
      <error statusCode="404"
redirect="ErrorPage.aspx?msg=404&handler=customErrors%20section%20-
%20Web.config"/>
    </customErrors>
  </system.web>
</configuration>

```

Unfortunately, when you use the configuration to redirect the user to a different page, you do not have the details of the error that occurred.

However, you can trap errors that occur anywhere in your application by adding code to the `Application_Error` handler in the `Global.asax` file.

```

void Application_Error(object sender, EventArgs e)
{
    Exception exc = Server.GetLastError();

    if (exc is HttpUnhandledException)
    {
        // Pass the error on to the error page.
        Server.Transfer("ErrorPage.aspx?handler=Application_Error%20-
%20Global.asax", true);
    }
}

```


6.1.2 Custom Error Handling in ASP.NET

Three possible error pages is displayed is based on two variables:

- The configuration information in the <customErrors> section, and
- Whether the user is visiting the site locally or remotely.

The <customErrors> section in Web.config has two attributes that affect what error page is shown: defaultRedirect and mode. The defaultRedirect attribute is optional. If provided, it specifies the URL of the custom error page and indicates that the custom error page should be shown instead of the Runtime Error YSOD. The mode attribute is required and accepts one of three values: On, Off, or RemoteOnly. These values have the following behavior:

- On - indicates that the custom error page or the Runtime Error YSOD is shown to all visitors, regardless of whether they are local or remote.
- Off - specifies that the Exception Details YSOD is displayed to all visitors, regardless of whether they are local or remote.
- RemoteOnly - indicates that the custom error page or the Runtime Error YSOD is shown to remote visitors, while the Exception Details YSOD is shown to local visitors.

Unless you specify otherwise, ASP.NET acts as if you had set the mode attribute to RemoteOnly and had not specified a defaultRedirect value. In other words, the default behavior is that the Exception Details YSOD is displayed to local visitors while the Runtime Error YSOD is shown to remote visitors. You can override this default behavior by adding a <customErrors> section to your web application's Web.config file.



Task Write a program to handle an exception using try-catch block.

6.2 Aspects of Error Handling

Following aspects are defined for error handling:

Tracing: Tracing the program execution at page level or application level.

Error handling: Handling standard errors or custom errors at page level or application level.

Debugging: Stepping through the program, setting break points to analyze the code.

Tracing

```
System.Diagnostics.Trace
<system.diagnostics>
<trace autoflush="true" indentsize="2">
<listeners>
<add name="myListener" type="System.Diagnostics.TextWriterTraceListener"
initializeData="log.txt" />
</listeners>
</trace>
</system.diagnostics>
Page.Trace

<system.web>
<trace enabled="true" pageOutput="true" traceMode="SortByTime"
mostRecent="true" />
```

Notes

```

</system.web>
Redirect System.Diagnostics.Trace to Page.Trace

<listeners>
<add name="WebPageTraceListener" type="System.Web.WebPageTraceListener,
System.Web, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"/>
</listeners>
Redirect Page.Trace to System.Diagnostics.Trace

<trace enabled="true" pageOutput="true" traceMode="SortByTime"
mostRecent="true" writeToDiagnosticsTrace="true" />
Bidirectional redirection is OK.

Switch

<configuration>
<system.diagnostics>
<switches>
<add name="ImportantSwitch" value="1" /><!-- This is for the BooleanSwitch
->
<add name="LevelSwitch" value="3" /><!-- This is for the TraceSwitch ->
<add name="SourceSwitch" value="4" /><!-- This is for the SourceSwitch ->
</switches>
</system.diagnostics>
</configuration>
Dim aSwitch As New BooleanSwitch("ImportantSwitch", "Show errors")
System.Diagnostics.Trace.WriteIf(aSwitch.Enabled, "The Switch is enabled!")

Dim tSwitch As New TraceSwitch("LevelSwitch", "Trace Levels")
System.Diagnostics.Trace.WriteIf(tSwitch.TraceInfo, "The Switch is 3 or
more!")

Dim sSwitch As New SourceSwitch("SourceSwitch", "Even More Levels")
System.Diagnostics.Trace.WriteIf(sSwitch.ShouldTrace(TraceEventType.Warning),
"The Switch is 3 or more!")

```

Exception and Error Handling

Catch what you expect: Use a Try/Catch around error-prone code. This can always catch specific exceptions that you can deal with, such as `System.IO.FileNotFoundException`.

Rather than catching exceptions around specific chunks of code at the page level, consider using the page-level error handler (`Page.OnError`) to catch specific exceptions that might happen anywhere on the page.

```

Protected Overrides Sub OnError(ByVal e As System.EventArgs)
    Dim AnError As System.Exception = Server.GetLastError()
    If (TypeOfAnError.GetBaseException() Is SomeSpecificException) Then
Response.Write ("Something bad happened!")
Response.StatusCode = 200

```

Notes

```
Server.ClearError()  
Response.End()  
End If  
End Sub
```

But prepare for unhandled exceptions: Set the Page.Error property if a specific page should show a specific error at page for any unhandled exception. This can also be done using the <%@ Page %> directive or the code behind the property.

Have default error pages for 400 and 500 errors set in your web.config.

```
<system.web>  
<customErrors mode = "On" >  
<error statusCode = "500" redirect = "FriendlyMassiveError.aspx" />  
</customErrors>
```

Have a boilerplate Application_OnError handler that takes into consideration both specific exceptions that you can do something about, as well as all unhandled exceptions that you may want logged.

```
Protected Sub Application_Error(ByVal sender As Object, ByVal e As  
System.EventArgs)  
Dim bigError As System.Exception = Server.GetLastError()  
'Example checking for HttpRequestValidationException  
If (TypeOfAnError.GetBaseException() Is HttpRequestValidationException)  
Then  
System.Diagnostics.Trace.WriteLine(bigError.ToString)  
Server.ClearError()  
End If  
End Sub
```

Debugging

An interesting CLR Internals trick: Call System.Diagnostics.Debugger.Launch within your assembly, even if the assembly was compiled via /debug:pdbonly, and the debugger pops up. The JIT compiler compiles code on the first call to a method, and the code that it generates is debuggable because JIT knows that a debugger is attached.

6.3 Levels of Error Handling

There are different levels where you could handle exceptions.

- Locally (method level), where exceptions could be thrown.
- Page level by handling the Page.Error event.
- Application level by handling the HttpApplication.Error event.
- HTTP Module level by handling the HttpApplication.Error event.

6.3.1 Local Error Handling

The try-catch statement consists of a try block followed by one or more catch clauses, which specify handlers for different exceptions. When an exception is thrown, the common language runtime (CLR) looks for the catch statement that handles this exception. If the currently executing method does not contain a catch block, the CLR looks at the method that called the current method, and so on, up the call stack. If no catch block is found, then the CLR displays an unhandled exception message to the user and stops execution of the program.

The following code example shows a common way of using try/catch/finally to handle errors.

Notes

```
try
{
    file.ReadBlock(buffer, index, buffer.Length);
}
catch (FileNotFoundException e)
{
    Server.Transfer("NoFileErrorPage.aspx", true);
}
catch (System.IO.IOException e)
{
    Server.Transfer("IOErrorPage.aspx", true);
}

finally
{
    if (file != null)
    {
        file.Close();
    }
}
```

In the above code, the try block contains the code that needs to be guarded against a possible exception. The block is executed until either an exception is thrown or the block is completed successfully. If either a `FileNotFoundException` exception or an `IOException` exception occurs, the execution is transferred to a different page. Then, the code contained in the finally block is executed, whether an error occurred or not.

6.3.2 Page Level

To add page-level error handling, create the `Page_Error` event handler in the `Default.aspx` file, as shown below. The 1st statement transfers the error response to the named URL. The boolean value specifies if `QueryString/Form` collections are to be cleared (`true`) or preserved (`false`). The 2nd statement indicates that the error has been handled and thus be cleared.

```
protected void Page_Error(object sender, EventArgs e)
{
    Server.Transfer("ErrorPage1.htm", true);
    Server.ClearError();
}
```

Create a dummy page called "ErrorPage1.htm" and include appropriate contents in the page.



Notes "You will want to be cautious when displaying detailed information to the end user of the application, especially when the application is running on the Internet. A more appropriate action would be to display a message to the user notifying them that an error has occurred, and then actually logging the specific error details in the event log."

Now, re-run the app. Note that the exception is now handled at the above error handler.

Notes

6.3.3 Application Level

If an error is uncaught at the page level, it is propagated to the application level. To setup application level error handling, unlike page-level error handlers, you create the Global.asax file. (Go to Project > Add New Item > select Global Application Class from the installed templates. Keep the name as Global.asax. Open the file and you will see various empty application-level event handlers, including one for handling errors. Create the above Global.asax file and add the code below.

```
protected void Application_Error(object sender, EventArgs e)
{
    Server.Transfer("ErrorPage2.htm", true);
    Server.ClearError();
}
```

Create a dummy page called "ErrorPage2.htm" and include appropriate contents in the page. Now, comment out the Page_Error method in the Default.aspx.cs file and re-run the app. Note that the exception is now handled at the above error handler.



Caution If you do not clear the error, the exception would propagate. It becomes difficult to handle the exception.

6.3.4 HTTP Module Level

Instead of handling application errors in global.asax, exceptions may also be handled by attaching an HTTP Module which would have a handler attached to the Application.Error event. This method would be triggered before the corresponding application handler would be invoked. Such an implementation would be beneficial if you have multiple projects with the same global error handling implementation. In such a scenario, you could create a module and attach it to each web application you have.



Did u know? In 1996 the Ariane V rocket exploded due in part to the Ada programming language exception handling policy of aborting computation on arithmetic error – a floating point to integer conversion overflow – which would not have occurred if IEEE 754 exception-handling policy had been used.

Self Assessment

Multiple Choice Questions:

1. Which of the following is NOT a .NET Exception class?
 - a. Exception
 - b. StackMemoryException
 - c. DivideByZeroException
 - d. OutOfMemoryException
 - e. InvalidOperationException
2. Which of the following statements is correct about an Exception?
 - a. It occurs during compilation.
 - b. It occurs during linking.
 - c. It occurs at run-time.

6.4 Web Application Error Handling in ASP.NET

All web applications are prone to throw two kinds of errors, conditional and unconditional. Conditional errors are related to the business logic specific rules, whereas unconditional errors may occur due to a network slowdown, database crash, server breakdown, etc. All these errors, if left untouched may manifest in a clumsy way threatening the audience. ASP.NET provides various error handling techniques to elegantly trap and handle the errors. This article talks about the use of the web configuration file to corner the errors and prop up customized error pages in the application.



Example: Classic ASP uses `Server.GetLastError` to return an `ASPErrors` object. You can and should still use `Server.GetLastError` in .NET, but this now returns a `System.Exception`.

6.4.1 The Problem

Errors will occur in the applications. We try to trap for most errors using try-catch blocks; however, we usually do not cover every possible exception. What goes on when an unhandled error occurs? Usually the user is brought to IIS's default error webpages (usually located in `c:\winnt\help\iishelp\common`). The downsides are you have no clue when this occurs and the actual page does not have your website's look and feel.

Errors are a development fact, but we strive to eliminate or handle them gracefully. With this in mind, we need to know:

1. When an error occurs
2. Where it occurred
3. What the error is

Having a central location such as the event log, database or some other log file to log errors is essential for debugging this problem later.

IIS provides great error-handling capabilities. There are some problems with these though. Sometimes we know an error will occur, and that we cannot always trap for it in a nice way without overriding the site's fall behind error redirection page. For example, upon access to a source that requires authentication, we should redirect to an application's login page. Also, a common problem exists with Web hosting companies. If you have a hosted Web site, you usually have no control over its IIS settings. Thus, setting up custom mistake pages can be next to impossible in traditional ASP. This really is eliminated with ASP.NET, while you will learn as you continue reading.

6.4.2 The Solution

For such a list of problems, the solution is actually pretty simple. There are three places in ASP.NET to define what happens to these unhandled errors in the actual order of error handling events:

1. On the `aspx` or associated codebehind page in the `Page_Error` sub.
2. In the `global.asax` file's `Application_Error` sub.
3. In the `web.config` file's `customErrors` section.

6.4.3 Using the Page_Error or on Error Sub

Notes

The Page_Error event handler provides a way to trap errors that occur at the page level. You can simply display error information (as the sample code to follow does), or you can log the event or perform some other action.

Note This example displays detailed error information in the browser only for demonstration purposes. You will want to be cautious when displaying detailed information to the end user of the application, especially when the application is running on the Internet. A more appropriate action would be to display a message to the user notifying them that an error has occurred, and then actually logging the specific error details in the event log.

This example throws a null exception, which forces an error to occur in the Page_Load event handler. Follow these steps to create the initial page that will demonstrate using the Page_Error event handler.

- Follow these steps to add a new file named PageEvent.aspx to your project:
 - ❖ Open Microsoft Visual Studio .NET.
 - ❖ In Solution Explorer, right-click the project node, point to Add, and then click Add Web Form.
 - ❖ In the Name text box, type PageEvent.aspx, and then click Open.

- Add the following code to PageEvent.aspx:

```
<script language=C# runat="server">
void Page_Load(object sender, System.EventArgs e)
{
    throw(new ArgumentNullException());
}

public void Page_Error(object sender, EventArgs e)
{
    Exception objErr = Server.GetLastError().GetBaseException();
    string err = "<b>Error Caught in Page_Error event</b><hr><br>" +
        "<br><b>Error in: </b>" + Request.Url.ToString() +
        "<br><b>Error      Message:      </b>" +
objErr.Message.ToString() +
        "<br><b>Stack Trace:</b><br>" +
    objErr.StackTrace.ToString();
    Response.Write(err.ToString());
    Server.ClearError();
}
</script>
```



Notes In this code sample, the AutoEventWireup attribute is not explicitly set. If you do not explicitly assign a value to the AutoEventWireup attribute, the default value true is used. If you are using Visual Studio .NET to develop your applications, the Web Form template code explicitly sets the AutoEventWireup attribute value to false. There is an important difference between the default value that ASP.NET uses, and the default value that the Visual Studio .NET template code assigns to this attribute. If the AutoEventWireup attribute value is set to false, the event handlers that are declared in the .ASPX page do not fire. This may be confusing if you do not know about this functionality.

Notes

3. From the File menu, click Save PageEvent.aspx.
4. Right-click the page, and then click View in Browser to run the page. Notice that the error is thrown and reported according to the code specifications.



Notes You may notice that the code issues a call to `Server.ClearError`. This prevents the error from continuing to the `Application_Error` event handler.

In addition, you should also take note of the `Inherits` attribute in the `@ Page` directive. If `Inherits` is set, you must build the project before you browse to the page. If you do not build the project first, you receive the following error message:

```
'Project.PageEvent' is not a valid type
```

6.4.4 Using the Global.Asax File

The `Global.asax` file is in the root application directory. While Visual Studio .NET automatically inserts it in all new ASP.NET projects, it's actually an optional file. It's okay to delete it—if you aren't using it. The `.asax` file extension signals that it's an application file rather than an ASP.NET file that uses `aspx`.

The `Global.asax` file is configured so that any direct HTTP request (via URL) is rejected automatically, so users cannot download or view its contents. The ASP.NET page framework recognizes automatically any changes that are made to the `Global.asax` file. The framework reboots the application, which includes closing all browser sessions, flushes all state information, and restarts the application domain.

6.4.5 Using the Web.Config File

In ASP, the configuration properties of the applications are stored in a binary encoded data format. They can be accessed either by the script or through the IIS Management Console. For example, if our website is hosted on a third party server and we want to provide customized error pages for our web site we need to contact that party, and pay to make them configure our web site using IIS. So one of the main problems with this particular way of configuration is portability. The configuration properties of the ASP applications have to be recreated and set manually when transferred from one machine to another using the IIS admin tool. This issue was resolved with the advent of ASP.NET. ASP.NET took a XML-based configuration approach. ASP.NET provides 2 configuration files: `machine.config` and `web.config`. The `machine.config` is the server configuration file that is common to all ASP.NET applications. By default, this gets installed in the `/WinNT/Microsoft.NET/Framework/version/config` folder, where `version` is the version of the .NET framework. On the other hand, `web.config` is the configuration file for the individual applications. Each ASP.NET application has a `web.config` file installed in the project directory. This file manages the properties of that particular web application it is associated with. As `machine.config` is common to all the ASP.NET applications, it requires changes only in very rare circumstances. It is the `web.config` file that is engaged in configuring the individual applications. This `web.config` file may be used to provide custom error pages for our application.

The section of `web.config` that is under consideration for customizing the error pages is shown below:

```
<configuration>
<system.web>
<customErrors>
```

```
<!-- Modify this part - >
</customErrors>
</system.web>
</configuration>
```

The customErrors element is to be modified for any application errors. The customErrors element has an attribute "mode" that sets the error mode for an application. The error mode can be on, off or RemoteOnly, the details of which are discussed below:

- <customErrors mode="On">... </customErrors>: Displays only user-friendly error messages and not the default error text.
- <customErrors mode="Off">... </customErrors>: Toggles off the display of user-friendly messages and only the default error messages appear.
- <customErrors mode="RemoteOnly">... </customErrors>: This is the default mode. Displays default/detailed error messages to users of the local web server machine. It displays user-friendly messages when accessed from elsewhere.

6.5 Error Handling in ASP.NET Web API

ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices. With WebAPI content negotiation, we can return data based on the client requests. What I mean is, if the client is requesting the data to be returned as JSON or XML, the WebAPI framework deals with the request type and returns the data appropriately based on the media type. By default WebAPI provides JSON and XML based responses.

WebAPI is an ideal platform for building pure HTTP based services where the request and response happens with HTTP protocol. The client can make a GET, PUT, POST, and DELETE request and get the WebAPI response appropriately.

In summary, the WebAPI is

- An HTTP Service
- Designed for broad reach
- Uses HTTP as an Application protocol, not a transport protocol

Now that's out of the way, let us take a look at what a typical WebAPI error's HTTP content looks like:

```
{
  "Message": "Product with id = 12 not found"
  "MessageDetail": "No product found that matches the controller named 'ID'."
}
```

So the error is a collection of key-value pairs that provide data about what went wrong. This collection is then sent back to the user in the specified content type either through HTTP content negotiation. In the example above, the content is actually JSON.

But if you had "text/xml" in your request's Accept header for example, you might get this response instead:

```
<Error>
<Message> Product with id = 12 not found 'http://localhost/Foo'.</Message>
<MessageDetail> No product found that matches the controller named 'ID'.</
MessageDetail>
< /Error>
```

Notes This simple format makes it easy for clients to understand what went wrong or to extract relevant error information for error logging or reporting.

6.5.1 HttpError

The `HttpError` object provides a consistent way to return error information in the response body. The following example shows how to return HTTP status code 404 (Not Found) with an `HttpError` in the response body:

```
public HttpResponseMessage GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        var message = string.Format("Product with id = {0} not found", id);
        HttpError err = new HttpError(message);
        return Request.CreateResponse(HttpStatusCode.NotFound, err);
    }
    else
    {
        return Request.CreateResponse(HttpStatusCode.OK, item);
    }
}
```

In this example, if the method is successful, it returns the product in the HTTP response. But if the requested product is not found, the HTTP response contains an `HttpError` in the request body. The response might look like the following:

HTTP/1.1 404 Not Found

Content-Type: application/json; charset=utf-8

Date: Thu, 09 Aug 2012 23:27:18 GMT

Content-Length: 51

```
{
  "Message": "Product with id = 12 not found"
}
```

Notice that the `HttpError` was serialized to JSON in this example. One advantage of using `HttpError` is that it goes through the same content-negotiation and serialization process as any other strongly-typed model.

Instead of creating the `HttpError` object directly, you can use the `CreateErrorResponse` method:

```
public HttpResponseMessage GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        var message = string.Format("Product with id = {0} not found", id);
        return Request.CreateErrorResponse(HttpStatusCode.NotFound, message);
    }
}
```

```

}
else
{
    return Request.CreateResponse(HttpStatusCode.OK, item);
}
}

```

CreateErrorResponse is an extension method defined in the System.Net.Http.HttpRequestMessageExtensions class. Internally, CreateErrorResponse creates an HttpError instance and then creates an HttpResponseMessage that contains the HttpError.

6.5.2 HttpResponseMessage

What happens if a Web API controller throws an uncaught exception? By default, most exceptions are translated into an HTTP response with status code 500, Internal Server Error.

The HttpResponseMessage type is a special case. This exception returns any HTTP status code that you specify in the exception constructor. For example, the following method returns 404, Not Found, if the id parameter is not valid.

```

public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        throw new HttpResponseMessage(HttpStatusCode.NotFound);
    }
    return item;
}

```

For more control over the response, you can also construct the entire response message and include it with the HttpResponseMessage:

```

public Product GetProduct(int id)
{
    Product item = repository.Get(id);
    if (item == null)
    {
        var resp = new HttpResponseMessage(HttpStatusCode.NotFound)
        {
            Content = new StringContent(string.Format("No product with ID = {0}",
            id)),
            ReasonPhrase = "Product ID Not Found"
        }
        throw new HttpResponseMessage(resp);
    }
    return item;
}

```



Task Write the code to http error handling.

6.5.3 Error Detail

You may have noticed the actual “includeErrorDetail” parameter earlier in this unit on the `HttpError` constructor. This is because typically, servers want to send different error data depending on whether the user is just consuming the service or whether the client requires additional debugging data to exactly know what went wrong on the server. By default, WebAPI will not really send error details to remote user and will provide these extra error details to user on the local machine. Returning to the first example, whereas a local user would see this:

```
{
  "Message": "Product with id = 12 not found"
  "MessageDetail": "No product found that matches the controller named 'ID'."
}
```

A remote client would only see this:

```
{
  "Message": "Product with id = 12 not found"
}
```

The difference between `Message` and `Message Detail` above is that `MessageDetail` contains error information that is WebAPI-specific that remote clients should not have to see in most cases. The precise meaning of error detail depends upon the case. For exceptions, error detail includes the exception message, exception type, the stack trace, and the inner exceptions. Only a vague “An error has occurred.” information is sent back to remote user for exceptions by default. For model states, any design error messages are sent to remote clients. Model error exceptions, however, are considered detail and won’t get sent to remote clients by default.

6.5.4 Applying Error Handling to Handle Invalid Model States

One of the most common use of WebAPI’s error handling is to instantly send back an error if the model state is invalid. This can be achieved easily by implementing the following action filter:

```
public class ValidationFilterAttribute : ActionFilterAttribute
{
    public override void OnActionExecuting( HttpActionContext actionContext)
    {
        if (!actionContext.ModelState.IsValid)
        {
            actionContext.Response=actionContext.Request.CreateErrorResponse(
                HttpStatusCode.BadRequest, actionContext.ModelState);
        }
    }
}
```



Caution Attempting to use the `JScript Error` object in an ASP.NET page may produce unintended results. This results from the potential ambiguity between the `JScript Error` object and the `Error` event of the ASP.NET page. Use the `System.Exception` class instead of the `Error` object for error handling in ASP.NET pages.



Case Study

California Gives More Than 38 Million People BI about Government Salaries with Microsoft

To give citizens information about state and local government employee salaries, the California State Controller's Office engaged SymSoft Solutions to help deploy a business intelligence (BI) solution based on the Microsoft platform. With it, the public can create custom BI using 15 gigabytes of data and 5 million records. The solution also facilitates activism, speeds efficiency, and delivers a scalable foundation to support future BI projects.

Business Needs

The California State Controller's Office manages the State's \$100 billion budget, which equates to the world's ninth-largest economy. Before 2010, the only insight that taxpayers had into state expenditures, including employee salaries, came from the media. After a high-profile controversy over excessive compensation, the State Controller decided to increase government transparency and reduce waste and fraud by giving citizens easy insight into public state and local government payroll data.

In 2010, IT staff designed and deployed an initial reporting solution in just three months based on the Microsoft platform. Known as the Government Compensation in California (GCC) web site, it listed the salaries for some positions in large tables. "The previous solution displayed about 500 rows of information per page," says Steve Champeau, Data Processing Manager at the California State Controller's Office. "If you wanted to find out who was the highest paid county employee in the state, you would have had to sort and merge the data from each of the 58 county screens." In addition, to create reports or visualizations, people had to manually cut, paste, and format data using applications such as Microsoft Excel.

To better meet its citizens' needs, the Controller's Office decided to create a new BI solution that improved insight into payroll information and provided greater scalability to support future requirements.

Questions:

1. State the Business needs of this project.
2. What benefits Company get after migrating to .Net platform?

Self Assessment

Multiple Choice Questions:

6. Which of the following statements are correct about exception handling in C#.NET?
 - a. try blocks cannot be nested.
 - b. In one function, there can be only one try block.
 - c. An exception must be caught in the same function in which it is thrown.
 - d. While throwing a user-defined exception multiple values can be set in the exception, object.
7. Exceptions can be thrown even from a constructor, whereas error codes cannot be returned from a constructor.
 - a. True
 - b. False

Notes

8. Which of the following is not an Exception?
 - a. Stack Over flow
 - b. Division By Zero
 - c. Insufficient Memory
 - d. Incorrect Arithmetic Expression
 - e. Arithmetic overflow or underflow
9. It is compulsory for all classes whose objects can be thrown with throw statement to be derived from System.Exception class.
 - a. True
 - b. False
10. Application level error handling can be done by
 - a. Using Application_Error Event Handler
 - b. Using Custom Error
 - c. Both (a) and (b)
 - d. None of the above
11. Error handling in ASP.Net has Following aspects
 - a. Tracing and Debugging
 - b. Tracing, Debugging and Error Handling
 - c. Debugging and Error Handling
 - d. Tracing and Error handling
12. Customization of error page can be implemented by adding a value for an attribute "defaultRedirect" in the <customErrors> tag of the file
 - a. Global.asas
 - b. Web.Config
 - c. None
 - d. Machine.Config
13. For page-level error handling, which one is not Correct
 - a. Set the "customErrors" setting in Global.asax
 - b. Add code to the page's Page_Error event to deal with the exception.
 - c. Create at least one custom error page.
 - d. Point your page's ErrorPage property to point to one of your custom error pages
14.object provides a consistent way to return error information in the response body.
 - a. HTTP Error
 - b. Http Response
 - c. Http Exception
 - d. Http_Error Handler
15. Trace object allows you to add custom information to the
 - a. Trace output
 - b. Compile error
 - c. Runtime
 - d. Logic error

6.6 Summary

- An exception is any error, condition, or unexpected behavior that an application encounters.

- In the .NET Framework, an exception is an object that inherits from the System.Exception class.
- When the error attribute is set to "Off", ASP.Net uses its default error page for both local and remote users in case of an error.
- Try-catch is a C# programming construct. The try block holds any code that may or may not produce error and the catch block catches the error.
- Error handling in ASP.NET therefore, may be split into two separate logics: Redirect the user to an error page when errors go unhandled and Handle exceptions when they get thrown.
- Two different possibilities the user will be redirected to, when errors go unhandled are page level and application level.
- Error Handling has three aspects: Tracing, Error Handling and Debugging.
- The Page_Error event handler provides a way to trap errors that occur at the page level.
- Web API is a brand new framework that makes it easy to build HTTP services. As such, it provides several features that make it easy to send back useful and informative error messages in a variety of cases.
- HttpError object provides a consistent way to return error information in the response body.
- The HttpResponseException type is a special case. This exception returns any HTTP status code that you specify in the exception constructor.
- The custom Errors element of the Web.config file is the last line of defence against an unhandled error.
- The difference between Message and Message Detail above is that Message Detail contains error information that is Web API-specific that remote clients should not have to see in most cases.

6.7 Keywords

ASP.NET: It is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services.

Exception Handling: It is the process of responding to the occurrence, during computation, of exceptions anomalous or exceptional situations requiring special processing often changing the normal flow of program execution.

Global.asax: Global.asax file is in the root application directory. The Global.asax file is configured so that any direct HTTP request (via URL) is rejected automatically, so users cannot download or view its contents.

Hypertext Transfer Protocol: The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.

Web API: A Web API (Application Programming Interface) is typically a defined set of HTTP request messages along with a definition of the structure of response messages, typically expressed in JSON or XML.

Web.Config: Web.config is the main settings and configuration file for an ASP.NET web application.

Notes



- Lab Exercise*
1. Create a Web.config file in Visualstudio.
 2. Prepare a flow chart to handle an exception.

6.8 Review Questions

1. What is tracing? Where is it used?
2. What is the difference between application exception and system exception?
3. Can multiple catch blocks be executed?
4. List down the commonly used types of exceptions in .Net.
5. What is difference between the “throw” and “throw ex” in .NET?
6. What happens if an ASP.NET server control with event-handling routines is missing from its definition?
7. Describe the application event handlers in ASP.NET.
8. Explain the concept of error handling in ASP.NET web API.
9. What is the custom error handling in ASP.NET?
10. Explain error handling using the page error or on error sub with an example.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (b) | 2. (c) | 3. (b) | 4. (e) |
| 5. (b) | 6. (d) | 7. (a) | 8. (d) |
| 9. (b) | 10. (c) | 11. (b) | 12. (b) |
| 13. (a) | 14. (a) | 15. (a) | |

6.9 Further Readings



Books

Robust ASP.Net Exception Handling, by Lee Dumond



Online links

<http://books.google.co.in/books?id=g6q4Cd66MdoC&printsec=frontcover&dq=Error+Handling+in+asp.net&hl=en&sa=X&ei=kRcAUJaFMie3rAeW4dGaBg&ved=0CDYQ6AEwAA#v=onepage&q=Error%20Handling%20in%20asp.net&f=false>

Unit 7: Advanced ASP.NET

Notes

CONTENTS

Objectives

Introduction

7.1 Communicating with the Browser

7.1.1 Bookmarklet

7.1.2 Communicating with a Desktop Application from JavaScript

7.1.3 Using the JavaScript Image Class to Make Asynchronous Cross-Domain Requests

7.1.4 Browser Speak: A Concrete Example

7.2 Web.Configuration

7.2.1 ASP.NET Web.Configuration File

7.3 Characteristics of ASP.NET

7.4 New Features in ASP.NET 3.5

7.5 Summary

7.6 Keywords

7.7 Review Questions

7.8 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain about communicate with browser
- Define web.configuration
- Define the characteristic of ASP.NET
- Explain the new features of ASP.NET

Introduction

ASP.Net is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web application for PC, as well as mobile devices.

ASP.Net works on top of the HTTP protocol and uses the HTTP commands and policies to set a browser-to-server two-way communication and cooperation.

ASP.Net is a part of Microsoft .Net platform. ASP.Net applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

Notes

The ASP.NET application codes could be written in either of the following languages:

- C#
- Visual Basic .Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls like text boxes, buttons and labels for assembling, configuring and manipulating code to create HTML pages. In order for an ASP.NET website to function properly, it must be published to a Web server that supports ASP.NET applications. Microsoft's Internet Information Services (IIS) Web server is by far the most common platform for ASP.NET web sites. While there are some open-source options available for Linux-based systems, these alternatives frequently provide less than full assistance for ASP.NET applications.

7.1 Communicating with the Browser

There are numerous scenarios exactly where it is helpful to incorporate a desktop function with the browser. Given that most people these days spend lot of their time surfing the web with browser of choice, it can make sense to provide some kind of incorporation with our desktop software. Often, this will be easy as long as a way to export the current URL or perhaps a selected block of text to the application. For this, we have formulated a very simple application that uses text to speech to opine loud the currently selected block of text in our web browser. Internet Explorer provides many hooks for integrating application logic to the browser, the most popular being support for adding custom tool bars. Chrome plug-in architecture which uses XML for the User interface layout and JavaScript for the application logic.

What about the additional browsers, Google Chrome, Safari, and Opera? Given there is no common plug-in architecture used by all browsers, we can see a huge development effort is required to provide a separate toolbar implementation for each browser.

It will be nice to be able to write one toolbar that could be used across all browsers. This is not possible today, but we can accomplish almost the same effect using Bookmarklet.

7.1.1 Bookmarklet

A bookmarklet is special URL that will run a JavaScript application when clicked. The JavaScript will execute in the actual context of the current web page. Like any other URL, it may be bookmarked and added to our Favourites menu or placed on a Favourites toolbar. Here is a simple example of a bookmarklet:

```
javascript:alert('HelloCountry.');
```

Here is a slightly more complex example that will display the currently selected text in a message box:

```
<a href="javascript:varq='';if(window.getSelection)
q=window.getSelection().toString();elseif(document.getSelection)
q=document.getSelection();elseif(document.selection)
q=document.selection.createRange().text;if(q.length>0)
alert(q);elsealert('please Select Some text First');">ShowSelected Text</
a>
```

Now, drag and drop the bookmarklet it on our Favourites plugin (for IE, we need to right click, select Add to Favourites, and then create this in the Favourites Bar). Get around to a new page,

select a block of text, and click the Show Selected Text button. Once more, the selected text will be shown in a message box. We are able to see the potential of bookmarklets. We can create a bookmarklet for each command of our application and display them on a web page. The user can then select the commands they want to make use of and add them to their own Favourites (either the toolbar or menu). The downside is it is slightly more effort to install than a single toolbar, but on the upside, it gives the user a lot of versatility. They need only choose the commands they are interested in and can choose whether they need them to be accessible from a toolbar or menu. From a creator's perspective, bookmarklets are great as they are supported by all the major browsers. The only thing we need to worry about is making sure our JavaScript code handles differences in browser implementations, something that is actually well documented and understood nowadays.

7.1.2 Communicating with a Desktop Application from JavaScript

Bookmarklets allow us to execute an arbitrary block of JavaScript code just at the click, but how do we use this particular to communicate with a desktop application? The answer is to build a web server into our desktop application as well as issue commands from our JavaScript code using HTTP requests.

Now, before all of us baulk at the idea of creating a web server into our application, it is actually very simple. We do not require a complete web server implementation. We just need to be able to process simple HTTP GET requests. A basic implementation is as follows:

1. Listen for a socket connection on a specific port (80 being the default for HTTP protocol, but we should make use of a different port for our application).
2. Accept a connection and read the actual HTTP GET request.
3. At the actual URL from the GET header and execute the associated command in our application.
4. Send an HTTP Response back to the browser.
5. Close the connection.

The .NET framework 2.0 has an `HttpListener` class and associated `HttpListenerRequest` and `HttpListenerResponse` classes that allow us to implement the above in a few lines of code. On the internet browser, we need a way of giving HTTP requests from our JavaScript code. There are a number of methods for issuing HTTP requests from JavaScript.

The easiest is to write a new Web address into the `document.location` property. This makes the browser to navigate to the brand new location. However, this is not what we should want. We do not want to direct the user to a new page when they click on bookmarklets. Instead, we just want to issue a command to our application while remaining on the same web page.

This sounds like a job with regard to AJAX and the `HttpXml Request`. AJAX provides a easy means of issuing requests to a web server in the background without affecting the present page. However, there is one essential restriction placed on the `HttpXmlRequest` called the same domain origin policy. Browsers restrict `HttpXmlRequests` to the same domain because that used to serve the current page. For example, if we are viewing a page from `abc.com`, we can only issue `HttpXmlRequests` in order to `abc.com`. A request to another domain (e.g., `google.com`) is going to be blocked by the browser. This is an important security measure that ensures malicious pieces of software cannot send information to a different server behind the scenes without the knowledge.

This restriction means that we cannot use an `HttpXmlRequest` to communicate with our desktop application. Remember that JavaScript bookmark lets are process in the context of the current

Notes

page. We need to be able to send a request through any domain (e.g. abc.com) to our desktop application which will be in the local host domain. Google must be able to do precisely this in order to gather analytics information for a site. If we are not familiar along with Google analytics, it can be accustomed to gather a multitude of information about the visitors to our web site, such as the number of visitors, where they originated from, and the pages on our site they visit. All this information is actually collected, transferred back to Google, and appears in various reports in our analytics account.

To add tracking to our site, we simply add a call to the Google analytics JavaScript at the bottom of every page of our site. Whenever a visitor lands on our page, the JavaScript runs and the visitor details are sent back to our Google analytics account. The question is when does Google do this? Surely, they can't use an XMLHttpRequest as it might break the same domain origin policy? They do not. Instead, they make use of what can only be described as a very clever technique.

7.1.3 Using the JavaScript Image Class to Make Asynchronous Cross-Domain Requests

The JavaScript Image class is a simple class that can be used in order to asynchronously load an image. To request an image, we simply set the origin property to the URL of the image. If the image loads successfully, the actual unload() method is called. If a bug occurs, the onerror() method is called. Unlike the XMLHttpRequest, there is no same domain origin policy. The source image can be located on any kind of server. It does not need to be located on the same website as the current page.

We can use this behaviour to send arbitrary requests to our desktop application (or any domain for that matter) if we realize the source URL can contain any data, including a query string. The only requirement is that it returns an image. Here is an example URL:

```
<a href= "http://localhost:60024/speaktext/dummy.gif?text=Hello">  
http://localhost:60024/speaktext/dummy.gif?text=Hello%20world</a>
```

We can easily map this URL to the following command in our application:

```
Public void speak text(string text);
```

In order to ensure the request completes without bug, a 1x1 pixel GIF image is sent back. This image is never actually displayed to the user. A tiny image is used to minimize the number of bytes being transmitted. The most essential point to realize is all communication is one way, from the internet browser to the desktop application. There is no way of sending information from the desktop application back to the internet browser. However, for many applications, this isn't a problem. Google uses the JavaScript Image technique to send visitor information to our pages (hosted on our domain.com) back to our Google analytics account (hosted on google.com).

Maximum URL Length

We need to be aware that URLs have a maximum length that varies from browser to browser (around 2K - check). This restricts the amount of information we may send in a single request. In the event that we need to send a large amount of data, we will need to split it up into smaller chunks as well as send multiple requests. The sample application, Browser Speak, uses this technique to talk arbitrarily large blocks of text.

Text Encoding

JavaScript will automatically encode the Web address we pass to Image.src as UTF-8. However, whenever passing arbitrary text as part of the Link, we will need to get away the " and '='

characters. These characters are utilized to delimit the name/value pairs (or arguments) that are passed in the query string portion of the Web address. This can be done using the JavaScript `escape()` function.

Avoiding the Cache

Web browsers will cache images (as well as many other resources) locally to prevent making multiple requests back to the actual server for the same resource. This particular behaviour is disastrous for our application. The first command will make this through to our desktop application, and the browser will cache `dummy.gif` locally. Following requests will never reach the desktop application as they can be fulfilled from the local cache.

There are a couple of alternatives to this issue. One solution is to set the cache expiry directives in the HTTP response to instruct the browser never to cache the result.



Did u know? The Uniform Resource Locator was created in 1994 by Tim Berners-Lee and the URI working group of the Internet Engineering Task Force (IETF) as an outcome of collaboration started at the IETF Living Documents “Birds of a Feather” session in 1992.

7.1.4 Browser Speak: A Concrete Example

It has some real world use. Create a sample web application that speaks in a web browser, using C#. Application will speak the textual content on a web page aloud. It can be used if we are tired of reading large passages of text on screen. It uses the System Speech Synthesis component found in the .NET Framework 3.0 for that text to speech functionality. Internet browser Speak provides the following instructions, available through its web user interface.

- Speak Text
- Stop Speaking
- Pause Speaking
- Resume Speaking

It also provides a Buffer Text command available on the internet interface. This command is used to send a block of textual content from the web browser to the desktop application. It splits the written text into 1500 byte chunks so it's not limited by the optimum size of a URL. It is used by the Speak Chosen bookmark let to transfer the selected text to the BrowserSpeak software prior to speaking.

7.2 Web.Configuration

Web.Config acts as the central position for storing the info to be accessed by webpages. This information could be a connection String stored at a federal place so that it could be utilized in a data-driven page. If the connection string changes it is just a matter of changing it at one place.

In classic ASP such global info was usually stored as an application variable. In the sample we will read the info from web configuration using ASP.NET and ASP as there might be a probability of project getting ASP and ASP.NET

Web.Config

```
<?xmlversion="1.0" encoding="utf-8"?>
<configuration>
```

Notes

```
<appSettings>
<addkey="ConnectionString1"value="server=localhost;uid=sa;pwd
=;database=northwind"/>
<addkey="ConnectionString2"value="server=localhost;uid=sa;pwd
=;database=pubs"/>
</appSettings>
</configuration
```

In ASP.NET

We can just using Configuration.AppSettings(<key>)gives the Value. (Namespace:System.Configuration)

VB.NET

```
DimConConnectionasString
ConConnection=ConfigurationSettings.AppSettings("ConnectionString1")
Response.Write(ConConnection)
```

C#

```
StringconConnection;
STR Connection = Configuration Settings .App Settings[
"ConnectionString1"];
Response.Write(conConnection);
```

InASP

Weneedtoiteratethroughthenodesinweb.comfit.

VBscript

```
setxmldoc=server.CreateObject("Microsoft.XMLDOM")
setxmlappSettings=server.CreateObject("Microsoft.XMLDOM")
setxmladd=server.CreateObject("Microsoft.XMLDOM")
xmldoc.async="false"
xmldoc.load(server.MapPath("webcomfit"))
SetxmlappSettings=xmldoc.GetElementsByTagName("app
Settings").Item(0)
Setxmladd=xmlappSettings.GetElementsByTagName("add")
Foreachxinxmladd
`CheckfortheAttributeValue
Ifx.getAttribute("key")="ConnectionString1"then
Response.write(x.getAttribute("value"))
Endif
Next
```

7.2.1 ASP.NET Web.Configuration File

ASP.NET Web.config allows you to define or revise the configuration settings at the time of developing the application or at the time of deployment or even after deployment. The following are brief points that can be understood about the Web.config file:

- Web.config files are stored in XML format which makes us easier to work with.

Notes

- You can have any number of Web.config files for an application. Each Web.config applies settings to its own directory and all the child directories below it.
- All the Web.config files inherit the root Web.config file available at the following location `systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\Web.config` location.
- IIS is configured in such a way that it prevents the Web.config file access from the browser.
- The changes in Web.config don't require the reboot of the web server.

Web.config Settings

Before we start working with configuration settings of ASP.NET, we see the hierarchy of the Web.config file.

```
<configuration>
  <configSections>
    <sectionGroup>
  </sectionGroup>
  </configSections>
  <system.web>
  </system.web>
  <connectionStrings>
  </connectionStrings>
  <appSettings>
  </appSettings>
  .....
  .....
  .....
  .....
  .....
</configuration>
```

So from the above tree structure, we can understand that the configuration tag is the root element of the Web.config file under which it has all the remaining sub elements. Each element can have any number of attributes and child elements which specify the values or settings for the given particular section. To start with, we'll see the working of some of the most general configuration settings in the web.config file.

system.web

In the configuration hierarchy, the most common thing we will work with is the system.web section. Now we look at some of the child sections of the system.web section of web.config file.

Compilation Settings

If you are using Visual Studio 2010, probably the only available section of Web.config file by default is Compilation section. If you want to specify the target framework or if you need to add an assembly from the Global Assembly Cache (GAC) or if you want to enable the debugging mode of the application, you can take Compilation settings as granted for these tasks. The following code is used to achieve the discussed settings:

```
<system.web
  <compilation
    debug="true" strict="true" explicit="true" batch="true"
```


Notes

```
optimizeCompilations="true" batchTimeout="900"
maxBatchSize="1000" maxBatchGeneratedFileSize="1000"
numRecompilesBeforeAppRestart="15" defaultLanguage="c#"
targetFramework="4.0" assemblyPostProcessorType="">
    <assemblies>
        <add assembly="System, Version=1.0.5000.0, Culture=neutral,
        PublicKeyToken=b77a5c561934e089" />
    </assemblies>
</compilation>
</system.web>
```

Under the assemblies element, you are supposed to mention the type, version, culture and public key token of the assembly. In order to get the public key token of an assembly, you need to follow the below mentioned steps:

- Go to Visual Studio tools in the start menu and open the Visual Studio command prompt.
- In the Visual Studio command prompt, change the directory to the location where the assembly or .dll file exists.
- Use the following command, `sn -T itextsharp.dll`.
- It generates the public key token of the assembly. You should keep one thing in mind that only public key token is generated only for the assemblies which are strongly signed.



Example:

```
C:\WINNT\Microsoft.NET\Framework\v3.5> sn -T itextsharp.dll
Microsoft (R) .NET Framework Strong Name Utility Version 3.5.21022.8
Copyright (c) Microsoft Corporation. All rights reserved.
Public key token is badfaf3274934e0
```

Explicit and sample attributes are applicable only to VB.NET and C# compiler however ignores these settings.

Page Settings

Ok, by this time, we have got familiar with the Web.config file and we have seen the settings of Compilation Sections, now we will see the settings of a page. As an ASP.NET application consists of several number of pages, we can set the general settings of a page like sessionstate, viewstate, buffer, etc., as shown below:

```
<pages buffer ="true" styleSheetTheme="" theme ="Acqua"
    masterPageFile ="MasterPage.master"
    enableEventValidation="true">
```

By using the MasterPageFile and theme attributes, we can specify the master page and theme for the pages in web application.

Custom Error Settings

The next section of Web.config file, we are going to look around is Custom Error settings, by the name itself it is clear that we can configure the settings for the application level errors in these section. Now we will see the description of the customErrors section of the Web.config from the below mentioned code snippet.

```
<customErrors defaultRedirect ="Error.aspx" mode ="Off">
```

```
<error statusCode ="401" redirect ="Unauthorized.aspx"/>
</customErrors>
```

The customErrors section consists of defaultRedirect and mode attributes which specify the default redirect page and the on/off mode respectively.

The subsection of customErrors section allows redirecting to specified page depending on the error status code.

- 400 Bad Request
- 401 Unauthorized
- 404 Not Found
- 408 Request Timeout

Location Settings

If you are working with a major project, probably you might have numerous numbers of folders and sub-folders, at this kind of particular situation, you can have two options to work with. First thing is to have a Web.config file for each and every folder(s) and Sub-folder(s) and the second one is to have a single Web.config for your entire application. If you use the first approach, then you might be in a smoother way, but what if you have a single Web.config and you need to configure the sub-folder or other folder of your application, the right solution is to use the "Location" tag of "system.web" section of Web.config file. However you can use this tag in either of the discussed methods.

The following code shows you to work with Location settings:

```
<location path="Login.aspx">
    <system.web>
        <authorization>
            <allow users="*" />
        </authorization>
    </system.web>
</location>
Collapse | Copy Code
<location path ="Uploads">
    <system.web>
        <compilation debug = "false">
    </system.web>
</location>
```

In a similar way, you can configure any kind of available settings for any file/folder using the location tag.

Session State and View State Settings

As we all know, the ASP.NET is stateless and to maintain the state we need to use the available state management techniques of ASP.NET. View state and session state are among them. For complete information about view state and Session State and how to work with, there are some excellent articles in CodeProject, which you can refer here:

- [Beginners Introduction to State Management Techniques in ASP.NET](#)
- [Exploring Session in ASP.NET](#)

Notes

Now we'll see the Web.config settings of View State and Session State:

View State can be enabled or disabled by using the following page settings in the web.config file.

```
<Pages EnableViewState="false" />
Session state settings for different modes are as shown below:
<sessionState mode="InProc" />
<sessionState mode="StateServer"
stateConnectionString= "tcpip=Yourservername:42424" />
<sessionState mode="SQLServer" sqlConnectionString="cnn" />
```

HttpHandler Settings

HttpHandler is a code that executes when an http request for a specific resource is made to the server. For example, request an .aspx page the ASP.NET page handler is executed, similarly if an .asmx file is requested, the ASP.NET service handler is executed. An HTTP Handler is a component that handles the ASP.NET requests at a lower level than ASP.NET is capable of handling.

You can create your own custom http handler, register it with IIS and receive notice whenever a request is made. For doing this, you just need to create a class which implements IHttpHandler and then you need to add the following section of configuration settings in the web.config file. For this demonstration, I have created a sample imagehandler class which displays a JPG image to the browser. You can go through the imagehandler class code in the sample download code.

```
<httpHandlers>
  <add verb="*" path="*.jpg" type="ImageHandler"/>
  <add verb="*" path="*.gif" type="ImageHandler"/>
</httpHandlers/>
```

HttpModule Settings

HttpModule is a class or an assembly that implements the IHttpModule interface that handles the application events or user events. You can too create your own custom HttpModule by implementing the interface and configure it with ISS. The following settings show the HttpModules configuration in the web.config.

```
<httpModules>
  <add type ="TwCustomHandler.ImageHandler"
name ="TwCustomHandler"/>
  <remove name ="TwCustomHandler"/>
  <clear />
</httpModules>
```

Authentication, Authorization, Membership Provider, Role Provider and Profile Provider Settings

These settings are directly available in the web.config file if you have created the ASP.NET application by using the Visual Studio 2010. I'm not going to elaborate them as there are lot of articles in CodeProject describing the functionality and use of these settings and for further information you can refer to them. Some of the links are here:

- [ASP.NET Membership and Role Provider](#)
- [Developing custom ASP.NET Membership and Role providers reading users from custom section in the web.config](#)

- ASP.NET Membership
- Membership and Role providers for MySQL
- Custom Membership, Role Providers, Website administration tool, and Role based access to individual files

Notes

Authentication Settings

```
<authentication mode="Forms">
  <forms cookieless="UseCookies" defaultUrl="HomePage.aspx"
  loginUrl="Unauthorized.aspx" protection="All" timeout="30">
  </forms>
</authentication>
```

Authorization Settings

```
<authorization
  <allow roles ="Admin"/>
  <deny users ="*/>
</authorization>
```

Membership Provider Settings

```
<membership defaultProvider="Demo_MemberShipProvider">
  <providers>
    <add name="Demo_MemberShipProvider"
    type="System.Web.Security.SqlMembershipProvider"
    connectionStringName="cnn"
    enablePasswordRetrieval="false"
    enablePasswordReset="true"
    requiresQuestionAndAnswer="true"
    applicationName="/"
    requiresUniqueEmail="false"
    passwordFormat="Hashed"
    maxInvalidPasswordAttempts="5"
    minRequiredPasswordLength="5"
    minRequiredNonalphanumericCharacters="0"
    passwordAttemptWindow="10" passwordStrengthRegularExpression="">
  </providers>
</membership>
```

Role Provider Settings

```
<roleManager enabled="true" cacheRolesInCookie="true"
cookieName="TBHROLES" defaultProvider="Demo_RoleProvider">
  <providers>
    <add connectionStringName="dld_connectionstring"
    applicationName="/" name="Demo_RoleProvider"
    type="System.Web.Security.SqlRoleProvider, System.Web,
    Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
  </providers>
</roleManager>
```

Notes

Profile Provider Settings

```
<profile defaultProvider="Demo_ProfileProvider">
  <providers>
    <add name="Demo_ProfileProvider" connectionStringName="cnn"
      applicationName="/" type="System.Web.Profile.SqlProfileProvider,
      System.Web, Version=2.0.0.0, Culture=neutral,
      PublicKeyToken=b03f5f7f11d50a3a"/>
  </providers>
  <properties>
    <add name="Name" type="String"/>
    <add name="DateofBirth" type="DateTime"/>
    <add name="Place" type="string"/>
  </properties>
</profile>
```

AppSettings

In the above section, we have seen the settings available in system.web tag, now we will see the available settings in appSettings section.

appSettings element helps us to store the application settings information like connection strings, file paths, URLs, port numbers, custom key value pairs, etc.

The following code snippet shows the example of appSettings Section:

```
<appSettings>
  <add key="AppKey" value="APLJI12345AFAF89999BDFG"/>
</appSettings>
```

connectionStrings

The most common section of web.config file the connectionStrings sections allows you to store multiple connection strings that are used in the application. The connectionStrings tag consists of child element with attributes name and connectionString which is used to identify the connectionstring and the other is used to connect to the database server respectively.

The general connectionstring settings are shown below:

```
<connectionStrings>
  <add name ="cnn" connectionString ="Initial Catalog = master;
    Data Source =localhost; Integrated Security = true"/>
</connectionStrings>
```

ConfigSections

ConfigSections helps you to create your own custom configuration section that can be used with the web.config file. We look at this in the later section of the article, for the time being, we can have look at the configsection settings. ConfigSections should be declared just below the configuration (parent element) otherwise it is going through you an error.

```
<configSections>
  <sectionGroup name="pageAppearanceGroup">
    <section
      name="pageAppearance"
      type="PageAppearanceSection"
```

```

allowLocation="true"
allowDefinition="Everywhere"
/>
</sectionGroup>
</configSections>

```

Notes



Task Add a Logging Database Connection String to the Web.config File.

Self Assessment

Multiple Choice Questions:

- How many web.config files can be there in an ASP.NET application?
 - One
 - Atleast one
 - More than One
 - None
- Is it possible for multiple aspx pages to use the same code-behind file, if required?
 - Yes
 - No
- Which statement about the Web.Config file is the most accurate?
 - The Web.Config file can be only placed in the root of a Web Site to override settings in the Machine.Config file for all applications in a particular Web Site
 - The Web.Config file can only be placed in the root of a particular virtual directory
 - The Web.Config file can be placed in the root of the Web Site and the root of a virtual directory. The settings from the file in the virtual directory overrides the settings from the file in the Web Site
 - The Web.Config file can be placed in the root of the Web Site and the root of a virtual directory and in any subdirectory of an application. The settings from a file at a particular level override the settings from the ones above it.
- The settings in the Web.Config file are case-sensitive.
 - True
 - False
- Can we store multiple connection strings in Web Config file?
 - Yes
 - No

7.3 Characteristics of ASP.NET

ASP.NET Web pages, known officially as Web Forms, are the main building block for application development. Web forms are contained in files with a “.aspx” extension; these files typically contain static (X)HTML markup, as well as markup defining server-side Web Controls and User Controls where the developers place all the content for the Web page. Additionally, dynamic code which runs on the server can be placed in a page within a block `<% — dynamic code — %>`, which is similar to other Web development technologies such as PHP, JSP, and ASP. With ASP.NET Framework 2.0, Microsoft introduced a new code-behind model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs or .aspx.fs file (depending on the programming language used).

Notes**Directives**

A directive is special instructions on how ASP.NET should process the page. The most common directive is `<%@ Page %>` which can specify many attributes used by the ASP.NET page parser and compiler.



Example:

```
Inline code
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 //EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
// Assign the datetime to label control
lb11.Text = DateTime.Now.ToLongTimeString();
}
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>Sample page</title>
</head>
<body>
<form id="form13" runat="server">
The current time is: <asp:Label runat="server" id="lb11" />
</form>
</body>
</html>
```

The above page renders with the Text "The current time is: " and the current time.

Code-behind solutions

```
<%@ Page Language="C#" CodeFile="SampleCodeBehind.aspx.cs"
Inherits="Website.SampleCodeBehind" AutoEventWireup="true" %>
```

The above tag is placed at the beginning of the ASPX file. The CodeFile property of the @ Page directive specifies the file (.cs or .vb or .fs) acting as the code-behind while the Inherits property specifies the Class from which the Page is derived. In this example, the @ Page directive is included in SampleCodeBehind.aspx, then SampleCodeBehind.aspx.cs acts as the code-behind for this page:

```
Source language C#:
using System;
namespace Website
{
public partial class SampleCodeBehind : System.Web.UI.Page
{
protected void Page_Load(object sender, EventArgs e)
{
Response.Write("Hello, world");
}
}
}
```

```

}
Source language Visual Basic.NET:
Imports System
Namespace Website
Public Partial Class SampleCodeBehind
Inherits System.Web.UI.Page
Protected Sub Page_Load(ByVal sender As Object, ByVal e As EventArgs)
Response.Write("Hello, world")
End Sub
End Class
End Namespace

```

In this case, the Page_Load() method is called every time the ASPX page is requested. The programmer can implement event handlers at several stages of the page execution process to perform processing.

User controls

User controls are encapsulations of sections of pages which are registered and used as controls in ASP.NET.etc.

Custom controls

Programmers can also build custom controls for ASP.NET applications. Unlike user controls, these controls do not have an ASCX markup file, having all their code compiled into a dynamic link library (DLL) file. Such custom controls can be used across multiple Web applications and Visual Studio projects.

Rendering technique

ASP.NET uses a visited composites rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template. Literal text goes into instances of the Literal control class, and server controls are represented by instances of a specific control class. The initialization code is combined with user-written code (usually by the assembly of multiple partial classes) and results in a class specific for the page.

State management

ASP.NET applications are hosted by a Web server and are accessed using the stateless HTTP protocol. As such, if an application uses stateful interaction, it has to implement state management on its own. ASP.NET provides various functions for state management. Conceptually, Microsoft treats "state" as GUI state. Problems may arise if an application needs to keep track of "data state"; for example, a finite-state machine which may be in a transient state between requests (lazy evaluation) or which takes a long time to initialize. State management in ASP.NET pages with authentication can make Web scraping difficult or impossible.

There are three ways of providing state management:

1. Application
2. Session
3. View

Notes




Task Write a Transform to Change the Environment and Logging Connection Strings in the Staging Web.config.

Although ASP.NET applications are automatically updated to use the installing version of ASP.NET if the preceding conditions are met, custom configuration settings in the current Machine.config file are not transferred to the installing Machine.config file. If your application uses custom configuration settings, be sure to either manually update the new Machine.config file or use the ASP.NET IIS Registration tool (Aspnet_regiis.exe) to remap the application to the previous version of ASP.NET.

7.4 New Features in ASP.NET 3.5

There are four new features worth noting in ASP.NET 3.5:

- **Integrated ASP.NET AJAX support:** With ASP.NET 3.5, however, the AJAX-related classes are built directly into the .NET Framework, making it easier to get started building rich, AJAX-enabled Web applications with ASP.NET. (For more information on using the ASP.NET AJAX framework
- **The ListView control:** The ListView control is an update to the old DataList and Repeater controls, displaying multiple records and providing functionality like the GridView, but allowing for a more flexible layout through the use of templates.
- **The DataPager control:** The DataPager control operates as a sort of free-standing paging interface. In short, it renders a paging user interface - next, previous, first, last buttons, for instance - and is tied to a data Web control. The DataPager only works with those controls that implement the IPagableItemContainer interface, which (currently) includes only the ListView control.
- **LINQ:** LINQ (Language Integrated Query) adds ancient data querying capability to C# and VB.NET along with the compiler and Intelligence assistance. LINQ is a component of .NET 3.5. LINQ defines operators that permit us to script the query in a consistent manner over data stores, objects and XML. The ASP.NET LinqDataSource control allows us to use LINQ to filter, order and group information before binding to the List controls.



Notes ASP.NET 3.5 introduce a new merge tool (aspnet_merge.exe). This tool lets us combine and manage assemblies created by aspnet_compiler.exe.



Case Study **Egyptian Ministry Facilitates Transparent, Open Elections with New Applications**

The Egyptian Ministry of State for Administrative Development (MSAD) was given the difficult task of creating the software to support transparent Egyptian elections in 2012. MSAD had a tight schedule and limited budget to develop highly scalable applications to support the 50 million eligible voters set to participate in the landmark election process that was watched by the whole world. MSAD chose Microsoft Visual

Contd....

Notes

Studio 2010 and Team Foundation Server 2010 to ensure the application was ready for the elections and able to handle the millions of voters using it at one time. MSAD is testing Visual Studio 2012 and sees potential to improve developer productivity even more.

Business Needs

The Arab Spring swept across Northern Africa and the Middle East in 2011, bringing about dramatic changes in governments. In Egypt the government leaders stepped down and were replaced with a temporary military tribunal that restored order and then worked to hold an open election.

Originally scheduled for late 2012, the elections were pulled forward to November 2011 for parliamentary elections and May 2012 for the presidential elections. The Ministry of State for Administrative Development (MSAD) was tasked with developing the software systems, called the National System of Egyptian Elections of 2011/2012, required for the elections.

MSAD was under national and international pressure to facilitate a system that would support transparent and fair elections.

With only three developers, a limited budget, and about one month to develop the application to scale to 50 million eligible electors, MSAD knew the choice of a development platform would be critical to the success of the project. MSAD chose to use .NET as the development framework because its developers had .NET experience and the IT team at MSAD believed it could develop the application rapidly with the Microsoft development framework. "We chose the Microsoft platform because it fulfilled all of our requirements and was easy to use," says SherifFahmi, Technical Consultant, MSAD. Microsoft .NET also provided a large ecosystem of .NET developers that MSAD could leverage if required.

Questions:

1. IS .NetTechnology suitable for limited budget projects?
2. Explain the needs of MSAD project?

Self Assessment

Multiple Choice Questions:

6. You require to create an ASP.NET page with the functionality to allow a user to upload a file to the server
 - a. You need to use the System.Web.Upload namespace
 - b. You need to use a COM component to save the file on the server
 - c. You need to use the SaveAs method of the HttpPostedFile class.
 - d. The ASP.Net application automatically loops through all the <input type="File"> and saves the uploaded files to a virtual folder called "uploads"
7. Your site has been restructured and the paths of few pages have changed. Which method would you use to redirect users requesting for a pages using the old URL
 - a. Create an ISAPI filter to do the above task
 - b. Create an entry in the <badlinks> section of the Web.Config file
 - c. Use the Application.Config file.
 - d. Handle the Application_BeginRequest event and use the RewritePath() method

Notes

8. The settings in the Web.Config file can be configured to apply to:
 - a. An application
 - b. An application or a particular directory
 - c. An application or a particular directory or even an individual file.
 - d. The Web.Config file always applies to all pages in the current directory and its subdirectories
9. Is there a way to prevent configuration settings in a Web.Config file from being overridden by a Web.Configfile located below it
 - a. Yes, you can use the allowOverride attribute in the <location> tag to prevent configuration settings in a Web.Config file from being overridden by a Web.Config file located below it
 - b. Yes, you can use the <PreventOverride> tag to prevent configuration settings in a Web.Config file from being overridden by a Web.Config file located below it
 - c. If a Web.Config file exist at a particular level then all settings in the Web.Configfile located above it are always ignored
 - d. You cannot have more than one Web.Config file in an ASP.NET application
10. You are part of a team that is using Visual Studio .NET to develop a Web application. You have placed a number of configuration settings for the application in the server's Machine.configfile. You learn in a meeting that one of your co-workers has created a Web.config file to store these settings, though in comparing notes you realize that you and your co-worker have used different settings.

Which of the following describes what will happen when you execute the application

 - a. The application will run correctly and will use the settings in the Web.config file, ignoring the Machine.config settings
 - b. The application will run correctly and will use the settings in the Machine.config file, ignoring the Web.config settings
 - c. The application will not run correctly until you either remove the Web.config file or delete the settings from the Machine.config file
 - d. The application will run correctly, using only the settings that are identical in the Web.config and Machine.config files
 - e. The application will not run correctly until you modify either the Web.config file or the Machine.config file so that the application settings are identical
11. helps you to create your own custom configuration section that can be used with the web.config file.
 - a. Config Sections
 - b. Connection String
 - c. App setting
 - d. Member settings
12. App settings is used for.....
 - a. database connection
 - b. coding
 - c. testing
 - d. None of these.

13. JavaScript code using HTTPRequests. Notes
- a. True b. False
14. We can manage states in ASP.NET application using
- a. Session Objects b. Application Objects
- c. Viewstate d. All of the above
15. Which of the following is not the way to maintain state?
- a. View state b. Cookies
- c. Hidden fields d. Request object

7.5 Summary

- ASP.NET is a set of Web development tools offered by Microsoft. Programs like Visual Studio.NET and Visual Web Developer allow Web developers to create dynamic websites using a visual interface.
- ASP.NET is more than the next version of Active Server Pages (ASP); it provides a unified Web development model that includes the services necessary for developers to build enterprise-class Web applications.
- A bookmarklet is special URL that will run a JavaScript application when clicked. The JavaScript will execute in the context of the current page.
- Bookmarklets allow us to execute an arbitrary block of JavaScript code just at the click.
- The .NET framework 2.0 has an `HttpListener` class and associated `HttpListenerRequest` and `HttpListenerResponse` classes that allow us to implement the JavaScript code.
- To add tracking to our site, we simply add a call to the Google analytics JavaScript at the bottom of every page of our site.
- `Web.Config` acts as the central position for storing the info to be accessed by webpages. This information could be a connection String stored at a federal place so that it could be utilized in a data-driven page.
- The `Web.config` it is an XML file, it can consist of any valid XML tags, but the root element should always be `<configuration>`.
- ASP.NET is stateless and to maintain the state we need to use the available state management techniques of ASP.NET. View state and session state are among them.
- The connection Strings tag of Web config file consists of child element with attributes name and connection string which is used to identify the connection string and the other is used to connect to the database server respectively.
- A directive is special instructions on how ASP.NET should process the page. The most common directive is `<%@ Page %>` which can specify many attributes used by the ASP.NET page parser and compiler.
- ASP.NET uses a visited composites rendering technique. During compilation, the template (.aspx) file is compiled into initialization code which builds a control tree (the composite) representing the original template.
- There are three ways of providing state management in Asp. Net-Application, Session and View State.

Notes

- .NET provides a visual interface for developers to create their applications, which makes .NET are as on able choice for designing Web-based interfaces as well.

7.6 Keywords


Authentication: It is the act of confirming the truth of an attribute of a datum or entity.

Authorization: It is the function of specifying access rights to resources, which is related to information security and computer security in general and to access control in particular.

Bookmarklet: A bookmarklet is unobtrusive JavaScript stored as the URL of a bookmark in a web browser or as a hyperlink on a web page.

Globalization: It is the process of international integration, required due to the increasing connectivity and interdependence of the world’s markets and businesses.

Web.config: It is the main settings and configuration file for an ASP.NET web application. The file is an XML document that defines configuration information regarding the web application.



Lab Exercise

1. Add Code to Populate Text Value of EnvName from a Value in the Web.config
2. Transforming a Web.Config File for Deployment.

7.7 Review Questions

1. Can I lock a configuration setting so that a Web.config file that appears lower in the hierarchy cannot override it?
2. How are ASP.NET configuration files secured against unauthorized access?
3. How many web.config files are there in 1 project?
4. How you can add an event handler?
5. Can we have a web application running without web.Config file?
6. What is the difference between web.config and machine.config?
7. How do you register JavaScript for webcontrols?
8. What is the appSettings Section in the *web.config* file?
9. What are HTTP handlers in ASP.NET?
10. What are the major built-in objects in ASP.NET?

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (b) | 2. (a) | 3. (d) | 4. (a) |
| 5. (a) | 6. (c) | 7. (d) | 8. (c) |
| 9. (a) | 10. (a) | 11. (a) | 12. (a) |
| 13. (a) | 14. (d) | 15. (d) | |

7.8 Further Readings

Notes



Books

ASP.Net 3.5 Website Programming: Problem Design Solution, by Chris Love



Online links

<http://books.google.co.in/books?id=av—5iDeXo4C&printsec=frontcover&dq=Advanced+ASP.NET+3.5&hl=en&sa=X&ei=5bb-T6DhJYXRrQeK6b3cBg&ved=0CFoQ6AEwBQ#v=onepage&q=Advanced%20ASP.NET%203.5&f=false>

Unit 8: Creating More Advanced ASP.NET

CONTENTS

Objectives

Introduction

8.1 Page Sub-classing

8.2 User Control

8.2.1 Overview

8.2.2 How to Convert Web Form Pages into ASP.NET User Control

8.2.3 How to Include a User Control in an ASP.NET Web Page

8.2.4 How to Create Instances of ASP.NET User Controls Programmatically

8.2.5 How to Create Templated ASP.NET User Controls

8.2.6 How to Create ASP.NET User Controls

8.2.7 How to Include ASP.NET User Controls in Web Pages

8.2.8 Creating Reusable Elements with ASP.NET User Controls

8.3 Data Binding

8.3.1 RadioButtonList

8.3.2 CheckBoxList

8.3.3 Dropdown List

8.3.4 Listbox

8.4 Understanding Two-Way Data Binding

8.4.1 Sample Two-Way Data Binding Application

8.5 Data-Bound Controls

8.5.1 An Item

8.5.2 ASP.NET Data-Bound Controls

8.6 Summary

8.7 Keywords

8.8 Review Questions

8.9 Further Readings

Objectives

After studying this unit, you will be able to:

- Define page sub-classing
- Explain user control
- Describe data binding
- Understand two-way data binding

Introduction

Notes

ASP.NET is above the actual advance version of Active Server Pages (ASP); it provides the combined Web development model which includes the services required for designers to build enterprise- class Web application. While ASP.NET is mainly syntax like-minded with ASP, it also provides a brand new programming replica and transportation for additional scalable and stable applications that help provide greater protection. You may feel free to augment your existing ASP applications by incrementally adding ASP.NET features to them. ASP.NET is a compiled, .NET-based environment; you can create applications in any .NET compatible programming language, including Visual Basic .NET, C#, and JScript .NET. Additionally, the entire .NET Platform is offered to any ASP.NET application. Developers can easily access the benefits of these types of technologies, which include the managed common language runtime environment, type safety, inheritance, and so on. ASP.NET has been made to work seamlessly with WYSIWYG HTML editors and other programming resources, including Microsoft Visual Studio .NET. Not only does this help to make Web designing easier, but it also provides all the benefits that these tools have to offer, including a GUI that programmers can use to drop server controls onto a Web page and fully integrated debugging support. Programmers may use Web Forms or XML Web services when developing an ASP.NET application, or combine these in any way they think fit. Each is backed by exactly the same infrastructure that permits you to use authentication schemes; cache frequently used data, or customizes your software's configuration, to name only a few possibilities.

- Web Forms allow you to build authoritative forms-based Web pages. When design these pages, you should use ASP.NET wine waiter controls to create common UI elements, and program them for common tasks. These controls allow you to definitely rapidly build a Web Form out of reusable built-in or custom components, simplifying the actual code of a page.
- A good XML Web service provides the means to access server features remotely. Using XML Web services, organizations can expose programmatic interfaces to their data or company logic, which in turn can be obtained and altered by client as well as server applications. XML Web services enable the exchange of information in client-server or server-server scenarios, using standards such as HTTP and XML messaging to maneuver data across firewalls. XML Internet services are not tied to a particular component technology or object-calling convention. As a result, applications written in any language, using any component model, and running on any operating system can access XML Web services.
- All these models can take full benefit of all ASP.NET features, as well as the power of the .NET Framework and .NET Framework CLR. These features and how you can use them are laid out as follows:

If you have ASP programming skills, the new ASP.NET programming model will seem really familiar to you. However, the ASP.NET object model has changed significantly from ASP, which makes it more structured and object-oriented. However this means that ASP.NET is not fully backwards compatible; almost all existing ASP pages will have to be modified to some extent in order to run under ASP.NET. In addition, major changes to Visual Basic .NET means that existing ASP pages coded with Visual Basic Scripting Edition generally will not really port directly to ASP.NET. In most cases, though, the required changes will involve only a few lines of code.

Accessing databases from ASP.NET applications is definitely an often-used technique for exhibiting data to Web site visitors. ASP.NET makes it easier than ever to access databases for this purpose. It also allows you to definitely manage the database from your code.

Notes

ASP.NET provides an easy model that permits Web developers to write logic that operates at the application level. Developers can write this code within the Global.asax text file or perhaps in a compiled class deployed as an assembly. This logic can consist of application-level events, but designers can easily extend this model to suit the needs of the Web application.

ASP.NET provides easy-to-use application and session-state facilities that are familiar to ASP developers and are readily compatible with all additional .NET Framework APIs.

- For advanced developers who require to use APIs as powerful as the actual ISAPI programming interfaces that were included with previous versions associated with ASP, ASP.NET offers the IHttpHandler and IHttpModule interfaces. Applying the IHttpHandler interface gives a means of interacting with the lower-level request and response providers of the IIS Web server and provides functionality much such as ISAPI extensions, but with a simpler programming model. Applying the IHttpModule interface allows you to include custom events that participate in most request made to your software.
- ASP.NET takes advantage associated with performance enhancements found in the actual .NET Framework and CLR. Additionally, it has already been designed to offer significant overall performance improvements over ASP and other Web development platforms. All ASP.NET code is compiled, rather than interpreted, which enables early binding, strong typing, and just-in-time (JIT) compilation to indigenously code, to name only a few of its advantages. ASP.NET is also easily factorable, meaning that designers can remove modules (a program module, for instance) that are not relevant to the application they are creating. ASP.NET also offers extensive caching services (both built-in services and caching APIs). ASP.NET also ships with performance counters that designers and system administrators can monitor to test new applications as well as gather metrics on existing programs.
- Writing custom debug statements to your Web page can help immensely in troubleshooting your application's code. However, they can trigger embarrassment if they are not really taken out. The difficulty is that removing the debug statements from your pages when your application is able to be ported to a production server can require significant effort. ASP.NET offers the TraceContext class, which allows you to definitely write custom debug statements to your pages as you develop them. They appear only when you have enabled tracing for a web page or entire application. Enabling tracing also appends details about a request to the page, or even, if you so specify, to some custom trace viewer that is actually stored in the root listing of your application.
- The .NET Framework and ASP.NET provide default authorization and authentication schemes for Web applications. You can certainly eliminate, add to, or replace these schemes, depending upon the requirements of your application.
- ASP.NET configuration settings are stored in XML-based files, which are human decipherable and writable. Each of your applications can have a distinct configuration file and you can extend the configuration scheme to suit your requirements.
- Applications are said to become running side by side when they are installed on the same pc but use different versions of the .NET Framework. To learn using different versions of ASP.NET for distinct applications on your own server, see Side-by-Side Support in ASP.NET.

8.1 Page Sub-classing

A subclass, their class, or even child class is a modular, derivative class that inherits one or additional properties from a different class. The properties in query vary from programming language to language, other than commonly include class data variables, properties, and

techniques or functions. Some languages offer the inheritance of other properties as well. For example, in Eiffel, contracts which define the specification of the class are also inherited by heirs. The superclass establishes a common interface and foundational functionality, that specialized subclasses can inherit, alter, and supplement. The software passed down by a subclass is regarded as reused in the subclass.

In some instances, a subclass may customize or redefine a method inherited in the superclass. A superclass method which may be changed in this way is called a virtual method.

This is the Page base class that I use for all the pages that contain the < form > element. This base class outline the process of collecting data, validate data, display error message, and save the data. Each step is implemented by a virtual method and developer can override these methods when it is needed. I have used this base class in many application and it has proven very helpful for me.

```
using System;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Collections;
using System.Collections.Specialized;
using System.Security.Principal;

namespace TuNguyen.WebLibrary
{

    public abstract class PageForm : Page {
        public const string STATUS_DEFAULT = "PostInitialDefault";
        public const string STATUS_POSTBACK = "Postback";
        public const string STATUS_ERROR = "PostbackError";
        public const string STATUS_POSTVIEW = "PostInitialView";
        public const string STATUS_POSTCOMPLETE = "PostbackComplete";
        protected Hashtable formError;
        protected Hashtable formData;
        protected bool bypassInternalProcess = false;
        protected override void OnInit(EventArgs e) {
            base.OnInit(e);
            formError = new Hashtable();
            formData = new Hashtable();
            OnAuthenticateRequest(e);
            OnSetFormStamp(e);
            OnInitializeState(e);
        }

        protected override void OnLoad(EventArgs e) {
            base.OnLoad(e);
            SetNoPageCache(e);
            if (IsPostBack) {
                OnVerifyFormStamp(e);
                GetFormData();
                SetFormStatus(PageForm.STATUS_POSTCOMPLETE);
                if (!bypassInternalProcess) OnPostBackRequest(e);
            }
        }
    }
}
```

Notes

```
else {
    SetFormStatus(PageForm.STATUS_DEFAULT);
    SetData("form_pass", "0");
    if (!bypassInternalProcess) OnInitialRequest(e);
}
if (!bypassInternalProcess) {
    OnBeginProcessForm(e);
    OnProcessForm(e);
    OnEndProcessForm(e);
}
}

protected virtual void OnInitializeState(EventArgs e) {}
protected virtual void OnSetFormStamp(EventArgs e) {}
protected virtual void OnInitialRequest(EventArgs e) {}
protected virtual void OnPostBackRequest(EventArgs e) {
    IncrementCount();
    OnValidateData(e);
}
protected virtual void OnVerifyFormStamp(EventArgs e) {}
protected virtual void SetNoPageCache(EventArgs e) {
    Response.Expires = 0;
    Response.AddHeader("Cache-Control", "no-cache");
    Response.AddHeader("Pragma", "no-cache");
}
protected void SetBypassInternalProcess() {
    bypassInternalProcess = true;
}
protected virtual void OnProcessForm(EventArgs e) {
    switch (GetFormStatus()) {
    case PageForm.STATUS_DEFAULT:
        OnFormStatusDefault(e);
        break;
    case PageForm.STATUS_POSTVIEW:
        OnFormStatusShowData(e);
        break;
    case PageForm.STATUS_ERROR:
        OnFormStatusValidationError(e);
        break;
    case PageForm.STATUS_POSTCOMPLETE:
        OnFormStatusProcessComplete(e);
        break;
    default:
        OnFormStatusOther(e);
        break;
    }
}
protected virtual void OnValidateData(EventArgs e) {}
protected virtual void OnBeginProcessForm(EventArgs e) {}
```

Notes

```

protected virtual void OnFormStatusDefault(EventArgs e) {}
protected virtual void OnFormStatusShowData(EventArgs e) {}
protected virtual void OnFormStatusValidationError(EventArgs e) {}
protected virtual void OnFormStatusProcessComplete(EventArgs e) {}
protected virtual void OnFormStatusOther(EventArgs e) {}
protected virtual void OnEndProcessForm(EventArgs e) {}
protected virtual void OnAuthenticateRequest(EventArgs e) {}
void GetFormData() {
    NameValueCollection form = Context.Request.Form;
    foreach (string fieldName in form.AllKeys)
        SetData(fieldName, form[fieldName]);
}
protected void SetError(string fieldName, string value) {
    if (IsErrorField(fieldName)) formError[fieldName] = value;
    else formError.Add(fieldName, value);
}
protected virtual Literal AddFormTrackingStatus()
{
    Literal formStatus = new Literal();
    formStatus.Text = "\n<input name=\"form_pass\" type=\"hidden\" value=\""
+ GetData("form_pass") + "\" />\n" +
    "<input name=\"form_status\" type=\"hidden\" value=\"" + GetFormStatus()
+ "\" />\n";
    return formStatus;
}
protected string GetError(string fieldName) {
    string value = "";
    if (IsErrorField(fieldName)) value = (string) formError[fieldName];
    return value;
}
protected bool IsErrorField(string fieldName) {
    return formError.ContainsKey(fieldName);
}
protected bool IsError {
    get {return (formError.Count > 0);}
}
protected int ErrorCount {
    get {return formError.Count;}
}
protected string GetData(string fieldName) {
    string value = "";
    if (IsDataField(fieldName)) value = (string) formData[fieldName];
    return value;
}
protected void SetData(string fieldName, string value) {
    if (IsDataField(fieldName)) formData[fieldName] = value;
    else formData.Add(fieldName, value);
}
protected bool IsDataField(string fieldName) {
    return formData.ContainsKey(fieldName);
}

```

Notes

```
}
protected bool IsStatusComplete {
get {
string status = GetData("form_status");
return (String.Compare(status, PageForm.STATUS_POSTCOMPLETE, true) == 0);
}
}
protected void SetFormStatus(string value) {
SetData("form_status", value);
}
protected string GetFormStatus() {
return GetData("form_status");
}
protected void IncrementCount() {
if (IsDataField("form_pass")) {
int count = Convert.ToInt32(GetData("form_pass"));
++count;
SetData("form_pass", count.ToString());
}
}
protected string GetDataText() {
string text = "";
foreach (string key in formData.Keys)
text += "<div>" + GetData(key) + "</div>";
return text;
}
protected string GetErrorText() {
string text = "";
foreach (string key in formError.Keys)
text += "<div>" + GetError(key) + "</div>";
return text;
}
public override void Dispose() {
formError.Clear();
formData.Clear();
formError = null;
formData = null;
base.Dispose();
}
}
```

For managing database connection, I have an abstract class that provide many convenient API. Typically, the business object would utilize this abstract class. I also have use this class in many application and it has save me a lot of time reinventing the wheel. To use this abstract class, you have to override the abstract method `GetConnectionString()`;

```
using System;
using System.Data;
using System.Data.OleDb;
```

```
using System.Configuration;
using System.Web;

namespace TuNguyen.WebLibrary {

    public abstract class DbConnection : IDbConnection {
        string connString = "";
        OleDbConnection conn = null;
        DataSet ds = null;
        public DbConnection() {
            connString = GetConnString();
        }
        public abstract string GetConnString();
        public OleDbConnection GetConnection() {
            if (connString == "")
                throw(new Exception("Unable to open database connection"));
            if (conn == null) conn = new OleDbConnection(connString);
            if (conn.State == ConnectionState.Closed) {
                conn.Open();
                if (conn.State != ConnectionState.Open)
                    throw(new Exception("Unable to open database connection"));
            }
            return conn;
        }
        public void CloseConnection() {
            if (conn != null && conn.State == ConnectionState.Open) conn.Close();
        }
        public OleDbDataReader GetDataReader(string sql) {
            OleDbDataReader reader = null;
            OleDbCommand cmd = new OleDbCommand(sql, GetConnection());
            try {reader = cmd.ExecuteReader();}
            catch (Exception e) {
                CloseConnection();
                throw(e);
            }
            finally {cmd.Dispose();}
            return reader;
        }
        public OleDbDataReader GetDataReader(OleDbCommand cmd) {
            OleDbDataReader reader = null;
            cmd.Connection = GetConnection();
            try {reader = cmd.ExecuteReader();}
            catch (Exception e) {
                CloseConnection();
                throw(e);
            }
            return reader;
        }
        public int ExecuteSql(string sql) {
```

Notes

```
int count = 0;
OleDbCommand cmd = new OleDbCommand(sql, GetConnection());
try {count = cmd.ExecuteNonQuery();}
catch (Exception e) {throw(e);}
finally {
CloseConnection();
cmd.Dispose();
}
return count;
}

public int ExecuteSql(OleDbCommand cmd) {
int count = 0;
cmd.Connection = GetConnection();
try {count = cmd.ExecuteNonQuery();}
catch (Exception e) {throw(e);}
finally {CloseConnection();}
return count;
}

public DataSet GetDataSet() {
if (ds == null) ds = new DataSet();
return ds;
}

public DataView GetDataView(OleDbCommand cmd) {
cmd.Connection = GetConnection();
OleDbDataAdapter da = new OleDbDataAdapter(cmd);
try {da.Fill(GetDataSet());}
catch (Exception e) {throw e;}
finally {
da.Dispose();
CloseConnection();
}
DataView view = ds.Tables[0].DefaultView;
return view;
}

public DataView GetDataView(string sql) {
OleDbCommand cmd = new OleDbCommand(sql, GetConnection());
OleDbDataAdapter da = new OleDbDataAdapter(cmd);
try {da.Fill(GetDataSet());}
catch (Exception e) {throw e;}
finally {
da.Dispose();
cmd.Dispose();
CloseConnection();
}
DataView view = ds.Tables[0].DefaultView;
return view;
}

public void Dispose() {
CloseConnection();
```

```

if (conn != null) {
    conn.Dispose();
    conn = null;
}
if (ds != null) {
    ds.Clear();
    ds.Dispose();
    ds = null;
}
GC.SuppressFinalize(this);
}
}
}

```



Did u know? The IIS 6.0 uses a new process model called worker process isolation mode, which is different from the process model used in previous versions of IIS. ASP.NET uses this process model by default when running on Windows Server 2003.

8.2 User Control

Server controls are one from the possessions that create creating along with ASP.NET so easy as well as influential at the similar period. We have discuss HTML as well as web server controls and possess showed you how to rely on them in your ASP.NET pages, but what if present is not a control that do precisely what you want to do? Like most all in ASP.NET there's really no magic concerned along with server controls. In fact, you can build your own controls and employ them on your pages just like you use the ones that deliver with .NET. Controls that you build are called user controls and they are the subject of this unit.

User Control Structure

ASP.NET User Controls Overview

- **How to:** Convert Web Forms Pages into ASP.NET User Controls
- **How to:** Include a User Control in an ASP.NET Web Page
- **How to:** Create Instances of ASP.NET User Controls Programmatically
- **How to:** Create Templated ASP.NET User Controls
- **How to:** Create ASP.NET User Controls
- **How to:** Include ASP.NET User Controls in Web Pages
- **Walkthrough:** Creating Reusable Elements with ASP.NET User Controls

8.2.1 Overview

An ASP.NET Web user control is similar to an entire ASP.NET Web page (.aspx file), with both a user interface page and program code. You create the user control in much the same way it is created in ASP.NET web page and then add the markup and child controls that you need. A user control can include code to manipulate its contents like a page can, such as performing tasks such as information binding.

Notes

A user controls differs from an ASP.NET Web page in these ways:

- User control file name extension is .ascx.
- Instead of a Page directive, the user control consists of an Control directive that defines configuration and other qualities.
- User controls cannot be executed as a stand-alone files. Rather, you must add them to ASP.NET pages, as you would any kind of control.
- The user control does not have HTML, body, or form components in it. These elements should be in the hosting web page.
- You can use the exact same HTML elements (except the html, body, or form elements) as well as Web controls on a user control that you do on an ASP.NET Web page. For example, if you are developing a user control to use as a toolbar, you can put a series of Button Web server controls on to the control and create event handlers for the buttons.

The following example shows a complete user control. The user control displays a read-only text box with a number in it and two arrows that users can click to increment and decrement the value in the text box. The control exposes three properties, MinValue, MaxValue, and CurrentValue, that can be used in the hosting page.

```
<% @ Control Language="C#" ClassName="Spinner" %>

<script runat="server">
privateint m_minValue;
privateint m_maxValue = 100;
privateint m_currentNumber = 0;
publicint MinValue
{
get
{
return m_minValue;
}
set
{
if(value >= this.MaxValue)
{
thrownew Exception("MinValue must be less than MaxValue.");
}
else
{
m_minValue = value;
}
}
}

publicint MaxValue
{
get
{
return m_maxValue;
}
}
```

Notes

```
set
{
if(value <= this.MinValue)
{
thrownew
Exception("MaxValue must be greater than MinValue.");
}
else
{
m_maxValue = value;
}
}
}

publicint CurrentNumber
{
get
{
return m_currentNumber;
}
}

protectedvoid Page_Load(Object sender, EventArgs e)
{
if(IsPostBack)
{
m_currentNumber =
Int16.Parse(ViewState["currentNumber"].ToString());
}
else
{
m_currentNumber = this.MinValue;
}
DisplayNumber();
}

protectedvoid DisplayNumber()
{
textNumber.Text = this.CurrentNumber.ToString();
ViewState["currentNumber"] = this.CurrentNumber.ToString();
}

protectedvoid buttonUp_Click(Object sender, EventArgs e)
{
if(m_currentNumber == this.MaxValue)
{
m_currentNumber = this.MinValue;
}
else
```

Notes

```
{
    m_currentNumber += 1;
}
DisplayNumber();
}
protectedvoid buttonDown_Click(Object sender, EventArgs e)
{
    if(m_currentNumber == this.MinValue)
    {
        m_currentNumber = this.MaxValue;
    }
    else
    {
        m_currentNumber -= 1;
    }
    DisplayNumber();
}
</script>
<asp:TextBox ID="textNumber" runat="server"
    ReadOnly="True" Width="32px" Enabled="False" />
<asp:Button Font-Bold="True" ID="buttonUp" runat="server"
    Text="^" OnClick="buttonUp_Click" />
<asp:Button Font-Bold="True" ID="buttonDown" runat="server"
    Text="v" OnClick="buttonDown_Click" />
```

The example user control includes code to define three properties, `MinValue`, `MaxValue`, and `CurrentValue`. Properties in user controls must be public. In this example, all three properties are created with get and set accessors to enable the control to check for values outside of an acceptable range. However, you can create a property by simply declaring a public member.

The `MinValue` and `MaxValue` properties can be set declaratively in the containing page using syntax such as the following:

```
<uc:Spinner ID="Spinner1" runat="server" MinValue=0 MaxValue=10 />
```

When the user control is initialized as part of the containing page, ASP.NET parses the declared property values and sets them in the user control automatically.

As with pages, the user control is re-initialized with each postback. Property values therefore must be stored in a persistent location between postbacks. A typical location to store property values is view state. During the Load event, you must check for a postback and reload the property values from the store. For more information about view state, see ASP.NET View State.

8.2.2 How to Convert Web Form Pages into ASP.NET User Control

If you have developed an ASP.NET Web page and would like to access its functionality throughout your application, you can make a few minor alterations to the page to change it to a user control.

To convert a single-file ASP.NET Web page into a user control

1. Relabel the control file name extension to `.ascx`.
2. Remove the actual HTML, body, and form elements from the web page.

3. Transform the Page directive to a Control directive.
4. Eliminate all attributes of the Control directive except Language, AutoEventWireup (if present), CodeFile, and Inherits.
5. Incorporate a className feature in the Control directive. This permits the user control to be highly typed when it is actually added to a web page.

Notes

To convert a code-behind ASP.NET Web page into a user control

1. Relabel the .aspx file so the actual file name extension is actually .ascx.
2. Rename the code-behind file to have the file name extension .ascx.vb or .ascx.cs, depending on which coding language the code-behind file is within.
3. Open the code-behind file and change the class from which it inherits from Page to User Control.
4. In the .aspx file, do the next:
 - ❖ Eliminate the HTML, body, and form elements from the page.
 - ❖ Change the Page directive to an Control directive.
 - ❖ Eliminate all features of the Control directive except Language, AutoEventWireup (if existing), CodeFile, and Inherits.
 - ❖ Within the Control directive, modify the CodeFile attribute to indicate the renamed code-behind file.
5. Include a className feature in the actual Control directive. This allows the user control to be strongly typed when it's added to a web page.

8.2.3 How to Include a User Control in an ASP.NET Web Page

To use a user control, you include it in an ASP.NET Web page. When a request arrives for a page and that page contains a user control, the user control goes through all of the processing stages that any ASP.NET server control performs.

To Include a user Control in a Web Forms Page

The first thing we have to do, is declare our UserControl. It can be done either in each page where it's used, or globally in the web.config file. There is no performance difference, but when declaring UserControls in the web.config file, the controls have to reside in a different directory than the page(s) using it.

For now, let's just declare it within the page. Add the following line below the standard page declaration:

```
<%@ Register TagPrefix="My" TagName="UserInfoBoxControl" Src="~/
UserInfoBoxControl.ascx" %>
```

Make sure that the Src value matches the path to your UserControl file. Now you may use the UserControl in your page, like any other control. For instance, like this:

```
<My:UserInfoBoxControl runat="server" ID="MyUserInfoBoxControl" />
```

If you look at the page now, you will see our UserControl in action, although the information will be a bit... limited. We will have to set a value for the properties we defined, for things to get just a bit more interestingly. Fortunately, it's very easy:

Notes

```
<My:UserInfoBoxControl runat="server" ID="MyUserInfoBoxControl"
UserName="John Doe" UserAge="45" UserCountry="Australia" />
```

You see, every public or protected member can be accessed declaratively, allowing easy access to them when we use our control. However, with this specific UserControl, chances are that you will be receiving the information from an external resource, like a database, and then populating the UserControl from there. This usually involves the CodeBehind of the page, so how can we do that? Pretty simple, actually. In the CodeBehind of the page, try something like this:

```
protected void Page_Load(object sender, EventArgs e)
{
    // These values can come from anywhere, but right now, we just hardcode
    them
    MyUserInfoBoxControl.UserName = "Jane Doe";
    MyUserInfoBoxControl.UserAge = 33;
    MyUserInfoBoxControl.UserCountry = "Germany";
}
```

8.2.4 How to Create Instances of ASP.NET User Controls Programmatically

To Create an Instance of a user Control Programmatically

In the user control, be certain that the actual @Control directive contains the ClassName attribute that assigns a class to the user control. The following example sets the ClassName attribute to strongly type a user control.

```
<%@ Control class Name= "MyUserControl" %>
```

Within the page where you wish to work with the user control, create a reference to the user control using the @Reference directive.

When you create the user control programmatically, the strong type for your user control is offered to the ASP.NET Web page only after you have created a reference to it. For example, the following code generates a reference to a user control produced in the MyUserControl.ascx file.

Create a case variable for the user control, using the control's class name. The class will be part of the ASP namespace.

```
<%@ Page Language="C#" %>
<%@ Reference Control="~/Controls/Spinner.ascx" %>
<script runat="server">
private ASP.Spinner Spinner1;
protectedvoid Page_Load(object sender, EventArgs e)
{
    Spinner1 = (ASP.Spinner)LoadControl("~/Controls/Spinner.ascx");
    // Set MaxValue first.
    Spinner1.MaxValue = 20;
    Spinner1.MinValue = 10;
    Placeholder1.Controls.Add(Spinner1);
}

protectedvoid Button1_Click(object sender, EventArgs e)
{
    Label1.Text = Spinner1.CurrentNumber.ToString();
}
```

```

}
</script>

<html>
<head id="Head1" runat="server">
<title>Load User Control Programmatically</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:Placeholder runat=server ID="Placeholder1" />
<br />
<asp:Button ID="Button1" runat="server"
  Text="Button"
  OnClick="Button1_Click" />
<br />
<br />
<asp:Label ID="Label1" runat="server" Text=""></asp:Label>
</div>
</form>
</body>
</html>

```

8.2.5 How to Create Templated ASP.NET User Controls

You can create user controls that implement templates, an ASP.NET feature that enables the separation of control data from its presentation. A templated control does not provide a UI. Instead, it is written to apply the naming container and to include a class whose attributes and methods are accessible to the host page.

The user interface for the user control is supplied by a page programmer at design time. The developer creates templates of the type defined by the user control, and can then add controls and markup to the templates.

To Create a Template User Control

1. Add an ASP.NET Placeholder control in .ascx file where you want the template to appear.
2. Apply a property of type ITemplate in the user control's code.
3. Implements the INamingContainer interface in server control class as a container to create an instance of the template. This is known as the template's naming container.
4. Implement the TemplateContainerAttribute to the property that apply ITemplate and pass the type of the template's naming container as the argument to the attribute's constructor.
5. Repeat the below mentioned steps one or more time in the control's Init method:
 - ❖ Build an event of the naming container class.
 - ❖ Build an event of the template in the naming container.
 - ❖ Add the naming container case to the Controls property of the Placeholder server control.

8.2.6 How to Create ASP.NET User Controls

You create ASP.NET user controls in very similar way you design ASP.NET webpages. You can use exactly the same HTML elements and controls on a user control that you do on a standard ASP.NET web page. However, the user control does not have HTML, body, and form elements; and the file name extension must be .ascx.

To Create an ASP.NET User Control

- Create or open a website project in which you want to add user controls.
- Click Add New Item in Website menu.
- Add New Item dialog box will appears.
- Click Web User Control in the Add New Item dialog box, under Visual Studio installed templates.
- Type a name for the control in the Name box.
- .ascx file name extension is added to the control name that you type.
- Select the programming language that you want to use from the Language list.
- Select the Place code in separate file check box if you want to keep any code for the user control in a separate file,\.
- Click Add.

8.2.7 How to Include ASP.NET User Controls in Web Pages

Adding an ASP.NET user control to a Web page is same as adding other server controls to the page. However, you must be certain to follow the sequence of steps beneath so that all the necessary elements are put into the page.

To add an ASP.NET user Control to a Web Page

1. Open the Web page to which you want to add the ASP.NET user control in Visual Web Developer.
2. Toggle to Design view.
3. Select and drag your custom user control file in Solution Explorer onto the page.

8.2.8 Creating Reusable Elements with ASP.NET User Controls

ASP.NET user controls allow you to encapsulate the features of multiple server controls in a unit. User controls are made up of one or more ASP.NET server controls – Button controls, TextBox controls, and so forth – along with any code that's needed for the controls to carry out the operation that you want to accomplish. The user control can also include custom properties or techniques that reveal the features from the user control to a container, which is usually an ASP.NET page. In this walkthrough, you will create an ASP.NET user control that acts as the picker control. The picker control has two lists, with a set of choices in one list (the source). Users can select items within the SourceList list, and then include the items to the Target List.

It has three parts, as follows:

1. By adding controls and code create the basic user control.
2. Add the user control in a new ASP.NET page (the host page).
3. In order to interact with the control from the host page, add custom properties and methods to the user control.

Tasks illustrated in this include the following:

1. Create a user control and add ASP.NET server controls to the user control.
2. Declare properties and a method in the user control.
3. Add a user control to a host page means the page you want your user control to run with.
4. Add code to the host page to handle user control.

Creating the User Control

Creating a user control is comparable to creating an ASP.NET Web page. In fact, a user control is a subset of an ASP.NET page and it may contain most of the kinds of components that you put on an ASP.NET page.

To create a user control

- Create or open a website project in which you want to add user controls.
- Click Add New Item in Website menu.
- Add New Item dialog box will appear.
- Click Web User Control in the Add New Item dialog box, under Visual Studio installed templates.
- Type a name for the control in the Name box.
- .ascx file name extension is added to the control name that you type.
- Select the programming language that you want to use from the Language list.
- Select the Place code in separate file check box if you want to keep any code for the user control in a separate file, \.
- Click Add.

The new control is produced and is opened in the actual design. The markup for the control is very similar to of the webpage, with one important exception: there is no Page directive at the top of the page. Rather, there is a Control directive, which identifies the actual file to ASP.NET like a user control.

Adding Server Controls to the User Control

Here we will add the controls that make up the user interface for the user control.

To add server controls

1. Toggle to Design view.
2. Click Insert Table on the Table menu.

Notes

3. Create a table with one row and three columns using Insert Table dialog box, and then click OK.
4. Type Available in the left column of the table, and then press ENTER to create a new line.
5. Type Selected in the right column, and then press ENTER to create a new line.
6. From the Standard group in the Toolbox, drag the following controls onto the table and set their properties as indicated.

Limitations of User Controls in ASP.NET

As the user controls are not compiled into assemblies, they have the following limitations:

1. Each web application must contain a copy the control in which the control is used.
2. User controls must create them by dragging the control from Solution Explorer to the Web form as it cannot be loaded in the Visual Studio .NET Toolbox.
3. Until web form load event the user control property values are not updated as user control code initialized after web form load.

Advantages of Using User Controls in ASP.NET

1. User controls give more features than the ordinary controls since you can combine another controls to create user controls. User controls can be created that are unique to the application that you create.
2. To help you add unique functionalities to the actual control. These user controls are easy to add to any web page once created. So there is no need to code again for the same functionality if it is used in another page of the application. It is also feasible to transform the whole web pages into a user control with a few minor alterations and use it anywhere.
3. By using the user control for some features you are able to separate the design work and also the back-end work so that you can engage your design and programming team separately and efficiently.
4. If you use user control the development is modularized and that means you have various features available by using just a simple tag inside your web page. This speeds in the development process.



Caution When you develop a custom server control, you must include it in a namespace. If you do not, it will not be accessible from an ASP.NET page.

Self Assessment

Multiple Choice Questions:

1. element in the web.config file to run code using the permissions of a specific user.
 - a. < credential>
 - b. < authentication>
 - c. < authorization>
 - d. < identity>
2. Which one is the Default scripting language in ASP?
 - a. EcmaScript
 - b. VBScript
 - c. PERL
 - d. JavaScript

Notes

```
</asp:RadioButtonList>
</form>
</body>
</html>
```

However, with data binding we might use a separate source, just like a database, an XML file, or perhaps a script to fill the list with selectable items.

By using an imported source, the data is separated from the HTML, and any changes to the items are made in the separate data source.

8.3.1 RadioButtonList

The RadioButtonList control issued to create a group of radio buttons.

Each selectable item in a RadioButtonList control is defined by a List Item element!

```
<script runat="server">
Sub submit(senderAsObject, eAsEventArgs)
label1.Text="You selected"&radiolist1.SelectedItem.Text
EndSub
</script>
<!DOCTYPE html>
<html>
<body>
<form runat="server">
<asp:RadioButtonList id="radiolist1" runat="server">
<asp:ListItem selected="true">Item1</asp:ListItem>
<asp:ListItem>Item2</asp:ListItem>
<asp:ListItem>Item3</asp:ListItem>
<asp:ListItem>Item4</asp:ListItem>
<asp:ListItem>Item5</asp:ListItem>
</asp:RadioButtonList>
<br/>
<asp:Button text="Submit" OnClick="submit" runat="server"/>
<p><asp:Label id="Label1" runat="server"/></p>
</form>
</body>
</html>
```

8.3.2 CheckBoxList

The CheckBoxList control issued to create a multi-selection check box group.

Each selectable item in a CheckBoxList control is defined by a List Item element!

```
<script runat="server">
Sub Check(senderAsObject, eAsEventArgs)
dim i
mess.Text=""<p>SelectedItem(s) :</p>"
for i=0 to check1.Items.Count-1
if check1.Items(i).Selected then
mess.Text+=check1.Items(i).Text+"<br/>" endif
next
EndSub
```

```

</script>
<!DOCTYPEhtml>
<html>
<body>
<formrunat="server">
<asp:CheckBoxListid="check1"AutoPostBack="True"
TextAlign="Right"OnSelectedIndexChanged="Check"runat="server">
<asp:ListItem>BMW</asp:ListItem>
<asp:ListItem>Mercedes</asp:ListItem>
<asp:ListItem>Jaguar</asp:ListItem>
<asp:ListItem>Audi</asp:ListItem>
<asp:ListItem>Volkswagen</asp:ListItem>
<asp:ListItem>Fiat Linea</asp:ListItem>
</asp:CheckBoxList>
<br/>
<asp:Labelid="mess"runat="server"/>
</form>
</body>
</html>

```

8.3.3 Dropdown List

The Dropdown List control issued to create a drop-downlist.

Each selectable item in a Dropdown List control is defined by a List Item element!

```

<scriptrunat="server">
Subsubmit(senderAsObject,eAsEventArgs)
mess.Text="Youselected"&drop1.SelectedItem.TextEndSub
</script>
<!DOCTYPEhtml>
<html>
<body>
<formrunat="server">
<asp:DropDownListid="drop1"runat="server">
<asp:ListItem>BMW</asp:ListItem>
<asp:ListItem>Mercedes</asp:ListItem>
<asp:ListItem>Jaguar</asp:ListItem>
<asp:ListItem>Audi</asp:ListItem>
<asp:ListItem>Volkswagen</asp:ListItem>
<asp:ListItem>Fiat Linea</asp:ListItem>
</asp:DropDownList>
<asp:ButtonText="Submit"OnClick="submit"runat="server"/>
<p><asp:Labelid="mess"runat="server"/></p>
</form>
</body>
</html>

```

8.3.4 Listbox

The Listbox control issued to create a single-or multi-selection drop-downlist.


Each selectable item in a Listbox control is defined by a List Item element!

Notes

```

<script runat="server">
Subsubmit (SenderAsObject, eAsEventArgs)
mess.Text="Youselected"&drop1.SelectedItem.Text
EndSub
</script>
<!DOCTYPEhtml>
<html>
<body>
<form runat="server">
<asp:ListBox id="drop1" rows="3" runat="server">
<asp:ListItem selected="true">BMW</asp:ListItem>
<asp:ListItem>Mercedes</asp:ListItem>
<asp:ListItem>Jaguar</asp:ListItem>
<asp:ListItem>Audi</asp:ListItem>
<asp:ListItem>Volkswagen</asp:ListItem>
<asp:ListItem>Fiat Linea</asp:ListItem>
</asp:ListBox>
<asp:Button Text="Submit" OnClick="submit" runat="server" />
<p><asp:Label id="mess" runat="server" /></p>
</form>
</body>
</html>

```



Task Write a program for code binding.

8.4 Understanding Two-Way Data Binding

DataSet/DataAdapter grouping is one common example of two-way data binding. You can update the data in the DataSet, and then update the underlying information trigger using the DataAdapter, or, if changes occur to the underlying data, you can refresh the actual DataSet using the Fill() method.

With ASP.NET 2.0's bi-directional data binding, you can eliminate the code required to update a data store by binding the data controls in your presentation layer to the data sources. Since the binding is bi-directional, whenever users alter the data within the controls the underlying data sources update when users post back to the server. Typically, you have had to use code similar to the following to manage displayed data binding:

```

//Populate the data controls from a business object
txtid.Text=emp.id;
txtName.Text=emp.Name;
//Populate a business object from the data controls emp.id=txtid.Text;
emp.Name=txtName.Text;
<%#Bind("FieldName")%>

```

8.4.1 Sample Two-Way Data Binding Application

ASP.NET databinder works with reflection. In simple terms Eval("PropertyName") get the value from the data object and put it on the relevant control, while Bind ("PropertyName") get the value from the data object put it in the control and one the update event it get the updated control value pass it to the relevant method to update.

Very simple example:**Notes**

Person class:

```
public class Person
{
    public string Name { get; set; }
    public List<Person> GetPeople()
    {
        List<Person> people = new List<Person>();
        for (int i = 0; i < 10; i++)
            people.Add(new Person { Name = "Person " + i });
        return people;
    }
    public void SetPerson(string Name)
    {
        }
    }
}
```

Markup:

```
<%@ Page Language="C#" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head2" runat="server">
</head>
<body>
    <form id="form2" runat="server">
        <div>
            <asp:GridView
                runat="server"
                ID="grvPeople"
                AutoGenerateColumns="false"
                DataSourceID="odsPeople"
                AutoGenerateEditButton="true">
                <Columns>
                    <asp:TemplateField>
                        <EditItemTemplate>
                            <asp:TextBox runat="server" ID="txtItem" Text='<%= Bind("Name") %>'></
asp:TextBox>
                        </EditItemTemplate>
                        <ItemTemplate>
                            <asp:Label runat="server" ID="lblPerson"
                                Text='<%= string.Format("Mr. {0}", Eval("Name")) %>'></asp:Label>
                        </ItemTemplate>
                    </asp:TemplateField>
                </Columns>
            </asp:GridView>
            <asp:ObjectDataSource
                ID="odsPeople"
                runat="server"
                TypeName="ActiveTest.Person"
                SelectMethod="GetPeople">
```

Notes

```
UpdateMethod="SetPerson">
</asp:ObjectDataSource>
</div>
</form>
</body>
```

In this example, as you can see

1. I have object datasource "odsPeople" which has the type of 'Person'
2. I have Select Method as public method GetPeople
3. I have Update Method as public method SetPerson

Number of parameters pass to SetPerson method is equal to the number of Bind(...) expressions. Parameter names are exactly equal to bound property name which is 'Name' here.

Where to use: In a simple page like this, where we don't have templated or templated repeater controls like Repeater, GridView, FormView, DetailsView etc., using Bind(...) method does not make any sense,

```
<%@ Page Language="C#" %>
<html>
<head id="Head2" runat="server">
  <script runat="server">
    public Person Person { get; set; }
    protected override void OnLoad(EventArgs e)
    {
      base.OnLoad(e);
      if (!this.IsPostBack)
      {
        this.Person = new Person();
        this.txtName.DataBind();
      }
    }
  </script>
</head>
<body>
  <form id="form2" runat="server">
    <div>
      <asp:TextBox runat="server" ID="TextBox1" Text='<%= Bind("Person.Name") %>' /
      >
    </div>
  </form>
</body>
</html>
```

Because it is very simple like this.

```
<%@ Page Language="C#" %>
<html>
<head id="Head1" runat="server">
  <script runat="server">
    public Person Person { get; set; }
    protected override void OnLoad(EventArgs e)
```

```

{
base.OnLoad(e);
if (!this.IsPostBack)
{
this.Person = new Person();
this.txtName.Text = "Save";
}
}
protected void Save(object sender, EventArgs e)
{
string name = this.txtName.Text;
}
</script>
</head>
<body>
<form id="form1" runat="server">
<div>
<asp:TextBox runat="server" ID="txtName" />
<asp:Button runat="server" ID="btnSave" Text="Save" OnClick="Save" />
</div>
</form>
</body>
</html>

```

With LINQ to SQL

```

<%@ Page Language="C#" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head2" runat="server">
</head>
<body>
<form id="form2" runat="server">
<div>
<asp:GridView
runat="server"
ID="GridView1"
AutoGenerateColumns="False"
AutoGenerateEditButton="true"
AutoGenerateSelectButton="true"
AutoGenerateDeleteButton="true"
DataKeyNames="Id"
DataSourceID="ldsLinqDataSource">
<Columns>
<asp:BoundField DataField="Id"
HeaderText="Id" InsertVisible="False"
ReadOnly="True" SortExpression="Id" />
<asp:TemplateField>
<ItemTemplate>
<asp:Label runat="server" ID="lblUrl" Text='<%= Eval("Url") %>' />
</ItemTemplate>


```


Notes

```

<EditItemTemplate>
<asp:TextBox runat="server" ID="txtUrl" Text='<%# Bind("Url") %>' />
</EditItemTemplate>
</asp:TemplateField>
</Columns>
</asp:GridView>
<asp:LinqDataSource
ID="LinqDataSource1"
runat="server"
ContextTypeName="ActiveTest.ActiveTestDataContext"
EntityTypeName=""
TableName="Projects"
EnableUpdate="true"
EnableDelete="true"
EnableInsert="true">
</asp:LinqDataSource>
</div>
</form>
</body>
</html>

```



Task Create a two way binding application.

8.5 Data-Bound Controls

Most ASP.NET applications rely on some degree of data presentation from a back-end data source. Data-bound controls have been a pivotal part of interacting with data in dynamic Web applications. ASP.NET 2.0 introduces some substantial improvements to data-bound controls, including a new BaseDataBoundControl class and declarative syntax.

The BaseDataBoundControl acts as the base class for the DataBoundControl class and the HierarchicalDataBoundControl class. In this module, we will discuss the following classes that derive from DataBoundControl:

- AdRotator
- List controls
- GridView
- FormView
- DetailsView

Following classes that derive from HierarchicalDataBoundControl class are:

- TreeView
- Menu
- SiteMapPath

8.5.1 An Item

Notes

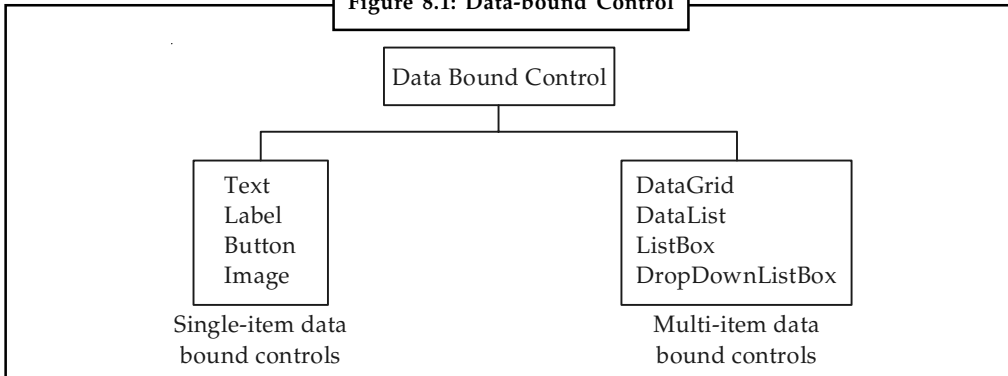
We use the multi-item data bound controls to display the entire or a partial table. These controls provide direct binding to the data source. We use the Data Source property of these controls to bind a database table to these controls (Table 8.1).

Table 8.1: Multiple-item Controls

AMAR	234, G. Road	12398
SUNIL	5443, NY	89433
AJAY	P.RD	54323

Some examples of multi-item data-bound controls are DataGrid, ListBox, DataList, DropDownList, and so on some of the data bound controls (See Figure 8.1).

Figure 8.1: Data-bound Control



In ASP.NET, you create these controls using a `<asp:controlName>` tag. describes some common data-bound server-side controls

8.5.2 ASP.NET Data-bound Controls

Databound Controls in asp.net

- **SqlDataAdapter.:** The `SqlDataAdapter`, serves as a bridge between a `DataSet` and SQL Server for retrieving and saving data. The `SqlDataAdapter` provides this bridge by mapping `Fill`, which changes the data in the `DataSet` to match the data in the data source, and `Update`, which changes the data in the data source to match the data in the `DataSet`, using the appropriate SQL statements against the data source.
- **SqlDataSource:** The `SqlDataSource` control enables you to use a Web server control to access data that is located in a relational database. This can include Microsoft SQL Server and Oracle databases, as well as OLE DB and ODBC data sources. You can use the `SqlDataSource` control with data-bound controls such as the `GridView`, `FormView`, and `DetailsView` controls to display and manipulate data on an ASP.NET Web page, using little or no code.
- **GridView:** ASP.NET `GridView` control available in Visual Studio allows data binding and editing without writing a single line of code.
- **DetailsView:** The `DetailsView` control is used to display one or more fields from a single data record. The unadapted version of the `DetailsView` control renders these fields as rows in an HTML. An adapter can be used to render an unordered list (`` and `` tags) instead.

Notes

- **FormsView:** Show how the FormView control can display data for a user to edit, delete and also add to the database.

Web.Config

- **DataList:** The DataList control is, like the Repeater control, used to display a repeated list of items that are bound to the control. However, the DataList control adds a table around the data items by default. The DataList control may be bound to a database table, an XML file, or another list of items. Here we will show how to bind an XML file to a DataList control.
- **Repeater:** The Repeater control is used to display a repeated list of items that are bound to the control. The Repeater control may be bound to a database table, an XML file, or another list of items. Here we will show how to bind an XML file to a Repeater control.
- **ListView:** The ASP.NET ListView control enables you to bind to data items that are returned from a data source and display them. You can display data in pages. You can display items individually, or you can group them.

The ListView control displays data in a format that you define by using templates and styles.



Case Study

Account Executives at Global Media Company Spend up to 60 Percent More Time with Sponsors

Account executives at Scripps Networks, LLC use video samples and slide presentations to help sell ad time. But finding the right content, copying it, and shipping it to advertisers was time-consuming and expensive. Scripps Networks turned to its video and presentation management vendor, PPTshare, which helped it to adopt a sales enablement solution based on PPTshare technology and Microsoft SQL Server 2008 R2 data management software. The customized content library and presentation management system reduced the time to locate content and get it to potential clients by 98 percent. Account executives now spend up to 60 percent more of their time with media buyers and advertisers—and still spend more time with their families. The solution has high availability, won rapid user adoption, and delivers business intelligence that Scripps Networks uses to burnish its sales effectiveness.

Situation

PPTshare was accustomed to helping its customers to organize and use their video and slide presentation decks. The challenge posed by Scripps Networks LLC, however, was a bit more complicated than that.

Scripps Networks—home to HGTV, Food Network, DIY Network, Travel Channel, Cooking Channel, and Great American Country, among other brands—was concerned that its video and presentation content management practices could become a drag on the company’s advertising revenue. It brought those concerns to PPTshare, its longtime video and presentation management vendor, and sought a solution.

The account executives at Scripps Networks relied on their company’s enormous library of video content—housed in New York City and Knoxville, Tennessee—to help make sales pitches. In particular, they used the thousands of video “vignettes” that the Scripps Networks television brands had produced for advertisers over the years. The vignettes

Contd....

Notes

were important sales aids because they gave prospective advertisers specific examples of how Scripps Networks could produce similar content for them. Account executives would generally bring the videos with them to client meetings or mail them to clients.

That is, they would when they could find them. Locating a video that had been produced for another advertiser a couple of years before might require tracking down the executive responsible for that account at the time and asking about the video's whereabouts. It wasn't always clear if the content was stored in Knoxville or New York. It could take a week to find, copy, and send a video to the regional office that had requested it. The account executive then mailed the video to a media buyer or advertiser—and wondered when, or if, the video had been viewed.

"Scripps Networks came to feel that it wasn't the most productive way for it to operate," says Alexandra Ontra, President of PPTshare. "Any time that the account executives spent looking for content was time that they weren't out meeting with clients and generating sales."

The time to track down and process video content was equally distracting to Scripps Networks. "Time spent responding to requests for existing video was time that the ad sales marketing department at Scripps Networks couldn't spend building new sales tools, producing new vignettes, or creating new partnership opportunities for advertisers," says Ontra.

Video wasn't the only medium creating a problem for Scripps Networks account executives. Slides—a simpler medium in many ways—were causing a greater headache. "Anyone can build a slide deck," points out Ontra. "And Scripps Networks had everyone from planners to vice presidents building them—but without any practical way to exchange them or to share best practices."

When account executives in New York City and Los Angeles, California, for example, were each pitching Scripps Networks vignettes to independent automotive companies, much of their presentations might be the same—but each presentation would be created separately, and redundantly.

"To cut down on the time spent recreating similar decks over and over, Scripps Networks wanted to produce 'master' sales presentations that were the result of repeated refinement by their account execs around the country," says Ontra. "And the presentations needed to be instantly accessible to everyone. Scripps Networks asked for our help and we agreed to provide it."

With that, all this wasn't just Scripps Networks' problem anymore. Now, it was PPTshare's problem, too.

Questions:

1. What are the services provided by Scripps Networks?
2. What are the benefits Company will get after adopted a video and presentation library and management solution by PPTshare, powered by Microsoft technologies?

Self Assessment

Multiple Choice Questions:

6. Aserves content in a number of languages. It contains multiple copies for its content and other resources, such as date and time, in different languages.
 - a. Multilingual Web site
 - b. Static Website
 - c. Dynamic Website
 - d. MultiLanguage Website

Notes

7. You can register a custom server control to a Web page using thedirective.
 - a. @Page
 - b. @Register
 - c. @language
 - d. @Control
8. How can you load an XML file directly into a DataSet
 - a. By setting the XML property of the DataSet class
 - b. By using the GetXML() method of the DataSet class
 - c. By using the ReadXml() method of the DataSet class
 - d. You cannot load an XML file directly into a DataSet
9. Which one is Not a Data bound Control
 - a. Data Grid
 - b. Image Button
 - c. CheckBox List
 - d. Repeater
10. You require to create an ASP.NET page with the functionality to allow a user to upload a file to the server
 - a. You need to use the System.Web.Upload namespace
 - b. You need to use a COM component to save the file on the server
 - c. You need to use the SaveAs method of the HttpPostedFile class
 - d. The ASP.NET application automatically loops through all the <input type="File"> and saves the uploaded files to a virtual folder called "uploads"
11. To dynamically add a TextBox control to a page and display it at specific location
 - a. Place a Label control at the required location in the page and add the control by manipulating the Text property of the Label control
 - b. Place a Placeholder control at the required location in the page. Use the Add method of the Controls collection of the Placeholder control to dynamically add the required control
 - c. Build the HTML for the entire page in a String and insert the HTML code for required control in required place
 - d. You cannot add a control to a page dynamically
12.grouping is one common example of two-way data binding.
 - a. Data Adapter
 - b. Data Grid
 - c. Data View
 - d. ListView
13. You are writing ASP.NET 2.0 Web site that collects lots of data from users, and the data collection forms spreads over multiple ASP.NET Web pages. When the user reaches the last page, you need to gather all of data, validate the data, and save the data to the SQL Server database. You have noticed that it can be rather difficult to gather the data that is spread over multiple pages and you want to simplify this application. What is the easiest control to implement that can be used to collect the data on a single Web page?
 - a. The View control
 - b. TheTextBox control
 - c. The Wizard control
 - d. TheDataCollection control

14. How to implement authentication via web.config? Notes
- a. Include the authentication element.
 - b. Include the authorization element.
 - c. Include the identity element.
 - d. Include the deny element.
15. Which DLL translate XML to SQL in IIS?
- | | |
|-----------------|---------------|
| a. SQLISAPI.dll | b. SQLXML.dll |
| c. LISXML.dll | d. SQLIIS.dll |

8.6 Summary

- ASP.NET provides easy-to-use application and session-state facilities that are familiar to ASP developers and are readily compatible with all other .NET Framework APIs.
- XML Web services enable the exchange of information in client-server or server-server scenarios, using standards such as HTTP
- The .NET Framework and ASP.NET provide default authorization and authentication schemes for Web applications.
- A subclass, their class, or even child class is a modular, derivative class that inherits one or additional properties from a different class.
- Server controls are one of the things that make developing with ASP.NET so simple and powerful at the same time.
- An ASP.NET Web user control is similar to a complete ASP.NET Web page (.aspx file), with both a user interface page and code.
- You can create user controls that implement templates, an ASP.NET feature that enables the separation of control data from its presentation.
- ASP.NET user controls allow you to encapsulate the features of multiple server controls in a unit.
- User controls are not compiled into assemblies.
- Data binding is simply a program's ability to bind some members of a target component to the members of a data source.
- To use this data binding framework you create an array of DataBindingRegister instances and populate them with the necessary data binding information in the Page_Load event of the web form.
- ASP.Net has several databound Controls like SQL data Adapter, Grid View, Details View, etc.

8.7 Keywords

Active Server Pages (ASP): ASP known as Classic ASP or ASP Classic, was Microsoft's first server-side script engine for dynamically generated web pages.

API: It stands for Application Program Interface. It is a way through which using computers becomes quite easy.

Notes

Data Binding: It is general technique that binds two data/information sources together and maintains synchronization of data.

Extensible Markup Language (XML): It is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

LINQ: Language-Integrated Query (LINQ) is a set of features introduced in Visual Studio 2008 that extends powerful query capabilities to the language syntax of C# and Visual Basic.

User Controls: User controls are one of ASP.NET methods to increase reusability of code, implement encapsulation and reduce maintenance.



- Lab Exercise*
1. Create a custom user control in ASP.NET.
 2. Write an XML program to create a DTD for the college.

8.8 Review Questions

1. Create a custom user control in ASP.NET.
2. State the advantage of user control over server controls.
3. What is actually returned from server to the browser when a browser requests an .aspx file and the file is displayed?
4. What is DataSet Object in ADO.Net?
5. Explain the concept of page sub-classing.
6. Give an example of data bound using Checkbox List.
7. Create a template for user control in ASP.NET.
8. Explain the need of User Control.
9. Describe the term "Data Binding".
10. What is Container? Explain using an example.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (d) | 2. (b) | 3. (a) | 4. (c) |
| 5. (d) | 6. (a) | 7. (b) | 8. (c) |
| 9. (d) | 10. (c) | 11. (b) | 12. (a) |
| 13. (c) | 14. (b) | 15. (a) | |

8.9 Further Readings



Books

Professional ASP.NET by Jon Galloway, Phil Haack, and Brad Wilson



Online links

Net: Asp the Complete Reference, by Matthew MacDonald www.telerik.com/products/aspnet-mvc.aspx

Unit 9: The Database Model

Notes

CONTENTS

Objectives

Introduction

9.1 Concept of Database Model

9.1.1 Data Repository

9.1.2 Data Dictionary

9.1.3 Database Software

9.1.4 Data Abstraction

9.1.5 Data Access

9.2 Hierarchical Database Model

9.2.1 Parent – Child Relationships

9.2.2 Root Segment

9.2.3 Physical Pointers

9.3 Network Database Model

9.3.1 Levels

9.3.2 Record Types

9.3.3 Relationships

9.3.4 Multiple Parents

9.3.5 Physical Pointers

9.4 Relational Database Model

9.4.1 Levels

9.4.2 Relations or Tables

9.4.3 Relationships

9.4.4 No Physical Pointer

9.5 Types of Databases

9.5.1 Centralized Database

9.5.2 Distributed Database

9.6 Creating a Basic Object Model

9.6.1 Model Precisions

9.6.2 Instance Variables

9.6.3 Methods

9.6.4 Abstractness

9.7 Creating the User Interface

9.7.1 Designing User Interface Components

9.7.2 User Interface Component Functionality

Contd....

Notes

9.7.3	New System Design Process
9.7.4	Methodology in Action
9.8	Summary
9.9	Keywords
9.10	Review Questions
9.11	Further Readings

Objectives

After studying this unit, you will be able to:

- Describe the concept of database model
- Discuss the hierarchical database model
- Explain the network database model
- Understand the relational database model
- Define the types of databases
- Explain that how create a basic object model
- Describe the creating the user interface

Introduction

An important aspect of most every business is record keeping. In our information society, this has become an important aspect of business, and much of the world’s computing power is dedicated to maintaining and using databases.

Databases of all kinds pervade almost every business. All kinds of data, from emails and contact information to financial data and records of sales, are stored in some form of a database. The quest is on for meaningful storage of less-structured information, such as subject knowledge.

A data model is a conceptual representation of the data structures that are required by a database. The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects. As the name implies, the data model focuses on what data is required and how it should be organized rather than what operations will be performed on the data. To use a common analogy, the data model is equivalent to an architect’s building plans.

A data model is independent of hardware or software constraints. Rather than try to represent the data as a database would see it, the data model focuses on representing the data as the user sees it in the “real world”. It serves as a bridge between the concepts that make up real-world events and processes and the physical representation of those concepts in a database.

The data model gets its inputs from the planning and analysis stage. Here the modeler, along with analysts, collects information about the requirements of the database by reviewing existing documentation and interviewing end-users.

The data model has two outputs. The first is an entity-relationship diagram which represents the data structures in a pictorial form. Because the diagram is easily learned, it is valuable tool to communicate the model to the end-user. The second component is a data document. This document that describes in details the data objects, relationships, and rules required by the database. The dictionary provides the detail required by the database developer to construct the physical database.

9.1 Concept of Database Model

Notes

Representation of a real world situation about which data is to be collected and stored in a database. A data model depicts the dataflow and logical interrelationships among different data elements. Before proceeding further, we explain some key expressions related to concept.

9.1.1 Data Repository

Data Repository is a logical (and sometimes physical) partitioning of data where multiple databases which apply to specific applications or sets of applications reside. For example, several databases (revenues, expenses) which support financial applications (A/R, A/P) could reside in a single financial Data Repository.

9.1.2 Data Dictionary

A DBMS component that stores metadata. The data dictionary contains the data definition and its characteristics and entity relationships. This may include the names and descriptions of the various tables and fields within the database. Also included are things like data types and length of data items. The inclusion of primary and foreign key also adds to the consistency of the database being built. Overall having a well designed data dictionary will help make it easier to build and maintain your database.

There are two other types of Data Dictionaries:

1. **Active Data Dictionary:** A data dictionary that is automatically updated by the DBMS every time the database is accessed.
2. **Passive Data Dictionary:** Similar to Active DD however it is not automatically updated and usually requires a batch process to be run.

9.1.3 Database Software

Database software is the phrase used to describe any software that is designed for creating databases and managing the information stored in them. Sometimes referred to as database management systems (DBMS), database software tools are primarily used for storing, modifying, extracting, and searching for information within a database. Database software is used for a number of reasons in any industry – from keeping your bookkeeping on task, compiling client lists to running your online web site.

Because they have so many uses, there are dozens of database software programs available. The options have gone beyond Oracle or Microsoft Access to encompass FileMaker, Avanquest and Delicious Monster Software for options tailored to a variety of needs. Some of the more popular database software applications include desktop solutions like Microsoft Access and FileMaker Pro and server solutions like MySQL, Microsoft SQL Server and Oracle.

9.1.4 Data Abstraction

Abstraction is the process of recognizing and focusing on important characteristics of a situation or object and leaving/filtering out the unwanted characteristics of that situation or object. Lets take a person as example and see how that person is abstracted in various situations:

- A doctor sees (abstracts) the person as patient. The doctor is interested in name, height, weight, age, blood group, previous or existing diseases etc. of a person

Notes

- An employer sees (abstracts) a person as Employee. The employer is interested in name, age, health, degree of study, work experience, etc., of a person.

So, you see that Abstraction is the basis for software development. Its through abstraction we define the essential aspects of a system. The process of identifying the abstractions for a given system is called as Modelling (or object modelling).

9.1.5 Data Access

Data access typically refers to software and activities related to storing, retrieving, or acting on data housed in a database or other repository. There are two types of data access, sequential access and random access.

Data Access is simply the authorization you have to access different data files. Data access can help distinguish the abilities of Administrators and users. For example, Admin’s may be able to remove, edit and add data while a general user may not be able as they don’t have the access to that particular file.

Historically, different methods and languages were required for every repository, including each different database, file system, etc., and many of these repositories stored their content in different and incompatible formats.

In more recent days, standardized languages, methods, and formats, have been created to serve as interfaces between the often proprietary, and always idiosyncratic, specific languages and methods. Such standards include SQL, ODBC, JDBC, XQJ, ADO.NET, XML, X Query, X Path, and Web Services.

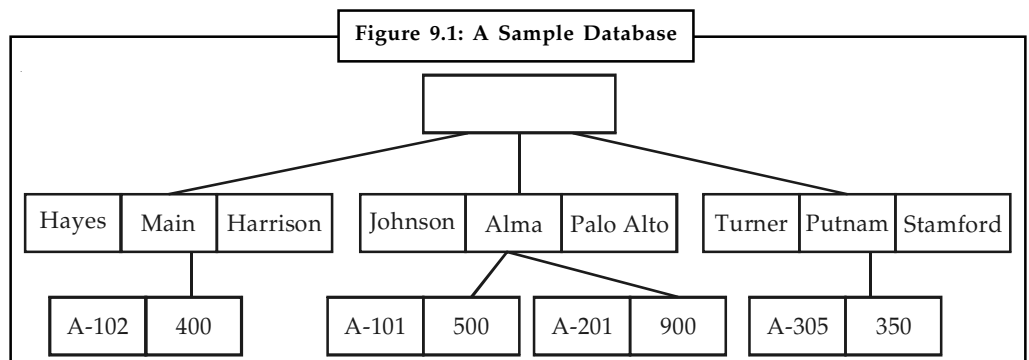
Some of these standards enable translation of data from unstructured (such as HTML or free-text files) to structured (such as XML or SOL).

9.2 Hierarchical Database Model

A hierarchical database consists of a collection of records that are connected to each other through links. A record is similar to a record in the network model. Each record is a collection of fields (attributes), each of which contains only one data value. A link is an association between precisely two records. Thus, a link here is similar to a link in the network model.

Consider a database that represents a customer-account relationship in a banking system. There are two record types: customer and account. The customer record type can be defined in the same manner as in Appendix A. It consists of three fields: customer name, customer street, and customer city. Similarly, the account record consists of two fields: account number and balance.

A sample database appears in Figure given below. It shows that customer Hayes has account A-102, customer Johnson has accounts A-101 and A-201, and customer Turner has account A-305.

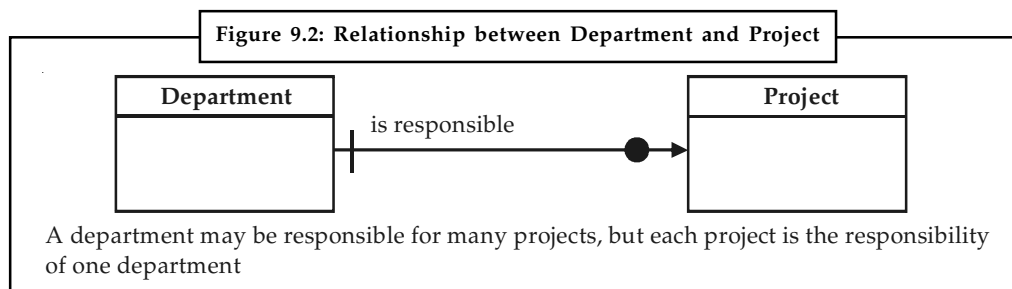


Note that the set of all customer and account records is organized in the form of a rooted tree, where the root of the tree is a dummy node. As we shall see, a hierarchical database is a collection of such rooted trees, and hence forms a forest. We shall refer to each such rooted tree as a database tree. The content of a particular record may have to be replicated in several different locations. For example, in our customer-account banking system, an account may belong to several customers. The information pertaining to that account, or the information pertaining to the various customers to which that account may belong, will have to be replicated. This replication may occur either in the Same database tree or in several different trees. Record replication has two major drawbacks:

1. Data inconsistency may result when updating takes place.
2. Waste of space is unavoidable.

9.2.1 Parent – Child Relationships

The relationship between each pair of data structures at levels after that to each other is a parent – child relationship. DEPARTMENT is considered the parent entity while PROJECT is the child. Reading from left to right, the diagram represents departments may be responsible for many projects. The optionality of the relationship reflects the “business rule” that not all departments in the organization will be responsible for managing projects. Reading from right to left, the diagram tells us that every project must be the responsibility of exactly one department.



9.2.2 Root Segment

The data segment at the top level node in the hierarchy is known as the root data segment means the node having no parent.

9.2.3 Physical Pointers

Physical pointers link records of the parent segments to those of the child segments by means of parent – child forward or backward relationship. It is of two types:

- Forward Pointer
- Backward Pointer

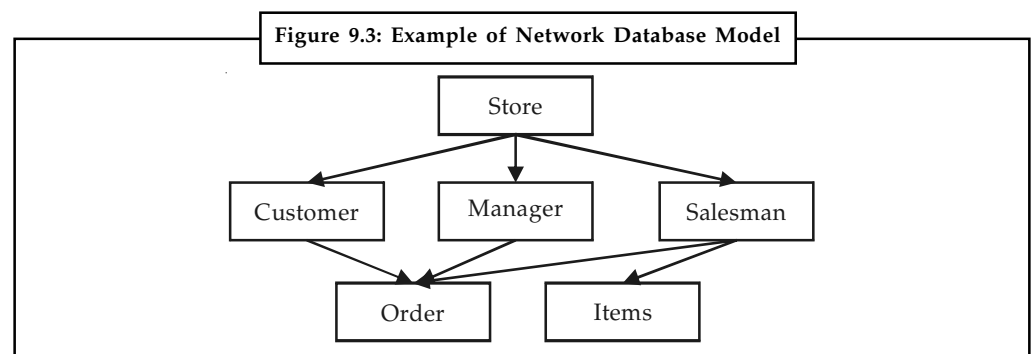
Forward and backward physical pointers link records of the same segment type but forward works in the next direction and backward works in the previous direction.

9.3 Network Database Model

The Network Database Model was created for three main purposes. These purposes include representing a complex data relationship more effectively, improving database performance, and imposing a database standard. The network model highly resembles the hierarchal model.

Notes

Another characteristic is the network database model is composed of at least two record types including an owner, which is a record type equivalent to the parent type in the hierarchal database model, and the member record type which resembles the child type in the hierarchal model. The schema, subschema, and data management language are a few of the key components that make this database model unique. The schema used for this model is conceptual organization of the entire database as the database administrator intends. The subschema defines the database portion as seen by the application programs that produce the information from the raw data that is contained in the given database. The network database model uses a data management language that defines data characteristics and the data structure in order to manipulate the data. The database management language uses a schema and subschema data definition language. The schema data definition language enables database administrators to define schema components. The subschema data definition language allows the application programs to define database components that will be used. There are a few advantages and disadvantages of using the network database model. Some advantages include conceptual simplicity, data access flexibility, conformance to standards, handle more relationship types, promote database integrity, and allows for data independence. The disadvantages of the network database model are the structure is difficult to change, this type of system is very complex, and there is a lack of structural independence. In summary the network database model is similar but different then the hierarchal database model. The network database model should be used when it is necessary to have a flexible way of representing objects and their relationships.



9.3.1 Levels

As in most real-world scenarios, no hierarchical levels can be found in the network model. The lines in a network data model purely connect the appropriate data structures wherever necessary with no constraint of connecting only successive levels as in the hierarchical model. Note the lines connecting the various data structures with no limitations.

9.3.2 Record Types

The basic data modeling construct in the network model is the set construct. A set consists of an owner record type, a set name, and a member record type. A member record type can have that role in more than one set, hence the multi-parent concept is supported. An owner record type can also be a member or owner in another set.

9.3.3 Relationships

The network data model expresses relationships between two record types by designating one as the owner record type and the other as the member record type. For each occurrence of an owner record type, there are one or more occurrences of the member record type. The owner

record type may be considered as the parent and the member record kind as the child. The owner record type “is the owner of” the corresponding member record kind. Each member type with its corresponding owner record type is actually a set. A set represents the connection between an owner and a member record type.

9.3.4 Multiple Parents

For ORDER member there are three parents or owners, namely, CUSTOMER, MANAGER and SALESMAN. Therefore for one event of CUSTOMER, one or more occurrences of ORDER exist.

9.3.5 Physical Pointers

Physical pointers link occurrences of an owner record type with the corresponding occurrences of the member record type. Within each record type itself the individual occurrences may be linked to one another by means of forward and backward pointers.

9.4 Relational Database Model

A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints. In such a database the data and relations between them are organised in tables. A table is a collection of records and each record in a table contains the same fields.

Properties of Relational Tables:

- Values Are Atomic
- Each Row is Unique
- Column Values Are of the Same Kind
- The Sequence of Columns is Insignificant
- The Sequence of Rows is Insignificant
- Each Column Has a Unique Name

Certain fields may be designated as keys, which means that searches for specific values of that field will use indexing to speed them up. Where fields in two different tables take values from the same set, a join operation can be performed to select related records in the two tables by matching values in those fields. Often, but not always, the fields will have the same name in both tables. For example, an “orders” table might contain (customer-ID, product-code) pairs and a “products” table might contain (product-code, price) pairs so to calculate a given customer’s bill you would sum the prices of all products ordered by that customer by joining on the product-code fields of the two tables. This can be extended to joining multiple tables on multiple fields. Because these relationships are only specified at retrieval time, relational databases are classed as dynamic database management system. The RELATIONAL database model is based on the Relational Algebra.

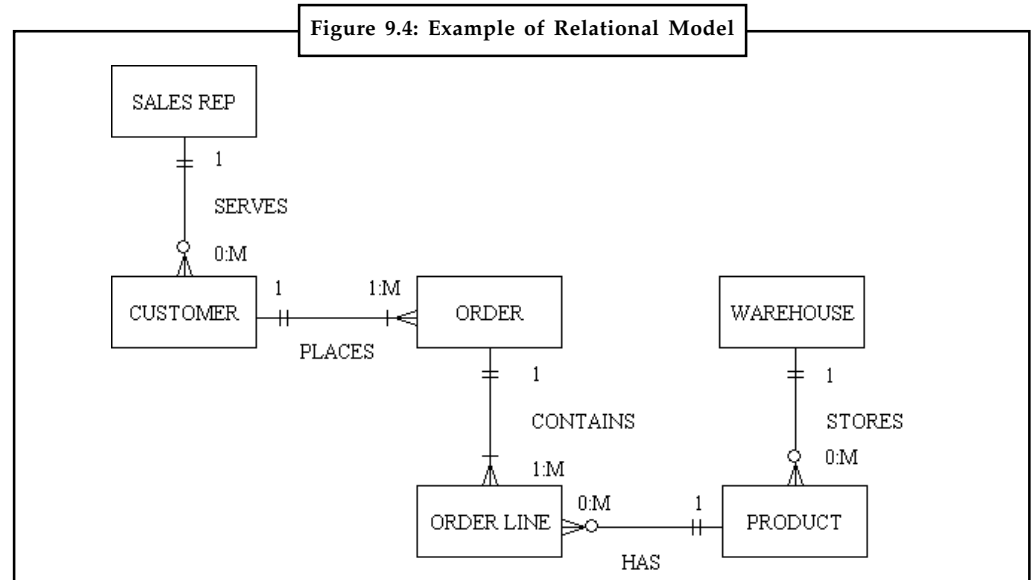
9.4.1 Levels

Just like the network data model, no hierarchical levels are present in the relational model. The actual lines in a relational data model basically indicate the relationships between the suitable data structures wherever required without the limitation of connecting only successive levels as in the hierarchical model. As in the network model, note the lines joining the different data structures with no restrictions.

Notes

9.4.2 Relations or Tables

The relational model consists of relations. Relation is a two-dimensional table of data observing relational rules. For example, the CUSTOMER relation represents the data related of all customers. The ORDER relation represents the data related of all orders.



9.4.3 Relationships

In a CUSTOMER and ORDER relationship for each customer one or more orders may exist. Therefore each customer occurrence should be connected to all the related order occurrences. In the relational database model, a foreign key field is included in the ORDER data structure. In each of the order occurrences relating to a certain customer, the foreign key contains the unique identification of that customer. When we are looking for orders of a particular customer then we search using foreign key column in ORDER which helps in pulling out all the related orders of particular customer.

9.4.4 No Physical Pointer

In relational model relationships between data structures is established using the concept of Foreign Key. This model does not use the concept of physical pointers like we use in Hierarchal and network model.



Did u know? Dr. E. F. Codd is the father of the relational model, stipulated the rules and put this model on a solid mathematical foundation.

Self Assessment

Multiple Choice Questions:

1. What term is used to describe a collection of information that you can use to build reports or discover facts about an environment?
 - a. Database
 - b. Web site
 - c. Wiki
 - d. Relation

2. Which of the following contains information about the structure of a database? Notes
- a. Database management system b. Data dictionary
c. Data repository d. Data warehouse
3. Which database level is closest to the users?
- a. External b. Internal
c. Physical d. Conceptual
4. is a logical (and sometimes physical) partitioning of data where multiple databases which apply to specific applications or sets of applications reside.
- a. Data Model b. Data Dictionary
c. Data Repository d. Network Model
5. Conceptual design
- a. Is a documentation technique.
b. Needs data volume and processing frequencies to determine the size of the database.
c. Involves modelling independent of the DBMS.
d. Is designing the relational model.

9.5 Types of Databases

By now you realize that information is a key business asset and that it needs to be managed, protected, and used like every other major asset. The business database that holds an business's data is the unique foundation for corporate information. Businesses are also faced with queries about how and where to contain the corporate data.

Where should a good enterprise hold its data? Really should the corporate data be kept centrally in one place? If that's the case, what are the advantages and disadvantages? What are the implications of the arrangement?

Organizations primarily adopt either of the approaches. If the entire data source is kept in one central location, this type of database is a centralized database. However, if fragments of the database are physically placed at various locations, this type of database is a distributed database. Each kind has its own benefits as well as shortcomings. Again, whether an business adopts a centralized or the distribute approach depends on the organizational setup and the information requirements. Let us review the two types.



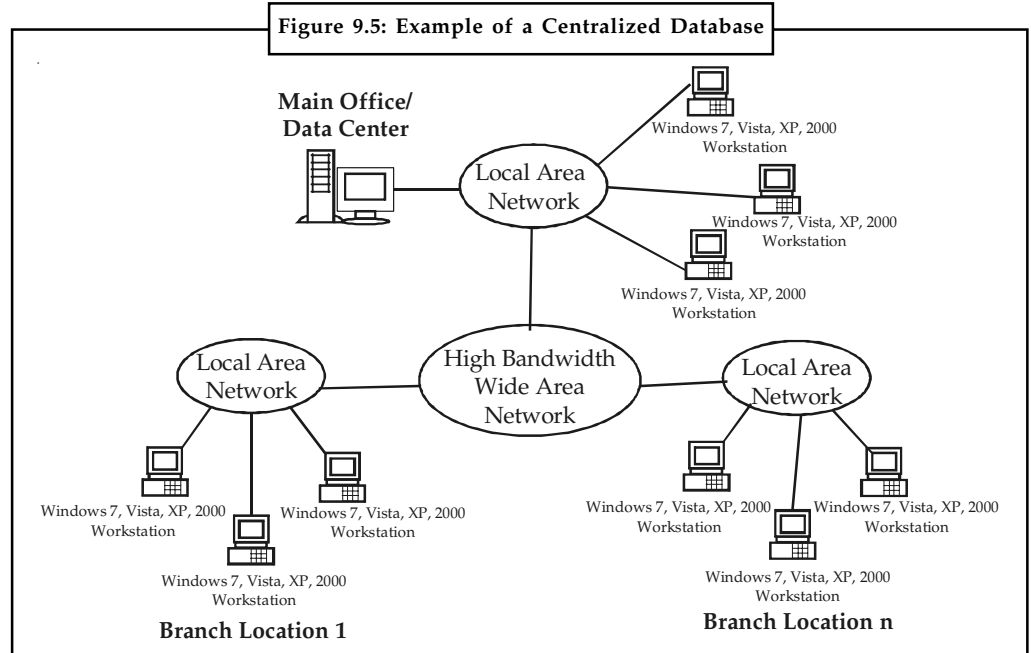
Caution The corporate data should be divided into suitable fragments and the pieces kept at different locations.

9.5.1 Centralized Database

In a centralized database, all the data of an organization is stored in a single place such as a mainframe computer or a server. Users in remote locations access the data through the Wide Area Network (WAN) using the application programs provided to access the data. The centralized database (the mainframe or the server) should be able to satisfy all the requests coming to the

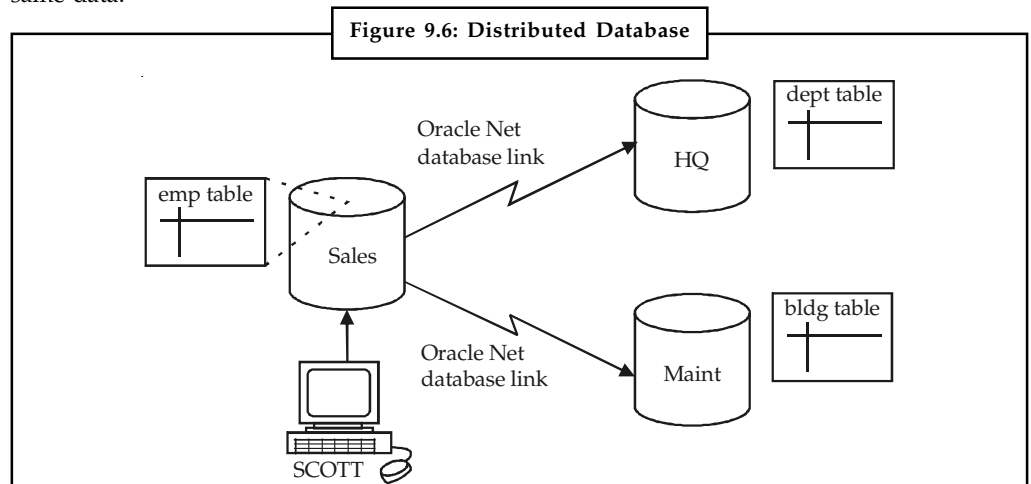
Notes

system, therefore could easily become a bottleneck. But since all the data reside in a single place it easier to maintain and back up data. Furthermore, it is easier to maintain data integrity, because once data is stored in a centralized database, outdated data is no longer available in other places.



9.5.2 Distributed Database

In a distributed database, the data is stored in storage devices that are located in different physical locations. They are not attached to a common CPU but the database is controlled by a central DBMS. Users access the data in a distributed database by accessing the WAN. To keep a distributed database up to date, it uses the replication and duplication processes. The replication process identifies changes in the distributed database and applies those changes to make sure that all the distributed databases look the same. Depending on the number of distributed databases, this process could become very complex and time consuming. The duplication process identifies one database as a master database and duplicates that database. This process is not complicated as the replication process but makes sure that all the distributed databases have the same data.



9.6 Creating a Basic Object Model

In object model, the data is stored in the form of objects, which are structures called classes that display the data within. The fields are instances of these classes.

The Rules of the Model

To design a object model we use a set of rules which are giving below:

The rules are the following ones:

Rule 1: Everything is an object.

Rule 2: Every object is an instance of a class.

Rule 3: Every class has a superclass.

Rule 4: A class defines it's behavior via public methods, and the structure of its instances via instance variables which are private to the instances.

Rule 5: Objects only communicate via message passing (i.e., method invocation). When an object receives a message, the corresponding method is looked up in the class of the receiver. If the method is not found in this class, the search continues in the receiver class's superclasses. In other words, method lookup follows the inheritance chain.

Rule 6: Every class is an instance of a metaclass. A metaclass is created automatically whenever a class is created. Metaclasses are implicit, since the programmer does not need to do anything explicit with them. A class and its metaclass are two separate classes, even though the class is an instance of the metaclass. The metaclass has exactly one instance. Metaclasses are anonymous. However, they can be referred to through the child class.

Rule 7: The metaclass hierarchy parallels the class hierarchy. For example, assume that a class X inherits from class Y. Then the class X is the sole instance of the metaclass of X and class Y is the sole instance of the metaclass of Y. In addition, the metaclass of X inherits from the metaclass of Y.

Rule 8: Every metaclass inherits from Class and Behavior.

Rule 9: Every metaclass is an instance of Metaclass.

Rule 10: The metaclass of Metaclass is an instance of Metaclass.

9.6.1 Model Precisions

Here is the classic example for implementing and understanding Model View Presenter pattern in an ASP.Net application:

Step 1: Create an interface for a business object (Model). For example:

```
public interface ICircleModel
{
    double getArea(double radius);
}
```

Step 2: Create a class for Model

```
public class CModel: ICircleModel
{
    public CModel(){}
}
```

Notes

```
public double getArea(double radius)
{
    return Math.PI * radius * radius;
}
}
```

9.6.2 Instance Variables

Instance variables are private towards the instance itself, contrary to Java or C++. Even instances of the same class cannot access the instance variables of an object if the one did not define accessors methods. Instance variables tend to be accessible by all the methods of the class and subclasses.

9.6.3 Methods

All the methods tend to be public. A method always precedes a value using the construct. When not specific explicitly using the construct, the return value of the method is the receiver of the message i.e., self.

Step 3: Create an interface for View

```
public interface IView
{
    string RadiusText { get; set;}
    string ResultText { get; set;}
}
```

Step 4: Create UI like below with a Label, TextBox and Button. Textbox for getting radius of the circle, Label is for displaying the area and Button is for calculating the Area.

```
<html xmlns="<a href="%22http://www.w3.org/1999/xhtml%22">http://www.w3.org/1999/xhtml</a">
<head runat="server">
<title>MVP-Easy Example for ASP.Net C#</title>
</head>
<body>
<form id="form1" runat="server">
<table>
<tr>
<th colspan="2">Calculate Area of Circle</th>
</tr>
<tr>
<td>Enter Radius</td>
<td><asp:TextBox ID="TextRadius" runat="server"></asp:TextBox></td>
</tr>
<tr>
<td>Result:</td>
<td><asp:Label ID="LabelResult" runat="server" ForeColor="red"></asp:Label></td>
</tr>
<tr align="right">
<td colspan="2"><asp:Button ID="ButtonResult" runat="server" Text="Get Area?"
OnClick="ButtonResult_Click" /></td>
```

```

</tr>
</table>
</form>
</body>
</html>

```

Step 5: Create a Presenter class for collecting user inputs from View and pass view details to the Model.

```

public class CPresenter
{
    IView mView;
    public CPresenter(IView view)
    {
        mView = view;
    }
    public double CalculateCircleArea()
    {
        CModel model = new CModel();
        mView.ResultText = model.getArea(double.Parse(mView.RadiusText)).ToString();
        return mView.ResultText.ToString();
    }
}

```

Step 6: Code-behind of ASPX page – View is communicating to the Model via Presenter.

```

public partial class _Default : System.Web.UI.Page, IView
{
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    protected void ButtonResult_Click(object sender, EventArgs e)
    {
        CPresenter presenter = new CPresenter(this);
        presenter.CalculateCircleArea();
    }
    public string RadiusText
    {
        get{return TextRadius.Text;}
        set{TextRadius.Text = value;}
    }
    public string ResultText
    {
        get { return LabelResult.Text; }
        set { LabelResult.Text = value; }
    }
}


```



Caution To access hidden methods of a superclass messages should be sent to super and not self.

9.6.4 Abstractness

To finish with the precisions, a class could be abstract. However, there is no dedicated construct for that. A class is regarded as abstract if one of its method contains the expression `self.subclassResponsibility` stating that subclasses have the responsibility to define the method. When such a method is executed an exception is raised.



Notes Nothing prevents you to create instance of the class having abstract methods. This will work until an abstract method will be invoked.

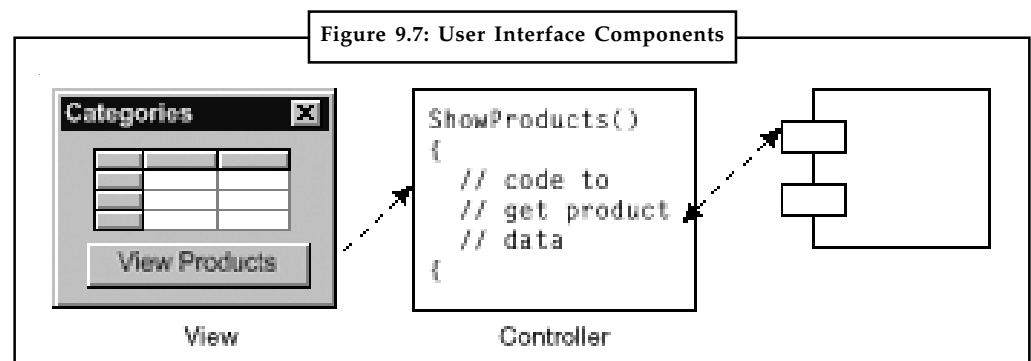
9.7 Creating the User Interface

The presentation layer contains the components that are required to enable user interaction with the application. The most simple presentation layers contain user interface components, such as Windows Forms or ASP.NET Web Forms. For more complex user interactions, you can design user process components to orchestrate the user interface elements and control the user interaction. User process components are especially useful when the user interaction follows a predictable flow of steps, such as when a wizard is used to accomplish a task.

In the case of the retail application, two user interfaces are required: one for the e-commerce website that the customers use, and another for the Windows Forms-based applications that the sales representatives use. Both types of users will perform similar tasks through these user interfaces. For example, both user interfaces must provide the ability to view the available products, add products to a shopping basket, and specify payment details as part of a checkout process. This process can be abstracted in a separate user process component to make the application easier to maintain.

9.7.1 Designing User Interface Components

You can implement user interfaces in many ways. For example, the retail application requires a Web-based user interface and a Windows-based user interface. Other kinds of user interfaces include voice rendering, document-based programs, mobile client applications, and so on. User interface components manage interaction with the user. They display data to the user, acquire data from the user, and interpret events that the user raises to act on business data, change the state of the user interface, or help the user progress in his task.



User interfaces usually consist of a number of elements on a page or form that display data and accept user input. For example, a Windows-based application could contain a DataGrid control

displaying a list of product categories, and a command button control used to indicate that the user wants to view the products in the selected category. When a user interacts with a user interface element, an event is raised that calls code in a controller function. The controller function, in turn, calls business components, data access logic components, or user process components to implement the desired action and retrieve any necessary data to be displayed. The controller function then updates the user interface elements appropriately.

9.7.2 User Interface Component Functionality

User interface components must display data to users, acquire and validate data from user input, and interpret user gestures that indicate the user wants to perform an operation on the data. Additionally, the user interface should filter the available actions to let users perform only the operations that are appropriate at a certain point in time.

User interface components:

- Do not initiate, participate in, or vote on transactions.
- Have a reference to a current user process component if they need to display its data or act on its state.
- Can encapsulate both view functionality and a controller.

When accepting user input, user interface components:

- Acquire data from users and assist in its entry by providing visual cues (such as tool tips), validation, and the appropriate controls for the task.
- Capture events from the user and call controller functions to tell the user interface components to change the way they display data, either by initiating an action on the current user process, or by changing the data of the current user process.
- Restrict the types of input a user can enter. For example, a Quantity field may limit user entries to numerical values.
- Perform data entry validation, for example by restricting the range of values that can be entered in a particular field, or by ensuring that mandatory data is entered.
- Perform simple mapping and transformations of the information provided by the user controls to values needed by the underlying components to do their work (for example, a user interface component may display a product name but pass the product ID to underlying components).
- Interpret user gestures (such as a drag-and-drop operation or button clicks) and call a controller function.
- May use a utility component for caching. In ASP.NET, you can specify caching on the output of a user interface component to avoid re-rendering it every time. If your application contains visual elements representing reference data that changes infrequently and is not used in transactional contexts, and these elements are shared across large numbers of users, you should cache them. You should cache visual elements that are shared across large numbers of users, representing reference data that changes infrequently and that is not used in transactional contexts.
- May use a utility component for paging. It is common, particularly in Web applications, to show long lists of data as paged sets. It is common to have a “helper” component that will keep track of the current page the user is on and thus invoke the data access logic component

Notes


“paged query” functions with the appropriate values for page size and current page. Paging can occur without interaction of the user process component.

When rendering data, user interface components:

- Acquire and render data from business components or data access logic components in the application.
- Perform formatting of values (such as formatting dates appropriately).
- Perform any localization work on the rendered data (for example, using resource strings to display column headers in a grid in the appropriate language for the user’s locale).
- Typically render data that pertains to a business entity. These entities are usually obtained from the user process component, but may also be obtained from the data components. UI components may render data by data-binding their display to the correct attributes and collections of the entity components, if the entity is already available. If you are managing entity data as DataSets, this is very simple to do. If you have implemented custom entity objects, you may need to implement some extra code to facilitate the data binding.
- Provide the user with status information, for example by indicating when an application is working in “disconnected” or “connected” mode.
- May customize the appearance of the application based on user preferences or the kind of client device used.
- May use a utility component to provide undo functionality. Many applications need to let a user undo certain operations. This is usually performed by keeping a fixed-length stack of “old value-new value” data for specific data items or whole entities. When the operation has involved a business process, you should not expose the compensation as a simple undo function, but as an explicit operation.
- May use a utility component to provide clipboard functionality. In many Windows-based applications, it is useful to provide clipboard capabilities for more than just scalar values – for example, you may want to let your users copy and paste a full customer object. Such functionality is usually implemented by placing XML strings in the Clipboard in Windows, or by having a global object that keeps the data in memory if the clipboard is application-specific.

9.7.3 New System Design Process

An examination of most business solutions based on a layered component model reveals several common component types. Figure 9.8 shows these component types in one comprehensive illustration.



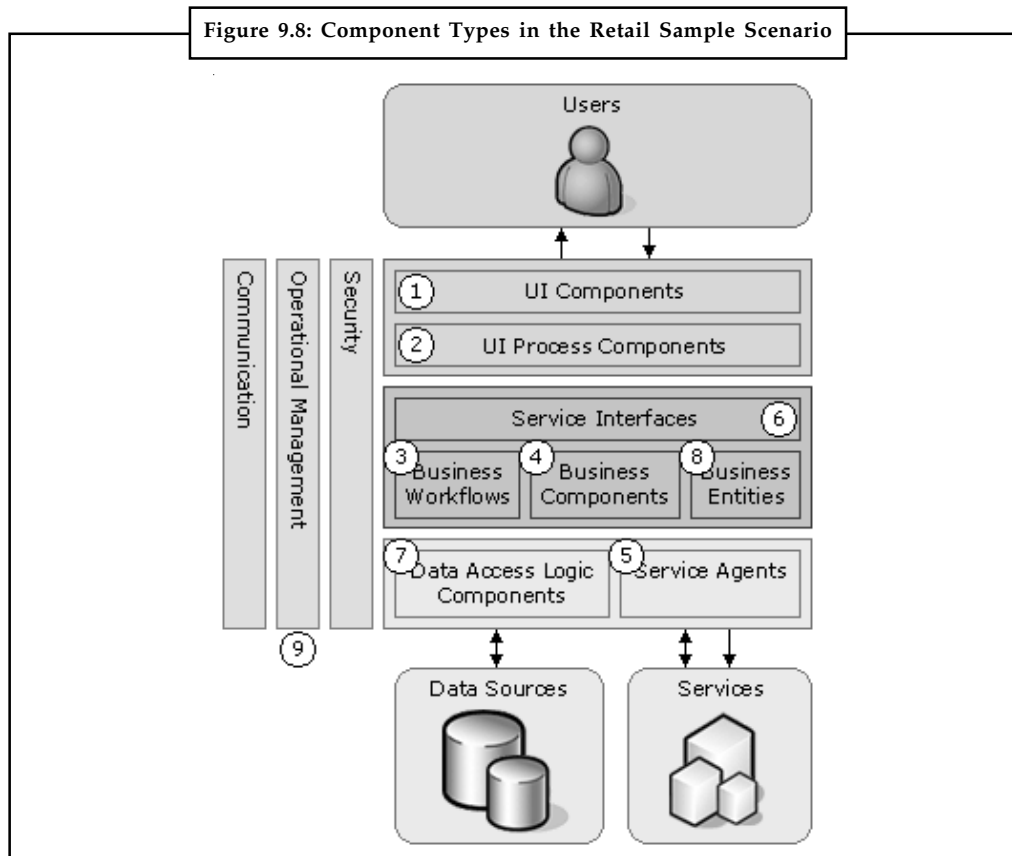
Notes The term component is used in the sense of a piece or part of the overall solution. This includes compiled software components, such as Microsoft .NET assemblies, and other software artifacts such as Web pages

Although the list of component types shown in Figure 9.8 is not exhaustive, it represents the common kinds of software components found in most distributed solutions. These component types are described in depth throughout the remainder of this unit.

The component types identified in the sample scenario design are:

Notes

1. **User interface (UI) components.** Most solutions need to provide a way for users to interact with the application. In the retail application example, a Web site lets customers view products and submit orders, and an application based on the Microsoft Windows® operating system lets sales representatives enter order data for customers who have telephoned the company. User interfaces are implemented using Windows Forms, Microsoft ASP.NET pages, controls, or any other technology you use to render and format data for users and to acquire and validate data coming in from them.



2. **User process components.** In many cases, a user interaction with the system follows a predictable process. For example, in the retail application you could implement a procedure for viewing product data that has the user select a category from a list of available product categories and then select an individual product in the chosen category to view its details. Similarly, when the user makes a purchase, the interaction follows a predictable process of gathering data from the user, in which the user first supplies details of the products to be purchased, then provides payment details, and then enters delivery details. To help synchronize and orchestrate these user interactions, it can be useful to drive the process using separate user process components. This way the process flow and state management logic is not hard-coded in the user interface elements themselves, and the same basic user interaction “engine” can be reused by multiple user interfaces.
3. **Business workflows.** After the required data is collected by a user process, the data can be used to perform a business process. For example, after the product, payment, and delivery details are submitted to the retail application, the process of taking payment and arranging delivery can begin. Many business processes involve multiple steps that must be performed

Notes

in the correct order and orchestrated. For example, the retail system would need to calculate the total value of the order, validate the credit card details, process the credit card payment, and arrange delivery of the goods. This process could take an indeterminate amount of time to complete, so the required tasks and the data required to perform them would have to be managed. Business workflows define and coordinate long-running, multi-step business processes, and they can be implemented using business process management tools such as BizTalk Server Orchestration.

4. **Business components.** Regardless of whether a business process consists of a single step or an orchestrated workflow, your application will probably require components that implement business rules and perform business tasks. For example, in the retail application, you would need to implement the functionality that calculates the total price of the goods ordered and adds the appropriate delivery charge. Business components implement the business logic of the application.
5. **Service agents.** When a business component needs to use functionality provided in an external service, you may need to provide some code to manage the semantics of communicating with that particular service. For example, the business components of the retail application described earlier could use a service agent to manage communication with the credit card authorization service, and use a second service agent to handle conversations with the courier service. Service agents isolate the idiosyncrasies of calling diverse services from your application, and can provide additional services, such as basic mapping between the format of the data exposed by the service and the format your application requires.
6. **Service interfaces.** To expose business logic as a service, you must create service interfaces that support the communication contracts (message-based communication, formats, protocols, security, exceptions, and so on) its different consumers require. For example, the credit card authorization service must expose a service interface that describes the functionality offered by the service and the required communication semantics for calling it. Service interfaces are sometimes referred to as *business facades*.
7. **Data access logic components.** Most applications and services will need to access a data store at some point during a business process. For example, the retail application needs to retrieve product data from a database to display product details to the user, and it needs to insert order details into the database when a user places an order. It makes sense to abstract the logic necessary to access data in a separate layer of data access logic components. Doing so centralizes data access functionality and makes it easier to configure and maintain.
8. **Business entity components:** Most applications require data to be passed between components. For example, in the retail application a list of products must be passed from the data access logic components to the user interface components so that the product list can be displayed to the users. The data is used to represent real-world business entities, such as products or orders. The business entities that are used internally in the application are usually data structures, such as DataSets, DataReaders, or Extensible Markup Language (XML) streams, but they can also be implemented using custom object-oriented classes that represent the real-world entities your application has to work with, such as a product or an order.
9. **Components for security, operational management, and communication:** Your application will probably also use components to perform exception management, to authorize users to perform certain tasks, and to communicate with other services and applications.

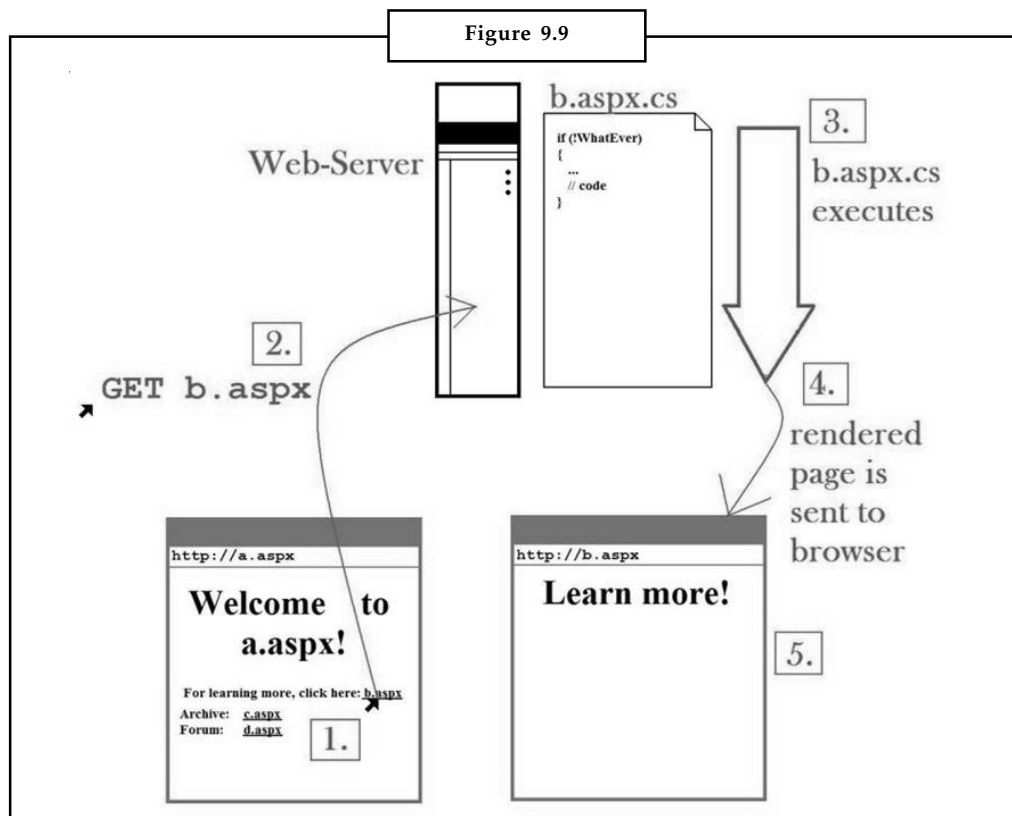
9.7.4 Methodology in Action

Notes

Once the system is made, the system information is going to be stored in the system database. The first step to use the system would be to retrieve the form info and rebuild it for the user. The next step is to get the data from the user as well as store the information within the database.

Form Displaying/Rendering Process

This process includes presenting the form to the user independent from the platform depending on the environment (See Figure 9.9).



One of the important results of this process is an XForm document that has the precise declaration of the original make use of cases in the design phase. This announcement is abstract as well as platform independent. As we saw in design phase, parser engine is accountable for converting declarative model of designed form to OO model. The other obligation of parser engine is to create the actual user interface in the stored information and in accordance to end user's platform.

The parser gets form's declarative definition in XForms format and then regenerates the user interface for the target platform from the logic and the user interface design information. For instance for regular web browsers it creates HTML codes while for PDAs it might generate WML codes or the windows form for PC desktop applications. To perform this task, parser uses the following parts of the system or information:

XSL files and XSLT files

These files have the main role in generating the interface and the look of the form for different platforms.

Notes

Database

The parser may need to determine a connection to the actual database to get some information about the sub forms or sibling forms and use them to generate the code.

Validation

In add-on to the user interface code, some auxiliary-codes maybe required to control/validate the input data like by injecting javascript control codes into the HTML forms.

Web Services

If a form requires to be connected to a different distributed system, then web services is the access method to such system.

In this way, the form will be displayed and is ready for the following stage.

Processing and Storing Data

The user fills the shown form and returns it to the system. Figure 9.9 shows the steps evidently (it assumed the form is an HTML one).

After filling the form by the user, the data will be sent to parser engine box. One of the other accountabilities of this box is converting user's data to the proper format that would end up being tangible by the system (it has different responsibilities in "designing form" and "displaying form" phases). It will recognize the relations between delivered data and form's fields through referring to the form's definition document (in XForms structure). This is the first part of importing user's data to program as a structured one. The next phase, which is so important, is actually data validation. In "displaying form" process, the validation was completed in partial mode but the main activity would be done following receiving user's data. In this step, user's data should be validated against the form's logic. In other words the declarative statements that would be supplied in form's logic description would be checked against the user's data to verify if they're constant with them or not. But as we mentioned previously, it may be impossible in order to validate the user's data at the same place where we show the form to the user. Then it is required for connecting to the other part of same system (or even others) to verify correctness of user's data. So as you see in the Figure 9.9, the validation box may connect with database or use some web services (to connect to others in distributed solutions) to see if the user's data are in line with other data or not. Observe that we can connect to other systems with different solutions like using RPC (Remote Procedure Call), CORBA (Common Object Request Broker Architecture), DCOM (Distributed COM) and so forth that are not mentioned on this diagram.

Here we have the consistency issue between distributed databases that should be resolved by the DBAs (Database Administrators). After user's data is validated and relations between data and form's components are found, the related objects would be created from the classifiers that were made in "designing form" phase. In other words, the user's data would be stored in some objects (within OO concepts). Therefore the user's data is able to circulate through the system easily as well as being processed wherever is needed because its format is identifiable for the code generator system. Briefly, the objects are the real instances of form's data in the format of the classes which present the form's schema.

Lastly these created objects should end up being stored in database. This procedure would be done by "Data Access Methods" part. This component receives the user's data in object format and then stores them in database as output result. This part is responsible for mapping these

received objects to such database objects that were created in “design form” phase and store received objects data into these database objects. Observe that the purpose for this mapping in current phase is because of input objects of this container are OO objects whereas the equivalent objects in database are Database.

Objects (like tables, views, stored procedures and so on). The received objects would be mapped to associated ones in database system as a result. Moreover, “Data Access Methods” routine ought to store related user’s data information to make it available for future access and also with regard to specifying the state of user’s data in system. This info usually comprises of one unique identification number, date of filling form, related actors and so on.

Representing the Filled Forms

This stage consists of representation of a designed form with data samples. In other words, it is the final stage of a designed form including data. As mentioned before we can divide this stage into two major steps:

- Form Representation
- Injecting form’s data to the rendered form

At the first step, design part of the form within XForms document (declarative, platform dependent) is converted into as implemented model in target platform. Certainly in this, conversion would be carried out by XSLT. Also for preparing required form data which are determined by other parts of the system such as other forms or even other software systems, we make use of web services. The second stage includes the following steps: depending on XForms, related entities in database are identified. Data Access Component does this task, which has been utilized in data store step, too. This element gets the XForms and complementary information as well as retrieves the filled forms data from the database.

Major parts of the gathered data in previous step are unprocessed. So it is necessary to retrieve more data from data source or web services to put together the rendered form’s data.

The DAC, creates the OO structure for rendered data by using XForms structure. After this phase the form’s structure would be accessible in system as a set of controls/components dependent on platform.

The rendered controls would be transfer to the parser engine. This engine:

- Renders the form for platform.
- Maps the prepared data to controls.



Case Study

Replacing CRM System Generates Large Savings and New Efficiencies for Green Company

Safety-Kleen Systems used several technologies, including Salesforce.com, to manage customer relationships and sales. Because of high licensing costs, employees outside of the sales force lacked access to Salesforce.com. It was also difficult to integrate it with other tools or adjust it to reflect evolving business needs. Working with disconnected software tools, however, caused inefficiencies and resulted in unreliable information. The company replaced Salesforce.com with Microsoft Dynamics CRM and integrated it with other systems in a very fast implementation. Giving many more employees access to the

Contd....

Notes

solution, Safety-Kleen Systems still saves more than US\$500,000 per year in licensing. With a single, unified tool to work with customers, sales activities are vastly more efficient. Decision makers have a single, reliable source of current information to direct the company.

Situation

With the goal to “make green work,” Safety-Kleen Systems serves customers in more than 270,000 locations throughout the United States, Canada, and Puerto Rico. The company’s 4,500 employees work within a large network of branch offices, oil collection and processing centers, recycling locations, oil refineries, recycling facilities, and manufacturing operations. These offices run the largest re-refining capacity of used motor oil in North America and transform 400 million gallons of waste into 300 million gallons of usable materials per year. They also manufacture cleaning and oil products and provide services relevant to waste management, emergency response, compliance, and equipment. In addition, Safety-Kleen Systems teams develop innovative equipment for specialized cleaning needs in many different industries.

Technology Restrictions and Cost Concerns

For Safety-Kleen Systems, the ability to serve customers effectively and with optimal responsiveness is paramount. The company employs a large organization of field sales and service professionals, including a national account team. To support the field, Safety-Kleen Systems used Salesforce.com as the technology tool for managing customer relationships and communications. However, the software presented certain limitations and challenges. As Carla Rolinc, VP of IT at Safety-Kleen Systems, explains, “Because of cost concerns, we had to limit the amount of licenses we acquired, which meant not all of the people who would benefit from a customer relationship management system had access to it. We also found it difficult to adjust Salesforce.com for our business requirements or integrate it with our other systems. For that reason, sales team members had to navigate five different systems to do their work and spent as much as 40 percent of their time on administrative duties. This was a drain on their effectiveness, especially if you consider they are all mobile, connecting with company resources from remote locations.”

Cost issues also were an important consideration for the company’s IT resources. “We found that IT professionals who can program for Salesforce.com command a much higher hourly rate than those with Microsoft .NET skills,” says Rolinc. “We already had Microsoft technology skills on our team but incurred significant expense whenever we needed somebody to work on Salesforce.com.”

Process Inefficiencies and Lacking Integrations

In addition to Salesforce.com, the field accessed discrete software tools to create and record customer quotes, manage orders, facilitate waste services, pass on customer needs and opportunities to the right person if the first responder was unable to address them, and ensure that national accounts received the right level of responsiveness. Many salespeople used Microsoft Outlook to track appointments and communicate with customers through email.

The company also owned a stand-alone mapping application to help route sales leads to the right person. Cathy King, Director of IT Applications at Safety-Kleen Systems, says, “In trying to overcome the restrictions we experienced with Salesforce.com, we created and acquired additional software tools and came up with workarounds to meet specific business needs. However, inefficiencies were common, information was not always consistent or reliable, and accountability could be obscure. For example, it was possible to close out a customer request and resolve customer issues in the software tools without taking the actual, appropriate actions.”

Contd....

Seeking Change

Salesforce.com and SAP, the company's system of record, exchanged updates in a daily batch process. People who used either one of these systems lacked access to the other and could not be certain that information they reviewed was not outdated. Company managers and the field called for more meaningful information; consolidated, reliable reporting; and more efficient ways to work. When Safety-Kleen Systems approached the end of its three-year contract with Salesforce.com, Rolinc and company leadership decided to find another customer relationship management solution instead of renewing for another three-year term.

Questions:

1. What are the technological restrictions in Safety-Kleen Systems?
2. What role does cost issue plays in Companies IT resources?

Notes

Self Assessment

Multiple Choice Questions:

6. In a one-to-many relationship, the entity that is on the one side of the relationship is called a(n) entity.
 - a. Parent
 - b. Child
 - c. Instance
 - d. subtype
7. The property/properties of a database is/are:
 - a. It is an integrated collection of logically related records.
 - b. It consolidates separate files into a common pool of data records.
 - c. Data stored in a database is independent of the application programs using it.
 - d. All of the above.
8. A..... is a conceptual representation of the data structures that are required by a database.
 - a. Functional Model
 - b. Network Model
 - c. Data Model
 - d. Business Model
9. Relational Algebra is:
 - a. Data Definition Language
 - b. Meta Language
 - c. Procedural Query Language
 - d. None of the above
10. Which of the following is another name for weak entity?
 - a. Owner
 - b. Child
 - c. Dominant
 - d. All of the Above
11. Hierarchical model is also called
 - a. Tree Structure
 - b. Plex Structure
 - c. Normalize Structure
 - d. Table Structure
12. Which of the following is record based logical model?
 - a. Network Model
 - b. Object Oriented Model
 - c. E-R Model
 - d. None of these

Notes

13. Given the basic ER and relational model which of the following is incorrect?
 - a. An entity can have more than one attribute.
 - b. An attribute of an entity can be composite.
 - c. In row of a relational table, an attribute have more than one value.
 - d. In row of a relational table, an attribute can have exactly one value or a NULL value.
14. A primary key if combined with a foreign key creates
 - a. Parent – Child relationship between the tables that connect them.
 - b. Many to many relationship between the tables that connect them.
 - c. Network model between the tables that connect them.
 - d. None of the above.
15. Which are the two ways in which entities can participate in a relationship?
 - a. Passive and active
 - b. Total and partial
 - c. Simple and Complex
 - d. All of the above

9.8 Summary

- A data model is a conceptual representation of the data structures that are required by a database.
- The data structures include the data objects, the associations between data objects, and the rules which govern operations on the objects.
- The data model has two outputs – Entity relationship Diagrams and Data Document.
- All data in the database reside in a data repository.
- The data repository contains the actual data.
- Data Dictionary is a DBMS component which store metadata.
- Database software is the phrase used to describe any software that is designed for creating databases and managing the information stored in them.
- The database approach provides for data abstraction.
- Abstraction is the process of recognizing and focusing on important characteristics of a situation or object and leaving/filtering out the unwanted characteristics.
- Hierarchy model should show that an assembly contains subassemblies, a subassembly contains parts, and a part contains subparts. Immediately you can observe that this data model must be hierarchical in nature, diagramming the assembly at the top with subassembly, part, and subpart at successive lower levels.
- Hierarchical data model represents well any business data that inherently contains levels one below the other.
- The relationship between each pair of data structures at levels after that to each other is a parent – child relationship.
- Physical pointers link records of the parent segments to those of the child segments by means of parent-child relationship.
- The network data model expresses relationships between two record types by designating one as the owner record type and the other as the member record type.

- A relational database allows the definition of data structures, storage and retrieval operations and integrity constraints.
- In a centralized database, all the data of an organization is stored in a single place such as a mainframe computer or a server.
- In a distributed database, the data is stored in storage devices that are located in different physical locations.
- User interface components must display data to users, acquire and validate data from user input, and interpret user gestures.
- Once the system is built, the system information will be stored in the system database.
- Database Factory is responsible for transformation of generated structures to the equivalent model in a database.

9.9 Keywords

Database: It is an ordered collection of related data elements intended to meet the information needs of an organization and designed to be shared by multiple users.

Hierarchical Data Model: Represents well any business data that inherently contains levels one below the other.

Network Data Model: It is more representative of real-world information requirements than the hierarchical model.

Parser: May need to establish a connection to the database to get some information about the sub-forms or sibling forms and use them to generate the code.

Physical Pointers Link: It occurrences of an owner record type with the corresponding occurrences of the member record type.

Relation: It is a two-dimensional table of data observing relational rules.

User Interface (UI): Forms are essential in capturing data requirements, UI prototype can be used by the developer as a source for developing the classes and objects.

Validation: In addition to the user interface code, some auxiliary-codes maybe needed to control validate the input data like by injecting javascript control codes into the HTML forms.



Lab Exercise

1. Construct Lookup of messages sent to self starts in the class of the receiver.
2. Create a basic object model which follows all the rules.

9.10 Review Questions

1. What is the difference between logical data model and physical data model?
2. How are Data Models Used in Practice?
3. You are hired by one client for designing a database for Air line transaction. The data base has to capture the detailed level of data related to the tickets booked by the user and the updations made by them with timestamp. Which database model do you prefer?

Notes

4. State the differences between the hierarchical database model and the relational database model.
5. What is the difference between Database and Relational Database?
6. Explain the term Physical Pointers.
7. Explain the difference between Centralized and distributed Database.
8. What is Relational Database Model? List out the properties of Relational Database Model.
9. What is Data abstraction? Explain with example.
10. Which type of database management system uses physical pointers to connect table?

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (a) | 2. (b) | 3. (a) | 4. (c) |
| 5. (c) | 6. (a) | 7. (d) | 8. (c) |
| 9. (b) | 10. (a) | 11. (a) | 12. (a) |
| 13. (c) | 14. (a) | 15. (b) | |

9.11 Further Readings



Books

Creating User Interfaces by Demonstration, by Brad A. Myers.



Online links

http://www.iam.unibe.ch/_ducasse/FreeBooks.html.

Unit 10: Web Services

Notes

CONTENTS

Objectives

Introduction

10.1 Web Services

10.2 XML Web Services

10.2.1 SOAP

10.2.2 WSDL

10.2.3 UDDI

10.2.4 General Principles of Web Services XML

10.3 Using for Web Services

10.4 Summary

10.5 Keywords

10.6 Review Questions

10.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Define the XML Web Services
- Explain the Uses for Web Services

Introduction

With the emergence of computer networks, the paradigm of distributed computing was born. Applications were split first into two parts with one part, the client, initiating a distributed activity, and the other part, the server, carrying out that activity. This decentralization minimized bottlenecks by distributing the workload across multiple systems. It provided flexibility to application design formerly unknown on centralized hosts. But this two-tier architecture had its limits. For failover and scalability, issues a third tier was introduced, separating an application into a presentation part, a middle tier containing the business logic, and a third tier dealing with the data. This three-tier model of distribution has become the most popular way of splitting applications. It makes application systems scalable. The foundation for the communication between the distributed parts of an application is the remote procedure call (RPC). To keep developers from the low-level tasks like data conversion or the byte order of different machines, a new layer of software hit the market. This middleware masks the differences between various kinds of hosts. It sits on top of the host's operating system and networking services and offers its services to the applications above.

The first middlewares, like DCE, were based on a procedural programming model and were superseded by the introduction of the object oriented programming model by middlewares like CORBA®, DCOM, or RMI, which are the most popular middlewares at present. Roughly speaking,

Notes

Web Services are applications that can be published, located, and invoked across the Internet. Typical examples include:

- Getting stock price information.
- Obtaining weather reports.
- Making flight reservations.

These Web Services may use other Web Services in order to perform their task.

So far, there's no difference between a Web Service and a server in a distributed application. The differences lie in the underlying layers for performing the application logic and data manipulation.

The points mentioned in the conclusion of the last part are some of the main reasons why those middlewares don't qualify for Web Services. On the worldwide Web, there are heterogeneous environments on both the server and the client side, and it is not known in advance which kind of middleware each side of the connection uses. Thus, a new approach is needed for Web Services "Web Services are encapsulated, loosely coupled contracted functions offered via standard protocols" where:

- "Encapsulated" means the implementation of the function is never seen from the outside.
- "Loosely coupled" means changing the implementation of one function does not require change of the invoking function.
- "Contracted" means there are publicly available descriptions of the function's behavior, how to bind to the function as well as its input and output parameters.

10.1 Web Services

A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

Web Service messages are formatted as XML, a standard way for communication between two incompatible system. And this message is sent via HTTP, so that they can reach to any machine on the Internet without being blocked by firewall.

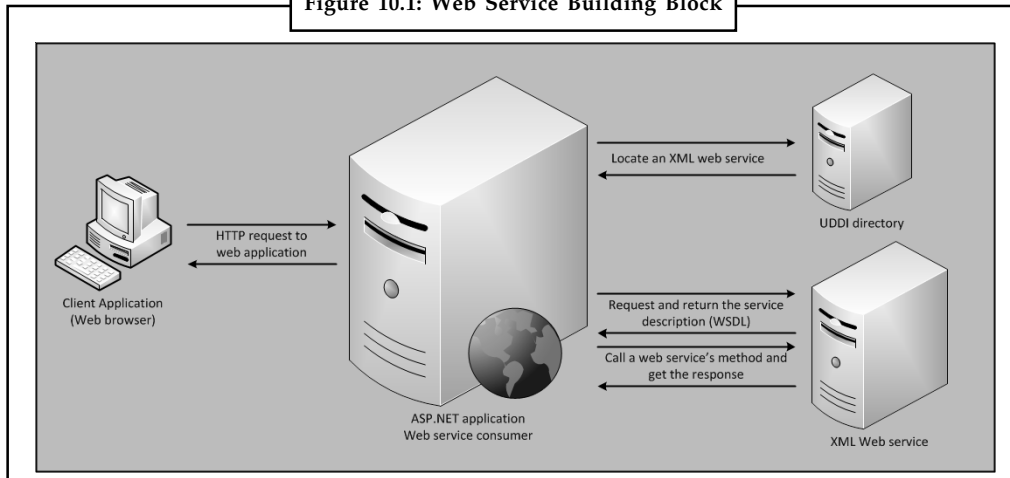


Example: Web Service

- **Weather Reporting:** You can use Weather Reporting web service to display weather information in your personal website.
- **Stock Quote:** You can display latest update of Share market with Stock Quote on your web site.
- **News Headline:** You can display latest news update by using News Headline Web Service in your website.

Web Services however use XML on top of HTTP. Thus, no problems with firewalls will occur. Usually the firewalls don't block the HTTP port. If you look at the definition, you will see that Web Services don't have to use HTTP. The use of SMTP or FTP can be considered. XML is a widely accepted format for exchanging data and its corresponding semantics. It is a fundamental building block for nearly every other layer that is used for Web Services.

Figure 10.1: Web Service Building Block



The concepts used in Web Services hinge on the three following acronyms:

- SOAP (Simple Object Access Protocol) is a standard XML based protocol that communicated over HTTP. We can think of SOAP as message format for sending messages between applications using XML. It is independent of technology, platform and is extensible too.
- Web Services Description Language document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages, which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations.
- Universal Description, Discovery and Integration (UDDI) is a directory service where businesses can register and search for Web services. UDDI behaves like a Web Service itself, its methods being called using the SOAP protocol.

A Web Service Example

Following is our First Web Service example which works as a service provider and exposes two methods (add and SayHello) as Web Services to be used by applications. This is a standard template for a Web Service. .NET Web Services use the .asmx extension. Note that a method exposed as a Web Service has the WebMethod attribute. Save this file as FirstService.asmx in the IIS virtual directory (as explained in configuring IIS; for example, c:\MyWebServices).

FirstService.asmx

```
<%@ WebService language="C" class="FirstService" %>

using System;
using System.Web.Services;
using System.Xml.Serialization;

[WebService(Namespace="http://localhost/MyWebServices/")]
public class FirstService : WebService
{
    [WebMethod]
    public int Add(int a, int b)
    {
        return a + b;
    }
}
```

Notes

```
[WebMethod]
public String SayHello()
{
    return "Hello World";
}
}
```

To test a Web Service, it must be published. A Web Service can be published either on an intranet or the Internet. We will publish this Web Service on IIS running on a local machine. Let's start with configuring the IIS.

- Open Start->Settings->Control Panel->Administrative tools->Internet Services Manager.
- Expand and right-click on [Default Web Site]; select New ->Virtual Directory.
- The Virtual Directory Creation Wizard opens. Click Next.
- The "Virtual Directory Alias" screen opens. Type the virtual directory name—for example, MyWebServices – and click Next.
- The "Web Site Content Directory" screen opens. Here, enter the directory path name for the virtual directory – for example, c:\MyWebServices – and click Next.
- The "Access Permission" screen opens. Change the settings as per your requirements. Let's keep the default settings for this exercise. Click the Next button. It completes the IIS configuration. Click Finish to complete the configuration.

To test that IIS has been configured properly, copy an HTML file (for example, x.html) in the virtual directory (C:\MyWebServices) created above. Now, open Internet Explorer and type <http://localhost/MyWebServices/x.html>. It should open the x.html file. If it does not work, try replacing localhost with the IP address of your machine. If it still does not work, check whether IIS is running; you may need to reconfigure IIS and Virtual Directory.

To test our Web Service, copy FirstService.asmx in the IIS virtual directory created above (C:\MyWebServices). Open the Web Service in Internet Explorer (<http://localhost/MyWebServices/FirstService.asmx>). It should open your Web Service page. The page should have links to two methods exposed as Web Services by our application. Congratulations; you have written your first Web Service!!!

Client/Server Model

Web Services use the conventional client server model: a, possibly remote, server provides resources that are requested and consumed by a client whom the user interacts with. A simple example of this is surfing around the web where the web browser is a client that displays pages and allows the user to interact with the pages, and the web server(s) provide the data for all pages (HTML, images, etc.) when it's requested by the client.

Synchronous and Asynchronous Access

Synchronous means that every time a client accesses a Web service application, the client receives a SOAP response. Synchronous is request – response operation. Synchronous services are designed when client applications require a more immediate response to a request. Web services that rely on synchronous communication are usually Remote Procedure Call (RPC)-oriented.

Asynchronous means that the client which invokes a Web service, does not or can not wait for a response. Thus, asynchronous is one-way operation. The client sends a request in the form of an

XML message. The Web service receives the message and processes it, sending the results when it completes its processing.



Example: Creating Web Service in .Net

This Web Service will retrieve CustomerList Country Wise and return as dataset to client application for display.

Step 1: Create a Web Service Application by File > New > Web Site > Asp.net Web Services Named the web service, for example here I have chosen name "WSGetCustomerCountryWise".

Step 2: Rename the default Service.asmx file to proper name, you also need to switch design view and change the class name with the same name you use to rename the service.asmx.



Example: "WSGetCustomerCountryWise.asmx" and switch to design view and change the class="Service" to class="WSGetCustomerCountryWise".

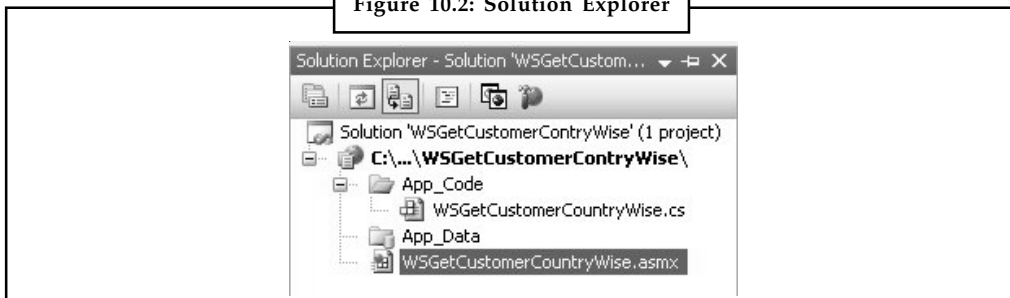
Step 3: Rename the Service. CS File to proper name, you need to change the class name and constructor name too.



Example: "WSGetCustomerCountryWise.CS" and switch to code view and change the class and constructor name to "WSGetCustomerCountryWise".

After three steps your solution explorer looks as shown in Figure 10.2:

Figure 10.2: Solution Explorer



Step 4: Create a Logic for Web Service:

- Create a Method "GetCustomerCountryWise".



Notes You need to specify [WebMethod] before method definition, if you want it to be accessible public, otherwise the method would not be accessible remotely.

- Specify proper argument and return type for method in web service.
- It is also good practise to specify the use "Description" attribute to tell what method is meant for.



Example: Here I need to access data of northwind customers and want to return customer list country wise, so add namespace for

```
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
```

Notes

[WebMethod(Description="It will generate Customer List, Country Wise")] public System.Xml.XmlElement GetCustomerCountryWise(stringsCountry)

```

{
string sConn = ConfigurationManager.ConnectionStrings["connStr"].ToString();

string sSQL = "select CustomerId, CompanyName, ContactTitle, City " +
" from Customers where country = '" + sCountry + "'";

SqlConnection connCustomer = new SqlConnection(sConn);

DataSet dsCustomer = new DataSet();

SqlDataAdapter daCustomer = new SqlDataAdapter(sSQL, sConn);

daCustomer.Fill(dsCustomer, "Customers");

//Known bug while return "DataSet" is "Data source is an invalid type.
It must be either an IListSource, IEnumerable, or IDataSource."
//For more details on Error: http://support.microsoft.com/kb/317340

//So to access data we need to make use of XmlElement.

// Return the DataSet as an XmlElement.
System.Xml.XmlDataDocument xdd = new System.Xml.XmlDataDocument(dsCustomer);
System.Xml.XmlElement docElem = xdd.DocumentElement;
return docElem;
}

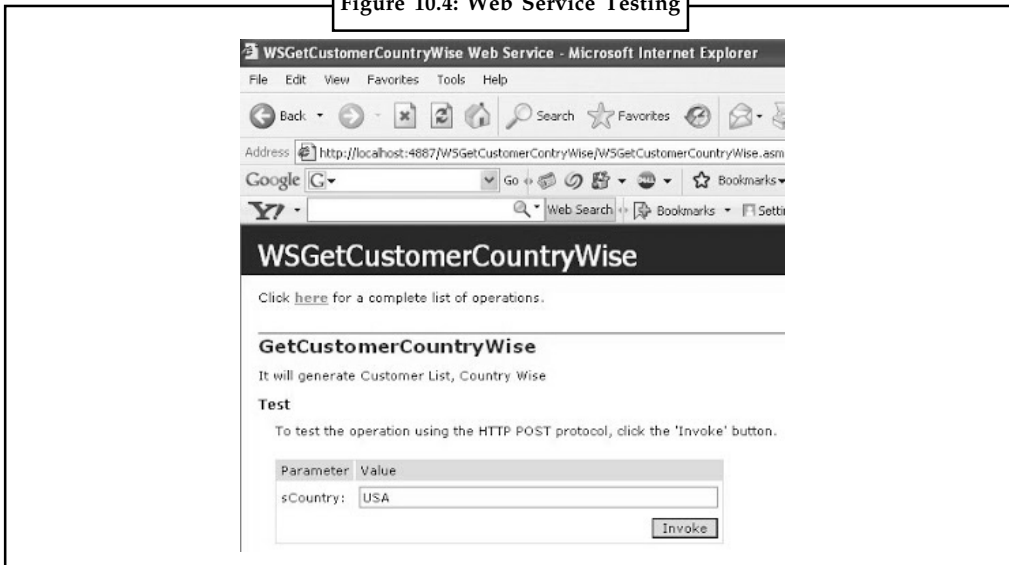
```

Step 5: Build Web Service and Run the Web Service for testing by pressing F5 function key.

Figure 10.3: Web Service Testing



Figure 10.4: Web Service Testing



By pressing “Invoke” button will generate XML File.

So you are done creating web service application.



Example: Testing Web Service in .Net

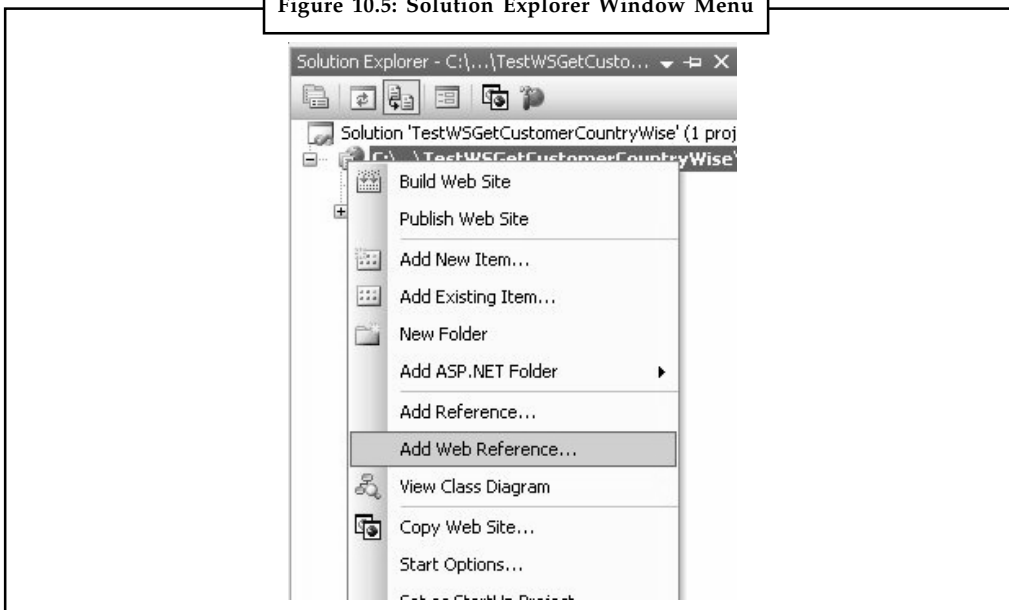
This Web Service will display the information which had been retrieved from Remote computer by accessing public method “GetCustomerCountryWise”.

Step 1: Create a Test Web Site by File > New > Web Site > Asp.net Web Site Named the web site, for example here I have choosen name “TestGetCustomerCountryWise”.

Step 2: Displaying data in gridview, so drag the gridview on to the form.

Step 3: Right Click Solution Explorer and Choose “Add Web Reference”.

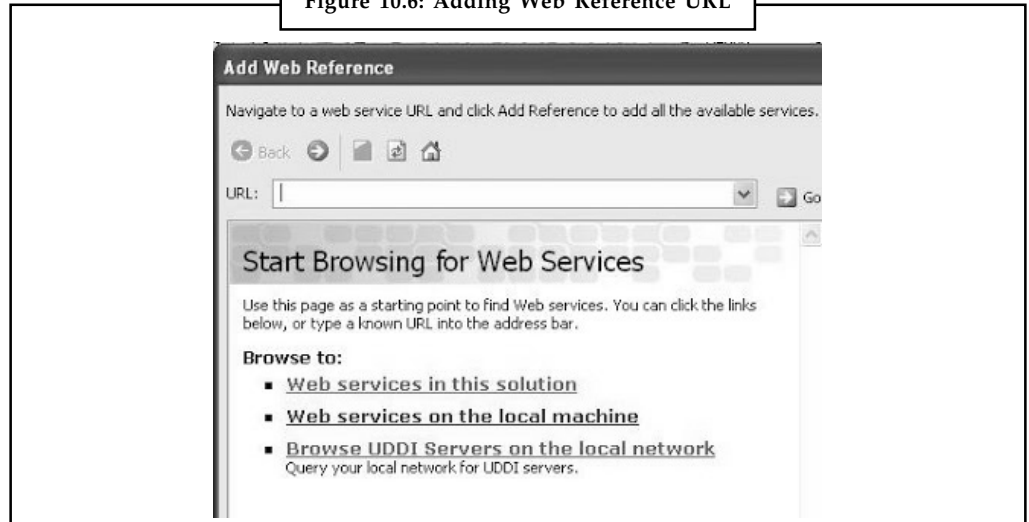
Figure 10.5: Solution Explorer Window Menu



Notes

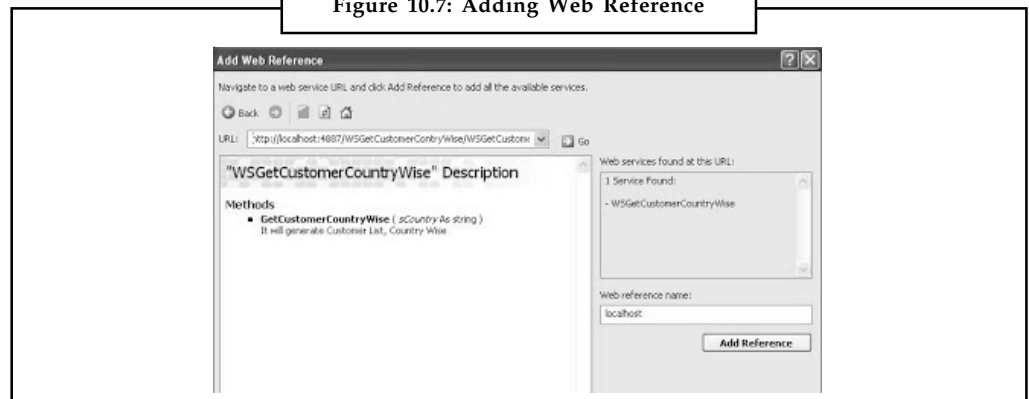
Step 4: Choose the option Web Service on the local machine or you can enter the .WSDL File address directly in URL space and press Go button.

Figure 10.6: Adding Web Reference URL



Step 5: Press “Add Reference button”.

Figure 10.7: Adding Web Reference



Step 6: Writing Code for Displaying data in GridView.

Here note: I have Pass “USA” as parameter in Country Field.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        //Create object of WSGetCustomerCountryWise Object
        localhost.WSGetCustomerCountryWise objGetCustomerCountryWise
        = new localhost.WSGetCustomerCountryWise();

        DataSet dsCustomer = new DataSet();

        // Get the data from Webservice.
        XmlElement elem = objGetCustomerCountryWise.GetCustomerCountryWise("USA");

        // Load the XML to the Typed DataSet that you want.
        XmlNodeReader nodeReader = new XmlNodeReader(elem);
```

```

dsCustomer.ReadXml(nodeReader, XmlReadMode.Auto);

GridView1.DataSource = dsCustomer;

GridView1.DataBind();
}
}

```

Notes

Step 7: Output as data displayed in GridView.

Figure 10.8: Output of the Program

CustomerId	CompanyName	ContactTitle	City
GREAL	Great Lakes Food Market	Marketing Manager	Eugene
HUNGC	Hungry Coyote Import Store	Sales Representative	Elgin
LAZYK	Lazy K Kountry Store	Marketing Manager	Walla Walla
LETSS	Let's Stop N Shop	Owner	San Francisco
LONEP	Lonesome Pine Restaurant	Sales Manager	Portland
OLDWO	Old World Delicatessen	Sales Representative	Anchorage
RATTC	Rattlesnake Canyon Grocery	Assistant Sales Representative	Albuquerque
SAVEA	Save-a-lot Markets	Sales Representative	Boise
SPLIR	Splirt Rail Beer & Ale	Sales Manager	Lander
THEBI	The Big Cheese	Marketing Manager	Portland
THECR	The Cracker Box	Marketing Assistant	Butte
TRAIH	Trail's Head Gourmet Provisioners	Sales Associate	Kirkland
WHITC	White Clover Markets	Owner	Seattle

Few facts about Web Service in .Net:

- Each Response from web service is a new object, with a new state.
- Web Service are asynchronous because the request object from the client application and the response object from the web service are unique SOAP envelopes that do not require shared connection.
- This allow client application and the web service to continue processing while the interaction is ongoing.
- Instead of a user interface, it provides a standard defined interface called a contract.

10.2 XML Web Services

An XML Web service is a programmable entity that provides a particular element of functionality, such as application logic, and is accessible to any number of potentially disparate systems using ubiquitous Internet standards, such as XML and HTTP. XML Web services depend heavily upon the broad acceptance of XML and other Internet standards to create an infrastructure that supports application interoperability at a level that solves many of the problems that previously hindered such attempts.

Notes

An XML Web service can be used internally by a single application or exposed externally over the Internet for use by any number of applications. Because it is accessible through a standard interface, an XML Web service allows heterogeneous systems to work together as a single web of computation.

Instead of pursuing the generic capabilities of code portability, XML Web services provide a viable solution for enabling data and system interoperability. XML Web services use XML-based messaging as a fundamental means of data communication to help bridge the differences that exist between systems that use incongruent component models, operating systems, and programming languages. Developers can create applications that weave together XML Web services from a variety of sources in much the same way that developers traditionally use components when creating a distributed application.

One of the core characteristics of an XML Web service is the high degree of abstraction that exists between the implementation and the consumption of a service. By using XML-based messaging as the mechanism by which the service is created and accessed, both the XML Web service client and the XML Web service provider are freed from needing any knowledge of each other beyond inputs, outputs, and location.

XML Web services are enabling a new era of distributed application development. It is no longer a matter of object model wars or programming language beauty contests. When systems are tightly coupled using proprietary infrastructures, this is done at the expense of application interoperability. XML Web services deliver interoperability on an entirely new level that negates such counterproductive rivalries. As the next revolutionary advancement of the Internet, XML Web services will become the fundamental structure that links together all computing devices.

XML Web services must be agnostic regarding the choice of operating system, object model, and programming language to succeed in the dissimilarity of the Web. Also, for XML Web services to benefit from the same widespread adoption as other Web-based technologies, they must be:

- **Loosely Coupled:** Two systems are considered loosely coupled if the only mandate imposed on both systems is to understand the aforementioned self-describing, text-based messages. Tightly coupled systems, on the other hand, impose a significant amount of customized overhead to enable communication and require a greater understanding between the systems.
- **Ubiquitous Communication:** It is unlikely that anyone builds an operating system now or in the near future that does not incorporate the ability to connect to the Internet, therefore providing a ubiquitous communication channel. As such, the ability to connect almost any system or device to the Internet ensures such systems and devices are universally available to any other system or device connected to the Internet.
- **Universal Data Format:** By adopting existing, open standards over proprietary, closed-loop communication methods, any system that supports the same open standards is capable of understanding XML Web services. Utilizing self-describing, text-based messages that XML Web services and their clients can share without knowing what constitutes each underlying system enables communication between autonomous and different systems. XML Web services achieve this capability using XML.

XML Web services employ an infrastructure that provides the following: a discovery mechanism to locate XML Web services, a service description for defining how to use those services, and standard wire formats with which to communicate.

While much work has been done to make XML Web services a reality, more is needed. Today, people are having success with XML Web services, but there are still things that are left as an exercise for the developer (e.g. security, operational management, transactions, reliable

messaging). Pending and future specifications (e.g., WS-Security, WS-Inspection, WS-Routing, WS-Referral) will help complete these outstanding requirements, as well as, enhance the Web Service architecture. In the future, some of the most interesting XML Web services will support applications that use the Web to do things that cannot be done today. For example, one of the services that XML Web Services would make possible is a calendar service. If your dentist and mechanic exposed their calendars through this XML Web service, you could schedule appointments with them on line or they could schedule appointments for cleaning and routine maintenance directly in your calendar if you like. With a little imagination, you can envision hundreds of applications that can be built once you have the ability to program the Web.

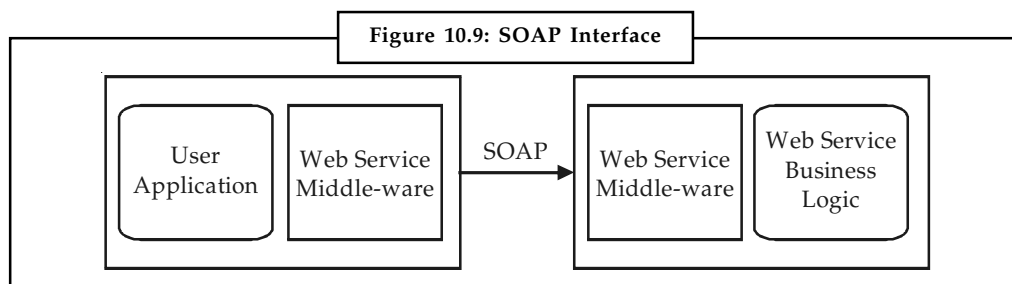
10.2.1 SOAP

SOAP and XML created the solution for the problem that developers were facing before. SOAP is a standard XML based protocol that communicated over HTTP. We can think of SOAP as message format for sending messages between applications using XML. It is independent of technology, platform and is extensible too.

We have SOAP and XML to give us connectivity between applications. Does it mean that I have to write XML and SOAP specific things myself to facilitate this communications? I could do that but that will be very time consuming and sometimes error prone too.

Where does Web Services come in picture? Well, Web services is the mechanism that ASP.NET framework provides to make it easy for us to write code to facilitate connectivity between applications. As ASP.NET developer, if I need an application that will be used by many other applications then I can simply decide to write a web service for it and ASP.NET framework will take care of doing the low level SOAP and XML work for us.

The other side of the coin is, if our ASP.NET application wants to use a web service, i.e., an application that is providing me SOAP based interface for communication. So what we are going to do now is write a small Web service to see how we can have our application communication-ready for other applications and secondly, we will try to consume a web service to understand how we can use other applications from our application.



10.2.2 WSDL

As communications protocols and message formats are standardized in the web community, it becomes increasingly possible and important to be able to describe the communications in some structured way. WSDL addresses this need by defining an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages. WSDL service definitions provide documentation for distributed systems and serve as a recipe for automating the details involved in applications communication.

A WSDL document defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions: messages,

Notes

which are abstract descriptions of the data being exchanged, and port types which are abstract collections of operations. The concrete protocol and data format specifications for a particular port type constitutes a reusable binding. A port is defined by associating a network address with a reusable binding, and a collection of ports define a service. Hence, a WSDL document uses the following elements in the definition of network services:

- **Types:** A container for data type definitions using some type system (such as XSD).
- **Message:** An abstract, typed definition of the data being communicated.
- **Operation:** An abstract description of an action supported by the service.
- **Port Type:** An abstract set of operations supported by one or more endpoints.
- **Binding:** A concrete protocol and data format specification for a particular port type.
- **Port:** A single endpoint defined as a combination of a binding and a network address.
- **Service:** A collection of related endpoints.

It is important to observe that WSDL does not introduce a new type definition language. WSDL recognizes the need for rich type systems for describing message formats, and supports the XML Schemas specification (XSD) as its canonical type system. However, since it is unreasonable to expect a single type system grammar to be used to describe all message formats present and future, WSDL allows using other type definition languages via extensibility.

In addition, WSDL defines a common binding mechanism. This is used to attach a specific protocol or data format or structure to an abstract message, operation, or endpoint. It allows the reuse of abstract definitions.

10.2.3 UDDI

Universal Description, Discovery and Integration (UDDI) is a directory service where businesses can register and search for Web services.

UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.

- UDDI stands for Universal Description, Discovery and Integration
- UDDI is a directory for storing information about web services
- UDDI is a directory of web service interfaces described by WSDL
- UDDI communicates via SOAP
- UDDI is built into the Microsoft .NET platform

UDDI uses World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) Internet standards such as XML, HTTP, and DNS protocols.

UDDI uses WSDL to describe interfaces to web services.

Additionally, cross platform programming features are addressed by adopting SOAP, known as XML Protocol messaging specifications found at the W3C Web site.

Any industry or businesses of all sizes can benefit from UDDI.

Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Nor was there a method of how to integrate into each other's systems and processes.

Problems the UDDI specification can help to solve:

Notes

- Making it possible to discover the right business from the millions currently online
- Defining how to enable commerce once the preferred business is discovered
- Reaching new customers and increasing access to current customers
- Expanding offerings and extending market reach
- Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
- Describing services and business processes programmatically in a single, open, and secure environment

10.2.4 General Principles of Web Services XML

The general principles of XML Web Services are given below:

Principle of Core Content

If you consider the information in contention to be part of the essential material that is being expressed or communicated in the XML, put it in an element. For human-readable documents this generally means the core content material that is being communicated to the reader. For machine-oriented records formats this normally means the data that comes straight from the problem domain.

If you consider the info to be peripheral or incidental to the main communication, or purely intended to help programs process the main communication, make use of attributes. This avoids cluttering in the core content with auxiliary material. For machine-oriented records formats, this normally means application-specific notations on the main data from the problem-domain.

Principle of Structured Information

If the information is expressed in a structured form, especially if the structure may be extensible, use elements. On the other hand: If the information is indicated as an atomic token, make use of attributes. Elements are the extensible engine for expressing structure within XML. Almost all XML processing tools are designed around this fact, and if you break up structured information appropriately into elements, you will find that your own processing tools accentuate your design, and that you thereby gain productivity and maintainability. Attributes are designed for expressing simple properties of the information represented in an element. If you work against the basic architecture of XML by shoehorning organised information into attributes you might gain some specious terseness as well as convenience, but you will most likely pay in maintenance costs.

Dates are a good example: A date has fixed structure and generally acts as a single token, so it makes sense as an attribute (preferably stated in ISO-8601). An individual name has surprisingly variable structure (in some cultures you may cause confusion or offense by omitting honorifics or assuming an order of parts of names). An individual name is also rarely an atomic token. As an instance, sometimes you may want to search or sort by a first name and sometimes by a surname. Thus:

```
customer>
<name>Johny Parker</name>
<occupation>Doctor</occupation>
</customer>
```

Notes

is not much better than:

```
<customer name= "Johny Parker">
  <occupation>Doctor</occupation>
</customer>
```

Principle of Readability

If the information is supposed to have been read and understood by a person, use elements. In general this guideline places prose in element content. If the information is most readily understood and digested by a machine, use attributes. In general this guideline means that information tokens that are not really natural language go in attributes.

In some cases, people may decipher the information being represented but need a machine to make use of it properly. URLs are a great example: People have learned to see URLs through exposure in Browsers and e-mail messages, however a URL is usually not much use without the computer to retrieve the referenced resource. Some database identifiers are also quite readable (although established database management best practice discourages using IDs that could have business meaning), but such IDs are often props for machine processing.

Principle of Element/Attribute Binding

Use an element if you'll need its value to be altered by another attribute. XML determines a very strong conceptual relationship between an attribute and the actual element in which it appears. An attribute provides some property or modification of that specific element. Processing tools for XML tend to follow this concept and it is almost always a terrible idea to have one attribute modify another. For example, if you are designing the format for a restaurant menu and you include the portion sizes of items on the menu, you may determine that this isn't really important to the typical reader of the menu format so you apply the Principle of core content and make it an attribute.

A simple XML event instance

```
<event>
<title>XML Seminar Planning Sessions</title>
<abstract>This meeting will cover the upcoming seminar</abstract>
<dateTime>January 20, 2003 at 7:00 PM</dateTime>
<duration>2 hours</duration>
<address>Ranier (Room 25), Building C</address>
</event>
```

However, there are a number of problems with a simple schema like this. The first is the fact that it is usually necessary for the computer to be able to differentiate between multiple events – two events may have the same title but be completely unrelated.

A second problem comes if a meeting is scheduled to start at a certain time, go for a certain duration, break for lunch, then start again at a later time. The <dateTime> and <duration> elements actually make up a single unit – call it a schedule element – that may be repeated multiple times within the event.

Similarly, what happens when the address of the meeting is outside of the same building? The structure needs to be more granular with this information — indicating which building is referenced, or even a total mailing address for people who are coming into the meeting from out of town.

The final problem with the above outline deals with the way that times and durations are specified. The values for <dateTime> and <duration> are useful for humans to read, but require

some serious parsing in order to be understood by a computer. By conforming to a distinct standard that XML languages are capable of understanding, you can minimize the amount of processing necessary in consumers of this XML.

A revised <event> item, taking into account these factors, may look more like the following:

Listing 2. Getting a little more complex

```
<event id="event-schema-00011521">
<title>XML Seminar Planning Sessions</title>
<abstract>This meeting will cover the upcoming seminar</abstract>
<scheduledTime>
<dateTime>2003-01-20T19:00:00</dateTime>
<duration>PT2H</duration>
</scheduledTime>
<address>
<room>Ranier (Room 25)</room>
<building>C</building>
</address>
</event>
```

The id attribute on the event contains a unique string that identifies the event; the degree of that uniqueness will depend upon the scope of the event – if the events being discussed are local to a group, then uniqueness could be something as simple as keeping track of how many meetings have been held. On the other hand, if the event was part of a distributed hierarchy sent out on the Internet as part of an RSS feed (as just one example), the id would have to be very unique, and would probably end up relying on something like a UUID (Java) or GUID (Microsoft) to be able to identify the event absolutely.

The <dateTime> element may look a little odd if you're not that familiar with XML schema, but it is actually a very convenient and compact notation for representing time. By giving the time in year, month, date, hour, minute and second order (assuming two digits for all but the year) you have a fixed-length string that can be queried to retrieve specific values in a uniform manner. Moreover, if you order the dates alphanumerically, they will also be in chronological order.

Similarly, the duration element "PT3H" gives a period ("P") of 2 hours ("H"). Periods are given in descending value of time. Thus a period of two days, six hours and eleven minutes is rendered as "P2DT6H11N". Again, while not immediately legible, this format is easy to parse. The T indicates that the values following are granular units of time less than one day.

Web Services Classification

Classification of XML Appliances

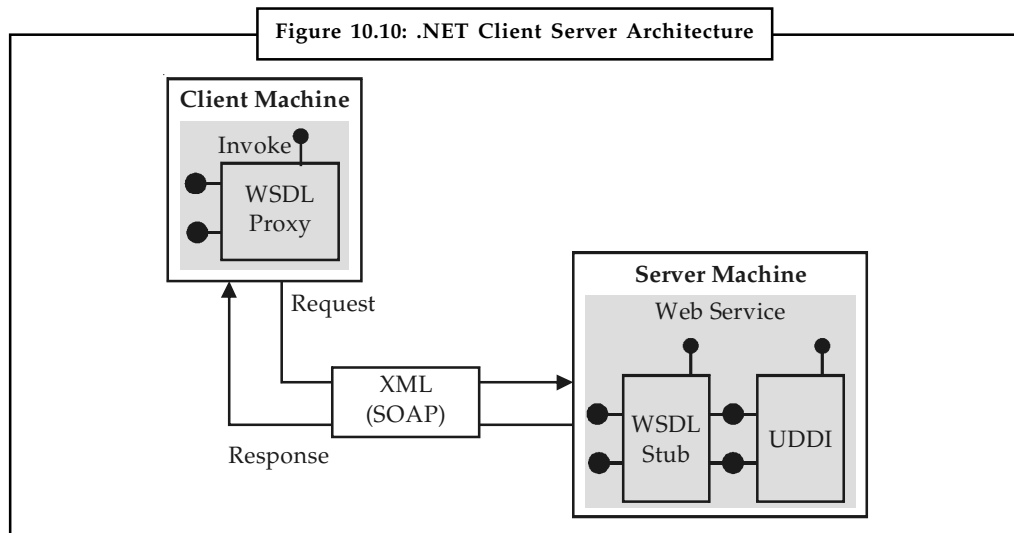
An XML appliance is a special purpose network device used to secure, manage and mediate XML traffic. They are most popularly implemented in Service Oriented Architectures to control XML based Web Services traffic, and increasingly in cloud oriented computing to help enterprises integrate on premise applications with off premise cloud hosted applications.

The following are alternative names used for XML Appliances:

XML accelerators: These devices primarily focused on XML processing and acceleration. SOA network appliance is used to off-load and act as co-processor device, providing overall increased performance and application server requirements for back-end applications.

about how the service is implemented. Unlike current component technologies, Web Services are not accessed via object-model-specific protocols, such as DCOM, RMI, or IIOP. Instead, Web Services are accessed via ubiquitous Web protocols (e.g., HTTP) and data formats (e.g., XML).

An ideal .NET web server and client would be designed as figured below:



Why Use Web Services

- **Exposing the existing function on to network:** A Web service is a unit of managed code that can be remotely invoked using HTTP, that is, it can be activated using HTTP requests. So, Web Services allows you to expose the functionality of your existing code over the network. Once it is exposed on the network, other application can use the functionality of your program.
- **Connecting Different Applications i.e. Interoperability:** Web Services allows different applications to talk to each other and share data and services among themselves. Other applications can also use the services of the web services. For example VB or .NET application can talk to java web services and vice versa. So, Web services is used to make the application platform and technology independent.
- **Standardized Protocol:** Web Services uses standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description and Service Discovery layers) uses the well defined protocol in the Web Services protocol stack. This standardization of protocol stack gives the business many advantages like wide range of choices, reduction in the cost due to competition and increase in the quality.
- **Low Cost of communication:** Web Services uses SOAP over HTTP protocol for the communication, so you can use your existing low cost internet for implementing Web Services. This solution is much less costly compared to proprietary solutions like EDI/B2B. Beside SOAP over HTTP, Web Services can also be implemented on other reliable transport mechanisms like FTP etc.

Advantages of Web Services

- Web services provide interoperability between various software applications running on disparate platforms/operating systems.

Notes

- Web services use open standards and protocols. Protocols and data formats are text-based where possible, making it easy for developers to comprehend.
- By utilizing HTTP, web services can work through many common firewall security measures without requiring changes to the firewall filtering rules. Other forms of RPC may more often be blocked.
- Web services allow software and services from different companies and locations to be combined easily to provide an integrated service.
- Web services allow the reuse of services and components within an infrastructure. Web services are loosely coupled thereby facilitating a distributed approach to application integration.

Disadvantages of Web Services

- Web services standards features such as transactions are currently non-existent or still in their infancy compared to more mature distributed computing open standards such as CORBA. This is likely to be a temporary disadvantage as most vendors have committed to the OASIS standards to implement the Quality of Service aspects of their products.
- Web services may suffer from poor performance compared to other distributed computing approaches such as RMI, CORBA, or DCOM. This is a common trade-off when choosing text-based formats.



Case Study

MP Education Department Deploys MIS Portal Solution to Facilitate Proactive and Transparent Governance

The School Education Department is one of the largest departments in terms of engagement of human resources in the government sector in Madhya Pradesh. To administer more than one lakh schools, 3.5 lakh teachers and staff, maintain and monitor a database of more than 1.60 crore students was a gigantic task. To overcome these challenges, the government conceived the idea of developing a bilingual, database driven, dynamic portal in technical collaboration with National Informatics Centre, Madhya Pradesh, Bhopal. Powered by Microsoft SQL Server 2005, the portal is addressed to all stakeholders of the state education system namely students, teachers, citizens and educational managers enabling them to communicate in real time. It provides administrative transparency, scalability for 500 concurrent portal users, and tools to improve collaboration between students, teachers, parents, and administrators.

Situation

Madhya Pradesh (MP) is the second largest state of India with largest tribal population amongst all states. School education is the largest sector in the state in terms of number of beneficiaries, geographical reach, number of institutions, and engagement of human resources. It is also the most complex sector with multiple departments and local bodies exercising control over the work force engaged in it. Over the last two decades, the workload of the departments, institutions and offices dealing with school education has increased manifold without corresponding increase in the quantity and capability of the supervisory staff.

Contd....

Key Entities/Attributes of the School Education Department	
Geographical reach	More than 1 lakh habitations
Beneficiaries	1.60 crore students and their parents
Establishments	1.15 lakh schools
Human resources	Nearly 3.5 lakh teachers and staff

Geographical reach	More than 1 lakh habitations
Beneficiaries	1.60 crore students and their parents
Establishments	1.15 lakh schools
Human resources	Nearly 3.5 lakh teachers and staff

Implementation of the Right of Children to Free and Compulsory Education Act 2009 requires the state to provide free elementary education and ensure compulsory admission, attendance and completion of elementary education by all children between 6–14 years. The inception of RTE Act in 2009 has levied additional expectations on the state-school education departments, to be able to provide transparent and accountable governance apart from ensuring quality education to the masses.

The major challenges faced by MP government in achieving the objective of quality education and proactive governance is to ensure enrolment of over 1.60 crore children, retention and achievement of minimum learning levels, management and administration of schools, tracking of movement of employees and timely payments of salaries. In addition, the government also needs to manage, coordinate, and synchronize multiple administrative units under multiple departments.

Considering the complex scenario and spread of the activities at habitat level, there were no reliable and integrated systems to facilitate the functioning in an efficient and integrated manner. The state did a SWOT analysis with department officials and teachers and analyzed the key issues, problems, strengths and shortcomings of the prevailing system of governance, and opportunities and potential threats.

It emerged that effective management and monitoring of such a large sector required an integrated, common and concurrent system that works in real-time. There was need for universal provisioning of basic schooling; and ensuring quality education in schools by increasing capacity of teachers, students and education managers.

Questions:

1. Discuss the IT issues in this Project.
2. What are the business needs of this project?

Self Assessment

Multiple Choice Questions:

6. You are creating a Web service that will include a Web method. You want the method to return its data to the caller as it is created and put into a serial stream, rather than waiting for all of the data to be prepared before returning it. Which of the following WebMethod attributes should you set to enable this
 - a. <WebMethod(EnableSession:=True)>
 - b. <WebMethod(BufferResponse:=False)>
 - c. <WebMethod(EnableSession:=False)>
 - d. <WebMethod(BufferResponse:=True)>
7. What are Transport methods in SOAP?
 - a. HTTP
 - b. FTP
 - c. POST
 - d. HTTPS

Notes

8. A web service is bind with three different protocols such as HTTP/POST, HTTP/GET, and SOAP.
 - a. True
 - b. False
9. A standard for creating business registries that catalog companies, the web services they provide, and the corresponding URLs for their WSDL contracts.
 - a. UDDI
 - b. HTTP
 - c. SOAP
 - d. DISCO
10. Because web services use HTTP, they can pass through firewalls without explicit configuration.
 - a. True
 - b. False
11. Theclass wraps the calls to the web service’s methods. It generates SOAP message format and manages the transmission of the messages over the network (using HTTP).
 - a. WebMethod
 - b. Web
 - c. Proxy
 - d. Synchronous
12. Thecomputing allows partitioning of application logic into units and spreading the unit over different computers of a network or across different networks.
 - a. Parallel
 - b. Distributed
 - c. Centralized
 - d. Mobile
13. In which of these WSDL document uses the following elements in the definition of network services:
 - a. Port Type
 - b. Messenger
 - c. Port
 - d. Operation
14. In the following statement which one is not correct about UDDI.
 - a. UDDI stands for Universal Description, Discovery and Integration
 - b. UDDI is a directory for storing information about web application
 - c. UDDI is a directory of web service interfaces described by WSDL
 - d. UDDI communicates via SOAP
15. In which of these are alternative names used for XML Appliances
 - a. XML accelerators
 - b. XML firewalls
 - c. Both (a) and (b)
 - d. None of these

10.4 Summary

- The W3C defines a “Web service” as “a software system designed to support interoperable machine-to-machine interaction over a network”.
- Web Services are applications that can be published, located, and invoked across the Internet.
- Web Services may use other Web Services in order to perform their task.
- Web Service messages are formatted as XML, a standard way for communication between two incompatible system.

- The infrastructure for XML Web services is built to conform to industry standards such as SOAP, XML and WSDL, and this allows clients from other platforms to interoperate with XML Web services.
- When you build an XML Web service using ASP.NET, it automatically supports clients communicating using the SOAP, HTTP-GET, and HTTP-POST protocols.
- Web services are distributed application components that are externally available. You can use them to integrate computer applications that are written in different languages and run on different platforms.
- This directive allows you to refer to objects in the System.Web.Services namespace without having to fully qualify the request.
- To test a Web Service, it must be published over IIS.
- Web Services use the conventional client server model and can be accessed via Synchronous or Asynchronous call.
- Open standards and the focus on communication and collaboration among people and applications have created an environment where XML Web services are becoming the platform for application integration.
- WSDL file and generate the code required to communicate with an XML Web service.

10.5 Keywords

Application Programming Interface (API): The word to really pay attention to is “Interface”. If you have any experience at all with programming, all kinds of abstractions and contracts must be coming to your mind when you hear the word “interface” but we are more interested in the classical meaning of the term.

CORBA: The acronym for Common Object Request Broker Architecture, is a widely used communications model for building distributed (multi-tier) applications that connect both cross-platform and cross-language clients to server-based services.

MSMQ: It is a message queuing framework used for applications that use messaging infrastructure. MSMQ is a tool for sending and receiving messages.

Service Oriented Architecture (SOA): A service oriented architecture can be defined as a group of services, which communicate with each other. The process of communication involves either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

SOA Gateways: The term “SOA Governance” was used to describe everything from design time management of test and design artifacts, monitoring, enforcement, UDDI registry storage.

SOAP: Simple Object Access Protocol is a standard XML based protocol that communicated over HTTP.

Stateless: That is, information that is generated on one page is not normally accessible by any other page.

UDDI: Universal Description, Discovery and Integration is a directory service where businesses can register and search for Web services.

WSDL: Web Services Description Language document defines services as collections of network endpoints, or ports.

Notes

10.6 Review Questions

1. What is the standard you use to wrap up a call to a Web service?
2. Define Protocols that helps Web Services in ASP.NET.
3. Can you define basic element of Web Services and explain any one from them?
4. What do you think Web services only written in .NET?
5. Is it possible to use datareader in Web Services?
6. What is XML RPC?
7. Is it possible to generate the source code for an ASP.NET Web service from a WSDL?
8. Discuss advantage and disadvantage of using Web Service.
9. What is ASP.NET Web Services?
10. Explain when do we required ASP.NET Web services.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (a) | 2. (a) | 3. (c) | 4. (b) |
| 5. (d) | 6. (b) | 7. (a) | 8. (a) |
| 9. (a) | 10. (a) | 11. (c) | 12. (b) |
| 13. (b) | 14. (b) | 15. (c) | |

10.7 Further Readings



Books

Professional ASP.NET 3.5, by Bill Evjen



Online links

XML Web Services with ASP.NET, by Bill Evjen <http://dotnetguts.blogspot.in/2007/09/all-about-web-service-in-net.html>

Unit 11: Web Services in Visual Studio .NET

Notes

CONTENTS

Objectives

Introduction

11.1 Creating Web Services

11.2 Expanding Web Application with Web Services

11.2.1 Web Applications Defined

11.2.2 How do Web Applications Work?

11.2.3 Types of Web Applications

11.2.4 Technologies Used to Build Web Applications

11.3 Summary

11.4 Keywords

11.5 Review Questions

11.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain about creating web services
- Understand expanding web application with web services

Introduction

While .NET provides us with a robust and powerful framework for building all kinds of applications, one of the most significant shifts is to what is called Web services. When we say that .NET gives us a way to use object-oriented functionality across the Web, as opposed to just Web pages, we are talking about Web services. In simple terms, a Web service is a specialized Web application, which runs on a Web server. Instead of serving pages with a human-usable user interface, Web services serve methods and objects, complete with properties. The protocol used for using these objects across the network is SOAP and is heavily tied to XML.

SOAP uses XML as the default format of the objects, properties, and data when communicating via Web services. Instead of fulfilling requests for pages by a browser, or other simple rendering client, Web services are designed to fulfil requests made by other applications that will use the served objects in their runtime processing.

For a simple example, think of a shipping company. In the past, this company would most likely provide its prices and services over the Web via a standard Web application, meant to be viewed by a human using a browser, complete with a formatted user interface to display the data. With .NET, this company can still provide a browser-readable Web application as before, but it also has an additional means to allow access to its data. By creating a Web service, the company can now provide its shipping prices and services as objects and properties to other applications.

Notes

The Web service would not format the data or add a user interface to it at all. It would provide business objects, loaded with data, so the calling program could display or use the information as needed. This provides a whole new level of power and flexibility both for the company providing the Web services and for the calling application. Even simple functionality fits nicely into the Web services model. For example, instead of keeping a local database of ever-changing area codes or zip codes and city names, a company could call a Web service that offers this up-to-date data as objects over the Web and consume them in their applications.

When we install .NET on a Windows server, enhancements are added to Internet Information Server, which allow us to create and serve Web services easily. Likewise, we can consume Web services easily from our applications written for .NET by utilizing the fully featured SOAP client classes in the .NET Framework. Microsoft's .NET strategy will be key to moving away from the current client/server-based world of information to a truly distributed network architecture, all the way down to the application level.

We all use distributed networks everyday by accessing our company's customer accounts while at work or checking our bank statements over the Internet at home. We use a client application such as a browser, which is installed locally, to read and write data on a Web server. For example, when you transfer money between your bank accounts over the Internet, your browser reads the stream of HTML data from the server and displays the data about your current account balances on the screen. Then, you would make changes in the fields and press a button to invoke the changes back on the server. At this point in the game, all the work is done on the server – the browser only sends a stream of data back to the server, where the real work will happen to actually change your balance.

The difference between today's technology and .NET is on what level the distribution occurs. With .NET, there is a shift from having large silos of data on servers to having real functionality on the remote systems. With .NET, we will no longer be limited to requesting a stream of formatted data from the Web server; instead, we can actually call functions on the server in an object-oriented way. Think of the banking example just described. With .NET, the bank could make not only this data, but also the functionality and logic related to it, available as a Web service. Then, it could provide a user interface in the form of a Web application as before. If the system is exposed as services, the bank could also create a rich client application for its users that would provide more advanced features than can easily be built in Web browser output. Or, maybe the bank would create an application that plugs into a personal manager program, so that you can check your bank balance from there. These are just some of the things that Web services can provide; we will discuss how to create and consume Web services later in the book.

While Web services are the underlying backbone to .NET's idea of a massive distributed system, the Web as most of us know it today will remain unchanged for a long time. We will still be accessing applications by using our Web browsers to request pages of information. For these reasons, Web sites must become more feature rich while at the same time support a diverse set of client devices, each with its own abilities and limitations. This is where ASP.NET steps in, and this is what this book will focus on, and particularly how it is used when a Web browser is the client. In the very near future, many people will be accessing the Web from their cell phones, cars, and pocket computers, but those topics are outside the scope of this book. With ASP.NET, Web programming has taken a large evolutionary step in the right direction. And with the release of .NET, ASP has been upgraded to a serious programming tool that just happens to support all of the other great features of .NET and the distributed systems of tomorrow.

The .Net Web Services that run over HTTP can be called in three different ways:

- **HTTP GET Operation:** You can pass parameters to a Web Service by calling the ASMX page with query string parameters for the method to call and the values of simple parameters to pass.

Notes

- **HTTP POST Operation:** Works the same as GET Operation except that the parameters are passed as standard URL encoded form variables. If you use a client such as wwIPStuff you can use AddPostKey () to add each parameter in the proper parameter order.
- **SOAP:** This is the proper way to call a Web Service in .Net and it is also the way that .Net uses internally to call Web Services.

The GET and POST operations are useful if you need to call a Web Service quickly and no SOAP client is readily available. For example, in a browser based client application it may be easier to use GET and POST instead of constructing and parsing the more complex SOAP headers that are passed back and forth in a SOAP request. But with a proper SOAP client in place SOAP provides the full flexibility of the protocol, where GET and POST operations have to stick to simple inputs and outputs. Among other things that you can do with SOAP is pass complex objects and data over the wire and for these operations to work you need to use SOAP.

Behind the scenes there are three major components that make up a Web Service:

1. The Web Service on the Server side.
2. The client application calling the Web Service via a Web Reference.
3. A WSDL Web Service description that describes the functionality of the Web Service.

11.1 Creating Web Services

1. Start Visual Studio .NET or Visual Studio 2005.
2. Create a new Active Server Pages ASP.NET Web service project. Name the Web service MathService and point the location to an appropriate Web server that is running ASP.NET if necessary.
3. Change the name of the Solution file to MathService for consistency.
4. Change the name of the default Web service that is created from Service1.asmx to MathService.asmx.
5. Click here to switch to code view in the designer environment to switch to code view.
6. Change the name of the class from Public Class Service1 to Public Class MathService.
7. Define methods that encapsulate the functionality of your service. Each method that will be exposed from the service must be flagged with a WebMethod attribute in front of it. Without this attribute, the method will not be exposed from the service.



Notes Not every method needs to have the WebMethod attribute. It is useful to hide some implementation details called by public Web service methods or for the case in which the WebService class is also used in local applications. A local application can use any public class, but only WebMethod methods will be remotely accessible as Web services.

Add the following method to the MathServices class that you just created:

```
<WebMethod()> Public Function Add(a As Integer, b As Integer) As Integer
    Return(a + b)
End Function
```

```
<WebMethod()> Public Function Subtract(A As System.Single, B As
```

Notes

```
System.Single) As System.Single
    Return A - B
End Function
```

```
<WebMethod()> Public Function Multiply(A As System.Single, B As
System.Single) As System.Single
    Return A * B
End Function
```

```
<WebMethod()> Public Function Divide(A As System.Single, B As
System.Single) As System.Single
    If B = 0
    Return -1
    End If
    Return Convert.ToSingle(A / B)
End Function
```

8. Click Build on the Build menu to build the Web service.
9. Browse to the MathService.asmx Web service page to test the Web service. If you set the local computer to host the page, the URL is <http://localhost/MathService/MathService.asmx>.

The ASP.NET runtime returns a Web Service Help Page that describes the Web service. This page also enables you to test different Web service methods.

How to Use a Web Service

1. Start Visual Studio .NET or Visual Studio 2005.
2. Create a new Console Application project.
3. Add a reference for the MathService Web Service to the new console application.

This step creates a proxy class on the client computer. After the proxy class exists, you can create objects based on the class. Each method call that is made with the object then goes out to the uniform resource identifier (URI) of the Web service (usually as a SOAP request).

 - (a) On the Project menu, click Add Web Reference.
 - (b) In the Add Web Reference dialog box, type the URL for the Web service in the Address text box and press ENTER. If you set the local computer to host the Web service, the URL is <http://localhost/MathService/MathService.asmx>.
 - (c) Click Add Reference.
 - (d) Expand the Web References section of Solution Explorer and note the namespace that was used.
4. Create an instance of the proxy object that was created. Place this code in the Main procedure of the Module1 module:

```
Dim myMathService As localhost.MathService = New localhost.MathService()
```
5. Invoke a method on the proxy object created in the previous step:

```
Console.WriteLine("2 + 4 = {0}", myMathService.Add(2,4))
```
6. Close and save the project.



Task Create a Web service.



Did u know? The SOAP was designed as an object-access protocol in 1998 by Dave Winer, Don Box, Bob Atkinson, and Mohsen Al-Ghosein for Microsoft, where Atkinson and Al-Ghosein were working at the time.

Self Assessment

Multiple Choice Questions:

- A significant advantage of web services, relative to other, is the support of firewalls.
 - WSDL
 - Access Protocol
 - distributed architectures
 - SOAP
- A is programmable application logic accessible via standard Web protocols.
 - Web Application
 - Windows Application
 - Desktop Application
 - Web Service
- and Integration is the yellow pages of Web services.
 - FDDI
 - UDDI
 - WSDL
 - None of these.
- .NET framework provides aand code to consume web services.
 - Collection technologies
 - HTTP protocols
 - Protocols
 - Sophisticated set of tools
- SOAP uses as the default format of the objects, properties, and data when communicating via Web services.
 - CSS
 - XHTML
 - XML
 - HTML

11.2 Expanding Web Application with Web Services

Write a Web-based consumer as given below. Call it WebApp.aspx. Note that it is an ASP.NET application. Save this in the virtual directory of the Web Service (c:\MyWebServices\WebApp.aspx).

This application has two text fields that are used to get numbers from the user to be added. It has one button, Execute, that, when clicked, gets the Add and SayHello Web Services.

WebApp.aspx

```
<%@ Page Language="C#" %>
<script runat="server">
void runSrvce_Click(Object sender, EventArgs e)
{
FirstService mySvc = new FirstService();
Label1.Text = mySvc.SayHello();
}
```

Notes

```

Label2.Text = mySvc.Add(Int32.Parse(txtNum1.Text),
Int32.Parse(txtNum2.Text)).ToString();
}
</script>
<html>
<head>
</head>
<body>
<form runat="server">
<p>
<em>First Number to Add </em>:
<asp:TextBox id="txtNum1" runat="server"
Width="43px">4</asp:TextBox>
</p>
<p>
<em>Second Number To Add </em>:
<asp:TextBox id="txtNum2" runat="server"
Width="44px">5</asp:TextBox>
</p>
<p>
<strong><u>Web Service Result -</u></strong>
</p>
<p>
<em>Hello world Service</em> :
<asp:Label id="Label1" runat="server"
Font-Underline="True">Label</asp:Label>
</p>
<p>
<em>Add Service</em> :
& <asp:Label id="Label2" runat="server"
Font-Underline="True">Label</asp:Label>
</p>
<p align="left">
<asp:Button id="runSrvic" onclick="runSrvic_Click"
runat="server" Text="Execute"></asp:Button>
</p>
</form>
</body>
</html>

```

After the consumer is created, we need to create a proxy for the Web Service to be consumed. This work is done automatically by Visual Studio .NET for us when referencing a Web Service that has been added. Here are the steps to be followed:

- (a) Create a proxy for the Web Service to be consumed. The proxy is created using the wsdl utility supplied with the .NET SDK. This utility extracts information from the Web Service and creates a proxy. Thus, the proxy created is valid only for a particular Web Service. If you need to consume other Web Services, you need to create a proxy for this service as well. VS .NET creates a proxy automatically for you when the reference for the Web Service is added. Create a proxy for the Web Service using the wsdl utility supplied with the .NET SDK. It will create FirstSevice.cs in the current directory. We need to compile it to create FirstService.dll (proxy) for the Web Service.

```
c:> WSDL http://localhost/MyWebServices/  
a.FirstService.asmx?WSDL  
c:> csc /t:library FirstService.cs
```

- (b) Put the compiled proxy in the bin directory of the virtual directory of the Web Service (c:\MyWebServices\bin). IIS looks for the proxy in this directory.
- (c) Create the service consumer, which we have already done. Note that I have instantiated an object of the Web Service proxy in the consumer. This proxy takes care of interacting with the service.
- (d) Type the URL of the consumer in IE to test it (for example, <http://localhost/MyWebServices/WebApp.aspx>).

11.2.1 Web Applications Defined

A web application is any application that uses a web browser as a client. The application can be as simple as a message board or a guest sign-in book on a website, or as complex as a word processor or a spreadsheet.

What is a Client?

The 'client' is used in client-server environment to refer to the program the person uses to run the application. A client-server environment is one in which multiple computers share information such as entering information into a database. The 'client' is the application used to enter the information, and the 'server' is the application used to store the information.

What are the Benefits of a Web Application?

A web application relieves the developer of the responsibility of building a client for a specific type of computer or a specific operating system. Since the client runs in a web browser, the user could be using an IBM-compatible or a Mac. They can be running Windows XP or Windows Vista. They can even be using Internet Explorer or Firefox, though some applications require a specific web browser.

Web applications commonly use a combination of server-side script (ASP, PHP, etc.) and client-side script (HTML, Javascript, etc.) to develop the application. The client-side script deals with the presentation of the information while the server-side script deals with all the hard stuff like storing and retrieving the information.

How Long Have Web Applications Been Around?

Web Applications have been around since before the web gained mainstream popularity. For example, Larry Wall developed Perl, a popular server-side scripting language, in 1987. That was seven years before the Internet really started gaining popularity outside of academic and technology circles.

The first mainstream web applications were relatively simple, but the late 90s saw a push toward more complex web applications. Nowadays, millions of Americans use a web application to file their income taxes on the web.

What is the Future of Web Applications?

Most web applications are based on the client-server architecture where the client enters information while the server stores and retrieves information. Internet mail is an example of this, with companies like Yahoo and MSN offering web-based email clients.

Notes

The new push for web applications is crossing the line into those applications that do not normally need a server to store the information. Your word processor, for example, stores documents on your computer, and doesn't need a server.

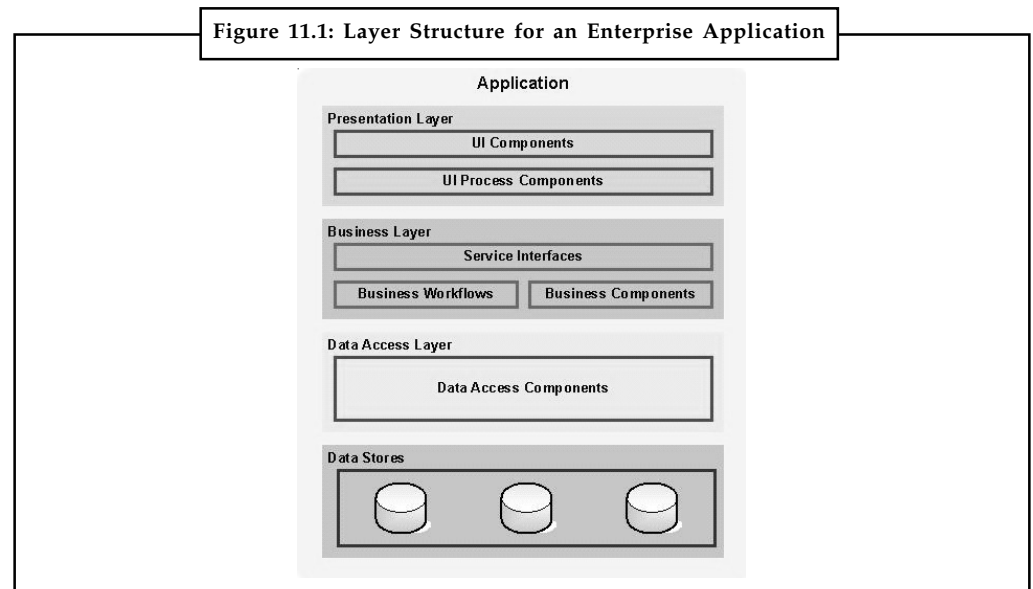
Web applications can provide the same functionality and gain the benefit of working across multiple platforms. For example, a web application can act as a word processor, storing information and allowing you to 'download' the document onto your personal hard drive.

If you have seen the new Gmail or Yahoo mail clients, you have seen how sophisticated web applications have become in the past few years. Much of that sophistication is because of AJAX, which is a programming model for creating more responsive web applications.

Google Apps, Microsoft Office Live, and WebEx WebOffice are examples of the newest generation of web applications.

11.2.2 How do Web Applications Work?

Modern enterprise applications are built using several components connected to one another, each providing a specific functionality. Components that perform similar types of functions are generally grouped into layers. These layers are further organized as a stack in which components in a higher layer use the services of components in the layer below. A component in a given layer will generally use the functionality of other components in its own layer or the layers below it. The Figure 11.1 shows a popular layer structure for an enterprise application.



- **Presentation Layer:** The presentation layer contains components needed to interact with the user of the application. Examples of such components are web pages, rich-client forms, user interaction process components etc.
- **Business Layer:** The business layer encapsulates the core business functionality of the application. Simple business functions can be implemented using stateless components whereas complex long running transactions can be implemented using stateful workflows. The business components are generally front-ended by a service interface that acts as a facade to hide the complexity of the business logic. This is commonly known as Service-Oriented Architecture (SOA).
- **Data Access Layer:** The data access layer provides a simple API for accessing and manipulating data. The components in this layer abstract the semantics of the underlying data access technology thus allowing the business layer to focus on business logic.

Each component typically provides methods to perform Create, Read, Update, and Delete (CRUD) operations for a specific business entity.

Notes

- **Data Stores:** Enterprise applications store their data in one or more data stores. Databases and file systems are two very common types of data stores.

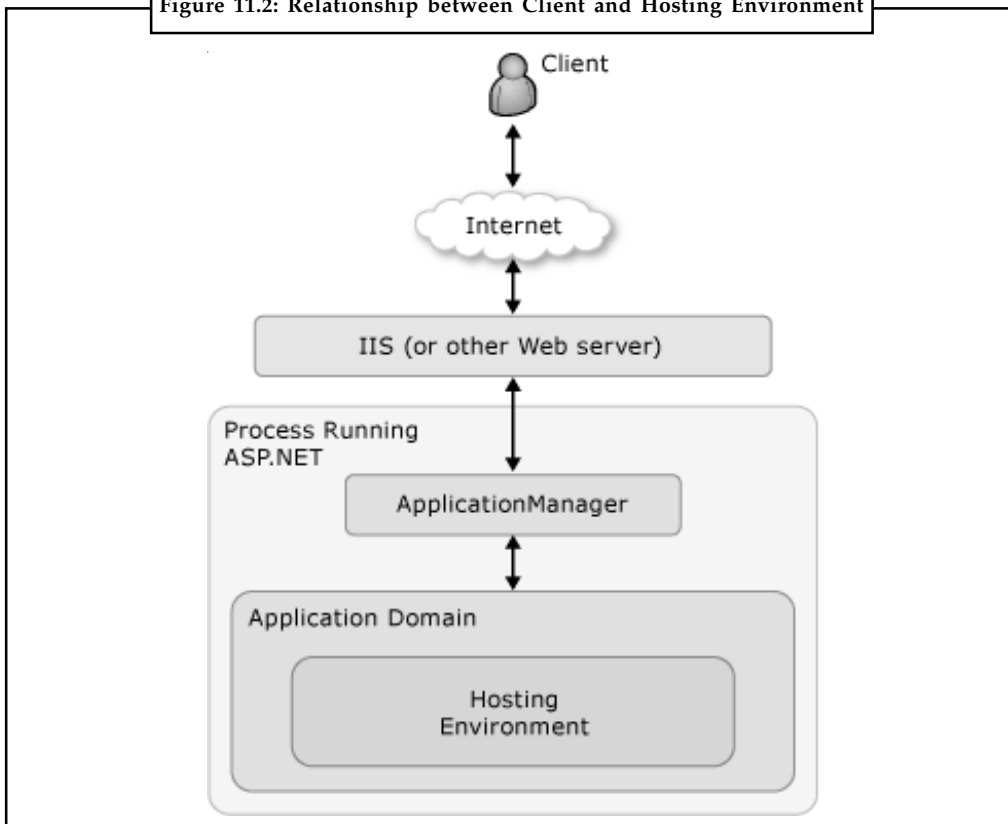


Notes Layers are simply logical groupings of components that make up an application. How these layers are deployed on physical machines can vary widely depending on several factors. In the very simplistic scenario all the layers can reside on one machine. In a slightly more complex scenario, the presentation layer can reside on one machine, the business and data access layers on a second machine and the database on a third machine. More elaborate scenarios are possible. For example, a high traffic web site can deploy the presentation layer on a web farm consisting of tens of machines.

When ASP.NET receives the first request for any resource in an application, a class named `ApplicationManager` creates an application domain. Application domains provide isolation between applications for global variables and allow each application to be unloaded separately. Within an application domain, an instance of the class named `HostingEnvironment` is created, which provides access to information about the application such as the name of the folder where the application is stored.

The following Figure 11.2 illustrates this relationship:

Figure 11.2: Relationship between Client and Hosting Environment



ASP.NET also compiles the top-level items in the application if required, including application code in the `App_Code` folder.

11.2.3 Types of Web Applications

An application in which all or some parts of the software are downloaded from the Web each time it is run. It may refer to browser-based applications that run within the user's Web browser or to "rich client" applications that do not use a browser.

Browser Based

In a browser-based Web application, JavaScript instructions are contained within the Web page that is retrieved from a Web site. Combined with the HTML code that determines the visual layout, the HTML and JavaScript on the Web page are executed via the browser. Alternatively, the Web page may cause the browser to launch a Java applet. Java is a full-blown programming language that is more comprehensive than JavaScript. The data for a Web application may be stored locally or on the Web, or in both locations. See Java.

Client Based

Web applications may also run without the browser. A client program, which is either installed in the user's computer or mobile device or is downloaded each session, interacts with a server on the Web using standard Web protocols. This is similar to the "client/server" architecture that prevailed in companies before the Internet exploded, except that today the server is often on the Internet rather than the local network. Just like browser-based applications, the data may be stored remotely or locally. See rich client, cloud computing, ASP and SaaS.



Task Prepare a Web application.

11.2.4 Technologies Used to Build Web Applications

The fact that Web applications are so much more interactive than standard websites means that it's not possible to create them with just HTML; instead, they require additional 'client-side' technology including Java, JavaScript, AJAX, Flash, Silverlight and HTML5.

In general, HTML, CSS, JavaScript and AJAX are used for the user-interface; Flash or Silverlight for an enhanced user experience; a Web-programming language such as ASP.NET or PHP to deliver bespoke functionality; and a database to reliably store all sorts of information including usage statistics, user/customer data and application-specific requirements.

If you're interested in the development technologies and techniques we use, take a look at our Web Software Development Technologies page – you'll find unbiased information on Java, JavaScript, AJAX, Flash vs. Silverlight vs. HTML5, ASP.NET vs. PHP & Drupal, and Databases.



Did u know? Perl was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier. Perl gained widespread popularity in the late 1990s as a CGI scripting language, in part due to its parsing abilities.



Case Study

Procurement System to Automate Procedures

Customer

The customer is a large Russian Federal Agency that is proactively using public purchases in its business activity.

Problem Overview

The customer was faced with new legislative regulations that imposed strict requirements on procurement process. The breach of regulations may result in procurement results' protestation and cancellation.

The procedures used customer personnel to prepare procurement documents were not automated. All documents were made by manually using Microsoft Office: requests for proposal – in Microsoft Word, main stages were tracked in Microsoft Excel, reports were written by hand on pre-printed forms.

Considering new rules that mainly set up new time-constraints and restricted procedures execution, the customer could not rely upon old outdated practices any longer as they might result in transactions cancellation and financial loss.

The new procurement system was designed to automate procedures, namely but not limited to:

- Registration of customer needs in goods and services;
- Automatic creation of procurement documents and requests for quotations based on predefined templates;
- Flexible documents and templates management;
- Documents workflow management;
- Results of purchase orders placement;
- Tracking of time-constraints fulfilment;
- Creation of reports on results of preliminary purchase plans execution;
- Automatic notifications about upcoming events;
- Implementation of managed internal goods and services listings.

Also the system was designed to be easily extended to support new methods of procurements that were introduced by new legislation.

Questions:

1. State the benefits of this project when developed using ASP.NET.
2. Why we need to automate Procurement process?

Self Assessment

Multiple Choice Questions:

6. The layer provides a simple API for accessing and manipulating data.
 - a. Presentation
 - b. Data Access
 - c. Data Store
 - d. Business

Notes

7. Web services using the HTTP protocol on port 80, which is generally not secured.
 - a. True
 - b. False
8. When ASP.NET receives the first request for any resource in an application, a class named creates an application domain.
 - a. ApplicationManager
 - b. ClassManager
 - c. ResourecManager
 - d. ProjectManager
9. Which one is correct about Web Application?
 - a. Web applications can provide the same functionality and gain the benefit of working across multiple platforms.
 - b. Web applications are based on the client-server architecture.
 - c. AJAX is a programming model for creating more responsive web applications.
 - d. All of the above.
10. On a site that has sensitive information somewhere on it, every time that site is accessed withinstead of, the user and the session will get exposed.
 - a. HTTP,HTTPS
 - b. HTTPS,HTTP
 - c. HTTP,SOAP
 - d. HTTPS,SOAP
11. Which one is not correct about Web Service?
 - a. Web Service is the web processing directive that specifies a class name.
 - b. Web Services are simple and easy to understand.
 - c. Web Service is the web processing directive that specifies a path name.
 - d. A Web Service in .NET consists of an .asmx page that either contains a class that provides the Web Service functionality or references a specific external class.
12. In a Web application, JavaScript instructions are contained within the Web page that is retrieved from a Website.
 - a. Browser Based
 - b. Client Based
 - c. Server Based
 - d. Program Based
13. Major component that make up a Web Service.
 - a. The Web Service on the Server side.
 - b. The client application calling the Web Service via a Web Reference.
 - c. A WSDL Web Service description that describes the functionality of the Web Service.
 - d. All of the above.
14. A can be defined as a group of services, which communicate with each other.
 - a. FDI
 - b. SOA
 - c. SOAP
 - d. XML
15. Which one is not correct about Web Service?
 - a. Web services are the underlying backbone to .NET's idea of a massive distributed system.

- b. Web services can work through many common firewall security measures without requiring changes to the firewall filtering rules.
- c. Web services provide interoperability between various software applications running on disparate platforms/operating systems.
- d. Web services does not use open standards and protocols.

Notes

11.3 Summary

- A Web Service is programmable application logic accessible via standard Web protocols.
- Web Services are simple and easy to understand.
- Web services are the underlying backbone to .NET's idea of a massive distributed system.
- The .Net Web Services that run over HTTP can be called in three different ways using HTTP Get Operation, HTTP Post Operation and through SOAP.
- A Web Service in .NET consists of an .asmx page that either contains a class that provides the Web Service functionality or references a specific external class that handles the logic in an external class file.
- Web Service is the web processing directive that specifies a class name; here service for a web service and the language used to create the web service is visual basic.
- A web application is any application that uses a web browser as a client.
- Web applications commonly use a combination of server-side script (ASP, PHP, etc.) and client-side script (HTML, Javascript, etc.) to develop the application.
- Web application allowing website visitors to submit and retrieve data to/from a database over the Internet.
- Popular layer structure for an enterprise application includes – Presentation Layer, Business Layer, Data Access Layer and Data Store.
- Web Applications are of two types: Browser based and client based.
- Web Applications required additional 'client-side' technology including Java, JavaScript, AJAX, Flash, Silverlight and HTML5.

11.4 Keywords

AJAX: An acronym for Asynchronous JavaScript and XML is a group of interrelated web development techniques used on the client-side to create asynchronous web applications.

Cascading Style Sheets (CSS): It is a style sheet language used for describing the presentation semantics (the look and formatting) of a document written in a markup language.

Dreamweaver: It is the perfect tool for web designers, coders, and application developers at all levels.

Enterprise Resource Planning (ERP): Systems integrate internal and external management information across an entire organization, embracing finance/accounting, manufacturing, sales and service, customer relationship management, and so on.

Flash: Adobe Flash (formerly called "Macromedia Flash") is a multimedia and software platform used for authoring of vector graphics, animation, games and Rich Internet Applications.

Notes

PHP: PHP is a server-side scripting language designed for web development but also used as a general-purpose programming language.

SilverLight: Silverlight is a powerful development tool for creating engaging, interactive user experiences for Web and mobile applications.



Lab Exercise

1. Create a flow chart to prepare a Web services.
2. Prepare a list of companies which provide online Web services.

11.5 Review Questions

1. How to Consume Web Service?
2. How .NET and non-.NET component communicate with each other when they are on different platform?
3. Can you give an example of when it would be appropriate to use a web service as opposed to a non-serviced .NET component?
4. When would you use .NET Remoting and when Web Services?
5. Name the tools and Technologies used in creating a web Application.
6. What is the difference between Website and Web Application?
7. Create a XML Web Service using ASP.NET.
8. Write the main steps of calling the web services.
9. Why we expand Web Applications with Web Services?
10. Explain the different Layers of a Web Application.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (c) | 2. (d) | 3. (b) | 4. (d) |
| 5. (c) | 6. (b) | 7. (b) | 8. (a) |
| 9. (d) | 10. (a) | 11. (c) | 12. (a) |
| 13. (d) | 14. (b) | 15. (d) | |

11.6 Further Readings



Books

Practical ASP.NET 3.5 Projects for Beginners, by B.M. Harwani



Online links

ASP.NET, by Yashwanth Kanethkar <http://www.w3schools.com/aspnet/default.asp>

Unit 12: Security and Membership

Notes

CONTENTS

Objectives

Introduction

12.1 ASP.NET IIS Security

12.1.1 ASP.NET Security

12.1.2 ASP.NET Infrastructure and Subsystem Relationships, as Related to Security

12.2 ASP.NET Authentication

12.2.1 Security Relationship between IIS and ASP.NET

12.3 Summary

12.4 Keywords

12.5 Review Questions

12.6 Further Readings

Objectives

After studying this unit, you will be able to:

- Understand the IIS security
- Discuss the ASP.NET authentication

Introduction

Forms Authentication in ASP.NET can be a powerful feature. With very little code and effort, you can have a simple authentication system that is platform-agnostic. If your needs are more complex, however, and require more efficient controls over assets, you need the flexibility of groups. Windows Authentication gives you this flexibility, but it is not compatible with anything but Internet Explorer since it uses NTLM, Microsoft's proprietary authentication system. Now you must choose how to manage your assets: provide multiple login pages/areas and force users to register for each, or assign groups to users and limit access to pages/areas to particular groups. Obviously, you must choose the latter.

Role-based security in Forms Authentication is one thing Microsoft left out in this round for .NET, but they didn't leave you high-and-dry. The mechanisms are there, they're just not intuitive to code. This tutorial will cover the basics of Forms Authentication, how to adapt it to make use of role-based security, and how to implement role-based security on your site with single sign-ons.

Updated: With ASP.NET 2.0, Microsoft introduced built-in support for role membership. If you're using ASP.NET 2.0 or newer it's recommended you read Managing Authorization using Roles on MSDN. You can use an abstract data provider or create your own.

ASP.NET Membership builds on the success of the Forms authentication model from ASP.NET 1.x. ASP.NET Forms authentication provides a convenient way to incorporate a login form into your ASP.NET application and validate users against a database or other data store. The members of the Forms Authentication class make it possible to handle cookies for authentication, check

Notes

for a valid login, log a user out etc. However, implementing Forms authentication in an ASP.NET 1.x application can require a fair amount of code.

Membership in ASP.NET 2.0 is a major advancement over using Forms authentication alone. (Membership is most robust when coupled with Forms authentication, but using Forms authentication is not a requirement.) As you'll soon see, you can use ASP.NET Membership and the login controls in ASP.NET 2.0 to implement a powerful membership system without writing much code at all.

Membership is implemented by following five steps. Keep in mind that there are many sub-steps that are involved as well as optional configuration that can be implemented as well. These steps are meant to illustrate the big picture of configuring membership.

1. Create your membership database (if SQL Server is used as the membership store).
2. Specify the membership options in your applications configuration files. (Membership is enabled by default).
3. Determine the type of membership store you want to use. Options are:
 - a. Microsoft SQL Server (version 7.0 or later).
 - b. Active Directory Store.
 - c. Custom membership provider.
4. Configure the application for ASP.NET Forms authentication. Once again, Membership is designed to take advantage of Forms authentication, but using Forms authentication is not a requirement.
5. Define user accounts for membership and configure roles if desired.

12.1 ASP.NET IIS Security

IIS has its own security configuration and even for any request reaching the ASP.NET runtime, IIS verifies the request with it's own security configuration. So the first gatekeeper in the ASP.NET security pipeline is actually IIS. So let us understand those security mechanisms which IIS implements:

Authentication: IIS Support Following Authentication Mechanism

Basic authentication:

- Digest authentication
- Passport authentication
- Window authentication
- Certificate authentication

Points to remember:

- Any authentication which IIS performs results into an authenticated window user, so this means that IIS supports authenticating window users only.
- If ASP.NET is configured to support form or window authentication, then configure IIS to support basic or digest authentication.
- If ASP.NET is configured to support form or custom authentication, then configure IIS to support anonymous access.

- With XP, it comes with IIS 5.x
- With Server 2003, it is IIS 6.0

12.1.1 ASP.NET Security

When we are working on applications where authentication and authorization is a key requirement, then we will find the ASP.NET roles and membership feature very useful. Authentication means validating users. In this step, we verify user credentials to check whether the person trying to log in is the right one or not. Authorization on the other hand is keeping track of what the current user is allowed to see and what should be hidden from him. It is more like keeping a register to what to show and what not to show to the user.

Whenever a user logs in, he will have to authenticate himself with his credentials. Once he is authenticated, he will be authorized to see resources/pages of the website. Mostly these two concepts go together and ASP.NET provides us with some server controls that provide a lot of boilerplate functionality out of the box. If we use ASP.NET's authentication and authorization mechanism, then we can focus on what should be authorized and who should be authenticated rather than worrying about how to do that.

For the authorization part, Roles is the mechanism that ASP.NET uses to authorize users. Each user belongs to one or many roles and the web pages of our site are configured against roles. So if a user belongs to a role that is allowed to view a certain page, he will be able to:

Two types of Authentication:

1. **Windows authentication:** In this type, the users are authenticated on their Windows username and password. This method is least recommended in an Internet scenario. In an Internet scenario, we should always use "Forms based authentication".
2. **Forms based authentication:** In this type of authentication, the user will explicitly have to provide his credentials and these credentials, once verified by the server, will let the user to log in.



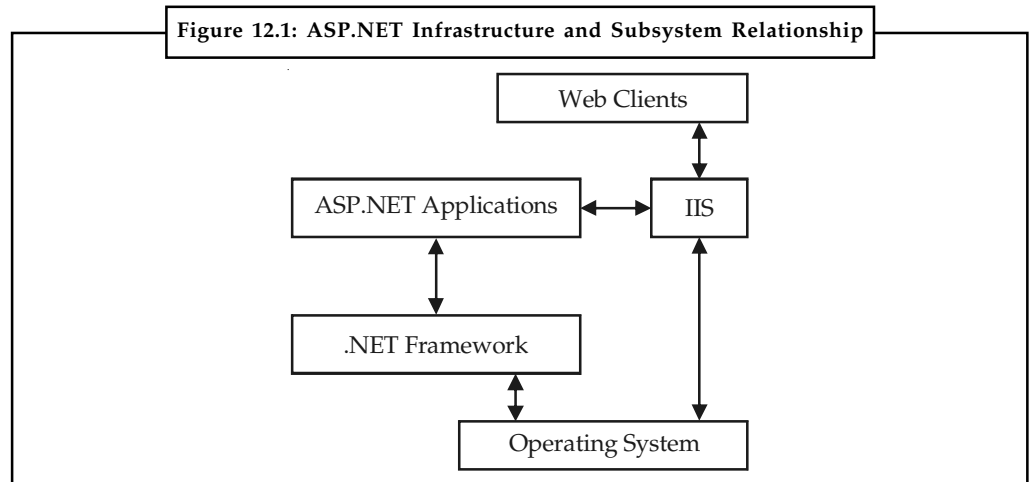
Task Write the steps to activate IIS server.

12.1.2 ASP.NET Infrastructure and Subsystem Relationships, as Related to Security

Web clients communicate with the Internet Information Services (IIS). The IIS decipheres and authenticates the user request. It examines whether the 'Allow Anonymous' property has been set to true authentication processes are bypassed, else they are set in motion. The IIS also searches for the requested resource and if the client is authorized it returns the resource else denies the resource. The IIS also assumes that a set of credentials is mapped to the Operating System account and uses them to authenticate the user. Different kinds of authentication is available in IIS 5.0 and 6.0 – the basic, the digest and the Integrated Windows Authentication. However, a detailed study on IIS authentication modes is beyond the purview of this tutorial and, therefore, only receives a mention here.

The security infrastructure and subsystem relationships of the ASP.NET are illustrated in Figure 12.1.

Notes



ASP.NET has its own security features and the application built on ASP.NET can have its own low level security features. When the IIS hands over the request to the ASP.NET application, the latter provides three kinds of authentication – forms authentication, Passport authentication and Windows authentication. The type of authentication required is declared in the configuration file of the application. If windows authentication is used the information about users and groups is stored in the Security Accounts Manager (SAM) database or in the Active Directory services. For Passport authentication, user information is stored in the internal passport database. Forms authentication allows the developer specify where to store the information.

Forms authentication is a system whereby unauthenticated requests are redirected to an HTML form using HTTP client-side redirection. The user provides his credentials and submits a form. If the data is authenticated by the application the system issues a ticket in a cookie that contains the credentials or key for reacquiring the identity. All subsequent requests are issued with the cookie in the request headers and they are authenticated and authorized by an ASP.NET handler using the validation method specified by the developer.

Passport authentication is a centralized authentication service provider that is made available by Microsoft. It offers a single logon and core profile services for member sites.

Windows Authentication is used in conjunction with Microsoft IIS authentication. This type of authentication is commonly used in Intranet scenarios. It provides a user interface and backend code needed to collect user inputs. The developer can be unaware of the data storage and validation of user roles. The identity of the application user is passed in from the IIS.

However, Passport or Windows authentication are not practical for real life scenarios where websites are extremely vulnerable to attacks of hackers and unauthorized entrants. The forms based authentication is considered the best protection for ASP.NET applications and as such has received much focus in ASP.NET 2.0.

On completion of the authentication by IIS, ASP.NET uses the authenticated identity to authorize access.

The sequence of events that occur when a authentication is sought is as under:

1. A client generates a request for a protected resource on the web.
2. The request is received by the IIS. The IIS checks whether the resource is authenticated by IIS or if Anonymous Access has been enabled for the resource. If yes, the request is passed on to ASP.NET application. If the ASP.NET application is set to forms authentication, IIS authentication is bypassed.

Notes

3. If the request does not have a cookie attached to it, ASP.NET redirects it to the Logon page. The configuration file of the application rests in this path. The client then, enters his credentials on the Logon page.
4. The credentials are checked by the application code using an event handler which checks the credentials. If authenticated a ticket is attached (a cookie) containing the username. If authentication fails the access denied message is sent to the user and the logon form is again presented to the user.
5. Once the ticket has been issued by the application, ASP.NET checks the ticket for validity using a message authentication check.
6. If the user has been authenticated, ASP.NET proceeds to check for authorization. It either provides access to the original resource requested or if the user does not have authorization for the resource, the application will redirect the user to another page which can be a custom authorization module where the credentials are again tested for authorization access. If authorization fails the user is redirected to the logon page.
7. If the user is authorized, access is granted.



Did u know? The first Microsoft web server was a research project at the European Microsoft Windows NT Academic Centre (EMWAC), part of the University of Edinburgh in Scotland, and was distributed as freeware.



Task Create a flow chart of ASP.NET authentication.

Self Assessment

Multiple Choice Questions:

1. In which of these is not a kind of basic Authentication?
 - a. Digest authentication
 - b. Passport authentication
 - c. Window authentication
 - d. Certified authentication
2. Which of the following is the process of granting access to the users based on identity?
 - a. Authentication
 - b. Authorization
 - c. Impersonation
3. The authorization settings in web.config overlap settings available in IIS.
 - a. True
 - b. False
4. authentication is a cookie/URL based authentication where username and password are stored on client machines as cookie files.
 - a. Passport
 - b. Forms
 - c. Windows
 - d. User
5. What is the default authentication mode for IIS?
 - a. Windows
 - b. Anonymous
 - c. Basic Authentication
 - d. None

12.2 ASP.NET Authentication

Authentication is the process of obtaining some sort of credentials from the users and using those credentials to verify the user's identity. Authorization is the process of allowing an authenticated user access to resources. Authentication is always precedes to Authorization; even if your application lets anonymous users connect and use the application, it still authenticates them as being anonymous.

ASP.NET provides flexible set of alternatives for authentication. You can perform authentication yourself in code or delegate authentication to other authorities (such as Microsoft Passport). In fact sometimes it seems ASP.NET authentication is a bit too flexible; it can be difficult for a new developer to know just where to start. In this article, we review the settings in ASP.NET and Internet Information Services (IIS) that control authentication and authorization in ASP.NET applications.

An ASP.NET application has two separate authentication layers. That is because ASP.NET is not a stand-alone product. Rather it is a layer on top of IIS. All requests flow through IIS before they are handed to ASP.NET. As a result, IIS can decide to deny access without the ASP.NET process even knowing that someone requested a particular page. Here is an overview of the steps in the joint IIS and ASP.NET authentication process.

1. IIS first checks to make sure the incoming request comes from an IP address that is allowed access to the domain. If not it denies the request.
2. Next IIS performs its own user authentication if it configured to do so. By default IIS allows anonymous access, so requests are automatically authenticated, but you can change this default on a per – application basis with in IIS.
3. If the request is passed to ASP.NET with an authenticated user, ASP.NET checks to see whether impersonation is enabled. If impersonation is enabled, ASP.NET acts as though it were the authenticated user. If not ASP.NET acts with its own configured account.
4. Finally the identity from step 3 is used to request resources from the operating system. If ASP.NET authentication can obtain all the necessary resources it grants the users request otherwise it is denied. Resources can include much more than just the ASP.NET page itself you can also use .Net's code access security features to extend this authorization step to disk files, Registry keys and other resources.

As you can see several security authorities interact when the user requests and ASP.NET page. If things are not behaving the way you think they should, it can be helpful to review this list and make sure you have considered all the factors involved.

Authentication Providers

Assuming IIS passes a request to ASP.NET, what happens next? The answer depends on the configuration of ASP.NET itself. The ASP.NET architecture includes the concept of and authentication provider a piece of code whose job is to verify credentials and decide whether a particular request should be considered authenticated. Out of the box ASP.NET gives you a choice of three different authentication providers.

1. The windows Authentication provider lets you authenticates users based on their windows accounts. This provider uses IIS to perform the authentication and then passes the authenticated identity to your code. This is the default provided for ASP.NET.
2. The passport authentication provider uses Microsoft's passport service to authenticate users.

3. The forms authentication provider uses custom HTML forms to collect authentication information and lets you use your own logic to authenticate users. The user's credentials are stored in a cookie for use during the session.

Selecting an authentication provider is as simple as making an entry in the web.config file for the application. You can use one of these entries to select the corresponding built in authentication provider:

```
<authentication mode="windows">
authentication mode="passport">
<authentication mode="forms">
```

ASP.NET also supports custom authentication providers. This simply means that you set the authentication mode for the application to none, then write your own custom code to perform authentication. For example, you might install an ISAPI filter in IIS that compares incoming requests to list of source IP addresses, and considers requests to be authenticated if they come from an acceptable address. In that case, you would set the authentication mode to none to prevent any of the .net authentication providers from being triggered.

Authentication using Windows accounts

If you plan to authenticate users using accounts maintained by a Microsoft Windows NT® domain controller or within Microsoft Windows® 2000 Active Directory™, you should use IIS Authentication coupled with the Windows Provider for ASP.NET. By using this approach, you do not need to write any specific authentication code. When authentication happens using this method, ASP.NET constructs and attaches a Windows Principal object to the application context based on the authenticated user. As a result, the ASP.NET thread can run as the authenticated user and can obtain the user's group membership.

In some cases, you may want to bypass IIS authentication and implement a custom solution. This is also possible with ASP.NET. For example, you can write a custom ISAPI filter that checks the user's credentials against Active Directory and the creation of the Windows Principal object would be performed manually. There are other methods besides this one that will work, but they all require more code than using the .NET provider directly.

Authentication using non-Windows accounts

If you are planning to authenticate users at the application level, and the users do not have Windows accounts, you will typically configure IIS to use Anonymous authentication. In this configuration, consider the following .NET authentication modules:

- **None:** Use when you are not authenticating users at all, or developing custom authentication code.
- **Forms:** Use when you want to provide users with a logon page.
- **Passport:** Use when you are using Passport services.

Impersonation and Delegation

With impersonation, ASP.NET applications can optionally execute with the identity of the client on whose behalf they're operating. Impersonation is usually performed for resource access control. You should carefully consider whether or not impersonation is required, as it consumes additional server resources. Delegation is a more powerful form of impersonation and allows remote resources to be accessed by the server process while acting as the client.

Notes

If impersonation is enabled, ASP.NET will receive the token to impersonate from IIS. You have more granular control of impersonation in a Web application when using ASP.NET in comparison to traditional Active Server Pages (ASP). This is controlled by specifying a value in the application's Web.config file.

You have the following three options when specifying the required impersonation setting:

1. **Impersonation enabled.** In this instance, ASP.NET will impersonate the token passed to it by IIS, which will either be an authenticated user or the anonymous Internet user account.

```
<identity impersonate="true" />
```

2. **Impersonation enabled, with a specific impersonation identity specified.** In this instance, ASP.NET will impersonate the token generated using the configured identity. In this case the client token, if applicable, is not used.

```
<identity impersonate="true" name="domain\user" password="pwd" />
```

3. **Impersonation is disabled.** This is the default setting for backward compatibility with ASP. In this instance, the ASP.NET thread will run using the process token of the application worker process, which by default is the IIS system account, regardless of which combination of IIS and ASP.NET authentication have been used.

```
<identity impersonate="false" />
```

If the application resides on a UNC share, ASP.NET will always impersonate the IIS UNC token to access that share unless a configured account is used. If an explicitly configured account is provided, ASP.NET will use that account in preference to the IIS UNC token.



Did u know? Web Site Administration tool was first introduced with ASP.NET 2.0 along with ASP.NET Microsoft Management Console (MMC) Snap-in.

12.2.1 Security Relationship between IIS and ASP.NET

IIS maintains security-related configuration settings in the IIS metabase. However, ASP.NET maintains security (and other) configuration settings in XML configuration files. While this generally simplifies the deployment of your application from a security standpoint, the security model adopted by your application will necessitate the correct configuration of both the IIS metabase and your ASP.NET application via its configuration file (Web.config).

IIS Security

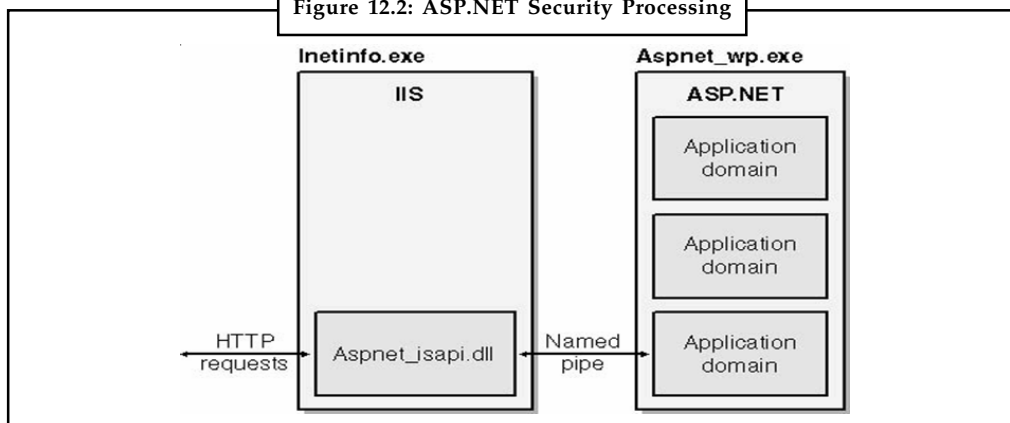
- IIS (Internet Information Services) Server:
 - ❖ a Web server
 - ❖ runs in process Inetinfo.exe as SYSTEM
 - ❖ accepts connections
 - ❖ responds to HTTP requests
- Web applications are deployed in application directories. Remote clients can't arbitrarily grab files outside application directories.
- IIS assigns every request an access token representing a Windows security principal. The access token enables the operating system to perform ACL checks on resources targeted.

- IIS supports IP address and domain name restrictions.
- IIS supports encrypted HTTP connections using the Secure Sockets Layer (SSL) family of protocols.
- Anonymous access (access by unauthenticated users).
- Request from anonymous users are tagged with IUSR_machinename's access token. IUSR_machinename is an Internet guest account created when IIS is installed, where machinename is usually the Web server's machine name.

ASP.NET Security

- Server Side Processing:
 - ❖ Client accesses .ASPX files =>
 - ❖ *Inetinfo.exe* (IIS) generates an access token => *Aspnet_isapi.dll* sends the request and the token through named pipe or local procedure calls (LPCs) =>
 - ❖ *Aspnet_wp.exe* (ASP.NET) makes ACL checks on the requested resource and passes access token to the targeted application =>
 - ❖ Targeted application uses a HTTP pipeline => HTTP modules => HTTP handlers (mapped in Machine.config).
- Two types of access tokens:
 - ❖ Authenticated user: authenticated security principal
 - ❖ Unauthenticated user: IUSR_machinename for anonymous login
 - ❖ Start->Settings->Control Panel->Administrative Tools->Computer Management ->Local Users and Groups->Users
 - ❖ Start->Settings->Control Panel->Administrative Tools->Computer Management ->Event Viewer->Security

Figure 12.2: ASP.NET Security Processing



Windows Integrated Authentication

Integrated Windows authentication uses Windows logon credentials to authenticate users. Rather than prompt a user for a user name and password and transmit them over HTTP, a browser asked to identify the user through integrated Windows authentication carries on a conversation with the Web server and identifies the user using that person's login identity on the client. In other words, if Bob logs in to his Windows PC, starts Internet Explorer, and requests a

Notes

resource protected by integrated Windows authentication, a handshake ensues between Internet Explorer and the Web server, and Bob's request executes as Bob on the server. Obviously, Bob has to be a valid account on the server (or in a domain that the server can authenticate against), or else access will be denied. Unless configured to do otherwise, the browser only asks for a user name and password if Bob is not a valid account on the Web server.

Integrated Windows authentication isn't an Internet standard; rather, it is a proprietary authentication protocol that permits Windows logon credentials to travel over HTTP. Its inner workings haven't been fully documented by Microsoft, although some details have been published by third parties. The details vary somewhat depending on the security provider being used, which can be either NTLM or Kerberos. In essence, however, the client and server negotiate a trust in a series of exchanges that involve user names, domain names, nonces, and hashes.

Here are the benefits regarding integrated Windows authentication. It provides a better user experience because it doesn't force users who have already logged into Windows to provide a user name and password again. Integrated Windows authentication is secure, even over unencrypted channels, because plain-text passwords are never transmitted.

There are two negatives to integrated Windows authentication. First, it works in Internet Explorer 2.0 and higher, but is unsupported by other browsers. Second, it's stopped dead in its tracks by most firewalls, primarily because the Windows security protocols that it negotiates – NTLM and Kerberos – rely on ports that are normally sealed off.

Integrated Windows authentication is a great solution for in-house networks that sit behind firewalls and whose browser clients can be carefully controlled – that is, restricted to Internet Explorer. It is poorly suited for general Internet use.

Passport Authentication

Passport authentication lets you to use Microsoft's passport service to authenticate users of your application. If your users have signed up with passport, and you configure the authentication mode of the application to the passport authentication, all authentication duties are off-loaded to the passport servers.

Passport uses an encrypted cookie mechanism to indicate authenticated users. If users have already signed into passport when they visit your site, they'll be considered authenticated by ASP.NET. Otherwise they'll be redirected to the passport servers to log in. When they are successfully log in, they'll be redirected back to your site.

To use passport authentication you have to download the Passport Software Development Kit (SDK) and install it on your server. It includes full details of implementing passport authentication in your own applications.

Default Authentication

Digest authentication is similar to basic authentication. When you attempt to access a resource guarded by digest authentication, the browser solicits a user name and password by popping up a dialog box. The Web server uses the credentials that you enter to assign an identity to the request. The big difference between basic and digest authentication is that digest doesn't transmit clear-text passwords. Instead, it passes an authentication token that is cryptographically secure. As a result, you can use it over unencrypted channels without fear of compromising your Web server.

The inner workings of digest authentication are also documented in RFC 2617. When the client first requests a resource guarded by digest authentication, the server returns a 401 error and

includes a “nonce” – a string of 1s and 0s – in a WWW-Authenticate header. The browser responds by prompting for a user name and password. It then transmits the user name back to the server, along with a hash or “digest” computed from the combined user name, password, and nonce. The server authenticates the request by performing its own hash on the user name, password, and nonce. The password the server uses doesn’t come from the client; it comes from the server itself (or from a connected server). If the hashes match, then the user is authenticated. Significantly, digest authentication never requires a plain-text password to be transmitted over an HTTP connection. It’s also compatible with proxy servers.

Digest authentication offers the following advantages. Like basic authentication, it provides an easily understood means for identifying callers, and it works with firewalls. In addition it’s far more secure over ordinary HTTP than basic authentication.

But there are three primary disadvantages. First, digest authentication requires a modern browser that supports digest authentication. For Microsoft Internet Explorer users, version 5.0 or higher is required. Second, it requires passwords to be stored in plain text (or in a reversible encrypted form that can be converted to plain text). This is contrary to the normal security model in Windows, which stores one-way password hashes in lieu of plain-text or encrypted passwords to protect the passwords if the server is compromised. Finally, like basic authentication, digest authentication uses pop-up dialog boxes to prompt for user names and passwords. Due to these restrictions, and because digest authentication doesn’t support delegation (the ability to make a call from one machine to another and have the call execute as the caller on the remote machine) on Windows 2000 servers, digest authentication is not widely used.

Forms authentication is a cookie/URL based authentication where username and password are stored on client machines as cookie files or they are sent encrypted on the URL for every request if cookies are not supported.

Below are the various steps which happen in forms authentication:

- **Step 1:** User enters “userid” and “password” through a custom login screen developed for authentication and authorization.
- **Step 2:** A check is made to ensure that the user is valid. The user can be validated from ‘web.config’ files, SQL Server, customer database, windows active directory and various other kinds of data sources.
- **Step 3:** If the user is valid then a cookie text file is generated on the client end. This cookie text file signifies that the user has been authenticated. Hence forth when the client computer browses other resources of your ASP.NET site the validation is not conducted again. The cookie file indicates that the user has logged in.

Forms Authentication Configuration

Add following Authentication setting into your web.config file under <system.web>.

```
< authentication mode = " Forms " >
< forms defaultUrl = " default.aspx " returnUrl = " ~/login.aspx "
slidingExpiration = " true " timeout = " 20 " ></ forms >
</ authentication >
```

For every user if you want to secure a particular folder, you can place setting for them either in parent web.config file (root folder) or web.config file of that folder.

Anonymous Authentication

With Anonymous authentication, the server does not request the client to send user credentials. It is a good choice when your site or service is publicly available and you do not need to know

Notes

the identity of the caller. Additionally, there are typically no browser restrictions which stem from incompatibilities with supported authentication mechanisms. When a site is configured for Anonymous authentication, all users are allowed access. It is important to note that although you may have IIS configured for Anonymous authentication, you may be authenticating at the ASP.NET layer, which is not true Anonymous authentication. This section assumes that both IIS and the application do not require a login.

Typical Usage Scenarios

You should consider Anonymous authentication when:

- You do not need to know the name and/or password of the caller for either login or business logic components.
- The information you are protecting is considered “public.”

You should not consider Anonymous authentication when:

- You are restricting your user base to require a login name and password.

Specify Role Settings for the Folder in Root web.config File (in This Case for Admin)

```
< location path = " Admin " >
< system.web >
< authorization >
< allow roles = " admin " />
< deny users = " * " />
</ authorization >
</ system.web >
</ location >
```

Write this code outside <system.web> but under <configuration> tag in the root’s web.config file. Here, I am specifying that if the path contains the name of folder Admin then only user with “admin” roles are allowed and all other users are denied.

Specify Role Settings for the Folder in Folder Specific web.config File (in This Case for User)

```
< system.web >
< authorization >
< allow roles = " User " />
< deny users = " * " />
</ authorization >
</ system.web >
```

Write this code into web.config file user folder. You can specify the setting for the user in root’s web.config file too, the way I have done for the Admin above. This is just another way of specifying the settings. This settings should be placed under <configuration> tag.

Specify setting for Authenticated user

```
< system.web >
< authorization >
< deny users = " ? " />
</ authorization >
</ system.web >
```

Write this code into web.config file of the Secure folder. This is specifying that all anonymous users are denied for this folder and only Authenticated users are allowed irrespective of their roles.



Caution No Authentication mode is not secure. If you enable No Authentication mode, debugging leaves your computer vulnerable to any user on the network. A hostile user can connect to your computer, launch applications on your computer, access data on your computer, and perform other mischievous or destructive actions by using a debugger.



Case Study

Advanced Search Mechanism for SharePoint-based Corporate Portal

Client

The Client is RosBusinessConsulting Group, large Russian company that operates on the mass media (an information agency, business television channel RBC TV, online newspapers, and marketing communications) and IT (RBC SOFT) markets.

Problem Background

The main goal of the project was to develop an efficient and convenient search mechanism based on corporate SharePoint (MS Windows SharePoint Services – WSS) portal and Yandex Server (most popular search engine in Russia) for RUSAL, United Company, the world's largest producer of aluminum and alumina.

The overwhelming majority of information resources of the customer are accumulated in the corporate portal based on Microsoft Windows SharePoint Services, portal PHP-based services and workflow system that uses Documentum enterprise content management platform. Microsoft SharePoint portal is a corporate standard at client's environment, and it is constantly extending and updating in the direction of gathering more information about staff personal data, phone numbers directory, results of staff tests, departments' workflow, projects documentation and a great deal of other corporate data.

The main points of Yandex Search – RUSAL project was a system development that allows company's staff to execute an effective information search within diverse corporate data resources, provide flexibility of adjustment to constant RUSAL IT infrastructure changes with the opportunity to expand functionality of search mechanism. The system should have a single entry point and be able to provide the required information with minimum delays and efforts.

Questions:

1. Give the name of Data Sources that should be used in search.
2. State the benefits for the customer.

Self Assessment

Multiple Choice Questions:

6. is the process of assigning a user account to an unknown user.
 - a. Impersonation
 - b. Authorization
 - c. Authentication

Notes

7. Which one of the authentication way identifies and authorizes users based on the server's user list?
 - a. Windows authentication
 - b. Forms authentication
 - c. Passport authentication
8. Which authentication type is best suited for multiple commercial web application?
 - a. Passport
 - b. Forms
 - c. Windows
 - d. Anonymous
9. Which authentication type should be used when you require to create your own database of users and validate the identity of those users when they visit your site?
 - a. Form
 - b. Passport
 - c. Windows
10. Which element of form authentication settings allows storing your user list in the web.config file?
 - a. Users
 - b. Credentials
 - c. Forms
 - d. Authentication
11. Digest Authentication provide following advantage.
 - a. it provides an easily understood means for identifying callers.
 - b. it works with firewalls.
 - c. it's far more secure over ordinary HTTP than basic authentication.
 - d. All of the above.
12. Which of the following is true about Windows Authentication in ASP.NET?
 - a. Automatically determines role membership.
 - b. Role membership determined only by user programming.
 - c. ASP.NET does not support Windows Authentication.
 - d. All of the above.
13. authentication is a centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites.
 - a. form
 - b. passport
 - c. windows
 - d. none
14. Which of the following is true about Windows Authentication in ASP.NET?
 - a. Automatically determines role membership.
 - b. Role membership determined only by user programming.
 - c. ASP.NET does not support Windows Authentication.
 - d. All of the above.
15. Passport authentication is a centralized authentication service, which Microsoft provides, that offers a single log on and core profile services for member sites.
 - a. True
 - b. False

12.3 Summary

Notes

- The user store contains user credentials such as user names and passwords, and in some cases, personalization information.
- Authentication is the process of obtaining identification credentials such as name and password from a user and validating those credentials against some authority.
- A system by which unauthenticated requests are redirected to an HTML form using HTTP client-side redirection. The user provides credentials and submits the form.
- Forms authentication provides us with a way to handle authentication using our own custom logic with in an ASP.NET application.
- ASP.NET Forms authentication provides a convenient way to incorporate a login form into your ASP.NET application and validate users against a database or other data store.
- IIS has its own security configuration and even for any request reaching the ASP.NET runtime.
- IIS support following authentication mechanism: Digest, Passport, Windows and Certificate Authentication.
- For the authorization part, Roles is the mechanism that ASP.NET uses to authorize users which is of two types: Windows Authentication and Form based Authentication.
- The ASP.NET helps control access to site information by comparing authenticated credentials, or representations of them, to NTFS file system permissions or to an XML file that lists authorized users, authorized roles (groups), or authorized HTTP verbs.
- When we install ASP.NET, the Machine.config file for server includes configuration elements that specify SQL Server membership providers.
- ASP.NET has its own security features and the application built on ASP.NET can have its own low level security features.
- Forms authentication is a system whereby unauthenticated requests are redirected to an HTML form using HTTP client-side redirection.
- Passport authentication is a centralized authentication service provider that is made available by Microsoft. It offers a single logon and core profile services for member sites.
- Windows Authentication is used in conjunction with Microsoft IIS authentication. This type of authentication is commonly used in Intranet scenarios.
- ASP.NET architecture includes the concept of an authentication provider a piece of code whose job is to verify credentials and decide whether a particular request should be considered authenticated.
- IIS maintains security-related configuration settings in the IIS metabase.
- With Anonymous authentication, the server does not request the client to send user credentials.

12.4 Keywords

Access-denied: If a request to ASP.NET application returns the error, "Denied Access to Directory Name directory. Failed to start monitoring directory changes.

Cookie: The topics in this topic describe how to create cookies in ASP.NET Web applications. Cookies are small text files that the server and browser exchange on each page request, and that we can use to store information that can help we customize your application for each user.

Notes

Cryptography: Means and methods for the transformation of data in order to hide its information content, prevent its undetected modification, or prevent its unauthorized use.

Internet Information Services (IIS): IIS is a group of Internet servers (including a Web or Hypertext Transfer Protocol server and a File Transfer Protocol server) with additional capabilities for Microsoft's Windows NT and Windows 2000 Server operating systems.

Lightweight: In information technology, the term lightweight is sometimes applied to a program, protocol, device, or anything that is relatively simpler or faster or that has fewer parts than something else.



- Lab Exercise*
1. Write the steps to authenticate a web form.
 2. Search about the authentication modes.

12.5 Review Questions

1. What is the authentication mode available in ASP.NET?
2. List out the difference between windows authentication and form authentication.
3. How do you impersonate the authenticated user in ASP.NET?
4. How do you set authentication mode in the ASP.NET application?
5. What happens when someone accesses a Web application that uses Forms authentication?
6. What are the advantages of storing user names and passwords in a database rather than a file?
7. What is the advantage of using Windows authentication in a Web application?
8. How do you allow or deny access to specific users using an authorization list from Web.config file, when using windows authentication?
9. Can you specify authorization settings both in Web.config and in IIS?
10. What is Passport Authentication? How does Passport authentication work?

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (d) | 2. (b) | 3. (a) | 4. (b) |
| 5. (b) | 6. (a) | 7. (a) | 8. (a) |
| 9. (a) | 10. (b) | 11. (d) | 12. (a) |
| 13. (b) | 14. (a) | 15. (a) | |

12.6 Further Readings



Books

Professional ASP.NET 3.5, by Bill Evjen



Online links

www.tutorialpoint.com/asp-net/asp.net-security.html

Unit 13: Adding E-commerce Essentials

Notes

CONTENTS

Objectives

Introduction

13.1 The XML Tools

13.2 Freight Calculations

13.2.1 Air Freight Calculation

13.2.2 Sea Freight Calculations

13.3 E-mail

13.3.1 Types of E-mail

13.3.2 How does E-mail Work?

13.3.3 Advantages of E-mail

13.3.4 Disadvantages of E-mails

13.4 Summary

13.5 Keywords

13.6 Review Questions

13.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Discuss the XML tools
- Define freight calculations
- Discuss the e-mail

Introduction

Contribution business information, maintain business associations and conduct business communication using computers associated to a telecommunication network is called E-Commerce.

There are mainly seven Essentials for Supporting E-Commerce:

Inventory Visibility and Synchronization

Nothing can spoil your relationship with your customers like allowing them to place an order for a product that is not available. One of the most important features of having an e-commerce site is to have your inventory accurately reflected in the shopping area so clients can purchase your products. Your fulfilment partner's warehouse management system must be able to update your shopping cart software with current inventory levels.

Notes

Keeping track of your stock with a warehouse management system (WMS) is essential because your inventory can become depleted from many different sources. Purchases and shipments can be initiated by shopping cart orders, salesperson samples, retail store orders, and wholesale/distribution channel orders to name a few.

Since these sources could quickly exhaust supply, it is important to have an automated and regularly scheduled inventory update from the WMS to the shopping cart. Typically, inventory levels are updated once a day or once per week depending on the popularity of the products. This ensures that out-of-stock items are not displayed (or are listed as such) to consumers visiting your online store. By accurately reflecting your inventory on your website, you can manage backorder issues and customer expectations.

Real-time Communication of Order Details Between Shopping Cart Software and WMS

Growth of online retail sales has been seen all year. From mega-sites like Amazon.com and e-Bay to hosting companies that service one person operations, online retailers have streamlined the buying process by using virtual “shopping carts” with a convenient “checkout” button to complete a transaction. Shopping cart software saves time and expenses by automating the buying/selling process.

Shopping cart software and warehouses primarily communicate through three methods:

1. E-mail Alerts.
2. Batch Processing by Manual Import/Export.
3. Automated Integrations (APIs and Web Service Calls) between systems.

The most basic method is e-mail alerts that are sent to the fulfilment house. This one-way notification is fine for drop shipments and partners who do not ship regularly, though message delivery is not always guaranteed.

Batch processing is the second method of integration. This method allows the fulfilment house to process multiple transactions by manually importing and exporting data files between systems. This requires user interaction to import and export flat files to move data between systems. Batch processing is inexpensive and familiar to most experienced computer users. Flat file imports and exports can also be implemented very quickly. Keep in mind though: batch processing creates delays in the information pipeline.

Batch processing may require you to import and export a new file for each shopping cart. While retyping order data may not be necessary, you will have to manage the inbound and outbound file traffic for each of your customers, and this user interaction can delay fulfilment and status updates.

Application Programming Interfaces, aka: APIs or Web Services, provide a real-time way to integrate shopping carts with your warehouse management system that eliminates user interaction. APIs and Web Service Calls link your customers’ shopping carts with the warehouse management system in real time. As orders are filled, APIs create an automatic two-way link that sends order information to the WMS and updates the shopping cart with status and tracking information. Web Service calls are also seamless to the consumer because their systems are simultaneously updated. This process means a reduction the lag time for information sharing and an extension of your customers’ service and visibility.

Handling Larger Order Volumes

Notes

Operational inefficiencies can become more apparent as order volumes increase. Nothing difficult is done in the warehouse, but the sheer volume of orders inherently becomes challenging. When looking at your fulfilment partner, make sure they have solid Standard Operating Procedures (SOPs) for picking orders. Look at their warehouse management system, does it optimize workflow and enforce their picking rules? Limiting movement and repetition in the warehouse is key to improving productivity and ensuring that orders are picked in a timely fashion.

Another thing to look at are the pick line locations and the pick location priority settings. Do they direct the staff to the most logical items to pick based on shortest distance, least touches, package configuration or a number of other criteria? When set up correctly, the WMS can automatically allocate product from these optimized locations.

Increased Pick Accuracy (Barcode Scanning)

Today's customers expect to receive the correct product in a timely fashion. In a time when accuracy is not something to be compromised, it is good to know that the best fulfilment centres utilizing barcode scanning technology at the time of pick-and-pack can dramatically increase accuracy over the old pick-to-paper model.

Barcode scanning functionality directs a warehouse picker to the correct inventory location as well as verifying the picked product. It is critical to have the capability to immediately verify that the correct SKU was selected at the time the product is picked. Having this real-time verification based on the Item Number or UPC barcode provides immediate feedback to the picker. Barcode scanning is one of the best ways to increase shipment accuracy resulting in happier customers and consumers.

With the growth of online sales increasing substantially over the next 5 years, e-commerce is one of the best ways to make yourself and your business accessible all the time. Providing accurate book fulfilment services is key to keep customers happy and increase customer loyalty to your brand.

Custom Branded Packing Slips and Integration with Different Shipping Services

It is important to make the customer feel special when he or she chooses one of your products, because your brand is everything. One way to create a better brand experience is by utilizing customized cartons and packing lists. As an online retailer, you have only a few opportunities to touch the client. That is why it is so important, maybe even a requirement, to customize shipper cartons and packing slips with your company name, logo and website address. This ensures that each time one of your clients receives their order he or she will know who you are.

Order fulfilment is not complete until the packages have been shipped. In today's market, consumers want choices in shipping methods. As an e-commerce retailer, which shipper works best – UPS, FedEx or the United States Postal System (USPS)? Under certain conditions, each carrier has strengths that make them a more appropriate choice.

UPS and FedEx are the largest package carriers with competitive rates to commercial addresses. When it comes to residential addresses, the cost dramatically increases because the carriers do not offer the same discount structure for residential deliveries. USPS is very competitive in this arena.

If you are shipping lighter-weight cartons (under 1 pound, maximum of 15 pounds) other services such as UPS Mail Innovations or FedEx SmartPost are appropriate and convenient.

Notes

The fulfilment that you choose centre needs to have the flexibility to ship via the medium that is most beneficial to you and your clients.

Electronic Data Interchange

In making a selection of a fulfilment centre, double check to make sure they can support your electronic data interchange (EDI) initiatives. The fulfilment centre must be able to generate the SSCC-18 Carton License Plate and affix the label to the carton. From there, the fulfilment house must send you a data set to meet your customers' ASN requirements. To identify this file, know that it will contain the SSCC-18 license number and the contents of that specific carton. This completes the information required for the ASN. Finally, the fulfilment warehouse will generate the UCC-128 label to be affixed to the cartons completing the ASN transaction.

Automatic E-mail Notifications to Customers

E-mail notifications take many different forms. When someone places an order online at an e-commerce site, the expectation is that they will receive an order confirmation at the time the order is placed and a shipping notification when the order is fulfilled by the fulfilment centre.

As a client of the fulfilment centre, you will have a need for various e-mail notifications to fit your company's needs. Automatic e-mail notifications are generated:

1. When an order file is sent confirming receipt,
2. When new product arrives in the fulfilment centre,
3. Low stock notification to begin the reorder process,
4. Weekly to advise of any stock adjustment,
5. Monthly for inventory reports.

As an e-retailer, you know one of the core functions you and your customers rely on is the availability of real-time, accurate information. Customers demand immediate details and their individual expectations will not all be the same. Some will want information "pushed" to them via automatic e-mail updates, while others will want to "pull" it themselves by accessing their own online reporting. Make sure your fulfilment centre offers you the information you need in the ways that you want to see it.

Communication is at the heart of any relationship with your fulfilment centre, or 3PL, and good communication is the difference between a good provider and a great provider. In order to be on top of your game, your best defense is good communication, and the best communication is delivered automatically and sent at the time the task is completed.

13.1 The XML Tools

eXtensible Markup Language (XML) is a markup language used to structure data to make it easy to exchange data over the Internet. The compatibility of XML with various other languages, such as Java and C++, has given rise to various tools and utilities such as XML editors, parsers, and browsers that help structure data in an XML document. Most of these XML tools conform to the World Wide Web Consortium's XML 1.0 standards and specifications and can be downloaded free of charge.

The XML tools and utilities enable you to structure, validate, debug, read, and view an XML document and create XML schemas.

There are various tools of XML are given below:

Notes

Xerces

- Validating XML parser
- SAX support
- DOM support
- XML Schema support
- JAXP support

Xalan

- XSLT engine
- XPath support

JDOM

- Java interface to SAX or DOM based parsers

Other popular XML Processing Tools:

- Saxon – popular XSLT engine

Java XML Data Binding Tools

- Castor – open source, although lacks complete schema support
- JAXB – Java Community Process based, provides schema support

DTD Validation

The WFS Specification mandates the use of XML Schema as a minimum for all operations. Other output formats for Get Feature, such as DTD, are possible if specified by DescribeFeatureType and GetCapabilities.

DTD Validation Tools:

- XSDValid – contains dtdvalid DTD validation tool



Did u know? The XSD 1.0 specification was originally published in 2001, with a second edition following in 2004 to correct large numbers of errors. XSD 1.1 became a W3C Recommendation in April 2012.



Caution When specified, the generate XML command writes the resulting XML data to a file otherwise any existing file will be overwritten without warning.



Task Write an XML program to validating DTD.

13.2 Freight Calculations

A charge paid for wagon or transport of goods by air, land, or sea. Goods may be on cloud nine on freight-prepaid or freight-collect basis:

1. If the freight is paid by the consignor (as under C&F as well as CIF terms) the goods remain the consignor's property until their delivery is taken by the consignee upon their arrival at the destination, as well as payment of the consignor's invoice.
2. If freight is compensated by the consignee (as under FOB terms) the goods become the consignee's property when handed over towards the carrier against a waybill. It may be charged on the weight or volume of the delivery (depending upon its nature or density) and additionally varies according to the mode of shipment, for example bulk, break bulk, as well as containerized. It is also called freightage.

A freight forwarder, or even forwarding agent is a person or company that organizes shipments for people or corporations to get large orders from the manufacturer or producer to market or final point of distribution. Forwarders will contract with a carrier in order to facilitate the movement of products. A forwarder is not typically a carrier, but is a specialist in supply chain management. Quite simply, a freight forwarder is a "travel agent," for the freight industry, or a third-party (non-asset-based) logistics supplier. A forwarder will contract along with asset-based carriers to move cargo ranging from raw agricultural products to manufactured goods. Freight can be booked on a variety of carrier types, including ships, airplanes, trucks, and railroads. It is not unusual for a shipment to maneuver along its route on multiple carrier types.

International freight forwarders typically arrange cargo movement to an international destination. International freight forwarders, have the expertise that allows them to prepare and process the documentation as well as perform related activities pertaining to international shipments. Some of the normal information reviewed by a freight forwarder is the commercial invoice, shipper's export declaration, bill of lading, and other documents required by the carrier or country of export, import, or trans-shipment. Much of the information is now processed inside a paperless environment.

13.2.1 Air Freight Calculation

Air carriers that are members of the International Air Transport Association (IATA) are bound by their relationship in order to abide by tariff issued by IATA. Although since 11th September 2002, airfreight rates are now tremendously negotiable. Airfreight rates cover transport from the airport associated with loading to the airport of discharge.

These rates do not include the following:

- Collection of air cargo from the consignor's/exporters premises
- Delivery of cargo from the airport of destination to the consignee's premises
- Storage of cargo before or after loading
- Customs clearance in the country of destination
- Any duties and taxes that may have to be paid
- Insurance

Chargeable/Volumetric Weight

Airline freight prices are based on a "chargeable weight", since the volume or weight that may be loaded into an plane is limited. The chargeable weight of a shipment will be either the "actual gross mass " or even the "volumetric weight", whichever may be the highest. The chargeable

weight is calculated as comes after: 1 metric ton = 6 cubic metres. In order to establish if the actual cargo will be the weight or volumetric dependent shipment.

Step 1: Measure the parcel/cargo along the greatest length, width and height of that parcel. For example; 100 cm (L) × 100 cm (W) × 100 cm (H) = 1 000 000 cm³. Next, weigh the parcel; assume it weighs 150 kg.

Step 2: Now divide the 1 000 000 cm³ by 6,000 = 166,66 kg. You have now converted the centimeters (cm) into kilograms (kg).

Step 3: Now compare the weight to the volume. If the weight is 150 kg then the airline would base the freight on the higher amount being: 166,66 kg.

Calculations

The airline calculates freight based on weight or volume, whichever yields the greatest amount. Airlines quote freight rates based on the following rate structures:

- A basic minimum charge per shipment.
- General cargo rates quoted for per kilogram. This rate applies without reference to the nature or description of the parcel, which is to be freighted.
- Specific commodity rates apply to certain goods of specific descriptions, such as fresh produce. These rates are lower than the general cargo rate, and they provide breakpoints at which the level of the rate reduces further.



Example: 0 - 50 kg @ ₹ 22.00 per kg

50 - 100 kg @ ₹ 19.00 per kg

100 - 150 kg @ ₹ 17.00 per kg

Unit Load Device Charges

These rates are charged per container/ULD without reference to the commodity loaded therein. Calculation of freight rates:

Let us assume the following figures:

The freight rate is ₹ 18.00 per kg

The weight of the parcel is 300 kg

The dimensions are: 114,6 cm × 120,4 cm × 132,5 cm (round the cm's up or down)

Therefore: 115 cm × 120 cm × 133 cm = 1 835 400 divide by 6 000 = 305.9 kg (having converted cm's to kg's now round up the kg's to the next half a kilogram = 306 kg)

As the freight rate quoted by the airline is ₹ 18.00 per kg, we calculate the price as follows:

$$306 \text{ kg} \times ₹ 18/\text{kg} = ₹ 5 508.00$$

The freight rate will not be calculated on the actual mass 300 kg × ₹ 18.00 = ₹ 5 400.00 as the airline will always use the greater amount either the kg, or volumetric weight.

Consolidation

Consolidation is an economical method of moving cargo by employing a consolidator. The consolidator receives cargo from a number of suppliers/shippers and then combines these

Notes

cargoes into one consignment by packing the goods into a Unit Load Device. The consolidator then books the Unit Load Device with an airline. The supplier/shipper would have a contract of carriage with the consolidator of the cargo and in turn the airline would have a contract of carriage with the consolidator. The airline would issue an air waybill to the consolidator when accepting the Unit Load Device and in turn the consolidator would issue the supplier/shipper with a house air waybill.

The Air Waybill

The air waybill, unlike the ocean bill of lading is not a document of title to the goods described therein, however it does perform several similar functions these are:

- It is a receipt for the goods
- It is evidence of the contract of carriage between the exporter and the carrier
- It incorporates full details of the consignor/shipper, the consignee/receiver and the consignment/goods
- It is an invoice showing the full freight amount
- It must be produced, be it in an electronic format, at the airport of discharge for clearing purposes

All copies of the air waybill, together with the commercial invoice, packing list, certificate of origin and any other document which may be necessary for clearing the goods through customs, these documents are carried in the flight captain's bag.

13.2.2 Sea Freight Calculations

Seafreight calculations can broadly be divided into two main components; breakbulk and containerised. In this section we deal with how you should calculate the freight costs of both of these two types of seafreight.

Break Bulk Cargo Calculations

Break bulk cargo, is cargo that is unitised, palletised or strapped. This cargo is measured along the greatest length, width and height of the entire shipment. The cargo is also weighed. Shipping lines quote break bulk cargo per "freight ton", which is either 1 metric ton or 1 cubic metre, whichever ever yields the greatest revenue.



Example: A case has a gross mass of 2 Mt.

The dimensions of the cargo are: $2.5 \times 1 \times 2$ metres

The tariff rate quoted by the shipping line is: ₹ 110.00 weight or measure (freight ton).

Step 1: Multiply the metres $2.5 \times 1 \times 2 = 5$ metres. Compare to the mass = 2 Mt.

Step 2: Calculate the freight with the greater amount either the mass or the dimension.

$$5 \times ₹ 110.00 = ₹ 550.00$$

Freight would be paid on the measurement and not the weight. All shipping lines carrying cargo in a break-bulk form insist on payment based on a minimum freight charge which is equivalent to one freight ton, one cubic metre or one metric ton.

Full Container Load Calculations and Surcharges

Notes

Freight rates for containers are based on the container as a unit of freight irrespective of the commodity or commodities loaded therein, Freight All Kinds (FAK). The shipping lines quote per box (container) either a six or twelve metre container. From time to time, abnormal or exceptional costs arise in respect of which no provision has been made in the tariffs. For example a shipping line cannot predict the movement of the US Dollar or the sudden increase of the international oil price. These increases have to be taken into account by the shipping line in order to ensure that the shipping line continues to operate at a profit. These increases are called surcharges. All shipping lines accordingly retain the right to impose an adjustment factor upon their rates taking into account these fluctuations. All surcharges are expressed as a percentage of the basic freight rate. Surcharges are regularly reviewed in the light of unforeseen circumstances, which may arise and bring cause for a surcharge increase.

Bunker Adjustment Factor (BAF)

“Bunkers” is the generic name given to fuels and lubricants that provide energy to power ships. The cost of bunker oil fluctuates continually and with comparatively little warning.

Freight rate: Port Elizabeth to Singapore

Freight rate: ₹ 1 250.00 per 6-M container + BAF 5.2%

$₹ 1\ 250.00 \times 5.2\% = ₹ 65.00$

Add the two amounts together

Freight rate: ₹ 1 315.00

Currency Adjustment Factor (CAF)

The currency adjustment factor is a mechanism for taking into account fluctuations in exchange rates, these fluctuations occur when expenses are paid in one currency and monies earned in another by a shipping company. The currency adjustment factor is a mechanism for taking into account these exchange rate fluctuations. It is always expressed as a percentage of the basic freight and is subject to regular review.

Self Assessment

Multiple Choice Questions:

-functionality directs a warehouse picker to the correct inventory location as well as verifying the picked product.
 - Barcode scanning
 - Custom Branded Packing Slips
 - Handling Larger Order Volumes
 - None of these
-Tool is found for XML Schema validation.
 - WBS
 - SPY
 - SQC
 - None of these
- Which of these is not the rule in XML?
 - All attributes of an element must be enclosed within quotes
 - Tags should not overlap

Notes

- c. Tags are not case-sensitive
- d. Tags must be closed
- 4.specifies the SOAP extension to run when a service description is generated for all XML Web service within the scope of configuration file.
 - a. Service Description Format Extension Types
 - b. Soap Extension Types
 - c. Soap Extension Reflector Types
 - d. Soap Extension Importer Types
- 5. Which is not an XML tool?
 - a. Xerces
 - b. Serax
 - c. SQC
 - d. Xalan

13.3 E-mail

E-mail (electronic mail) is the exchange of computer-stored messages by telecommunication. (Some publications spell it e-mail; we prefer the currently more established spelling of e-mail.) E-mail messages are usually encoded in ASCII text. However, you can also send non-text files, such as graphic images and sound files, as attachments sent in binary streams. E-mail was one of the first uses of the Internet and is still the most popular use. A large percentage of the total traffic over the Internet is e-mail. E-mail can also be exchanged between online service provider users and in networks other than the Internet, both public and private.

E-mail can be distributed to lists of people as well as to individuals. A shared distribution list can be managed by using an e-mail reflector. Some mailing lists allow you to subscribe by sending a request to the mailing list administrator. A mailing list that is administered automatically is called a list server.

E-mail is one of the protocols included with the Transport Control Protocol/Internet Protocol (TCP/IP) suite of protocols. A popular protocol for sending e-mail is Simple Mail Transfer Protocol and a popular protocol for receiving it is POP3. Both Netscape and Microsoft include an e-mail utility with their Web browsers.

Interactions between e-mail servers and clients are governed by e-mail protocols. The three most common e-mail protocols are POP, IMAP and MAPI. Most e-mail software operates under one of these (and many products support more than one). The most important reason for knowing of their existence. To understand that the correct protocol must be selected, and correctly configured, if you want your e-mail account to work.

The Post Office Protocol (currently in version 3, hence POP3) allows e-mail client software to retrieve e-mail from a remote server. The Internet Message Access Protocol (now in version 4 or IMAP4) allows a local e-mail client to access e-mail messages that reside on a remote server. There's a related protocol called SMTP, which we also discuss below:

The Messaging Application Programming Interface (MAPI) is a proprietary e-mail protocol of Microsoft, that can be used by Outlook (Microsoft's e-mail client software) to communicate with Microsoft Exchange (its e-mail server software). It provides somewhat more functionality than an IMAP protocol; unfortunately, as a proprietary protocol, it works only for Outlook – Exchange interactions.

POP

Notes

POP is the older design, and hails from an era when intermittent connection via modem (dial-up) was the norm. POP allows users to retrieve e-mail when connected, and then act on the retrieved messages without needing to stay “on-line.” This is an important benefit when connection charges are expensive.

The basic POP procedure is to retrieve all inbound messages for storage on the client, delete them on server, and then disconnect. (The e-mail server functions like a mailbox at the Post Office – a temporary holding area until mail gets to its final destination, your computer.)

Outbound mail is generated on the client, and held for transmission to the e-mail server until the next time the user’s connection is active. After it’s uploaded, the server forwards the outgoing mail to other e-mail servers, until it reaches its final destination.

Most POP clients also provide an option to leave copies of e-mail on the server. In this case, messages are only removed from the server when greater than a certain “age” or when they have been explicitly deleted on the client. It’s the copies on the client that are considered the “real” ones, however, with those left on the server merely temporary backups.

IMAP

IMAP is the newer protocol and oriented toward a “connected” mode of operation. The standard IMAP procedure is to leave messages on the server instead of retrieving copies, so e-mail is only accessible when “on-line.”

IMAP is more suited to a world of always-on connections, particularly the fast connections offered by broadband mechanisms. Having to be connected to read your e-mail is a trivial obstacle when the connection is always available. (It’s a little like leaving your messages at the Post Office, and going there every time you want to read them. That might be difficult in the physical world, but it’s easy in the virtual one.)

Because messages remain on the server, until explicitly deleted by the user, they can be accessed by multiple client computers – an important advantage when you use more than one computer to check your e-mail.

IMAP does not preclude keeping copies on the client, but, in an inversion of the way POP works, it’s the server’s copies that are considered the “real” ones. That offers an important security benefit – you won’t lose your e-mail if, for some reason, your client computer’s storage media fails.

IMAP has other advantages over POP (detailed in the links provided below). It is the standard we recommend if you can’t use MAPI.

SMTP

At the risk of overloading you with information, you should know that strictly speaking it’s only the incoming mail that is handled by a POP or IMAP protocol. Outgoing mail for both POP and IMAP clients uses the Simple Mail Transfer Protocol (SMTP).

When you set up a POP or IMAP e-mail account on e-mail client software, you must specify the name of the (POP or IMAP) mail server computer for incoming mail. You must also specify the name of the (SMTP) server computer for outgoing mail. These names are typically in the same form as Web addresses (e.g., “imap.med.miami.edu”). Depending on the client, there may also be specifications for e-mail directories and searching.

Notes

MAPI

As noted, MAPI is Microsoft’s proprietary e-mail protocol. It provides greater functionality than IMAP for Outlook e-mail clients interacting with an Exchange e-mail server. It doesn’t work for anything else. (In Outlook you may simply see the connection option “Microsoft Exchange Server” rather than MAPI. It’s offering the same thing.)

Remote access using MAPI may require use of a VPN connection, because the ports (communications channels) that MAPI uses are otherwise blocked for security reasons. (That’s the case when accessing the medical campus Exchange system remotely.)

Web Browser e-mail Access

Many e-mail systems can now be accessed using only a Web browser. There is no need to install client e-mail software of any kind. Logically, Web browser interfaces to e-mail are like IMAP, in that all the messages remain on the server unless explicitly deleted. (Message copies can be saved on the client computer.)



Example: The medical campus Exchange e-mail system can be accessed by Outlook Web Access (OWA); it provides most of the functionality of an installed version of Outlook. OWA is compatible with most browsers, such as Firefox, Netscape, Opera or Safari, though it works best with Microsoft’s Internet Explorer browser.

13.3.1 Types of E-mail

There are three types of E-mail:

Marketing E-mails

Definition

Marketing (or Bulk) e-mails stimulate your clients and leads. They contain informative/incentive messages. The recipient must agree to receive such e-mails: opt-in is mandatory.

However, the recipient does not make an explicit request for a message in particular. For example: he doesn’t subscribe for the “November Newsletter”, he rather subscribes to the “Monthly Newsletter”.

There can be a periodicity, but the message can also be sent to a segment in particular. Nonetheless, the main point is that messages are sent independently from specific actions from the clients.

With Marketing e-mails, you don’t state: “such clients will receive such message if they perform such action”: it would then be a notification e-mail (see below). We would rather state: “such clients will receive such message”. The message is neither related to an “event” nor to an action from the recipient.



Example: Common examples of marketing e-mails:

- Newsletters
- Flash sales
- Sales/promotions announcements

Notification E-mails

Notes

Definition

Notification e-mail are also known as trigger, alert or auto-responder. They allow the user to be notified each time a particular event happens (or has happened). More generally, the notification e-mail may be used in order to celebrate and/or mark an event.

It is either the recipients themselves, or the sender that will establish criteria for an outgoing message to be triggered by an event.

From a marketer's point of view, it can be relevant to encourage the targets to opt in to receive notifications about the services being offered. Think of an e-mail such as "Mr. X is now following you on Twitter". This kind of message is more often opened and it motivates the recipient into checking their account.

You may also decide to create notifications by yourself. These will then be related to events occurring after a client's action. The purpose here is to mark a significant event in order to capitalize on it.



Example: Common examples of notification e-mails:

- Getting in touch a few days after registration
- Congratulations after a status change (first purchase, subscription...)
- Birthday e-mail
- Shopping Cart Abandonment e-mail
- Goods back in stock
- Discounts on recently browsed products
- Greetings after a purchase
- Feedback request after a purchase (product, service...)
- E-mail following up a purchase and proposing other items
- Series of greeting messages and/or hands-on account management messages

Transactional E-mails

Definition

This is an expected message and its content is information that the client wishes to check or confirm, and not "discover". This type of e-mail is not intended to optimize the customer relationship but to define it and mark it out. It is a point of reference in one's CRM.

Strong attention should be paid to this kind of e-mail. These e-mails must be specifically dealt with: wrongly delivered newsletters might impact leads, but an undelivered transactional e-mail will upset the customer.



Example: Common examples of transactional e-mails:

- Welcome message/Account opening
- Shipment tracking and order status

Notes

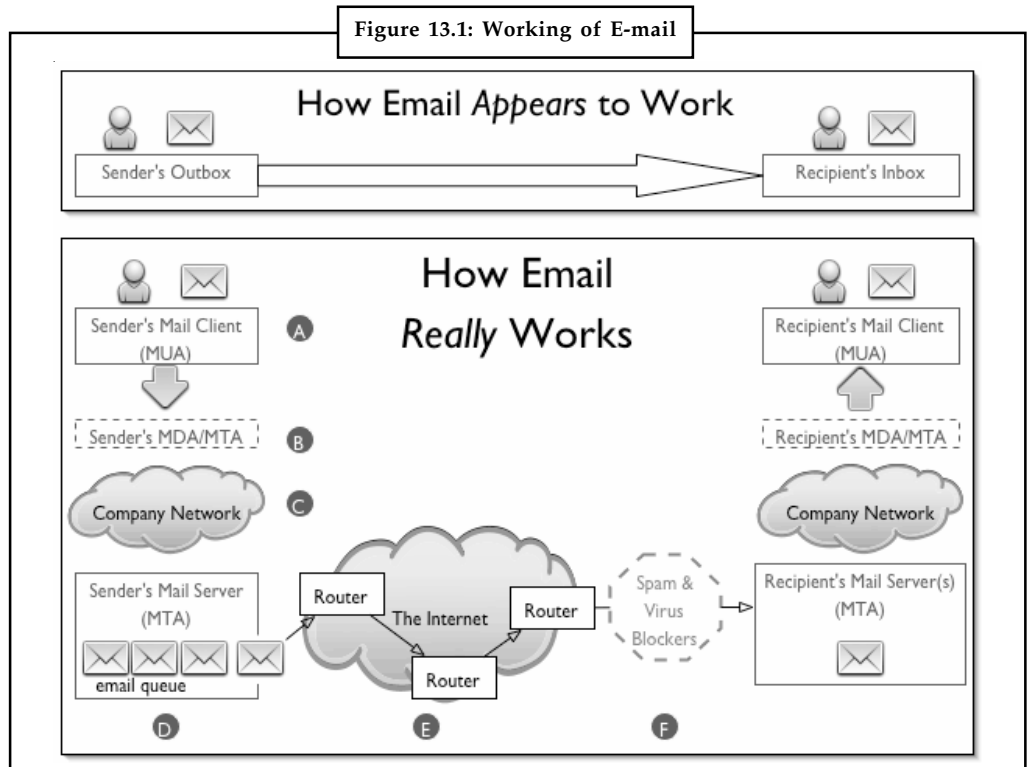
- Order shipment confirmation
- Account termination
- Payment confirmation
- Invoice

13.3.2 How does E-mail Work?

From the user standpoint, e-mail seems so simple. You set the e-mail address of the person to whom you want to send the e-mail, compose your message and click ‘Send’.

All done.

In reality, sending your message off into the network cloud is a bit like sending Little Red Riding Hood into the deep dark woods. You never know what might happen.



In Figure 13.1, the sender is a human being using their company account to send an e-mail to someone at a different company.

Step A: Sender creates and sends an e-mail

The originating sender creates an e-mail in their Mail User Agent (MUA) and clicks ‘Send’. The MUA is the application the originating sender uses to compose and read e-mail, such as Eudora, Outlook, etc.

Step B: Sender’s MDA/MTA routes the e-mail

The sender’s MUA transfers the e-mail to a Mail Delivery Agent (MDA). Frequently, the sender’s MTA also handles the responsibilities of an MDA. Several of the most common MTAs do this, including sendmail and qmail (which Kavi uses).

The MDA/MTA accepts the e-mail, then routes it to local mailboxes or forwards it if it isn't locally addressed.

In Figure 13.1, an MDA forwards the e-mail to an MTA and it enters the first of a series of "network clouds," labeled as a "Company Network" cloud.

Step C: Network Cloud

An e-mail can encounter a network cloud within a large company or ISP, or the largest network cloud in existence: the Internet. The network cloud may encompass a multitude of mail servers, DNS servers, routers, lions, tigers, bears (wolves!) and other devices and services too numerous to mention. These are prone to be slow when processing an unusually heavy load, temporarily unable to receive an e-mail when taken down for maintenance, and sometimes may not have identified themselves properly to the Internet through the Domain Name System (DNS) so that other MTAs in the network cloud are unable to deliver mail as addressed. These devices may be protected by firewalls, spam filters and malware detection software that may bounce or even delete an e-mail. When an e-mail is deleted by this kind of software, it tends to fail silently, so the sender is given no information about where or when the delivery failure occurred.

E-mail service providers and other companies that process a large volume of e-mail often have their own, private network clouds. These organizations commonly have multiple mail servers, and route all e-mail through a central gateway server (i.e., mail hub) that redistributes mail to whichever MTA is available. E-mail on these secondary MTAs must usually wait for the primary MTA (i.e., the designated host for that domain) to become available, at which time the secondary mail server will transfer its messages to the primary MTA.

Step D: E-mail Queue

The e-mail in the diagram is addressed to someone at another company, so it enters an e-mail queue with other outgoing e-mail messages. If there is a high volume of mail in the queue – either because there are many messages or the messages are unusually large, or both – the message will be delayed in the queue until the MTA processes the messages ahead of it.

Step E: MTA to MTA Transfer

When transferring an e-mail, the sending MTA handles all aspects of mail delivery until the message has been either accepted or rejected by the receiving MTA.

As the e-mail clears the queue, it enters the Internet network cloud, where it is routed along a host-to-host chain of servers. Each MTA in the Internet network cloud needs to "stop and ask directions" from the Domain Name System (DNS) in order to identify the next MTA in the delivery chain. The exact route depends partly on server availability and mostly on which MTA can be found to accept e-mail for the domain specified in the address. Most e-mail takes a path that is dependent on server availability, so a pair of messages originating from the same host and addressed to the same receiving host could take different paths. These days, it's mostly spammers that specify any part of the path, deliberately routing their message through a series of relay servers in an attempt to obscure the true origin of the message.

To find the recipient's IP address and mailbox, the MTA must drill down through the Domain Name System (DNS), which consists of a set of servers distributed across the Internet. Beginning with the root nameservers at the top-level domain (.tld), then domain nameservers that handle requests for domains within that .tld, and eventually to nameservers that know about the local domain.

Notes

DNS Resolution and Transfer Process

- There are 13 root servers serving the top-level domains (e.g., .org, .com, .edu, .gov, .net, etc.). These root servers refer requests for a given domain to the root name servers that handle requests for that tld. In practice, this step is seldom necessary.
- The MTA can bypass this step because it already knows which domain name servers handle requests for these .tlds. It asks the appropriate DNS server which Mail Exchange (MX) servers have knowledge of the subdomain or local host in the e-mail address. The DNS server responds with an MX record: a prioritized list of MX servers for this domain.
- An MX server is really an MTA wearing a different hat, just like a person who holds two jobs with different job titles (or three, if the MTA also handles the responsibilities of an MDA). To the DNS server, the server that accepts messages is an MX server. When transferring messages, it is called an MTA.
- The MTA contacts the MX servers on the MX record in order of priority until it finds the designated host for that address domain.
- The sending MTA asks if the host accepts messages for the recipient's username at that domain (i.e., username@domain.tld) and transfers the message.

Step F: Firewalls, Spam and Virus Filters

The transfer process described in the last step is somewhat simplified. An e-mail may be transferred to more than one MTA within a network cloud and is likely to be passed to at least one firewall before it reaches its destination.

An e-mail encountering a firewall may be tested by spam and virus filters before it is allowed to pass inside the firewall. These filters test to see if the message qualifies as spam or malware. If the message contains malware, the file is usually quarantined and the sender is notified. If the message is identified as spam, it will probably be deleted without notifying the sender.

Spam is difficult to detect because it can assume so many different forms, so spam filters test on a broad set of criteria and tend to misclassify a significant number of messages as spam, particularly messages from mailing lists. When an e-mail from a list or other automated source seems to have vanished somewhere in the network cloud, the culprit is usually a spam filter at the receiver's ISP or company. This explained in greater detail in Virus Scanning and Spam Blocking.

Delivery

In the Figure 13.1, the e-mail makes it past the hazards of the spam trap...er...filter, and is accepted for delivery by the receiver's MTA. The MTA calls a local MDA to deliver the mail to the correct mailbox, where it will sit until it is retrieved by the recipient's MUA.

RFCs

Documents that define e-mail standards are called "Request For Comments (RFCs)", and are available on the Internet through the Internet Engineering Task Force (IETF) website. There are many RFCs and they form a somewhat complex, interlocking set of standards, but they are a font of information for anyone interested in gaining a deeper understanding of e-mail.

Here are a couple of the most pertinent RFCs:

- **RFC 822:** Standard for the Format of ARPA Internet Text Messages
- **RFC 2821:** Simple Mail Transfer Protocol

13.3.3 Advantages of E-mail

Notes

- **Easy to Use:** Sending an e-mail frees us from the tedious work of managing huge chunks of data. An e-mail service helps manage our contacts, allows us to send mails quickly, helps maintain our mailing history and provides sufficient storage space. An e-mail can be sent from any terminal (computer) in the world with Internet access.
- **Environment Friendly:** Mails sent by postal service make use of paper. Thus, electronic mail prevents a large number of trees from getting axed. Using the electronic mailing service also saves the fuel that is consumed in the process of transporting letters.
- **Speed:** An e-mail can be delivered instantly and almost anywhere across the globe. No other mailing service matches the e-mail in terms of speed. You can send a message/mail simultaneously to multiple users; thus, e-mail service saves a lot of time.
- **Informal and Conversational:** Generally, the language used in e-mails is simple. It makes the process of communication informal. The process of sending and receiving e-mails doesn't require much time. Therefore, it can be used as a tool for conversing, just like chatting.
- **Data Storage:** The e-mail service providers offer their customers/users with enough space for storage of data. Also, the process of sorting and arranging mails, as per the subject or other criteria (date, sender, etc.), is made easier for users.
- **Provision of Attachments:** The feature of attachment allows users to send huge chunks of data in a single mail. Also, sending attachments with e-mails doesn't raise cost as in postal service.
- **Easier for Reference:** When a person has to reply to an e-mail, he/she can use the provision of attaching previously sent/received mails. Such mails can be used for the purpose of reference in the communication process. It helps the recipient understand what he/she is reading.
- **Automated E-mails:** It is possible to send automated e-mails with the help of specialized programs like autoresponders. Autoresponders reply only to those messages with a generalized, pre-written text.
- **Easy to Prioritize:** E-mails come with a subject line. Therefore, it becomes easy to prioritize them and ignore the ones that are unwanted. Thus, users can easily sort and filter the mails in their inbox.
- **Reliable and Secure:** Constant efforts are taken to improve the security needed for using an electronic mailing service. Today, electronic mailing service is considered as one of the most secure ways of communicating online.
- **Use of Graphics:** Users can send colorful greeting cards and interesting pictures through e-mails. This provision adds great value to the e-mail service.
- **Advertising Tool:** Nowadays, many individuals and business organizations are using the e-mail service to advertise their products, services, etc. Thus, e-mail can also be used as a marketing tool.
- **Cheap Service:** The expenses incurred in using an e-mail service are lesser in comparison to that in the traditional mailing service. However, the expenses also depend on whether you have an Internet connection at home or not. Those who don't have access to Internet at home can avail the services offered by internet cafes.

Notes

- **Advantages of Technological Development:** Development in computer technology has enabled users to send e-mails not only from their desktop computers, but also smartphones and other such devices. Thus, a user can send and receive e-mails even while he/she is traveling.

13.3.4 Disadvantages of E-mails

- **Spam:** E-mails when used to send unsolicited messages and unwanted advertisements create nuisance and are termed as spam. The activity of checking and deleting unwanted mails may consume a lot of user’s time. Therefore, it has become necessary to block or filter unwanted e-mails through spam filters. Generally, spamming is practiced by sending hoax e-mails. E-mail spoofing is another practice which is commonly used for spamming. The act of spoofing involves deceiving a recipient by altering e-mail headers or addresses from which a mail is sent.
- **Hacking:** The act of breaking into computer security is termed as hacking. In this form of security breach, e-mails are intercepted by hackers. An e-mail, before it is delivered to the recipient, “bounces” between servers located in different parts of the world; which is why e-mail accounts are vulnerable to hacking by professional hackers.
- **Not Suited for Business:** Many a time, business documents go unnoticed amidst large number of mails which get accumulated in the inbox. Thus, urgent transactions and especially those which require signatures cannot be easily managed through e-mails.
- **Viruses:** Viruses are computer programs which have the potential to harm computer systems. Viruses are known to copy themselves and further infect the computer system in question. Recipients are required to scan their mails because, viruses can be transmitted through them.
- **Crowded Inbox:** The e-mail inbox tends to get crowded with mails after a certain period of time. Thus, it becomes difficult for users to manage huge chunks of e-mails. Such kind of information overload often repels users from using the e-mail service.
- **Lacking that Personal Touch:** Even in today’s digital age, there are many who feel comfortable reading the handwritten word. E-mails lack that personal touch.
- **Internet Access is Required:** There are many parts in the world where people don’t have access to Internet. The e-mail service doesn’t serve any purpose in such areas.
- **Misinterpretation:** One has to be careful while posting content through an e-mail. If typed in a hurry, the matter can be misinterpreted by readers.
- **Lengthy Mails:** If a mail is too long or not drafted properly by the sender, the reader may lose interest; he/she may not read the mail till the end.
- **Checking the Inbox Regularly:** In order to stay updated, one has to check his/her e-mail account regularly. If a person, due to his/her busy daily routine, does not check the inbox, he/she can miss out on some of the important and urgent messages.



Did u know? In 1971 the first ARPANET e-mail was sent, and through RFC 561, RFC 680, RFC 724 – and finally 1977’s RFC 733, became a standardized working system.

Task Prepare a flow chart to sending process of an e-mail.



Case Study

Document Workflow System

Client

The client is University of Liverpool, large educational institution in United Kingdom.

Problem Background

The University has very intensive flow of documents on different aspects of its operations. The problem is that different departments have several different document management systems and very weak integration capabilities. The idea was to create a solid solution for documents flow management, which can be used by several departments with single entry point. There was already developed intranet portal on basis of MS SharePoint Portal Server, so it was decided to use the same platform for further development including ability to easily integrate new DocFlow system into this portal.

Questions:

1. What are the key features of this project?
2. Provide a technical Solution for this problem statement.

Self Assessment

Multiple Choice Questions:

6. The most popular way to materialize XML documents is to use:
 - a. DTD
 - b. XSLT
 - c. HTML
 - d. SOAP
7. System.Xml Namespace classes are
 - a. XmlDocument
 - b. XmlDataDocument
 - c. Both
 - d. None
8. Which DLL translate XML to SQL in IIS?
 - a. SQLISAPI.dll
 - b. SQLXML.dll
 - c. LISXML.dll
 - d. SQLIIS.dll
9. Which namespace used for Mail services to send?
 - a. System.Web.Mail
 - b. System.Web.E-mail
 - c. System.Web.Mails
 - d. System.Web.MailService
10. What are the parsers supported in XML?
 - a. non validating and validating
 - b. well documented
 - c. well formed
 - d. none of the above
11. e-mail are also known as trigger, alert or auto-responder.
 - a. Notification
 - b. Transaction
 - c. Marketing
 - d. Service

Notes

12. The mail server as defined in the text uses the protocol.
 - a. HTTP
 - b. FTP
 - c. POP
 - d. SMTP
13. What is the port number for (Post Office Protocol) POP3?
 - a. 21
 - b. 110
 - c. 120
 - d. 90
14. Which protocol is used to provide secure connections across the Internet?
 - a. ARP
 - b. HTTPS
 - c. NTP
 - d. POP3
15. Automatic e-mail notifications are not generated in which of the following condition:
 - a. When an order file is sent confirming receipt
 - b. When new product arrives in the fulfilment centre
 - c. High stock notification to begin the reorder process
 - d. Weekly to advise of any stock adjustment

13.4 Summary

- Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- The WFS Specification mandates the use of XML Schema as a minimum for all operations.
- A charge paid for wagon or transport of goods by air, land, or sea. Goods may be on cloud nine on freight-prepaid or freight-collect basis.
- Seafreight calculations can broadly be divided into two main components; breakbulk and containerised.
- The currency adjustment factor is a mechanism for taking into account fluctuations in exchange rates.
- E-mail (electronic mail) is the exchange of computer-stored messages by telecommunication. E-mail messages are usually encoded in ASCII text.
- E-mail is one of the protocols included with the Transport Control Protocol/Internet Protocol (TCP/IP) suite of protocols.
- There are three types of E-mail: Marketing E-mails, Notification E-mails and Transactional E-mails.
- XML spy is an XML development environment for designing and editing XML, XML Schema, XSL/XSLT, SOAP, WSDL and Web Service technologies.
- An e-mail encountering a firewall may be tested by spam and virus filters before it is allowed to pass inside the firewall.
- Documents that define e-mail standards are called "Request For Comments (RFCs)", and are available on the Internet through the Internet Engineering Task Force (IETF) website.

13.5 Keywords

DNS: The Domain Name System is a hierarchical distributed naming system for computers, services, or any resource connected to the Internet or a private network.

DTD: A Document Type Definition is a specific document defining and constraining definition or set of statements that follow the rules of the Standard Generalized Markup Language (SGML) or of the Extensible Markup Language (XML).

EDI: Electronic data interchange is a method for transferring data between different computer systems or computer networks.

IMAP: The Internet Message Access Protocol is an Application Layer Internet protocol that allows an e-mail client to access e-mail on a remote mail server.

MAPI: Messaging Application Programming Interface is a messaging architecture and a Component Object Model based API for Microsoft Windows.

Post Office Protocol (POP): It is an application-layer Internet standard protocol used by local e-mail clients to retrieve e-mail from a remote server over a TCP/IP connection.

RFC: A Request For Comments document defines a protocol or policy used on the Internet. An RFC can be submitted by anyone.

SMTP: Simple Mail Transfer Protocol is an Internet standard for electronic mail (e-mail) transmission across Internet Protocol (IP) networks.

WFS: Web Feature Service provides an interface allowing requests for geographical features across the web using platform-independent calls.

WMS: A warehouse management system (WMS) is a software application that supports the day-to-day operations in a warehouse.

XML: XML stands for Extensible Markup Language and is a markup language much like HTML which was designed to carry data, not to display data.



Lab Exercise

1. Write a program to calculate freight of air.
2. Write a program to create an XML DTD.

13.6 Review Questions

1. What is Freight calculation?
2. What is the difference between POP and IMAP protocol?
3. State the need of E-mail.
4. What are XML tools? Why we need them?
5. What Necessitates a Different Protocol for E-mail?
6. Is it possible to add mail protocol?
7. Which protocol is used to download an e-mail?
8. Explain the difference between XML DTD and XML Schema.

Notes

9. Explain the term E-mail Queue.
10. E-mail is an application of E-commerce. Justify your answer.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (a) | 2. (c) | 3. (c) | 4. (c) |
| 5. (b) | 6. (b) | 7. (c) | 8. (a) |
| 9. (a) | 10. (a) | 11. (a) | 12. (c) |
| 13. (b) | 14. (b) | 15. (c) | |

13.7 Further Readings



Books

Essentials of E-Commerce Technology, by Rajaraman



Online links

[http://books.google.co.in/books?id=-8BsyuXuEOMC&pg=PA3&dq= Adding+E-Commerce+Essentials&hl=en&sa=X&ei=J5j-T43bKIIfQrQfm5sjPBg&ved=0CEQQ6AEwAg#v=onepage&q=Adding%20E-Commerce%20Essentials&f=false](http://books.google.co.in/books?id=-8BsyuXuEOMC&pg=PA3&dq=Adding+E-Commerce+Essentials&hl=en&sa=X&ei=J5j-T43bKIIfQrQfm5sjPBg&ved=0CEQQ6AEwAg#v=onepage&q=Adding%20E-Commerce%20Essentials&f=false)

Unit 14: Debugging and Optimization

Notes

CONTENTS

Objectives

Introduction

- 14.1 Debugging in an ASP.NET Application
 - 14.1.1 Creating an ASP.NET Application
 - 14.1.2 Download Cassini
 - 14.1.3 Debugging an ASP.NET Application
 - 14.1.4 Attach to Process
 - 14.1.5 Disable Debugging for an ASP.NET Application
- 14.2 Optimizing Using Caching
 - 14.2.1 ASP.NET Caching Features
 - 14.2.2 The Upside of Using the Cache
 - 14.2.3 The Downside of Extensive Caching
 - 14.2.4 Standardizing Cache Access
- 14.3 Optimizing via Performance Profiling
 - 14.3.1 ASP.NET Performance
 - 14.3.2 ASP.NET Pipeline Optimization
 - 14.3.3 ASP.NET Process Configuration Optimization
- 14.4 Summary
- 14.5 Keywords
- 14.6 Review Questions
- 14.7 Further Readings

Objectives

After studying this unit, you will be able to:

- Explain debugging and optimization application
- Define optimizing using caching
- Discuss the optimizing via performance profiling

Introduction

Debugging is the process of finding and correcting the errors in a program. In normal Active Server Pages (ASP) file Response. Write statement was the only real primary method to debug the actual errors and even Script Debugger was not sophisticated to debug, in comparison to the

Notes

actual debuggers that Windows developers experienced for Visual Basic and C++ code. All these deficiencies have been changed in .Net Framework since the Common Language Runtime provides incorporated debugging for all the programs. The primary objectives for debugging in ASP.NET include both in-process and out-of-process debugging, remote process debugging, managed and unmanaged code debugging, edit and continue, and mixed language support.

We may enable debugging in ASP.NET page by writing the `<%@Page debug= "True" %>` page directive towards the top of the page. This instruction requires the compiler to exclude debugging symbols into the compiled page, and allows the debugger to attach to the actual running program. But before while using debugger, we should also have to view the web page in the browser so that the debugging symbols are loaded for the page.

The SDK debugger which comes along with .NET SDK can be found in the Guidebook directory. This debugger offers all the features such as seeing contents of variables, setting breakpoints on specific exceptions, expressions, or specific lines, stepping into, out of, and over code, and viewing the call stack, threads and modules as that of Visual Studio debugger except which SDK debugger does not support remote debugging or Edit and Continue features. Therefore it can be presumed that SDK debugger is a read-only debugger that doesn't allow us to edit code in-line. We can just edit the code externally and to view the new changes made in the application we have to rerun the application.

Just-In-Time debugging is another method in order to debug a program that is started outside of Visual Studio. In the event that we have enabled Just-In-Time debugging, we can view a dialog box when an accident occurs. Debugging a traditional ASP application generally involves placing Response. Write statements throughout our code to track variable values as well as execution paths. If we fail to remove debugging statements before deploying our application, the statements are visible to users. ASP.NET makes Web application debugging much simpler to make use of. We can use tracing statements to debug our application in a way that is similar to using Response. Write statements. However, we can keep the tracing statements in our application, even after deployment.

We can configure the Web application to allow tracing at either the page level or the application level, and we can easily switch it on and off. We can also control the destination of the actual tracing Output to allow just certain users to see the debugging Output.

ASP.NET incorporates a variety of features and tools that allow us to design and implement high-performance Web applications. These features include:

- A process model that has improvements over the model in ASP.
- A feature that automatically detects changes in ASP.NET pages, dynamically compiles changed pages, and stores the newly compiled files for reuse in subsequent requests.
- ASP.NET-specific performance counters.
- Web application testing tools.

14.1 Debugging in an ASP.NET Application

Debugging a traditional ASP application generally involves placing Response. Write statements throughout your code to track variable values and execution paths. If you fail to remove debugging statements before you deploy your application, the statements are visible to users.

ASP.NET makes Web application debugging much simpler to use. You can use tracing statements to debug your application in a way that is similar to using Response. Write statements. However, you can leave the tracing statements in your application, even after deployment.

You can configure the Web application to allow tracing at either the page level or the application level, and you can easily turn it on and off. You can also control the destination of the tracing output to allow only certain users to see the debugging output.

14.1.1 Creating an ASP.NET Application

The following procedure creates a simple ASP.NET Web application that performs a calculation on two numbers and displays the results on the screen.

1. Start Microsoft Visual Studio .NET.
2. Create a new Visual Basic ASP.NET Web Application project. Name it ASP Debugging Example.
3. In WebForm1.aspx, switch to HTML view.
4. Type or paste the following code between the opening and closing FORM tags.



Notes If you paste the code, paste it as HTML code. To do this, click Paste as HTML on the Edit menu.

When you run the code, it prompts you to type two numbers and displays a command button. When you click the command button, the code adds the two numbers and displays the result on your screen.

```
<TABLE id="Table1" cellSpacing="1" cellPadding="1" width="300" border="0">

<TR><TD>
<asp:Label id="Label1" runat="server">First Number:</asp:Label>
</TD>
<TD>
<asp:TextBox id="txtFirstNumber" runat="server"></asp:TextBox>
</TD></TR>
<TR><TD>
<asp:Label id="Label2" runat="server">Second Number:</asp:Label>
</TD>
<TD>
<asp:TextBox id="txtSecondNumber" runat="server"></asp:TextBox>
</TD></TR>
<TR><TD>
<asp:Label id="Label3" runat="server">Sum:</asp:Label>
</TD>
<TD>
<asp:TextBox id="txtSum" runat="server"></asp:TextBox>
</TD></TR>
<TR><TD>
<asp:Button id="Button1" runat="server" Text="Add Numbers"></asp:Button>
</TD>
<TD>
</TD></TR>
</TABLE>
```

Notes



Task Create login page for any institute using an ASP.NET Application.

14.1.2 Download Cassini

Cassini is a “simple, fully managed Web server that hosts ASP.NET”. It’s a lightweight “server” you get when you create a new ASP.NET application in Visual Studio.

Cassini has been pitched to those who don’t have IIS on their computers but want to develop in ASP.NET. This works great for hobbyists. However, Cassini is no IIS. It’s much slower, it hangs and crashes easily.

The reason I’m writing this is to convince you not to use Cassini within an organization. Sooner or later you will deploy your app on a real web server only to realize that something is badly broken. The problem never came up because Cassini is not a real web server and therefore issues were simply swept under the carpet.

14.1.3 Debugging an ASP.NET Application

The following procedure creates a simple ASP.NET Web application that performs a calculation on two numbers and displays the results on the screen. When you add a breakpoint to the Add function, notice how the new interactive debugging capabilities of ASP.NET behave:

1. Start Microsoft Visual Studio .NET.
2. Create a new Visual Basic ASP.NET Web Application project. Name it ASP Debugging Example.
3. In WebForm1.aspx, switch to HTML view.
4. Type or paste the following code between the opening and closing FORM tags.



Notes If you paste the code, paste it as HTML code. To do this, click Paste as HTML on the Edit menu.

When you run the code, it prompts you to type two numbers and displays a command button. When you click the command button, the code adds the two numbers and displays the result on your screen.

```
<TABLE id="Table1" cellSpacing="1" cellPadding="1" width="300" border="0">

<TR><TD>
<asp:Label id="Label1" runat="server">First Number:</asp:Label>
</TD>
<TD>
<asp:TextBox id="txtFirstNumber" runat="server"></asp:TextBox>
</TD></TR>
<TR><TD>
<asp:Label id="Label2" runat="server">Second Number:</asp:Label>
</TD>
<TD>
```

```

<asp:TextBox id="txtSecondNumber" runat="server"></asp:TextBox>
</TD></TR>
<TR><TD>
<asp:Label id="Label3" runat="server">Sum:</asp:Label>
</TD>
<TD>
<asp:TextBox id="txtSum" runat="server"></asp:TextBox>
</TD></TR>
<TR><TD>
<asp:Button id="Button1" runat="server" Text="Add Numbers"></asp:Button>
</TD>
<TD>
</TD></TR>
</TABLE>

```

5. Click the Design button in the lower-left corner of the Code window to switch to Design view.
6. Double-click Add Numbers to view the code and write an event handler for the command button's Click event.

7. Add the following code to the event handler.

```

Dim intTotal as integer
Dim intFirstNumber as integer
Dim intSecondNumber as integer

' Get the values from the input boxes.
intFirstNumber = CInt(txtFirstNumber.Text)
intSecondNumber = CInt(txtSecondNumber.Text)

' Get the total and display it.
intTotal = intFirstNumber + intSecondNumber
txtSum.Text = CStr(intTotal)

```

8. Select the following line of code in the Button1_Click routine.

```
intTotal = intFirstNumber + intSecondNumber
```

Press F9 to set a breakpoint on this line.

A breakpoint indicator appears on the line of code.

9. Save the file.
10. Build and then run the application. To do this, click Start on the Debug menu.
WebForm1 appears on the screen. The input controls and the submit button also appear on the form.
11. Type a number in each of the first two input controls, and then click Add Number.

The Visual Studio .NET Integrated Development Environment (IDE) comes into focus and the code halts when the breakpoint is reached. You can use the Locals window to examine the contents of the variables and make any changes you want. You can add one or more watches to trace the values of the variables in the routine. Or, you can continue running the application.

Notes

12. On the Debug menu, click Continue to continue running the application.
WebForm1 appears on the screen and the result appears in the third input box.
13. Close the Web browser and return to the Visual Studio .NET IDE.
14. On the Debug menu, click Clear All Breakpoints to remove all breakpoints from the application.
Verify that it works
When you click Add Number, the Visual Studio .NET IDE comes into focus and the code stops running at the point where the breakpoint is specified.

14.1.4 Attach to Process

It is possible to attach the Visual Studio debugger to processes that are already running. However, there are some limitations, principally that the assembly to be monitored must be compiled in debug mode, not release mode. The key requirements for attaching to a running program are:

- The software was compiled in debug mode and the program database file (*.PDB) is available.
- The source code for the assembly, as referenced in the PDB file, is available and matches that of the compiled code.
- The version of Visual Studio that you are using supports attaching to processes. This facility is not available in the Express editions.

If the conditions are met, you can attach Visual Studio to the running process. You can then add breakpoints or tracepoints to halt the code at an important location and use the usual tools such as the immediate, locals and watches windows to assist in finding problems or confirming the correct operation of the code.

Showing the Attach to Process Dialog Box

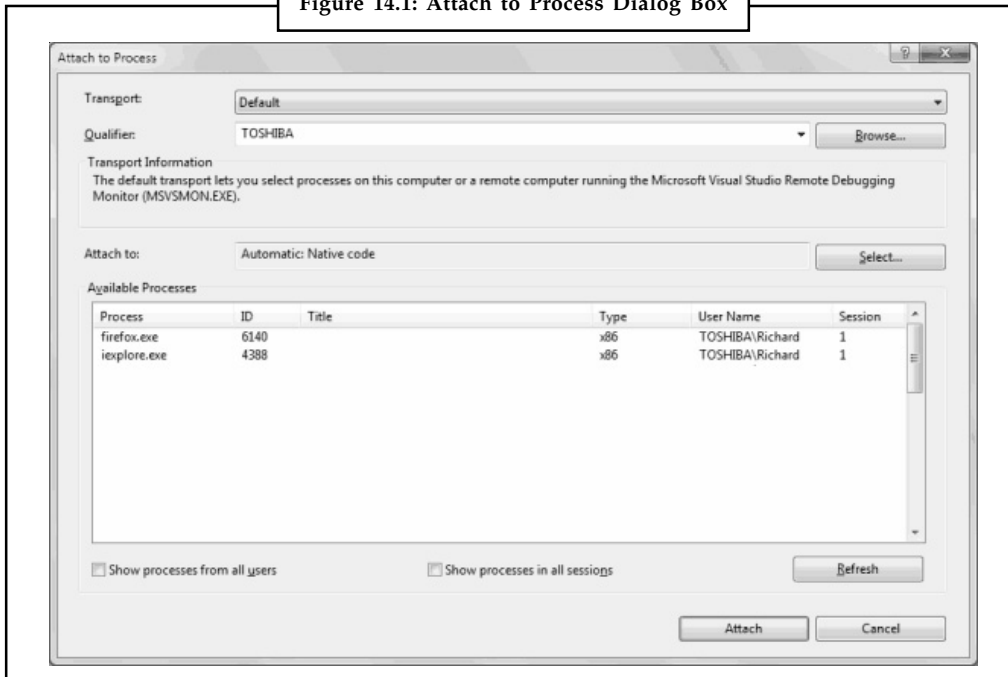
To debug a running program you can use the Attach to Process dialog box. To display the dialog box when using Visual Studio with an open project, select "Attach to Process" from the Debug menu. If you do not currently have a project open, the Debug menu will not be visible. In this case, select "Attach to Process" from the Tools menu.

The main area of the window shows the processes that are executing in the context of the current user. If you wish to debug a program that is running with different credentials, tick the "Show processes from all users" checkbox. The list of processes is loaded when the dialog box is first displayed and does not update in real time. You can, however, refresh the list by clicking the refresh button.

Above the list of available processes is the "Attach to" box. This shows the type of process that you will be connecting to. For example, managed code or native code. By default, the selection is made automatically. As you select processes in the list, the contents of the box change to show the appropriate type. If you wish to be more specific about the process types that you wish to debug, you can click the Select button and choose from a list.

Once you have located and selected the process, or several processes, that you wish to debug, click the Attach button. The Visual Studio screen will change to show the debugging controls as if you had started debugging from within the IDE. You can then use the debugging toolbar and commands as well as all of the other usual debugging utilities.

Figure 14.1: Attach to Process Dialog Box



When you have completed the debugging session, you can detach from the processes individually using the Processes window, which is displayed by selecting Processes from the Debug menu's Window submenu. Simply right-click a process in the list and select "Detach Process". You can also disconnect from all attached processes using the "Detach All" command from the Debug menu. Note that detaching does not terminate the process.

14.1.5 Disable Debugging for an ASP.NET Application

To improve the performance of our ASP.NET application, a recommended practice is to make sure that the application is not deployed with `Debug=True` attribute.

One of my clients had a production server which hosted multiple websites. Although some of the applications were deployed keeping `debug=false`, however the rest of the applications had the attribute set to true. They now wanted to control the setting from the `machine.config` file which would apply to all the applications running on that server.

Here's a technique that most of the users do not know about. In your `machine.config`, locate the `<system.web>` section and add the following entry

```
<deployment retail="true" />
```

That's it. Now this setting will automatically disable the debug mode and tracing on your applications.

14.2 Optimizing Using Caching

Caching is the most effective technique you can use to improve the performance of your ASP.NET web application. Designing your application with caching in mind, improves both the performance and the scalability of that application.

Caching is about storing data in memory the first time it is requested and then re-using it for the following requests for a specified period of time.

Notes

ASP.NET provides you with two types of caching that you can use to create a high performance web application. The first type is application caching in which you can store application's data in server memory for subsequent requests without recreating them each time. This saves the time and the resources of your application. The second type is output caching. In this type you can save or let us say you can cache dynamic pages and/or user controls for subsequent requests without the need to execute them everytime.

14.2.1 ASP.NET Caching Features

Caching is defined as temporary storage of data for faster retrieval on subsequent requests. In ASP.NET 2.0, the caching support is integrated with the DataSource controls to cache data in a web page. ASP.NET 2.0 also now includes automatic database server cache invalidation. This powerful and easy-to-use feature allows developers to aggressively output cache database-driven page and partial page content within a site, and have ASP.NET automatically invalidate these cache entries and refresh the content whenever the back-end database changes. ASP.NET 2.0 also introduces the Substitution control, which allows you to link dynamic and cached content in a web page.

Caching with the DataSource Controls

The DataSource controls enable you to cache database data while connecting a .NET application to a database. The DataSource control provides various properties, such as EnableCaching, which you can use to automatically cache the data represented by a DataSource control. The syntax to cache a database table in a memory for 120 seconds is:

```
<asp:SqlDataSource ID="SqlDataSource1" EnableCaching="True"
CacheDuration="120"
ConnectionString="Server=localhost;database=AdventureWorks;uid=user;pwd=word;"
SelectCommand="SELECT * FROM Production.ProductCategory" Runat="server" />
```

The above syntax caches a database table, ProductCategory, by setting the EnableCaching property of the DataSource control to True. The CacheDuration property of the DataSource control specifies the time, in seconds, for caching the data before it is updated in a database containing the ProductCategory table. The value of the Time parameter is set to 120 to cache data for two minutes.

Using SQL Cache Invalidation

The Cache API introduced with ASP.NET 1.x was a powerful feature that can be immensely useful in increasing the performance of a web application. The Cache API also allows you to invalidate items in the cache based on some predefined conditions, such as change in an XML file, change in another cache item, and so on. Using this feature, you can remove or invalidate an item from the cache when the data or another cached item changes. However, the Cache API in ASP.NET 1.x versions did not provide a mechanism to invalidate an item in the cache when data in a SQL Server database changed. This is a very common capability that many web applications require. Now with ASP.NET 2.0, Microsoft has introduced a new cache invalidation mechanism that works with SQL Server as well. Using this new capability, you can invalidate an item in the Cache object whenever the data in a SQL Server database changes. This built-in cache invalidation mechanism works with SQL Server 7.0 and above. However, with SQL Server 7.0 and 2000, only table-level cache invalidation mechanism is supported. The next release of SQL Server (named SQL Server 2005) will also feature a row-level cache invalidation mechanism, providing a finer level of accuracy over the cached data. To enable the SQL Server-based cache invalidation mechanism, you need to do the following:

1. Add a < caching > element to the Web.config file, and specify the polling time and the connection string information.
2. Enable SQL cache invalidation at the database and table levels by using either the aspnet_regsql utility or the SqlCacheDependencyAdmin class. This is not required if you are using SQL Server 2005 as your database.
3. Specify the SqlCacheDependency attribute in the SqlDataSource control.

Notes

That's all you need to do to leverage SQL Server cache invalidation from your ASP.NET pages.

Using the Substitution Control

ASP.NET 2.0 provides a new control called the Substitution control, which enables you to insert dynamic content into a cached web page. For example, you can display the name of an end user, which is dynamically generated in a cached web page containing some text or images. The Substitution control provides a property called MethodName, which represents the method called to return the dynamic content.

Partial Page Caching Using Substitution Control

```
<%@ Page Language="C#" %>
<%@ OutputCache Duration="6000" VaryByParam="none" %>
<script runat="server">
    static string GetRandomNumber(HttpContext context)
    {
        int randomNumber;
        randomNumber = new System.Random().Next(1, 10000);
        return randomNumber.ToString();
    }
</script>
<html>
<head>
<title>Use of Substitution control to implement Partial Caching</title>
</head>
<body>
<form id="form1" runat="server">
The random number generated is:
<asp:Substitution ID="Substitution1" MethodName="GetRandomNumber"
Runat="Server" />
<p>
The current time is <%= DateTime.Now.ToString("t") %>.
It never changes since the page is cached.
</p>
</form>
</body>
</html>
```

At the top of the page, the OutputCache directive is used to cache the contents of the page in memory. The Duration attribute of the OutputCache directive is set to 6000 milliseconds. The VaryByParam attribute indicates whether or not ASP.NET should consider the parameters passed to the page when caching. When VaryByParam is set to none, no parameters will be considered; all users will receive the same page no matter what additional parameters are supplied.

Notes

The `MethodName` attribute of the `Substitution` control is set to a method named `GetRandomNumber`, which simply returns a random number between 1 and 10,000. Note that the return value of the `GetRandomNumber` method is a string, because the `HttpResponseSubstitutionCallback` delegate always requires a return type of string. When you make a request for the page through the browser, you will find that the displayed current time always remains the same, whereas the portion of the page that is generated by the substitution control keeps changing every time. In this case, it displays a random number between 1 and 10,000 every time someone requests the page.



Did u know? Microsoft C 1.0, based on Lattice C, was Microsoft's first C product in 1983.

14.2.2 The Upside of Using the Cache

1. **Reduce hosting server round-trips:** When content is cached at the client or in proxies, it cause minimum request to server.
2. **Reduce database server round-trips:** When content is cached at the web server, it can eliminate the database request.
3. **Reduce network traffic:** When content is cached at the client side, it also reduces the network traffic.
4. **Avoid time-consumption for regenerating reusable content:** When reusable content is cached, it avoid the time consumption for regenerating reusable content.
5. **Improve performance:** Since cached content reduce round-trips, network traffic and avoid time consumption for regenerating reusable content which cause a boost in the performance.



Caution Do not create `MemoryCache` instances unless it is required. If you create cache instances in client and Web applications, the `MemoryCache` instances should be created early in the application life cycle.

14.2.3 The Downside of Extensive Caching

There are two main issues that can arise with caching:

1. **Keeping the cache current.** If you access a cache page that is not current then you are at risk of getting old and incorrect information. Some things you may never want to be cached, for example the results of a transactional database query. It's not that these problems are insurmountable, but there is always the risk that the data in cache will not be synchronized with changes.
2. **Volume.** There are some 100 millions of web sites out on the Internet alone. Each site contains upwards of several megabytes of public information. The amount of data is staggering and even the smartest caching scheme cannot account for the variation in usage patterns among users and the likely hood they will hit an un-cached page. If you have a diverse set of users it is unlikely the Cache will have much effect on a given day.

14.2.4 Standardizing Cache Access

You can programmatically manipulate the cache to place individual data items in it, thereby reducing the number of calls required to a backend database. For example, suppose an ASP.NET

Web page displayed data from a relational database that included lookup data populated in a drop-down list box. Since the lookup data is relatively static and is the same for all users, it is a good candidate for caching.

To access the ASP.NET caching engine programmatically, both the Page and UserControl classes – which are used to create the code-behind classes in a Web Form and Web User Control – include a Cache property that exposes the System.Web.Caching.Cache object. Using this property, a developer can write code to populate an item and place it in the cache, or simply pull it out of the cache if it already exists.

In this case, the cache is first checked in the Load or Init event to see if the item corresponding with the Categories key is in the cache. If not, an ADO.NET DataTable is created by calling a method in a custom data access class called WebData (not shown).

Both DataTables and DataSets are good candidates to place in the cache since they are totally disconnected from the underlying data source from which they were populated.

Once the DataTable has been retrieved, it is placed in the cache using the Insert method. In this case, the item is added to the cache with no dependencies using an absolute expiration of six hours and disregarding the sliding expiration interval. You could use the CacheItemRemovedCallback delegate to create a callback to be notified when the cached item expires.

On the other hand, if the item is in the cache, it is simply pulled out and cast back to the appropriate type using the CType method and bound to a DropDownList control called selCategory.

Visual C++ 2010 (known also as Visual C++ 10.0) was released on April 12, 2010, and it is currently the latest stable release.

Self Assessment

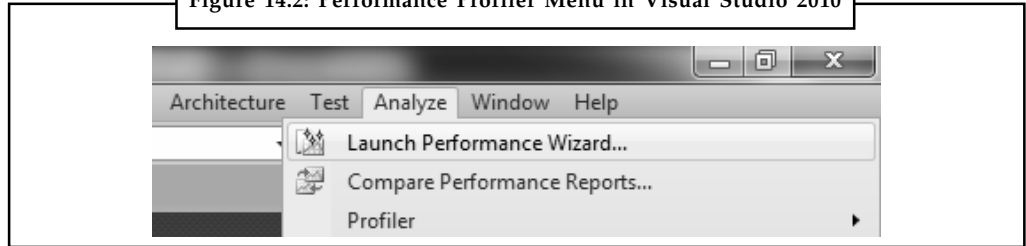
Multiple Choice Questions:

1. What debugging tools come with the .NET SDK?
 - a. CorDBG – command-line debugger
 - b. Dbg CLR – graphic debugger
 - c. Both A and B
 - d. None of the above
2. Caching is considered as the best way to enhance the performance of the application.
 - a. Yes
 - b. No
3. Which of the following can be used to debug .NET application?
 - a. Visual Studio .NET
 - b. Runtime Debugger
 - c. Systems.Diagnostics classes
 - d. All the Above
4. How do you trace the application_End event on runtime?
 - a. By Debugging
 - b. By Tracing
 - c. Can't be done
5. A process where items are removed from cache in order to free the memory based on their priority is called page posting.
 - a. true
 - b. False

14.3 Optimizing via Performance Profiling

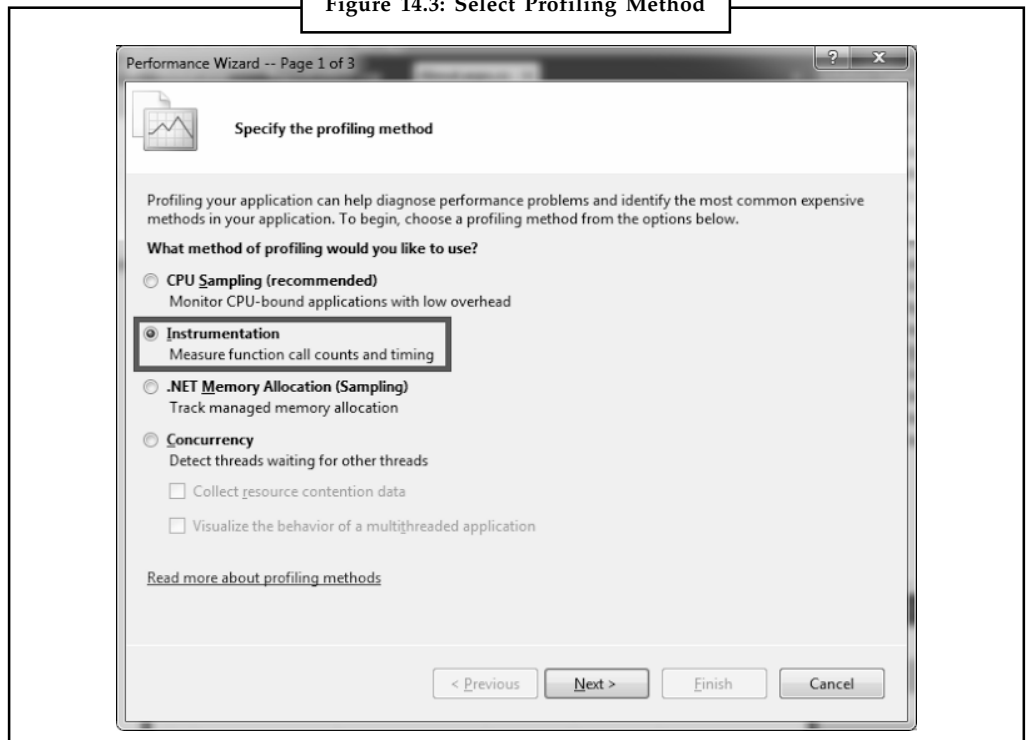
You want to ensure your application runs as fast as possible, and one of the most important thing is you need to know which part of your application is taking most amount of time (or, resources) to execute, so that, you can identify the areas where you need to optimize performance. Well, this wasn't easy, until Visual Studio introduced the "Analyze" menu in the top-right portion of the IDE.

Figure 14.2: Performance Profiler Menu in Visual Studio 2010



Assuming that you have opened your application on Visual Studio (In my case, I created a sample ASP.NET application) clicking on the "Launch Performance Wizard" menu will launch the performance profiler wizard for you with the following window:

Figure 14.3: Select Profiling Method

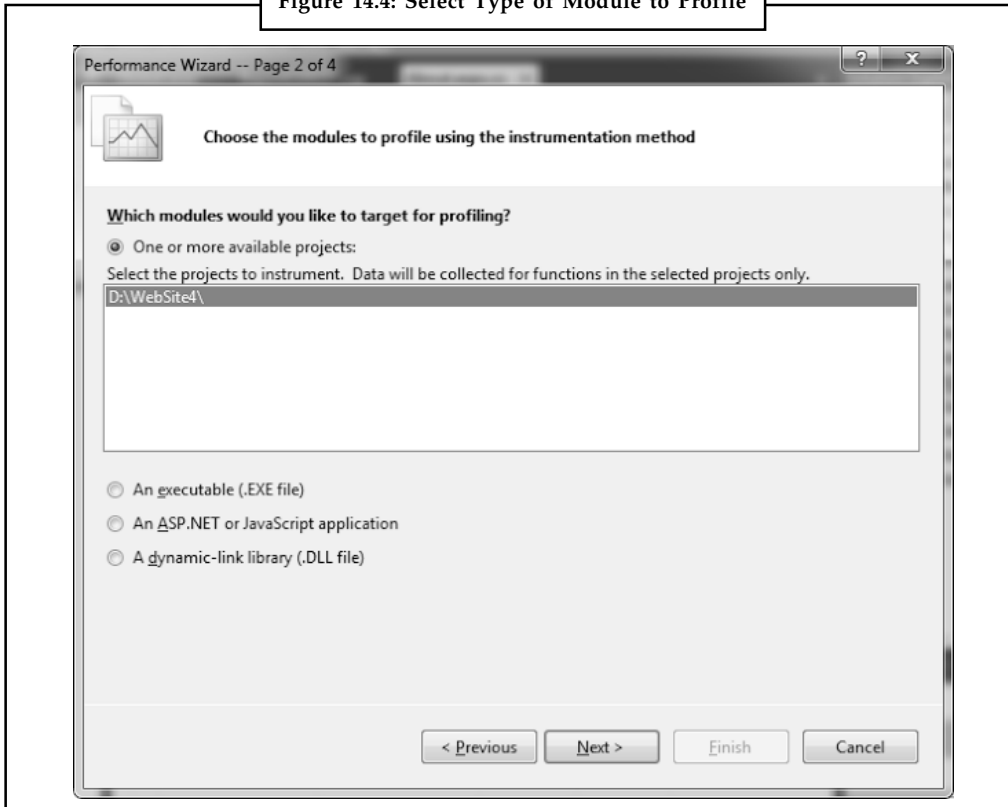


Note that, there are four options. Visual Studio recommends the CPU sampling (which periodically interrupts the target process and collects information from call stack to develop profiler information), but in our case we will most likely choose "Instrumentation" which injects some instrumentation codes within each method of the dlls (of the target application which we are profiling) for collecting performance data in terms of method level execution time. This profiling method is very handy for identifying the performance issues for database based applications (For example, you can measure performance of your DAL methods and identify which stored procedures are taking what amount of time in which conditions).

Selecting the “Instrumentation” option and clicking on the “Next” button would bring up the following wizard step:

Notes

Figure 14.4: Select Type of Module to Profile



This wizard step asks to select a “module” for profiling. You can profile any exe, dll or an ASP.NET web site/application and you need to provide the path to the exe or dll, or, you need to provide the URL of the ASP.NET web site/application.

In our case we are profiling a sample ASP.NET application which is currently opened in the visual studio. The sample ASP.NET application has an About.aspx page containing the following simple CodeBehind:

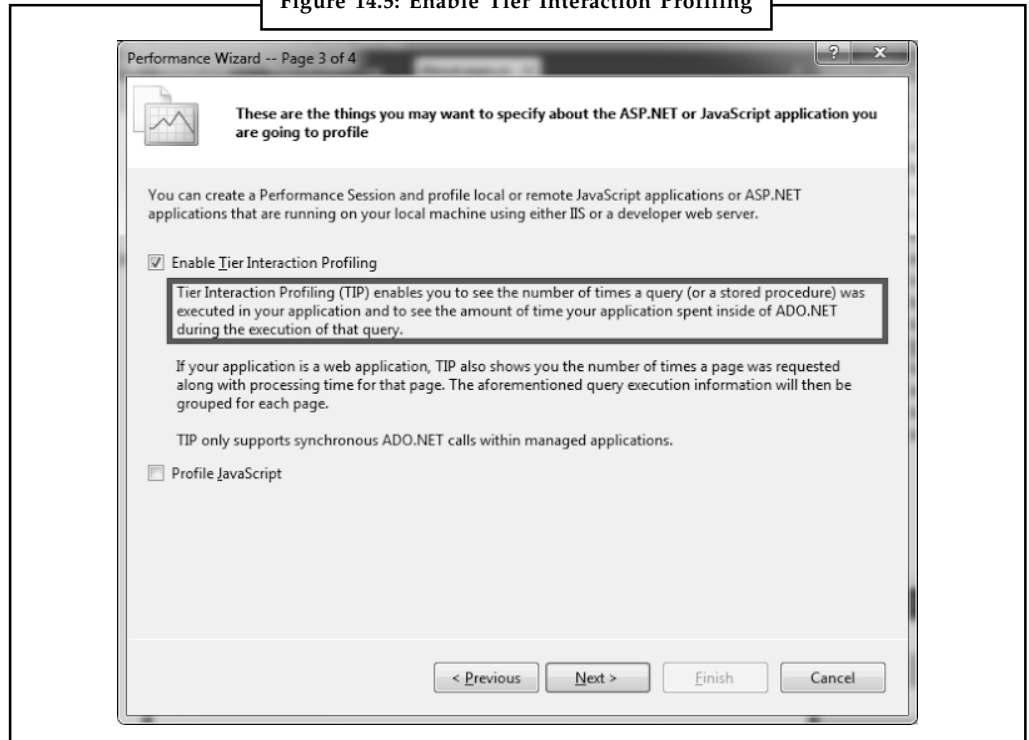
```
public partial class About : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        TestMethod();
    }

    private void TestMethod()
    {
        Thread.Sleep(300);
    }
}
```

Our target is to profile the performance of the methods within the CodeBehind class (TestMethod() in this case). So, we would proceed with the default selection of the web site and click “Next”, which would bring up the following wizard step:

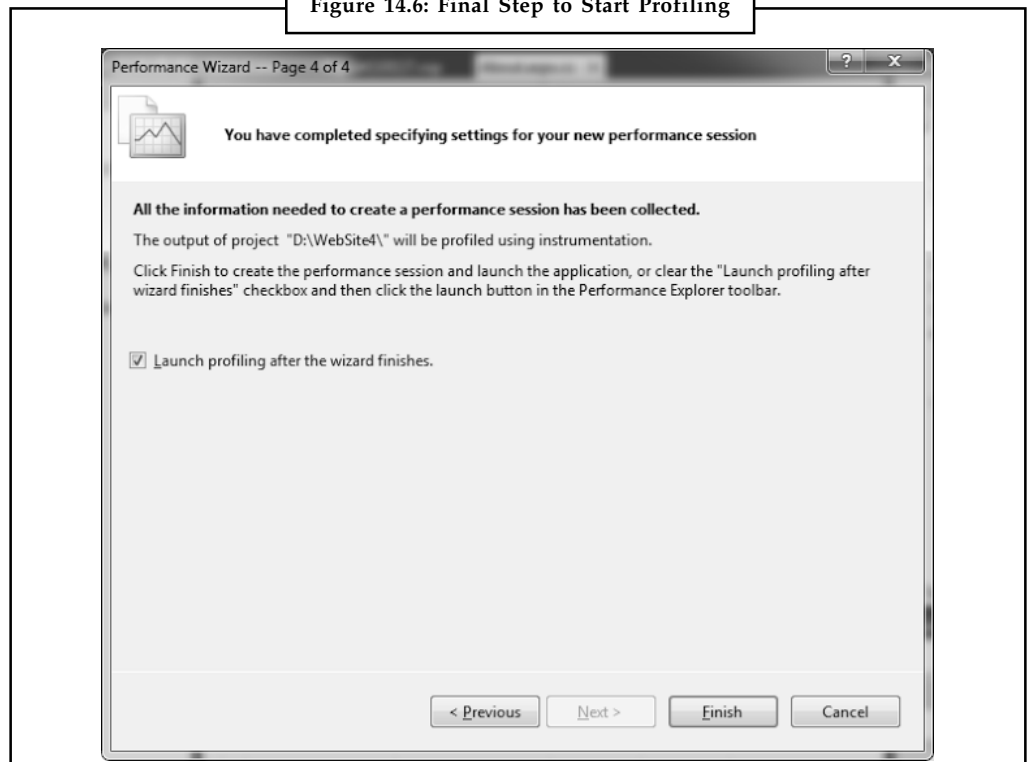
Notes

Figure 14.5: Enable Tier Interaction Profiling



This wizard step asks you to enable Tier Interaction Profiling (TIP) which lets you get the execution time the application spent on each Tier. We would like to check this check box and click "Next" to proceed the last wizard step:

Figure 14.6: Final Step to Start Profiling



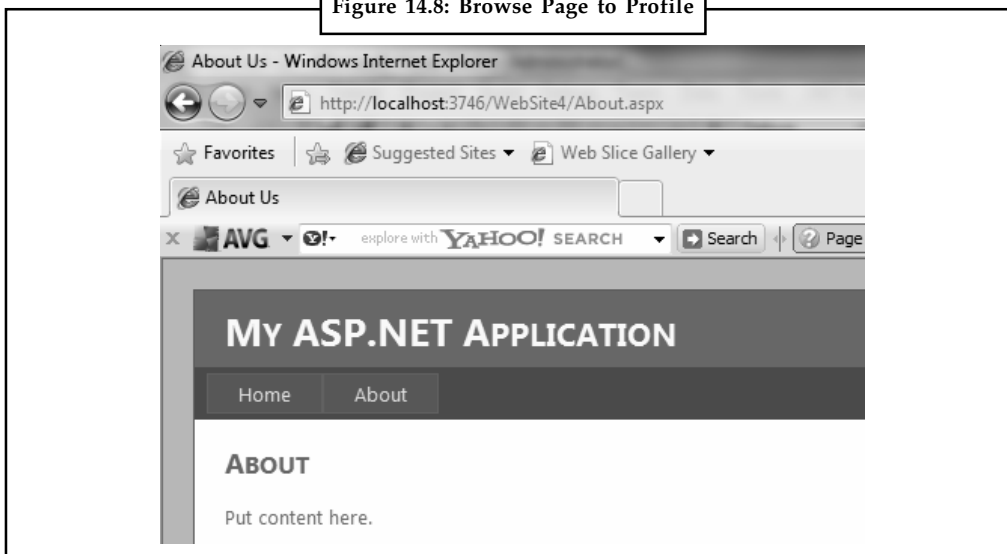
Click on “Finish” and Visual Studio will launch the application with hitting a URL of your application in the browser and will start profiling with showing the following activity in the screen:

Figure 14.7: Profiling Application



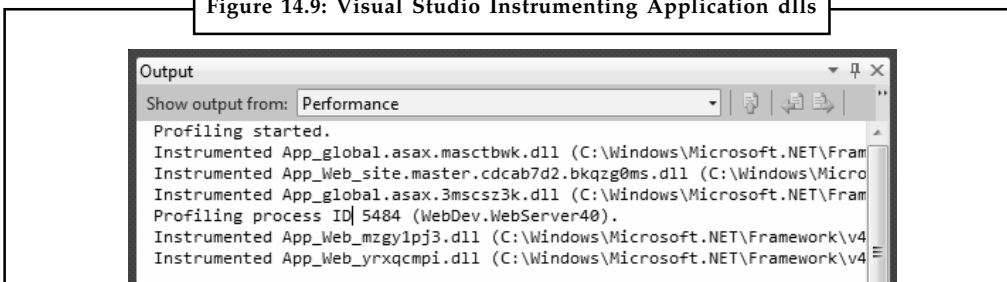
In order to profile the particular page functionality in About.aspx, I needed to hit the corresponding URL in the browser (because, Visual Studio didn't hit this URL automatically):

Figure 14.8: Browse Page to Profile



Visual studio will start profiling the page functionality and keep showing the following kind of information in the output window (regarding the instrumentation codes it is injecting within the dlls of the web application):

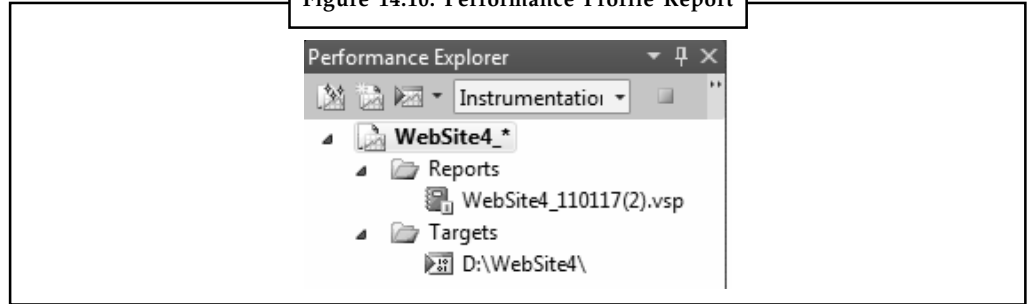
Figure 14.9: Visual Studio Instrumenting Application dlls



Notes

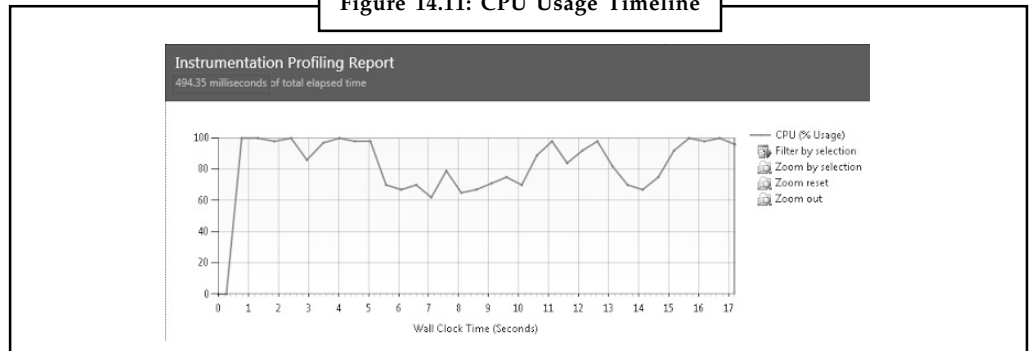
You need to perform the desired functionality in your target page (which you are profiling) so that Visual Studio can profile the related back-end methods. In my case, I just need to profile the methods which are executed when the page loads. So, as this is done, I need to close the page in the browser so that Visual Studio finishes gathering profile data and generates report. Alternatively, you can always stop or pause profiling and generate profile report if you want.

Figure 14.10: Performance Profile Report



The profile report will be generated and opened by Visual Studio and there will be three following sections being displayed in the Summary View of the report (which is default selected).

Figure 14.11: CPU Usage Timeline



The first section of the profile report shows CPU usage percentage in the timeline (the overall time taken by Visual Studio while profiling this page). This doesn't give necessary information we are seeking.

The second section of the profile report shows the "Hot Path" information. This means, it displays the most expensive call path of the method chain, and, elapsed time (Percentage) in terms of Inclusive and Exclusive. "Inclusive" means, the corresponding method execution time is being shown including the time taken by all child methods execution time. "Exclusive" means, the corresponding method executing time is being shown excluding the time taken by all child methods.

Figure 14.12: Hot Path

The most expensive call path based on execution times

Function Name	Elapsed Inclusive Time %	Elapsed Exclusive Time %
WebDev.WebServer40.exe	100.00	0.00
ASP.about_aspx.ProcessRequest(class System.Web.HttpContext)	99.18	0.00
System.Web.UI.Page.ProcessRequest(class System.Web.HttpContext)	99.18	33.42
About.Page_Load(object,class System.EventArgs)	60.90	0.02

Related Views: Call Tree Functions

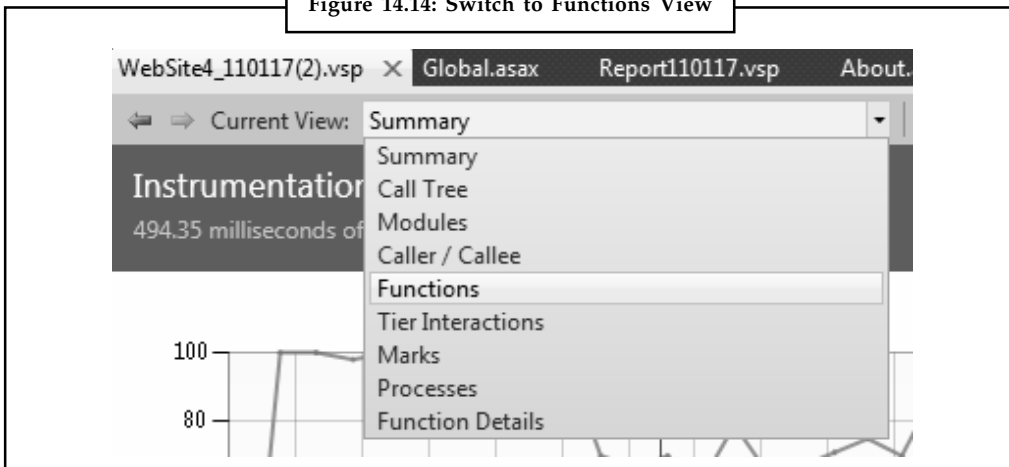
The third portion of the performance report shows methods/functions invocation information which had consumed most execution times (in terms of percentage among the overall page execution time).

Figure 14.13: Function Invocation with Most Execution Time

Functions with the highest exclusive application times	
Name	Exclusive Time %
System.Threading.Thread.Sleep(int32)	60.87
System.Web.UI.Page.ProcessRequest(class System.Web.HttpContext)	33.42
System.Web.UI.WebControls.MenuitemCollection.Add(class System.Web.UI.WebControls.Menuitem)	0.50
System.Web.UI.Page.ctor()	0.49
System.Web.UI.ParserAccessor.AddParsedSubObject(object)	0.49

Switch to the “Functions” view in the drop-down list to see the method level absolute performance (in terms of ms) of the related methods:

Figure 14.14: Switch to Functions View



The following Grid will appear containing the execution time of each method which were involved in overall execution of the page:

Figure 14.15: Function Execution Time

Function Name	Elapsed Inclusive Time	Elapsed Exclusive ...	A
ASP.FastObjectFactory_app_we	3.39	0.24	
_ASP.FastObjectFactory_app_we	0.61	0.26	
About.ctor()	2.46	0.03	
About.Page_Load(object,class Sy	301.04	0.11	
About.TestMethod()	300.94	0.00	
ASP.about_aspx.ctor()	3.15	0.11	
ASP.about_aspx._BuildControlBe	0.00	0.00	
ASP.about_aspx._BuildControlH	0.01	0.00	

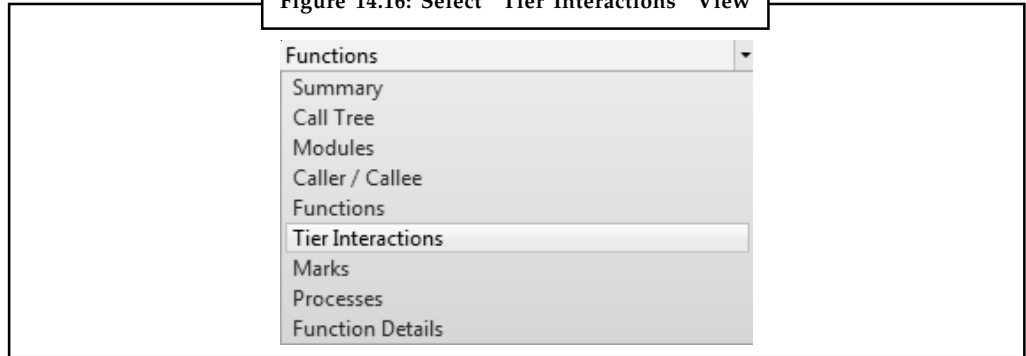


Notes You can sort the grid by the “Elapsed Inclusive Time” field to see which methods are consuming the most amount of execution time. You can also export the Grid in some format to analyze performance issues later.

Switch to “Tier Interactions” view from the drop-down list to see the execution time taken by the application in each different tier.

Notes

Figure 14.16: Select "Tier Interactions" View




The following Grid will display the execution time and the number of occurrence taken in each different tier:

Figure 14.17: Tier Interaction Report

Name	Database	Count	Total Elapsed Time
/WebSite4/About.aspx		1 Request	1,751.76
/WebSite4/		1 Request	208.41
/WebSite4/Styles/Site.css		1 Request	23.88

There are a number of factors affecting performance that operate independently and so must be tackled independently, often by different groups of people. Developers must tackle application coding issues by identifying the offending code blocks and then rewriting or optimizing them. System administrators must tackle server resource problems by examining the full set of tasks that the server is handling. Performance profiling has to be a cooperative task between different groups of people because at the end of the day, a user who experiences slow performance will be unhappy with the experience, regardless of whose piece of the puzzle is causing the issue. Performance profiling affects everyone on the technical team, and so it must engage everyone as well.



Task Search about the caching process. Differentiate between cache memory and RAM.

14.3.1 ASP.NET Performance

ASP.NET 2.0 has many secrets, which when revealed can give you big performance and scalability boost. For instance, there are secret bottlenecks in Membership and Profile provider which can be solved easily to make authentication and authorization faster. Furthermore, ASP.NET HTTP pipeline can be tweaked to avoid executing unnecessary code that gets hit on each and every request. Not only that, ASP.NET Worker Process can be pushed to its limit to squeeze out every drop of performance out of it. Page fragment output caching on the browser (not on the server) can save significant amount of download time on repeated visits. On demand UI loading can give your site a fast and smooth feeling. Finally, Content Delivery Networks (CDN) and proper use of HTTP Cache headers can make your website screaming fast when implemented properly. In this section, you will learn these techniques that can give your ASP.NET application a big performance and scalability boost and prepare it to perform well under 10 times to 100 times more traffic.

Execution Time: It is the time taken by the processor to process one instruction/request. Execution time directly affects the throughput calculation.

Response Time: It is the time taken by the system to respond to the instruction/request.

Scalability: It is the ability of the application to continue to perform well when it is changed in size or volume in order to meet the rising need.

Throughput: It can be define as the number of instruction/request a web application can process in given unit of time, generally that is measured in instruction/request per second.

The following techniques are:

- ASP.NET pipeline optimization
- ASP.NET process configuration optimization
- Things we must do for ASP.NET before going live
- Content Delivery Network
- Caching AJAX calls on browser
- Making best use of Browser Cache
- On demand progressive UI loading for fast smooth experience
- Optimize ASP.NET Profile provider
- How to query ASP.NET Membership tables without bringing down the site
- Prevent Denial of Service (DOS) attack

The above techniques can be implemented on any ASP.NET website, especially those who use ASP.NET Membership and Profile provider.

14.3.2 ASP.NET Pipeline Optimization

At times even after applying the best coding policies and practices you don't get the desired level of performance you are hoping from your ASP.NET application. This is because there are number other very important factors that directly affect ASP.NET applications. To get the best out of any system requires detail architectural, design, coding and deployment considerations.

Remove Unused HTTP Modules

There are various HTTP modules in ASP.NET that intercept each request sent to the server. Session State is a very commonly used HTTP module used to load session data in context object. It's referred with SessionStateModule name. HTTP modules kick in at each request and process them, therefore if you are not using the functionality provided by the module there is no use referring it as they would use additional CPU cycles. There is a list of HTTP Modules that your application automatically uses when it inherits config setting from web.config placed in \$WindowsFolder\Microsoft.NET\Framework\%version\CONFIG folder.

Below is a list of such entries:

```
<httpModules>
  <add name="OutputCache" type="System.Web.Caching.OutputCacheModule" />
  <add name="Session" type="System.Web.SessionState.SessionStateModule" />
    <add name="WindowsAuthentication"
type="System.Web.Security.WindowsAuthenticationModule" />
    <add name="FormsAuthentication"
```

Notes

```

type="System.Web.Security.FormsAuthenticationModule" />
    <add name="PassportAuthentication"
type="System.Web.Security.PassportAuthenticationModule" />
    <add name="RoleManager" type="System.Web.Security.RoleManagerModule" />
    <add name="UrlAuthorization"
type="System.Web.Security.UrlAuthorizationModule" />
    <add name="FileAuthorization"
type="System.Web.Security.FileAuthorizationModule" />
    <add name="AnonymousIdentification"
type="System.Web.Security.AnonymousIdentificationModule" />
    <add name="Profile" type="System.Web.Profile.ProfileModule" />
    <add name="ErrorHandlerModule" type="System.Web.Mobile.ErrorHandlerModule,
System.Web.Mobile, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a" />
    <add name="ServiceModel" type="System.ServiceModel.Activation.HttpModule,
System.ServiceModel, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089" />
</httpModules>

```

There are bunch of HTTP modules listed here and I am quite positive not all of them are being used by your application. Removing unused HTTP module can definitely give slight performance boost as there would be less work to be performed. Suppose one doesn't needs Windows authentication in application. To remove the inherited setting, under httpModules section in your web.config application add a remove element and specify name of the module that isn't required.



Example:

```

<httpModules>
<remove name="WindowsAuthentication" />
</httpModules>

```

```
<compilation debug="true"/> Killer
```

As a developer I have seen numerous incidents were the application is deployed to production with <compilation debug="true"/>. This is really a performance killer because:

- Compilation of ASP.NET pages take longer.
- Code execute slower as debug paths are enabled.
- More memory is used by the application.
- Scripts and images from WebResource.axd handler are not cached.

Always make sure that debug flag is set to false on production instances. You can override this by specifying following entry in machine.config for production instances:

```

<configuration>
<system.web>
<deployment retail="true"/>
</system.web>
</configuration>

```

This will disable the <compilation debug="true"/> for all applications deployed on the server.

14.3.3 ASP.NET Process Configuration Optimization

Notes

ASP.NET allows you to define many process level properties. By default these are set to auto config. This means that ASP.NET automatically configures `maxWorkerThreads`, `maxIoThreads`, `minFreeThreads`, `minLocalRequestFreeThreads` and `maxConnection` to achieve optimal performance. You can tailor these by specifying your own value to achieve better performance. Some of the major settings are:

A regular ASP.NET installation will create `machine.config` with the following configuration:

```
<system.web>
  <processModel autoConfig= "true"/>
```

We need to tweak this auto configuration and use some specific values for different attributes in order to customize the way ASP.NET worker process works.



Example:

```
<processModel enable= "true" Timeout= "Infinite"
idleTimeout= "Infinite"

shutdownTimeout= "00:00:05" requestLimit= "Infinite" requestQueueLimit=
"5000" restartQueueLimit= "10" memoryLimit= "60"
webGarden= "false" cpuMask= "0xffffffff" userName= "machine" password=
"AutoGenerate" logLevel= "Errors"
clientConnectedCheck= "00:00:05" comAuthenticationLevel= "Connect"
comImpersonationLevel= "Impersonate" responseDeadlockInterval= "00:03:00"
responseRestartDeadlockInterval= "00:03:00" AutoConfig= "false"
MaxWorkerThreads= "100"

maxIoThreads= "100" minWorkerThreads= "40" minIoThreads="30"
serverErrorMessageFile="" pingFrequency="Infinite" pingTimeout= "Infinite"
asyncOption= "20" maxAppDomains= "2000"
/>
```

Here all the values are default values except for the following ones:

maxWorkerThreads

The default value is 20 per process and it determines the maximum number for request that ASP.NET can process in a given second. For application that are not CPU intensive and most of time wait on database request or any external processing this can be increased to get better performance.

maxIOThreads

The default value is 20 per process and it determines the maximum number for I/O request that ASP.NET can process in a given second. If you have enough I/O resources you can increase this value for better results.

connectionManagement

This is a property of System.Net configuration and specifies the maximum parallel connections that can be established to a server. If your web application extensively connects to other server you can increase this value.

Notes

memoryLimit

The default is 60%. This is the max memory ASP.NET can use until worker process is refreshed. If you have a dedicated web server with no other services running you can increase this value for better results.

Besides the processModel, there's another very important section with the system.net where you can specify the maximum number of outbound requests that can be made to a single IP.

```
<system.net>
  <connectionManagement>
    <add address="*" maxconnection="100" />
  </connectionManagement>
</system.net>
```

Default is 2, which is just too low. This means you cannot make more than two simultaneous connections to an IP from your Web application. Sites that fetch external content a lot suffer from congestion due to the default setting. Here I have set it to 100. If your Web applications make a lot of calls to a specific server, you can consider setting an even higher value.



Case Study

Recruiting Portal

Client

Ajilon Consulting develops customized IT solutions for Fortune 1000, mid-tier, government and private organizations in every industry, including finance, healthcare, insurance, telecommunications, manufacturing, utilities and transportation.

Situation

It was required to develop two branches of the recruiting agency portal, having general architectural design, general database (one for all branches), but different UI design and its behavior.

Project Overview

The core essence for the system is realization of recruitment functions as a general help for jobseekers, employers, and recruitment agencies/offices. Jobseekers enter their CVs, detailed information, position requirements and gets through results needed.

In order to best assess what can be accommodated within this solution, it has been broken down into categories to allow a clear understanding of what can be achieved and to determine best how to resource this in the timeframe available.

The categories are as follows:

- General Website Development;
- Candidate Registration;
- Client Registration;
- Vacancy Registration;
- Vacancy Posting;

Contd....

- Candidate Job Applications;
- Basic MIS reporting.

Web Accessibility Initiatives (WAI) were taken into consideration when building the site to ensure the sites are user friendly to all customers, irrelevant to their connection method, i.e. PC, phone etc.

The system supports all commonly used browsers including PC and Mac browsers and to be more specific, all versions of IE 5.0 (sp1) and above, Netscape 6.0 and above. Search Engine Optimization was kept in mind while developing the system/pages to ensure that the Ajilon Brands are shown in the best light when outside people do searches on key search sites such as Google and Yahoo.

Questions:

1. Explain the problem background of this project.
2. State the benefits of customer through this project.

Self Assessment

Multiple Choice Questions:

6. A process where items are removed from cache in order to free the memory based on their priority.
 - a. Cache Priority
 - b. Cache Element Priority
 - c. Cache Item Priority
 - d. None of the Above
7. Select the caching type supported by ASP.NET
 - a. Output Caching
 - b. DataCaching
 - c. Both a & b
 - d. None of the above
8. Which of the following denote value that can be taken by Cache-Control of ASP.NET?
 - a. Public
 - b. Private
 - c. no-cache
 - d. All the Above
9. Performance profiling is a activity because application performance changes overtime.
 - a. space-dependent
 - b. space and time-dependent
 - c. time-dependent
 - d. None of these
10. What are the types of dependencies in Cache?
 - a. File Dependency
 - b. Time Based Expiration
 - c. Key Dependency
 - d. All of the above
11. You have a Web application that is configured for personalization. You need to access personalization data from one of the pages of the Web application by using the minimum amount of administrative effort. What should you do?
 - a. Access the personalization data from the Session property of the HttpContext object.
 - b. Access the personalization data from the Application property of the HttpContext object.

Notes

- c. Access the personalization data from the Cache property of the HttpContext object.
- d. Access the personalization data from the Profile property of the HttpContext object.
- 12. An e-shopping web portal has a shopping basket control where the list of available items meant for shopping is displayed. The item list remains unchanged for all users irrespective of scenarios and it always populates from a database look up table every time when the page is loaded. For getting better performance and memory effectiveness, it is appropriate to store the data in
 - a. Session
 - b. Cache
 - c. Server registry
 - d. All
- 13. Which object involves reading the data in a local cache?
 - a. Command
 - b. Dataset
 - c. Connection
- 14. Cache["dd"]="ASP.NET"; What is the timeperiod of this Cache object?
 - a. Infinite
 - b. 0 seconds
 - c. None of these
- 15. You are developing a web application that is retrieving historical library information from database server and displays it to the users of your application. What cache strategy will give you the best performance?
 - a. Use the output cache
 - b. Use a cache object
 - c. Use the ASP.NET central cache
 - d. Use the client cache

14.4 Summary

- Debugging is the process of finding and correcting the errors in a program.
- We can enable debugging in ASP.NET page by writing the `<%@Page debug= "True" %>` page directive at the top of the page.
- The SDK debugger that comes along with .NET SDK can be found in the Guidebook directory.
- Just-In-Time debugging is another technique to debug a program that is started outside of Visual Studio. If we have enabled Just-In-Time debugging, we can view a dialog box when a crash occurs.
- ASP.NET is a web application framework developed by Microsoft to allow programmers to build dynamic web sites and web applications. ASP.NET supports compiling applications in a special debug mode that facilitates developer troubleshooting.
- You can configure the Web application to allow tracing at either the page level or the application level.
- Cassini is a "simple, fully managed Web server that hosts ASP.NET".
- It is possible to attach the Visual Studio debugger to processes that are already running.
- To improve the performance of ASP.NET application make sure that the application is not deployed with `Debug=True` attribute.
- Caching is about storing data in memory the first time it is requested and then re-using it for the following requests for a specified period of time.

- In ASP.NET 2.0, the caching support is integrated with the DataSource controls to cache data in a web page.
- ASP.NET 2.0 provides a new control called the Substitution control, which enables you to insert dynamic content into a cached web page.
- The SqlDataSource control also has the EnableCaching property set to true, which results in the SqlDataSource automatically caching the data retrieved by the SelectCommand.
- You can programmatically manipulate the cache to place individual data items in it, thereby reducing the number of calls required to a backend database.
- ASP.NET application a big performance and scalability boost and prepare it to perform well under 10 times to 100 times more traffic.
- ASP.NET allows you to define many process level properties. By default these are set to auto config.

14.5 Keywords

Cache Keys: The cache key is the unique key that can be used to identify objects in the ICM server cache.

Pipeline: Computer processors can handle millions of instructions each second. Once one instruction is processed, the next one in line is processed, and so on. A pipeline allows multiple instructions to be processed at the same time.

Scalability Boot: Remove performance bottlenecks related to our data storage and databases and scale our .NET and Java applications to extreme transaction processing (XTP) with NCache.

SDK Debugger: This section details the software and hardware supported and required by the PayPal SDK, installation, and post-installation tasks. These samples can be installed in Microsoft Internet Information Server (IIS).

SqlDataSource Level: The SqlDataSource control enables us to use a Web server control to access data that is located in a relational database. This can include Microsoft SQL Server and Oracle databases.

Trump: Trump means to get the better of someone using a hidden resource.



Lab Exercise

1. Write C# program using SqlDataSource.
2. Write a C# program to create a data base connectivity.

14.6 Review Questions

1. What are the types of Caching in ASP.NET?
2. How will you differentiate between trace and debug in ASP.NET?
3. How to check or assess debugging skills of a person?
4. What are the difference between the Cache object and the Application object?
5. Which type of caching will be used if we want to cache the portion of a page instead of whole page?

Notes

6. How can we prevent browser from caching an ASPX page?
7. Is it possible to attach the Visual Studio debugger to processes that are already running?
8. How can we disable debugging in ASP.NET?
9. Discuss the upside and downside of using cache.
10. Discuss the techniques used for measuring performance of ASP.NET.

Answers: Self Assessment

- | | | | |
|---------|---------|---------|---------|
| 1. (c) | 2. (a) | 3. (d) | 4. (c) |
| 5. (b) | 6. (c) | 7. (c) | 8. (d) |
| 9. (c) | 10. (d) | 11. (d) | 12. (b) |
| 13. (b) | 14. (b) | 15. (b) | |

14.7 Further Readings



Books

Performance Tuning and Optimizing ASP.NET Applications, by Jeffrey Hasan, and Kenneth Tu



Online links

<http://www.wintrac.com/courses/dotnetTuning.asp>

LOVELY PROFESSIONAL UNIVERSITY

Jalandhar-Delhi G.T. Road (NH-1)

Phagwara, Punjab (India)-144411

For Enquiry: +91-1824-300360

Fax.: +91-1824-506111

Email: odl@lpu.co.in