Mahindra Satyam

Workbook for ASP.Net

A beginner's guide to effective programming

Why This Module

Static Web pages, which display the same content each time a user views them are not relevant in today's scenario. A Web page needs to display dynamic Web content in order to attract the Web community to revisit the Web page.

The content of dynamic Web pages changes every time they are displayed to a user. Some examples of dynamic Web pages are the pages that display the current weather forecast or display current sales items. Such Web pages display the most current information each time a visitor revisits the page.

Active Server Pages.NET (ASP.NET) is the language used for creating dynamic Web pages.

Active Server Pages allow the creation of Web pages containing a set of standard HTML elements, such as tables and text, and scripting elements, such as database fields and date/time information for dynamic generation of the Web page each time it is requested from a browser. When a browser requests an ASP.NET page, the page is processed by the Internet Information Server (IIS).

In this module, participant will be introduced to ASP.NET applications. They will learn about the components of an ASP.NET Web page. Then, students will learn to create Web applications for displaying dynamic data. In addition, you will also learn to configure, debug, and deploy ASP.NET Web applications

Contents

Conventions Used

Convention	Description
 **Topic**	Indicates the Topic to be discussed.
Estimated Time	Overview of estimated time needed to understand the Topic and complete the Practice session.
 **Presentation**	A brief introduction about the Topic.
 **Scenario**	An idea on real time situation where topics are used.
 **Demonstration/Code Snippet**	Topics implemented in real time code along with screenshots
Code in Italic	Italic framed lines are generated by the System related to that particular event.
// OR ‘	Topic is being described from complete program as code snippet
 **Context**	When the Topic should be used in an application.
 **Practice Session**	Participant should be able to implement the Topic to develop an Application in practice sessions.
 **Check list**	List of brief content of the Topic.
 **Common Errors**	List of common errors occurred while developing an Application.
 **Exceptions**	List of exceptions resulted from the execution of an Application.

 **Lessons Learnt**	Lessons learnt from the article of the workbook.
 **Best Practices**	Efficient development of the an Application through best ways
 **Notes**	Important information related to the Topic

1.0 Introduction to ASP .Net

Topics

- 1.1 ASP .Net Framework
- 1.2 Visual Studio .Net Overview
- 1.3 ASP .Net Application Folder
- 1.4 ASP .Net Architecture
- 1.5 Crossword

Topic: ASP .Net Framework 2.0

Estimated Time: 20 mins.

Objectives : This module will familiarize the participant with

- ASP .Net Framework and its features.

Presentation :

- ASP.NET is a technology for developing web applications**.** A web application is just a fancy name for a dynamic web site. Web applications usually (but not always) store information in a database, and allow visitors to the site to access and change that information.

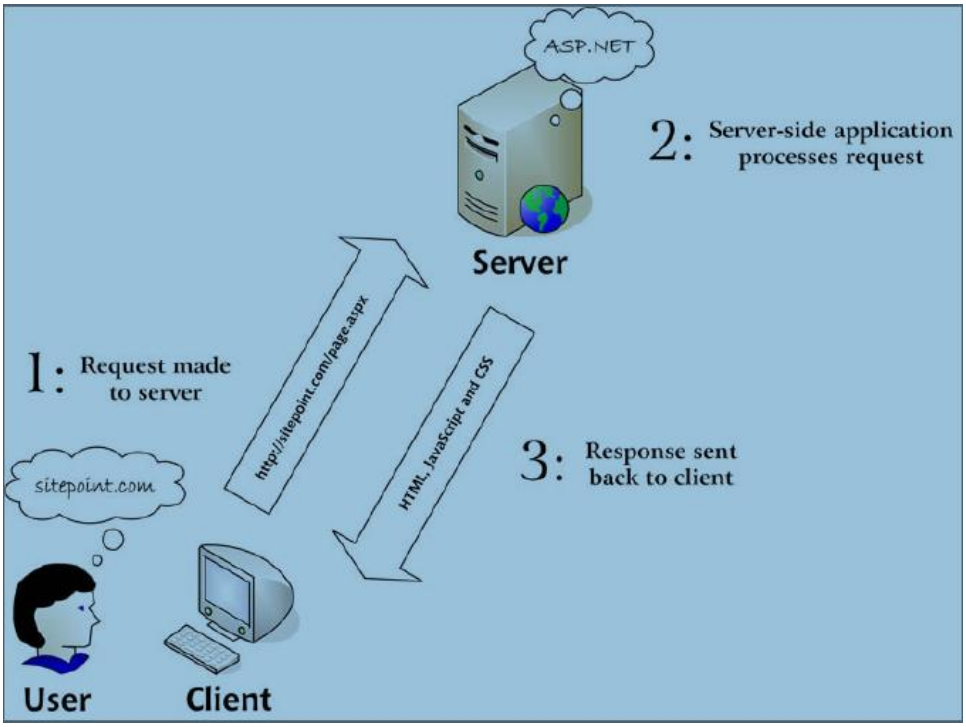


Figure 1.1-1: User interacting with Web Application.

- ASP.NET is a server-side technology for developing web applications based on the Microsoft .NET Framework.
- .NET is Microsoft's development model in which software becomes platform and device-independent, and data becomes available over the Internet.
- ASP.NET uses the Microsoft .NET Framework
- .Net is Language-Independent and Platform-Independent technology.

- The .Net platform consists of the following core technologies:
  - The .NET Framework
  - The .NET Enterprise Servers
  - Building Block Services
  - Visual Studio .NET

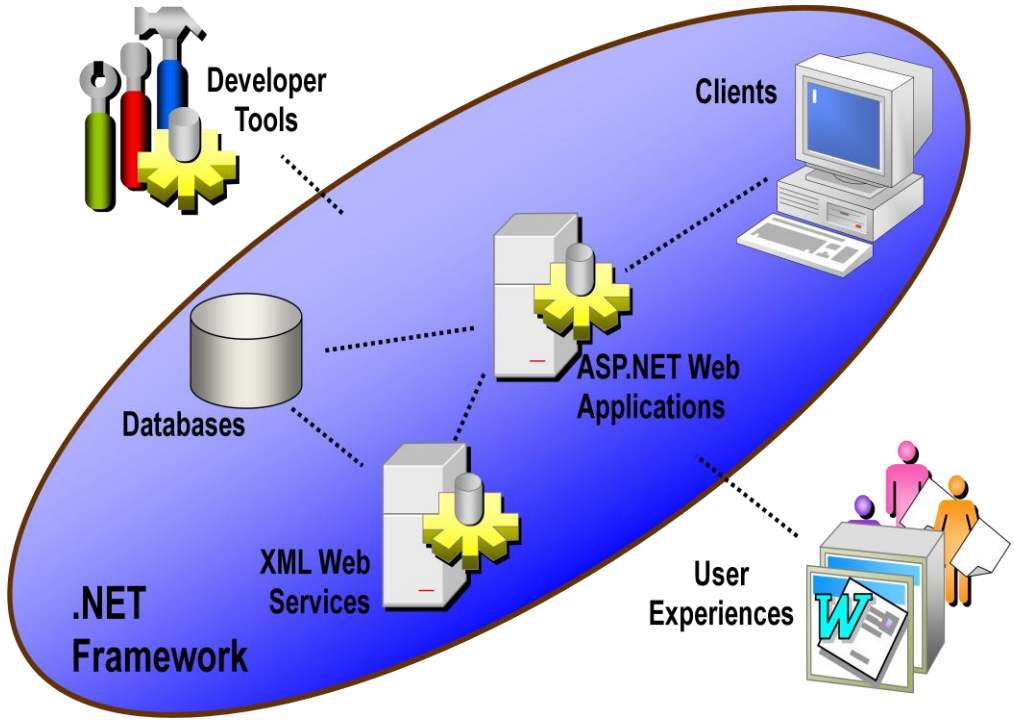


Figure 1.1-2: .Net Framework.

- Below figure illustrate the element of an ASP .Net application and how the elements fit in the broader context of the .NET Framework.

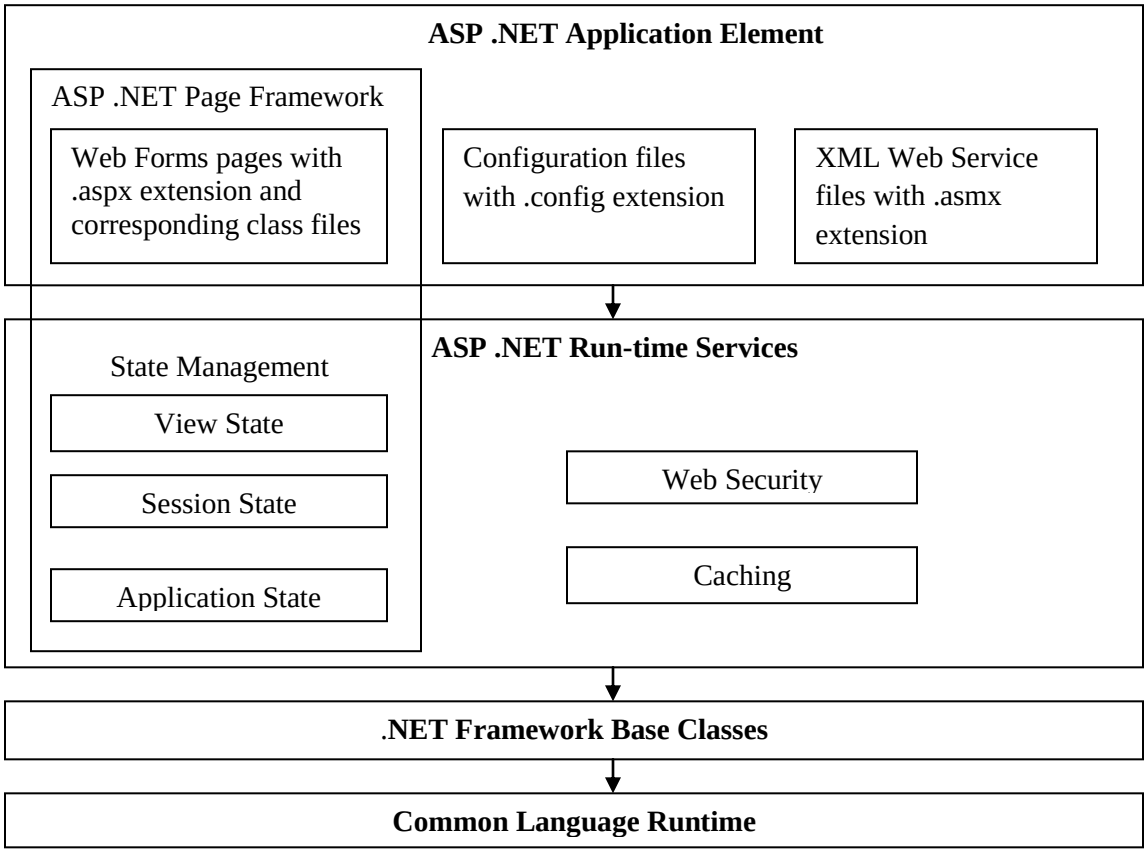


Figure 1.1-3: ASP .Net Application Elements.

- Elements of an ASP .Net application include Web Form, Configuration Files and XML Web Services Files.
- Web Forms enables you to include user interfaces, such as TextBox, ListBox controls and application logic of Web Applications.
- Configuration File enable user to store the configuration settings of an ASP .NET application.
- Web Service provides mechanism for programs to communicate over the Internet.
- The Common Language Runtime (CLR) in the .Net Framework manages the execution of the code and provides access to a variety of services that make the development process easier.
- .Net Frame Work Class library contains different type of Namespaces, Classes and Methods for different operation and to manipulate data. E.g. System, Web, Microsoft etc.
- Namespace has two basic functionality
  - Namespace Logically group types, example System.Web.UI logically groups
  - Namespace avoids class Name collision.

The .NET Framework provides the necessary compile-time and run-time foundation to

build and run .NET-based applications.

ASP .Net Features:

- The .NET Framework currently supports over 40 languages, and many of these may be used to build ASP.NET web sites.
- ASP.NET pages are **compiled**, not interpreted.
- ASP.NET has full access to the functionality of the .NET Framework. Support for XML, web services, database interaction, email, regular expressions, and many other technologies.
- ASP.NET allows user to separate the server-side code in pages from the HTML layout.
- ASP.NET makes it easy to reuse common User Interface elements in many web forms, as it allows user to save those components as independent web user controls.
- Visual Web Developer 2005 provides powerful visual editor that includes features such as code auto completion, code formatting, database integration functionality, a visual HTML editor, debugging, and more.
- .NET application has assemblies which is unit of deployment like EXE or a DLL. An assembly consists of one or more files (dlls, exe's, html files etc.), and represents a group of resources, type definitions, and implementations of those types. These resources, types and references are described in a block of data called a manifest. The manifest is part of the assembly, thus making the assembly self-describing. It is independent of registry.
- In Object Oriented programming many times its possible that programmers will use the same class name. By qualifying Namespace with class name this collision can be avoided.

In ASP.NET's predecessor, ASP, pages were interpreted: every time a user requested a page, the server would read the page's code into memory, figure out how to execute the code (that is, interpret the code), and execute it. In ASP.NET, the server need only figure out how to execute the code once. The code is compiled into efficient binary files, which can be run very quickly, again and again, without the overhead involved in re-reading the page each time.

Context :

- To have language interoperability.
- To develop Web applications, Windows applications, and XML Web services.

Practice Session/s :

- Identify the ASP.NET Application Elements.
- Identify the features of ASP.NET Framework.

Check List :

- Provide a code-execution environment that minimizes software deployment and versioning conflicts by using Namespaces.
- Make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- Build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

Common Error/s :

- Assuming .Net to be a language.
- Using C# .net for developing software drivers.
- Implementing low level API without checking for availability of functionality in .Net.

Exception/s :

- Trying to Access service provided by framework 2.0, whereas the system supports .Net version 1.1.

Lesson/s Learnt :

- ☑ Language independent and platform independent technology.
- ☑ The functionality of a .NET class is available from any .NET-compatible language or Programming model. Therefore, the same piece of code can be used by Windows Applications, Web applications, and XML Web services.

Best Practice/s :

- Language independent implementation of classes.
- To provide runtime error checking, security and managed execution.

Topic: ASP .Net Architecture

Estimated Time: 30 mins.

Objectives : This module will familiarize participant with the

- ASP.Net Architecture.

Presentation :

- ASP.NET is a sophisticated engine using Managed Code for front to back processing of Web Requests.
- Managed code runs inside the environment of CLR i.e. .NET runtime. In short all IL are managed code.
- ASP.NET is a request processing engine. It takes an incoming request and passes it through its internal pipeline to an end point where a developer can attach code to process that request.
- HTTP Runtime is a component that can be hosted in applications outside of IIS or any server side application altogether.

IIS Server:
IIS Server is Web Server for the Windows Platform. IIS Server enables user to access the ASP .NET application that user has developed.

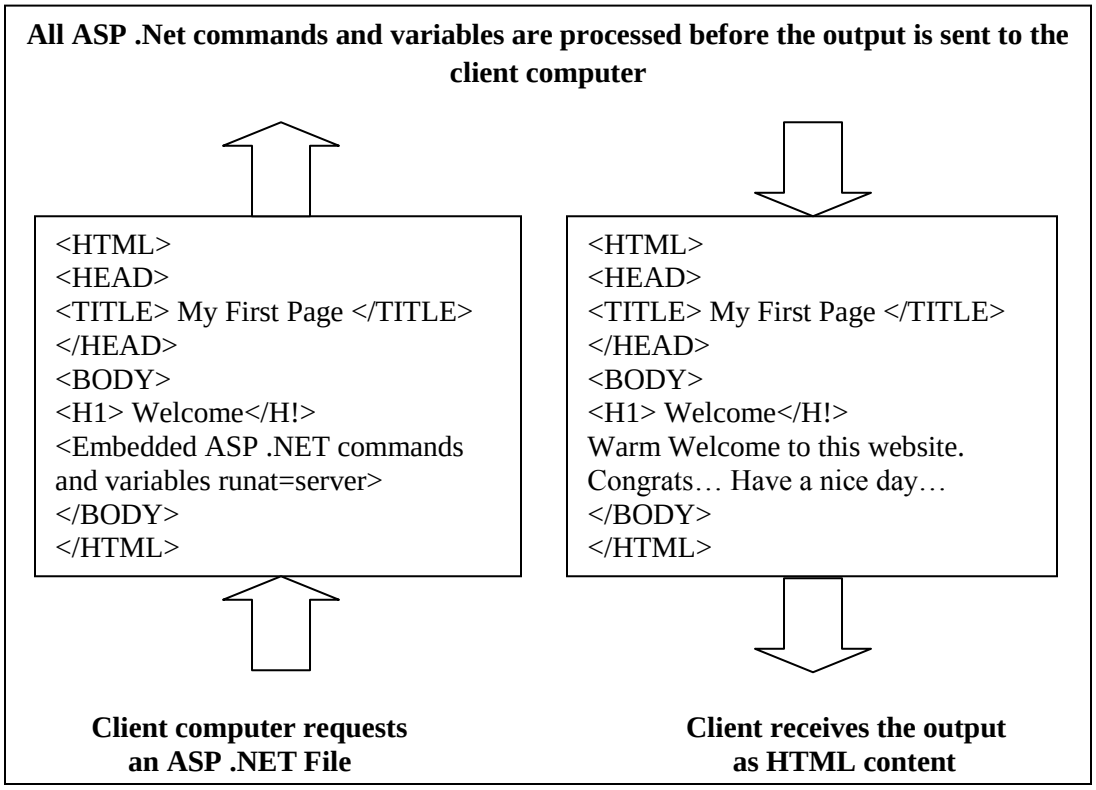


Figure 1.2-2: Processing the Request for an ASP .NET file by a Web Server.

Following steps are taken to execute an ASP .NET file:

1. A web browser sends a request for an ASP .NET file to a Web Server by using a URL.
2. The web server IIS receives the request and retrieves the appropriate ASP .NET file from the disk or memory.
3. The Web server forwards the ASP .NET file to the ASP .NET script engine for processing.
4. The ASP. NET script engine reads the file from top to bottom and executes any server-side scripts it contains.
5. The processed ASP .NET file is generated as an HTML document and the ASP .NET script engine sends the HTML page to the Web Server.
6. The Web Server the sends the HTML page to the client.
7. The Web Server interprets the output and displays it.

All the Asp .NET files need to be stored on an IIS server. An IIS Server processes ASP .NET files from top to bottom and then executes the scripts. Finally, results are sent back to the Web Browser that requested the ASP .NET file.

A Web Server generates and sends only the HTML output to the client. So, it helps in hiding the code of the ASP .NET file from users accessing an ASP .NET Web Page.

Source code	Versioning Info generated --------- > Compiler	IL (CPU Independenet Managed Code)	Code Verification ---------- > JIT Compiler	Native Code (Platform Independent)

- The .Net Framework architecture has two main components:
  - **CLR**
  - **Class Library**

CLR: CLR is a runtime environment that manages the execution of .NET program code.

- The CLR is a major component of the .NET framework.
- CLR is also known as the Virtual Execution System (VES).
- Following are the responsibilities of CLR

  - Garbage Collection

CLR automatically manages memory thus eliminating memory leaks. When objects are not referred GC automatically releases those memories thus providing efficient memory management.

- Code Access Security (CAS)

  CAS grants rights to program depending on the security configuration of the machine.

  Example:

  The program has rights to edit or create a new file but the security configuration of machine does not allow the program to delete a file. CAS will take care that the code runs under the environment of machines security configuration.

- Code Verification

  This ensures proper code execution and type safety while the code runs. It prevents the source code to perform illegal operation such as accessing invalid memory locations etc.

- Intermediate Language (IL)

  IL(Intermediate language)-to-native translators and optimizer's: CLR uses JIT and compiles the IL code to machine code and then executes. CLR also determines depending on platform what is optimized way of running the IL code.

Class Library:

- **Object**: - It's a basic unit of system. Object is an entity that has attributes, behavior, and identity. Objects are members of classes. Attributes and behavior of an object are defined by the class definition.
- **Class:** - A class describes all the attributes of objects, as well as the methods that implement the behavior of member objects. It's a comprehensive data type which represents a blue print of objects.
- **Class Library:** - ASP.NET includes an enormous class library which was built by Microsoft. Because this class library is so large, it encapsulates a huge number of common functions.

Context :

- To familiarize with the ASP.Net Architecture

Practice Session :

- Identify CLR and Class Library.
- Identify concept of IIS.
- Identify the process involved in executing an ASP .NET page.

Check List :

- CLR is a runtime environment that manages the execution of .NET program code.
- Runtime Host is a component that takes care of the functionalities of CLR.

Common Errors :

- Check whether IIS is installed and configured.

Exceptions :

- Running a web application without installing an IIS Server.

Lessons Learnt :

☑ CLR is an important component that manages the execution of .NET program code.

Best Practices :

- IIS Server must be installed on the computer to run the Web Application.

Topic: Visual Studio .Net Overview

Estimated Time: 40 mins.

Objectives : At the end of this module participant should understand

- Purpose of Visual Studio.NET
- VS.NET's Integrated Development Environment.
- Available windows in the IDE

Presentation :

- Microsoft Visual Studio is the main Integrated Development Environment (IDE) from Microsoft.
- It can be used to develop console and Graphical user interface applications along with Windows Forms Applications, Web Sites, Web Applications, and Web Services in both Native Code as well as Managed Code.
- Visual Studio .NET provides templates that support the creation of number of common Project Type.
- Creating a project in Visual Studio .NET, also create a larger container called a **Solution Explorer,** which is a graphical view to organize and manage all the projects and files that are needed to design, develop and deploy an application.

Scenario :

Satyam computer Ltd. wants to develop different types of application like windows application, web based application. To develop such kind of application visual studio is used.

Demonstration/Code Snippet :

Step 1: Start page is displayed each time Visual Studio is opened. The 'Start Page' which contains shortcut 'links' to the last few opened projects or websites(if any) appears.

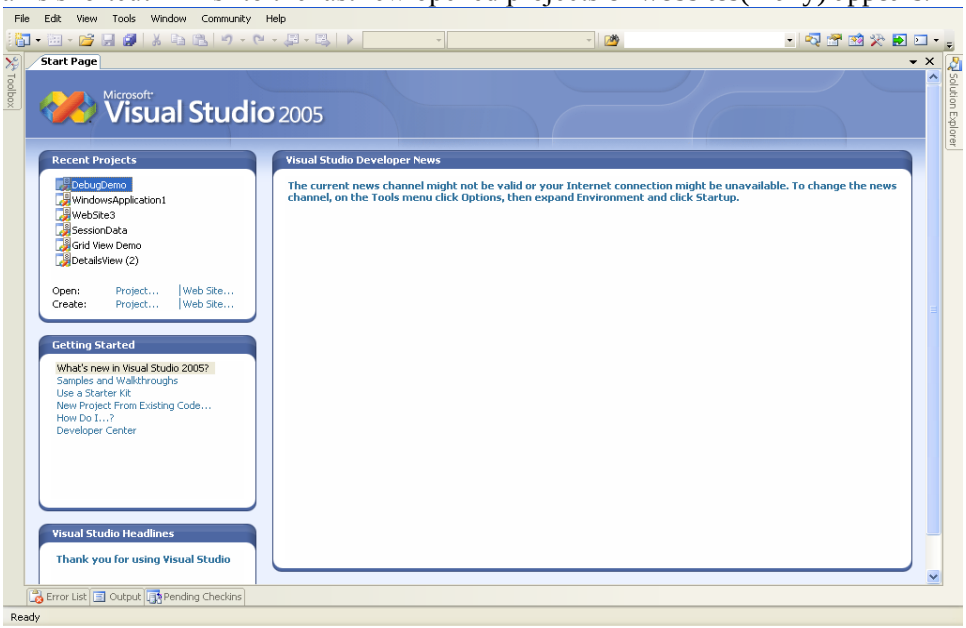


Figure 1.3-1: VS .Net IDE.

Step 2: Visual Studio contains different projects categories like Visual Basic, Visual C#, Visual J#, Visual J# and other project and create Windows and Console Application. Open a new project by selecting File > New > Project… from the menu. The New Project dialog appears. Assign Name, Location and Solution Name to the Project.

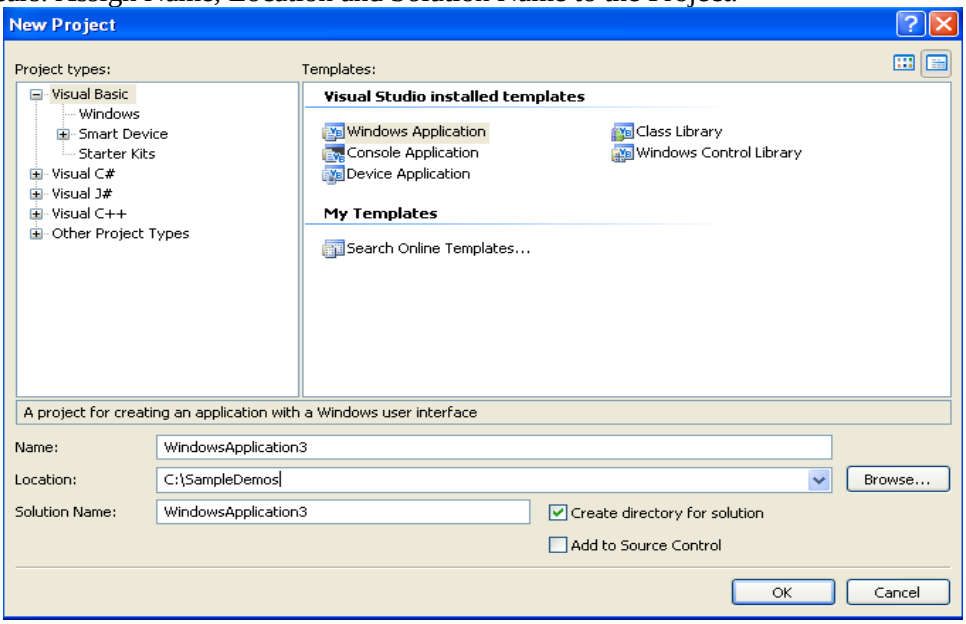


Figure 1.3-2: New Project Dialog box.

Using Visual Studio websites can be created like ASP .Net or ASP .Net Web Service using languages like Visual Basic, Visual C# and Visual J#. Open a new website by selecting File > New > Website… from the menu. The New Web Site dialog appears. Assign Location and Language of the Website.

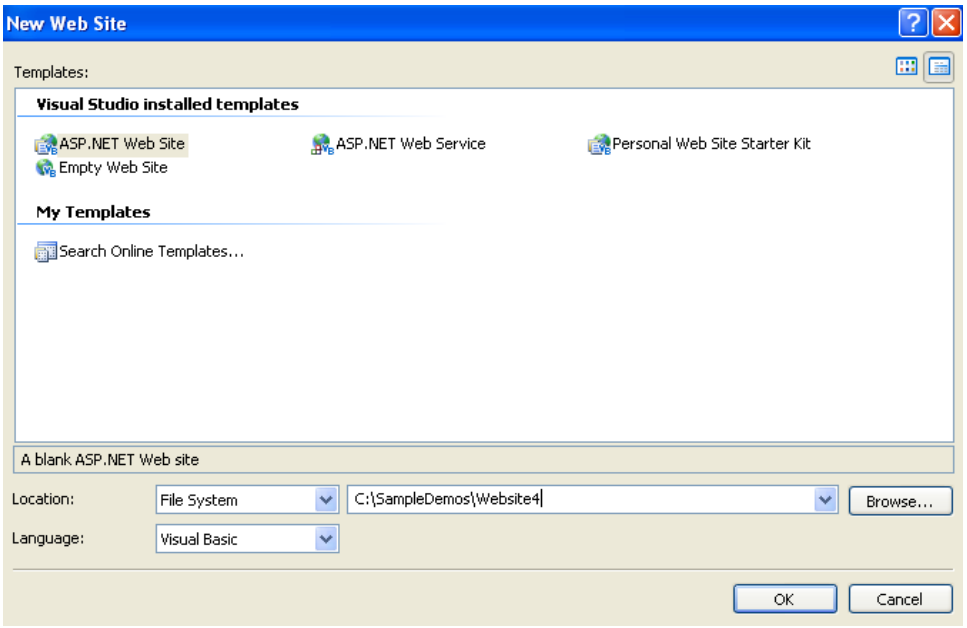


Figure 1.3-3: New Web Site Dialog box.

Step 3: The Visual Studio .NET IDE contains multiple windows that provide a variety of tools and services.

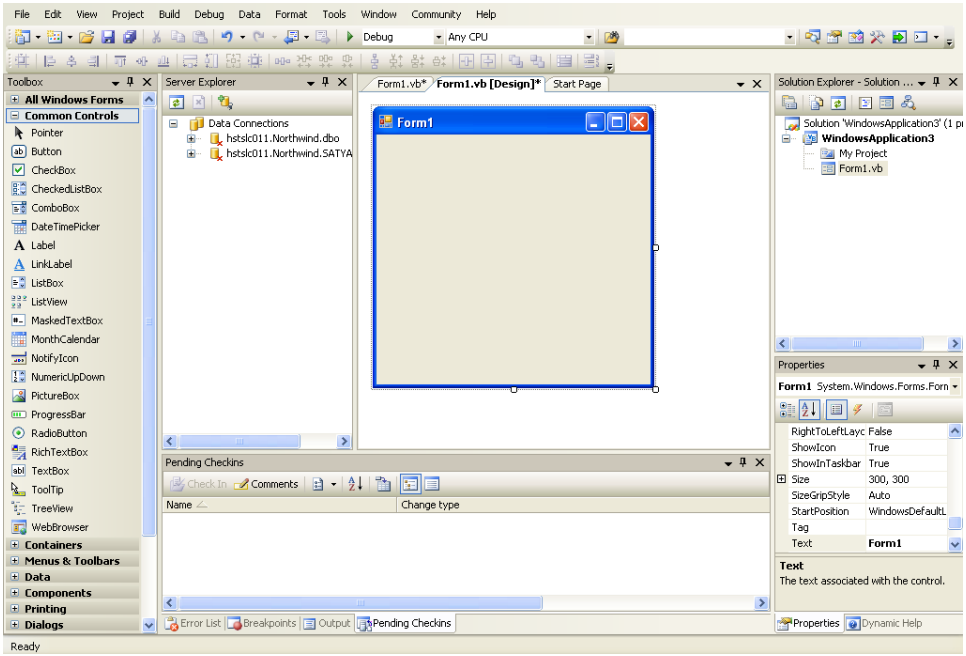


Figure 1.3-4: New Window Application Window.

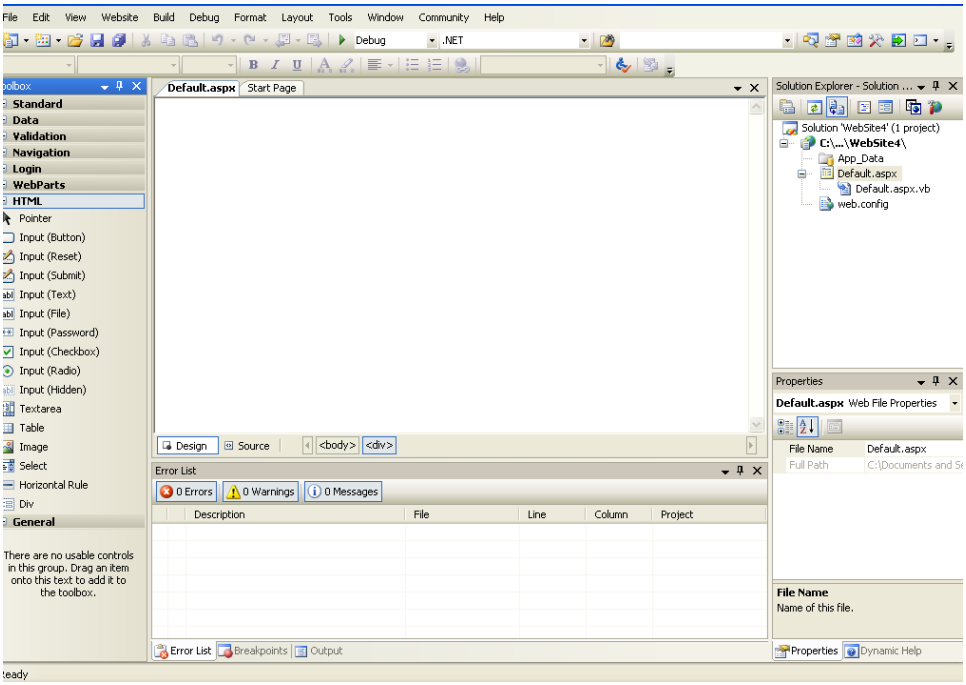


Figure 1.3-5: New Website Window.

Solution Explorer:

Solution Explorer window contains Project and all the Windows Forms added to the Project.

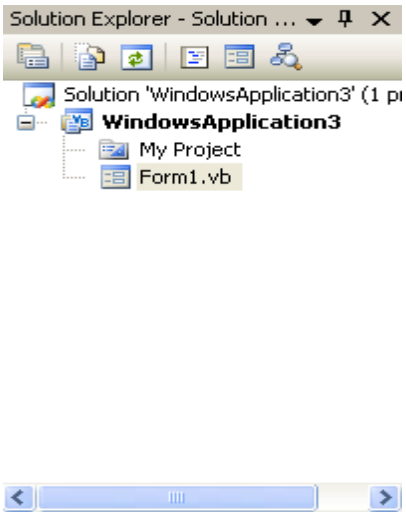


Figure 1.3-6: Solution Explorer.

Properties:

Property Window contains properties related to Controls

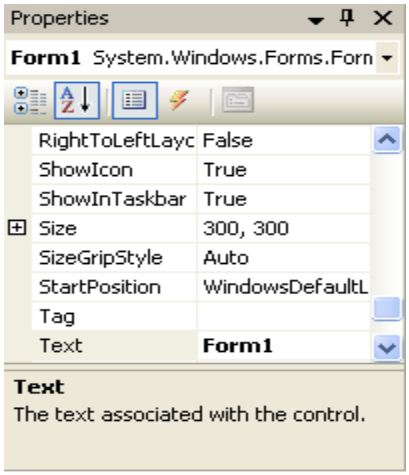


Figure 1.3-7: Property.

Server Explorer:

Server Explorer allows to access data from database and binding it with controls.

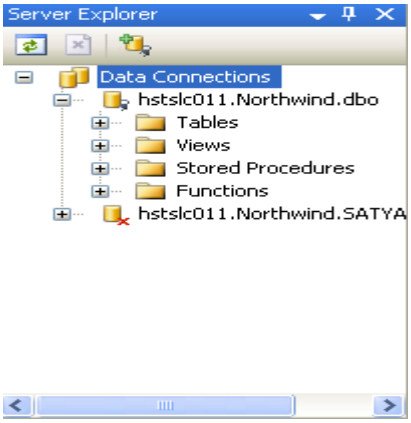


Figure 1.3-8: Server Explorer.

Toolbox:

Toolbox contains different categories of Controls which enables user to create Applications.

Figure 1.3-9: Toolbox.

Object Browser:

Display all the Design and Coding Window (.cs[Design] / .Designer.cs / .aspx / .aspx.cs)

Figure 1.3-10: Object Browser

Context :

- To familiarize with Visual Studio .Net IDE.

Practice Session :

- Review different project templates.
- Review all the windows available in Project/Website Window.

Check List :

- To create, build and view different types of application such as Window application, console application and other.

Common Errors :

- No Specific Exception

Exceptions :

- Visual Studio is not Backward Compatibility to its older versions.

Backward compatibility: Note that VS.NET 2005 is not 'backward compatible' with VS.NET 2003; a project or solution created using VS.NET 2003 can be opened in VS.NET 2005, but thereafter cannot be opened using VS.NET 2003 again. The reason is because VS.NET 2005 uses the newer .NET Framework version 2.0 compared to VS.NET 2003, which uses .NET Framework 1.1. When a VS.NET 2003 project is opened in VS.NET 2005, the project will be automatically converted to consume classes from the newer .NET Framework. A VS.NET 2005 project or solution thus cannot be opened in VS.NET 2003.

Lessons Learnt :

- The primary features of visual studio.NET for creating a Web application and Project.
- Navigate the Visual Studio .NET integrated development environment.

Best Practices :

- Never try to use the Project/Website created in newer VS.NET in older VS.NET.

Topic: ASP .Net Application Folders

Estimated Time: 20 mins.

Objectives : At the end of this module participant should be able to understand the

- Basics of ASP .Net Application Folders

Presentation :

App_Code:

- The App_Code folder is meant to store classes, .wsdl files, and typed datasets. Any of these items stored in this folder then is automatically available to all the pages in the solution.

App_Themes:

- Themes are used to define visual styles for web pages. Themes are made up of a set of elements: skins, Cascading Style Sheets (CSS), images, and other resources.

App_Data:

- It Contains application data stored in the form of files including MDF files, XML files, text file as well as other data store files.
- The user account utilized by the application will have read and write access to any of the files contained within the App_Data Folder.

App_GlobalResources:

- Application resources are stored outside of the application so that they can be recompiled and replaced without affecting and recompiling the application itself.
- Resource values in .resx files in the App_GlobalResources folders can be accessed programmatically from application code.
- Add Assembly Resource Files (.resx) to this folder and they are dynamically compiled and made part of the solution for use by all your aspx pages in the application

App_LocalResources:

- We can add resource files that are page-specific to the \App_LocalResources folder.
- We can define multiple .resx files for each page, each .resx file representing a different language or language/culture combination.

App_WebReferences:

- The App_WebReferences folder is a new name for the previous Web References folder used in previous versions of ASP.NET.
- Now you can use the \App_WebReferences folder and have automatic access to the remote Web services referenced from your application.

App_Browsers:

- The App_Browsers folder holds browser files, which are XML files used to describe characteristics and capabilities of these browsers.
- We can find a list of globally accessible browser files in the CONFIG\Browsers folder under the installation path.

Bin:

- It contains compiled assemblies (.DLL files) for controls, components, or other code that we want to reference in our application.
- Any DLL files found in the directory will be automatically linked in the application. This is also recognized by ASP.NET 1.x applications. This folder is mandatory.

Context :

- Use App_Code folder to store classes, .wsdl files, and typed datasets.
- Use App_Themes Folder for theme
- Use App_Data Folder to store MDF files, XML files, text file as well as other data store files.
- Use App_WebReferences folder to have automatic access to the remote Web services referenced from your application.
- Use App_Browsers folder to change any part of default browser definition files

Practice Session :

- Identify the special directories for application resources.

Check List :

- Use of special directories for application resources.

Common Errors :

- Files in the App_Data folder are not served by the Web server. Do not store any Web pages in the App_Data folder because users will see an error if they request a page from that folder.

Exceptions :

- Improper usage of App_Data may leads to an exception.

Lessons Learnt :

☑ Understanding Special directories for application folder.

Best Practices :

- To have appropriate arrangement of resources of your website use Application Folders.

Crossword: Unit-1 **Estimated Time: 10 mins.**

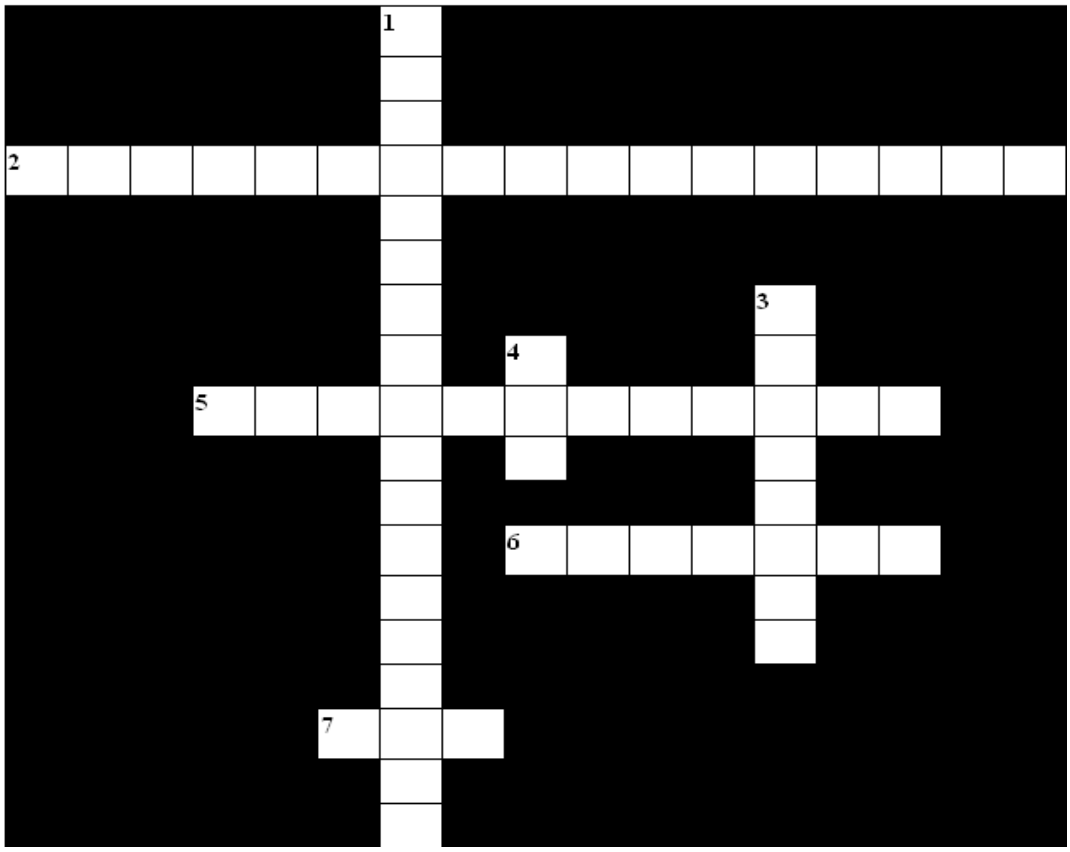


Fig.1-1

Across:

2. One of the major part of CLR, which is related to memory management.(17)

5. One of the component of .NET which provides reusable code for most common tasks, including data access, XML Web service development, and Web and Windows Forms.(12)

6. One of the application folder of ASP.NET which is meant to store classes, .wsdl files, and typed datasets. Any of these items stored in this folder then is automatically available to all the pages in the solution.(7)

7. One of the application folder of ASP.NET which contains compiled assemblies (.DLL files), any DLL files found in the directory will be automatically linked in the application.(3)

Down:

1. One of the responsibilities of CLR and a part of the .NET security model that determines whether or not a piece of code is allowed to run.(18)

3. .NET is independent of.(8)

4. It is a runtime environment that manages the execution of .NET program code, a major component of the .NET framework and also known as the Virtual Execution System (VES).(3)

2.0 Creating ASP .Net Web Form

Topics

- 2.1 Web Forms
- 2.2 Master Pages
- 2.3 Server Controls
- 2.4 Data Bound Controls
- 2.5 Themes and Skins
- 2.6 Crossword

Topic: Web Forms

Estimated Time: 40 mins.

Objectives : At the end of the activity, the participant should understand

- Designing Web Forms
- Choose appropriate controls based on the requirements

Presentation :

- Web forms enable to create user interface for Web applications.
- Web Forms can be designed and programmed using Rapid Application Development (RAD) tools.
- Web forms can be programmed using any of the .Net Framework languages such as c#, J# or visual basic .Net.
- Use server controls to design the user interface of Web applications and then write code, which will be executed at the server-side, to handle events triggered by these controls.
- A rich set of server-side controls that can detect the browser and send out appropriate markup language such as HTML.
- Less code to write due to the data binding capabilities of the new server-side .NET controls.
- A user interface consists of static HTML or XML elements and ASP.NET server controls.

Scenario :

Mr. Brown, the manager of X-Mart wants that in future the new recruites of their company should fill their Personal details online. So Mr. brown suggest developers to create a Web design which enables their employees to enter their personal details .

Demonstration/Code Snippet :

Design of the web page as shown below

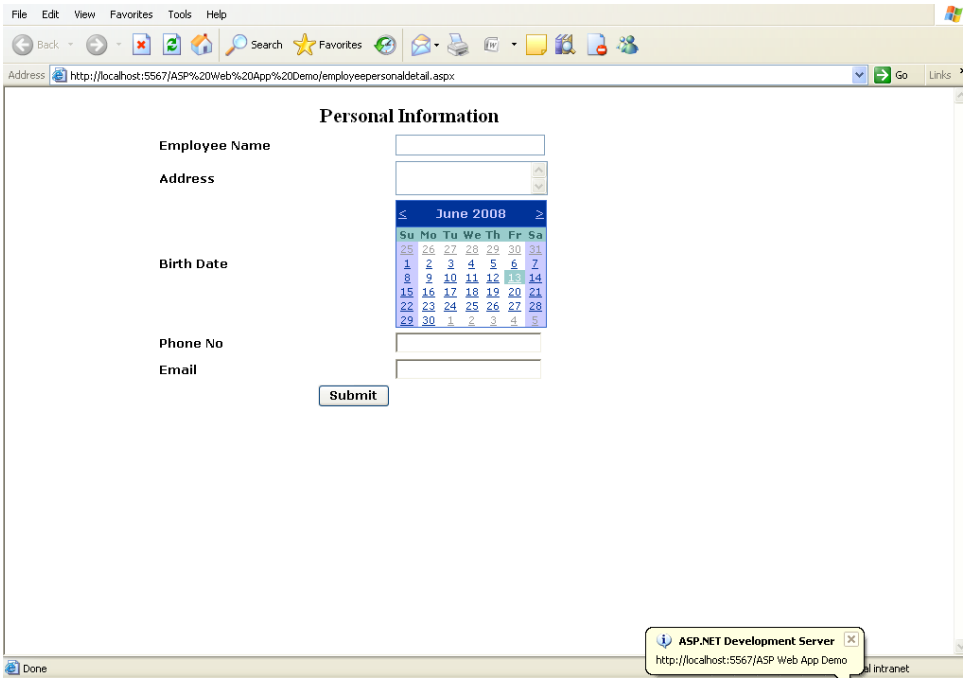


Figure 2.1-1 shows design of employeepersonaldetail.aspx page.

Step 1: Open Microsoft Visual Studio 2005. It appears as follows

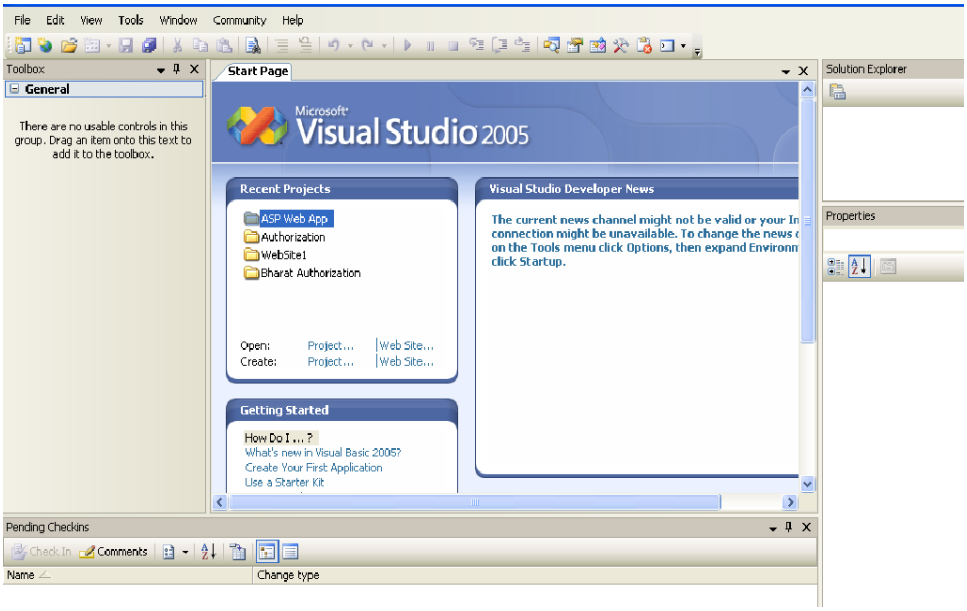


Figure 2.1-2: VS .Net IDE.

Step 2: Choose File/New Website from the menu bar.

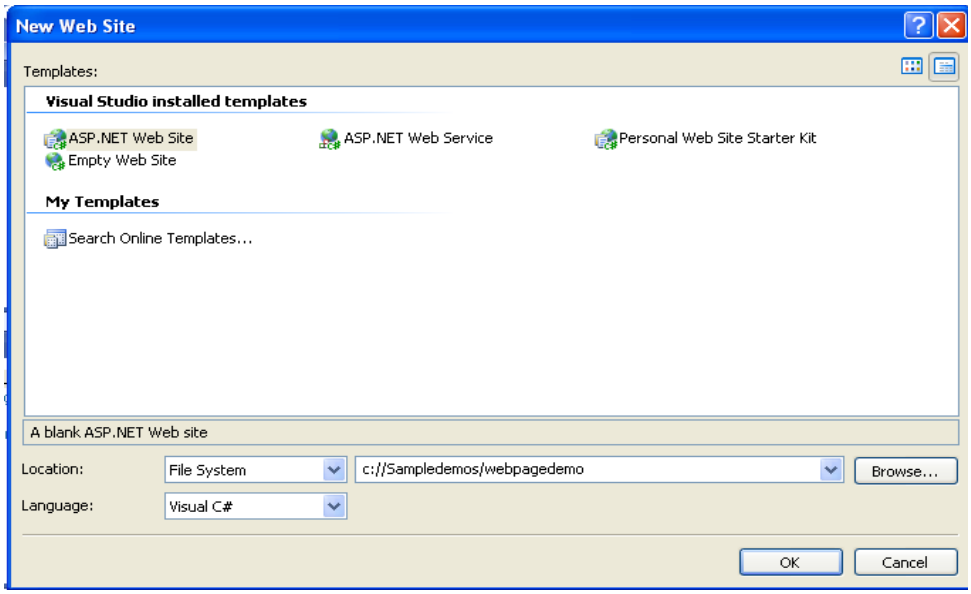


Figure 2.1-3: New Website dialog box.

Step 3: Browse to select the location to store the website being created and choose OK.

Step 4: Source code of default.aspx page is viewed as shown below. Rename Default.aspx to employeepersonaldetail.aspx in the solution explorer.

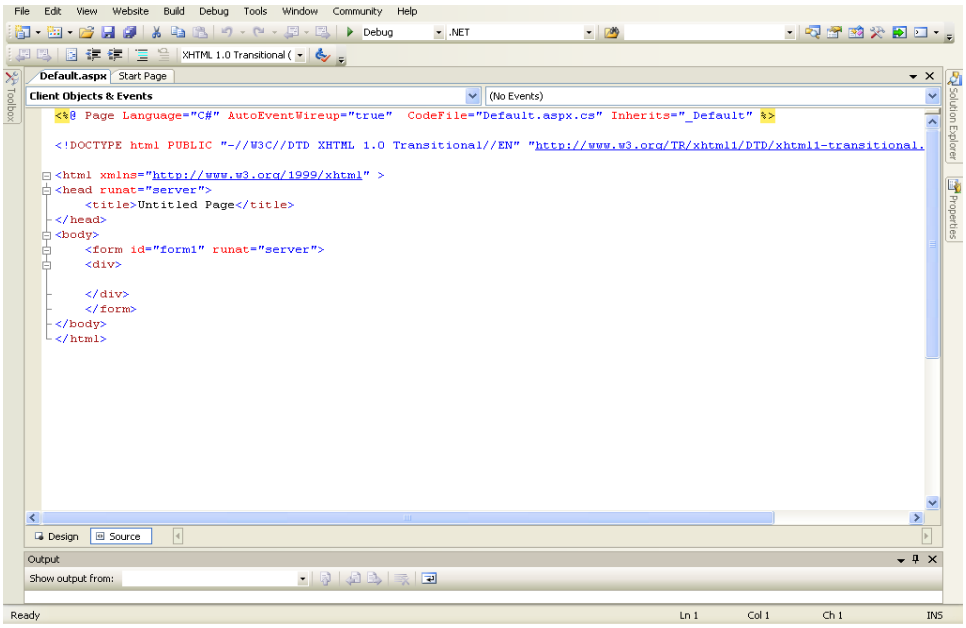


Figure 2.1-4: Source code of Default.aspx page.

Step 5: The design view provides the layout to design the web page.The source tab provides the code skeleton for the webform. Rename Default.aspx to employeepersonaldetail.aspx in the solution explorer.

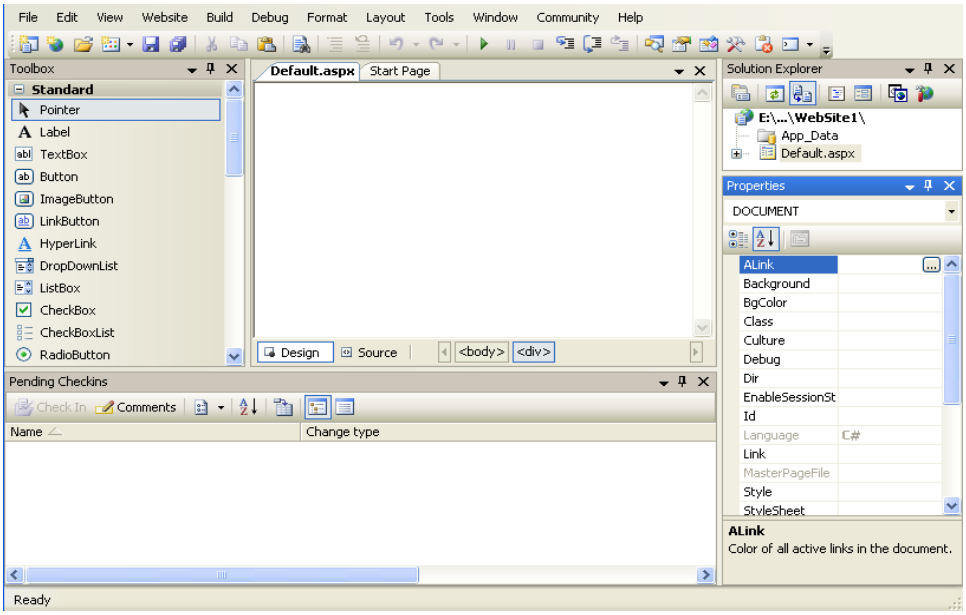


Figure 2.1-4: Design view of default.aspx.

Toolbox provides the list of controls. Additional controls can be added by choosing the "choose.items" from the context menu with a right click on the tool box

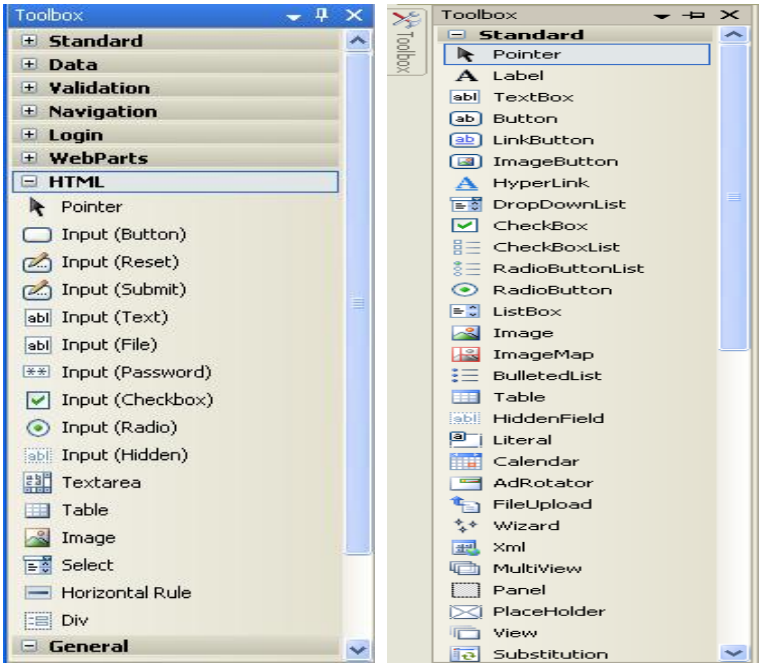


Figure 2.1-5: Toolbox html and standard server controls.

Solution Explorer, lists the resources for the website being developed, viz. WebPages,Configuration files etc.,

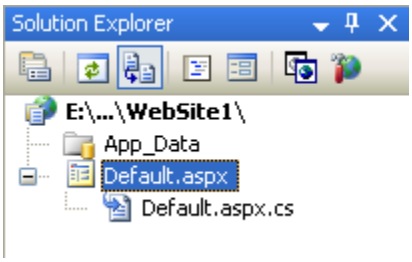


Figure 2.1-6: Solution explorer window.

Property Window for the Control in the design view lists the properties of Id, Text, Forecolor, Backcolor etc.

Figure 2.1-7: Property window for Controls.

Step 5: Insert HTML Table from toolbox in Login.aspx. Merge all cells in first row and Write Personal Information. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblName	Text:Employee Name
asp:label	lblAddress	Text:Address
asp:label	lblDob	Text:Birth Date
asp:label	lblPhno	Text:Phone No
asp:label	lblEmail	Text:Email
asp:textbox	txtName	-
asp:textbox	txtAddress	TextMode:Multiline
asp:textbox	txtPhno	-
asp:textbox	txtEmail	-
asp:calendar	calDob	-
asp:button	btnSubmit	Text:Submit

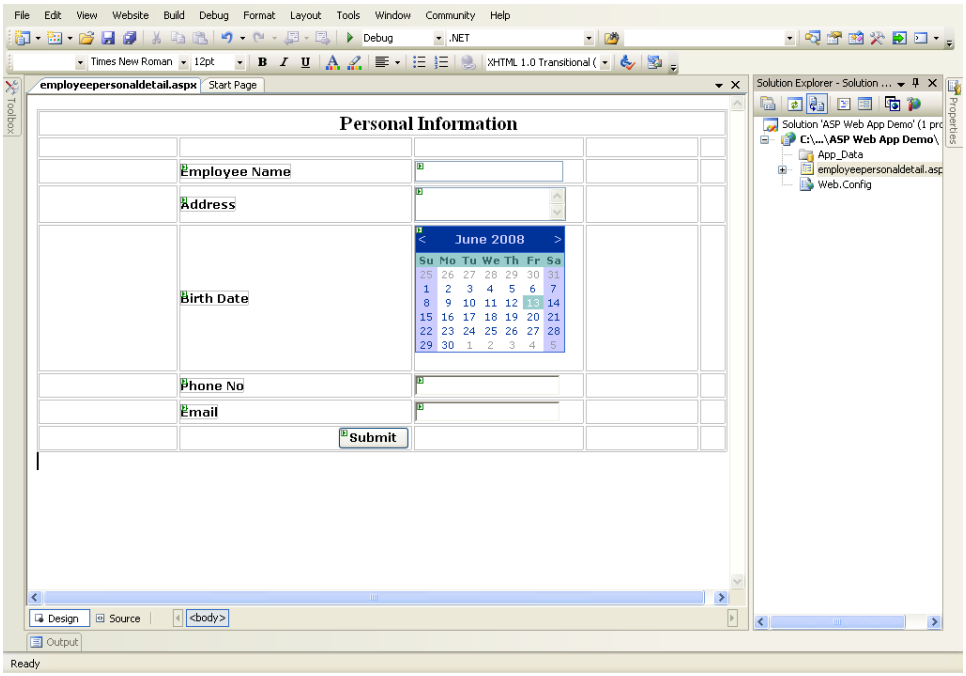


Figure 2.1-8: design of employeepersonaldetail.aspx page

F4: A useful shortcut to make the Properties pane appear is the F4 function key. Close the Properties pane. Select any control on your Web form, and hit F4.

Context :

- To build interactive web pages for an application

Practice Session :

Create a design to collect personal information of the employee. Employee should enter name, age, Email Id, Address, Phone no, Gender, hobbies. Employee should select city from drop down list, gender using radio buttons and hobbies using check boxes.

Check list :

- To design Web Forms using the controls as per the standard specification and guidelines.
- To choose appropriate controls and set their common properties such as name, text, color, font etc.

Common Errors :

- Inconsistent design elements such as fonts, labels, size, color etc.,
  Poor design with likes of scrollbars on the form

Exceptions :

- Browser and version compatibility of the controls should be verified.

Lessons Learnt :

- ☑ Understanding the basics of Web Forms.
- ☑ Understanding concept of controls toolbox.

Best Practices :

- Make appropriate design to make it user friendly

When a Web application is created, HTML or XML elements and server controls are saved in a file with .aspx extension. This file is called **Page File.**

Visual C# web forms cannot be included in visual basic.net or J# web project and vice versa.

Topic: Master Pages

Estimated Time: 60 mins.

Objectives : At the end of the activity, the participant should understand

- The utility of master pages in building websites

Presentation :

- A master page is a page template that can be applied to give many web forms a consistent appearance. For example, a master page can set out a standard structure containing the header, footer, and other elements that you expect to display in multiple web forms within a web application.

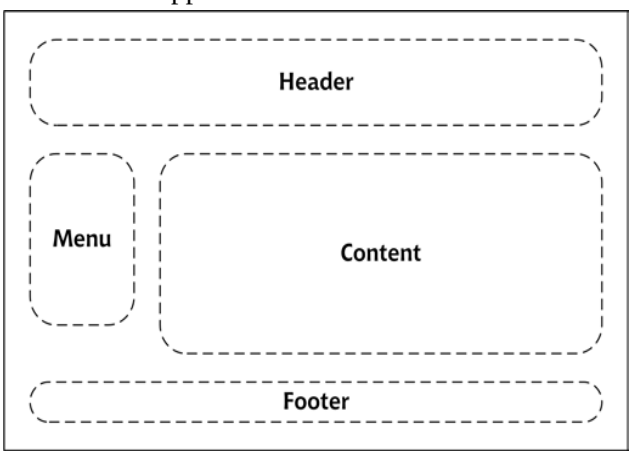


Figure 2.2-1: Simple Master Page Layout.

- Provides a template/layout and design with form elements for website with shared layout and functionality.
- When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.
- The Master Page name ends with the special extension .master
- Master page is built using HTML and controls, including the special Content Placeholder control.
- Content Placeholder is a placeholder that can be filled with content relevant to the needs of each web form that uses the master page.
- All master pages inherit from the class **System.Web.UI.MasterPage**.

Scenario :

Mr. Steve, the owner of WallMart wants to create a website which enables their customers to do online transactions. For this Steve wants the same look and feel for all the pages of the website. Mr. John, developer with Wallmart, suggests with the feature of Master page to accomplish the task.

Demonstration/Code Snippet :

Homepage inheriting master page is shown below

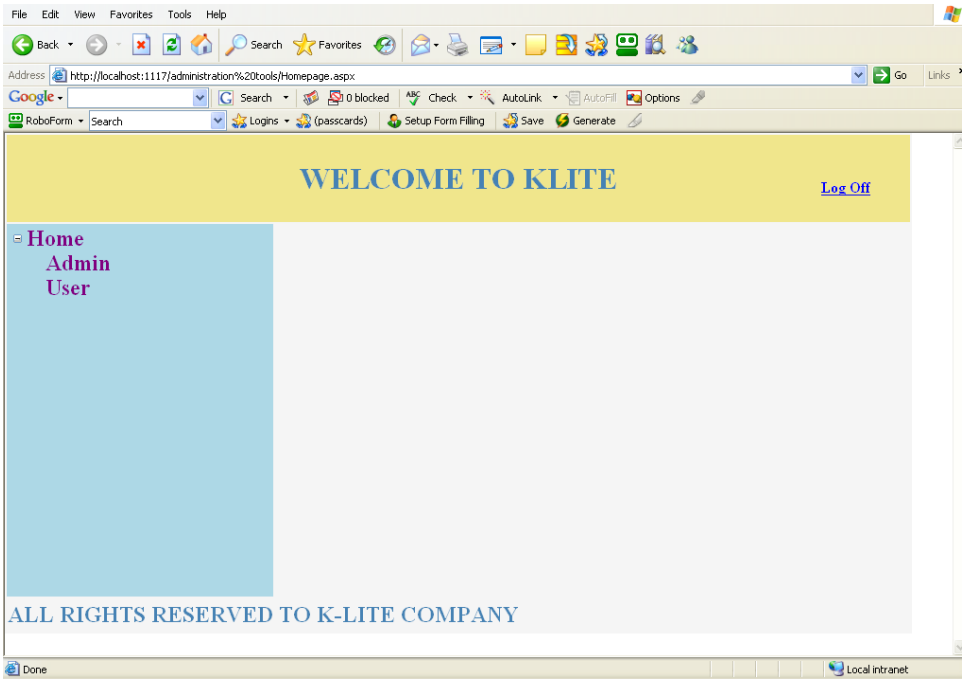


Figure 2.2-2: Homepage using master page.

Step 1: Create a new Web application project called Masterpagedemo using visual studio 2005. Rename the default.aspx to demopage.aspx

Step 2: In solution explorer Right click on Masterpage demo select Add New Item/Master page.

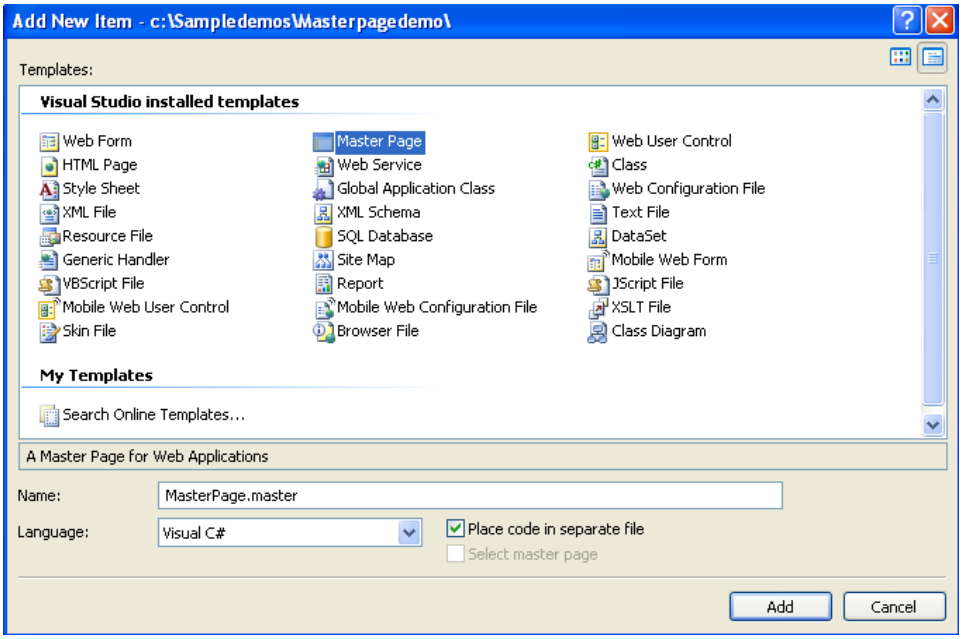


Figure 2.2-3: Creating Master Page.

Step 3: Insert a HTML Table, cut and paste the Content Place Holder in one of the cell. In the first row of the table merge all cells and write welcome to klite. Similarly in the last row of the table add the footer. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:treeview	trvNavigation	Nodes : Root/Text: Home Child/Text: Admin Child/Text: User
asp:linkbutton	lbtnLogoff	Text: LogOff

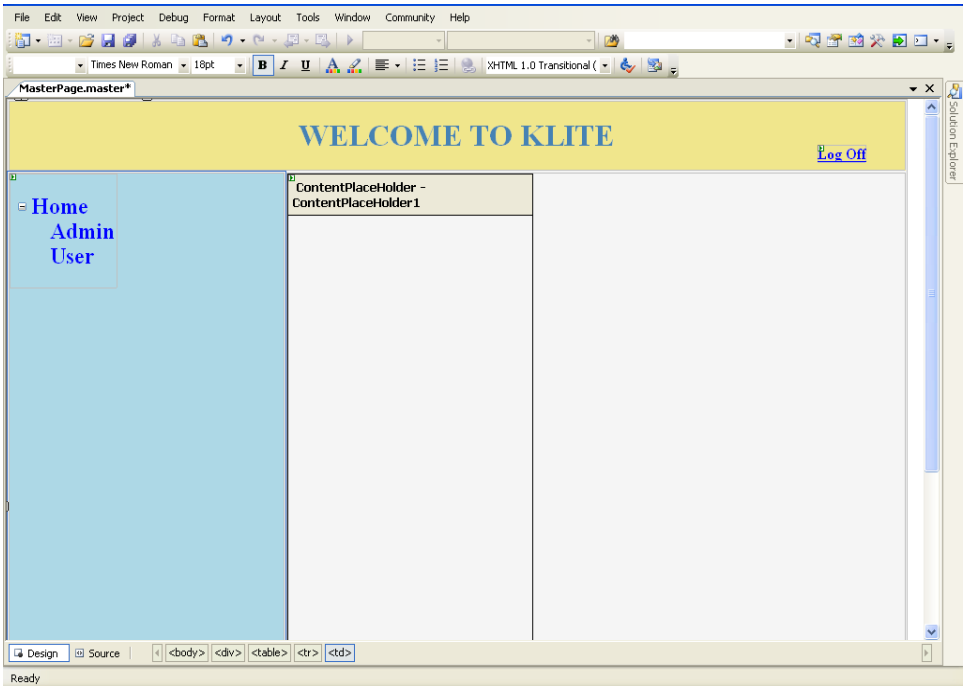


Figure 2.2-4: Master Page Design.

Step 4: Add a Web Form with "Select Master Page" option checked.

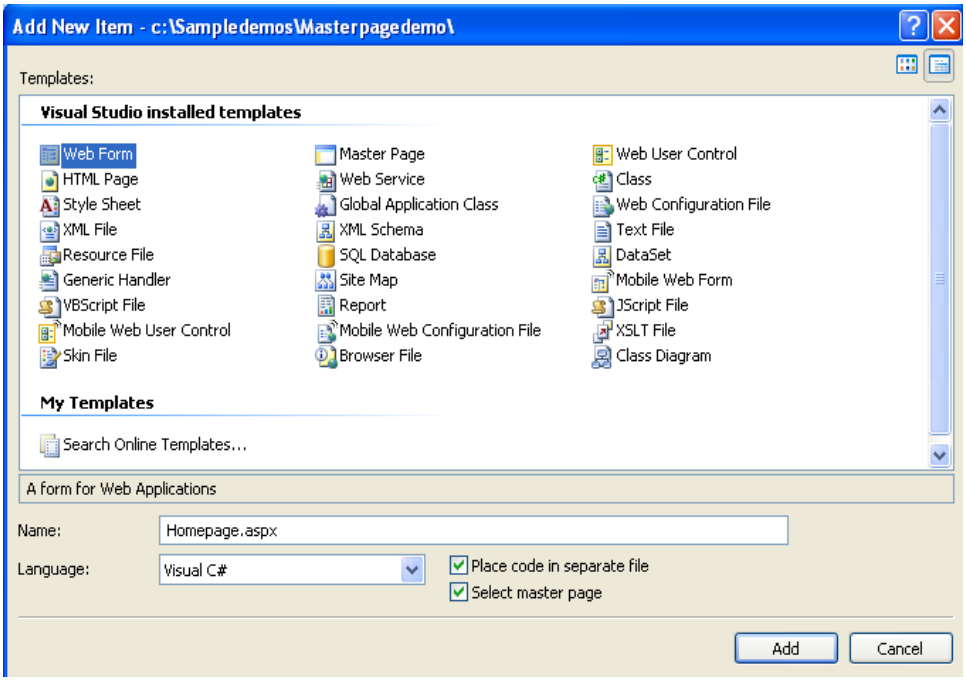


Figure 2.2-5: Create webform selecting masterpage.

Step 5: Choose the master page form the "select master page" window. This will add Master page to the New Web Page.

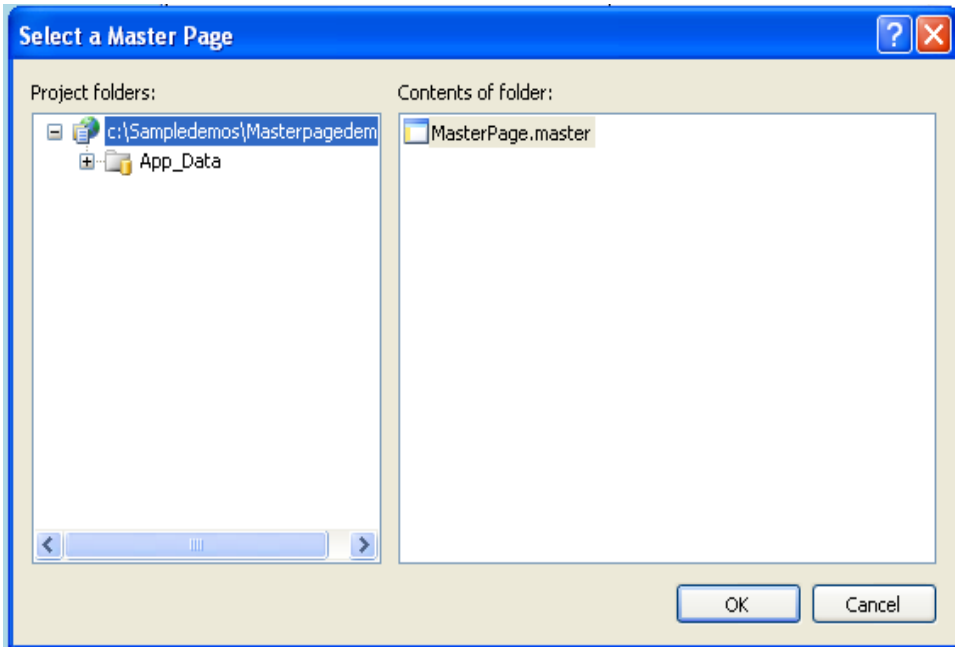


Figure 2.2-6: Select master page.

Step 6: To add master page to the existing page main.aspx copy all from div tag ( ) to its ending tag( ) including and .

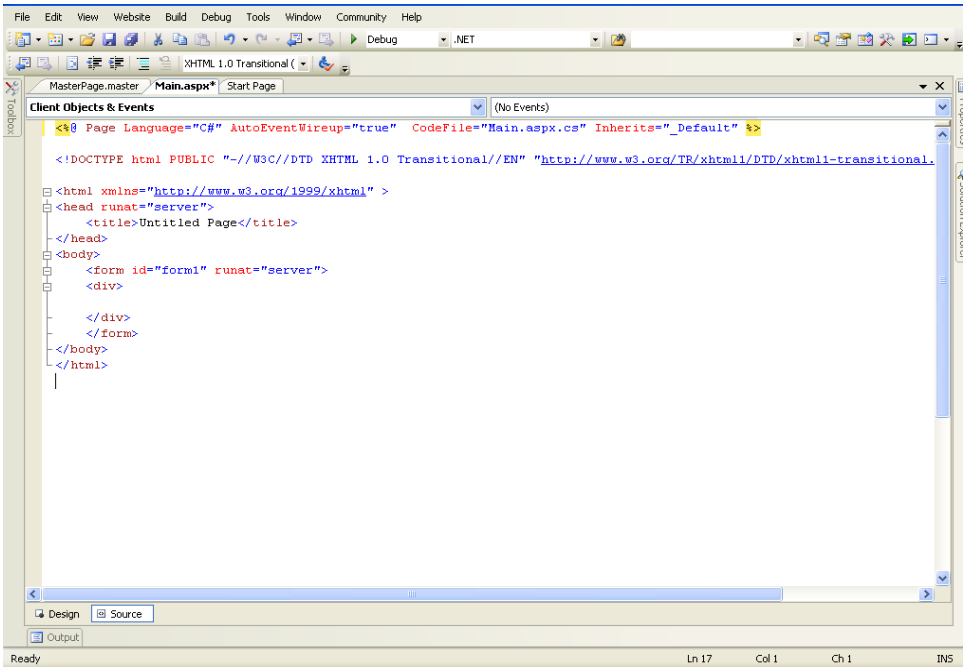


Figure 2.2-7: Mainpage.aspx source code.

Delete all except @Page directive from the form and <!doctype> . Add Masterpagefile =”masterpage.master” attribute/value pair tothe @page directive as shown below and then paste all u copied in it (to)

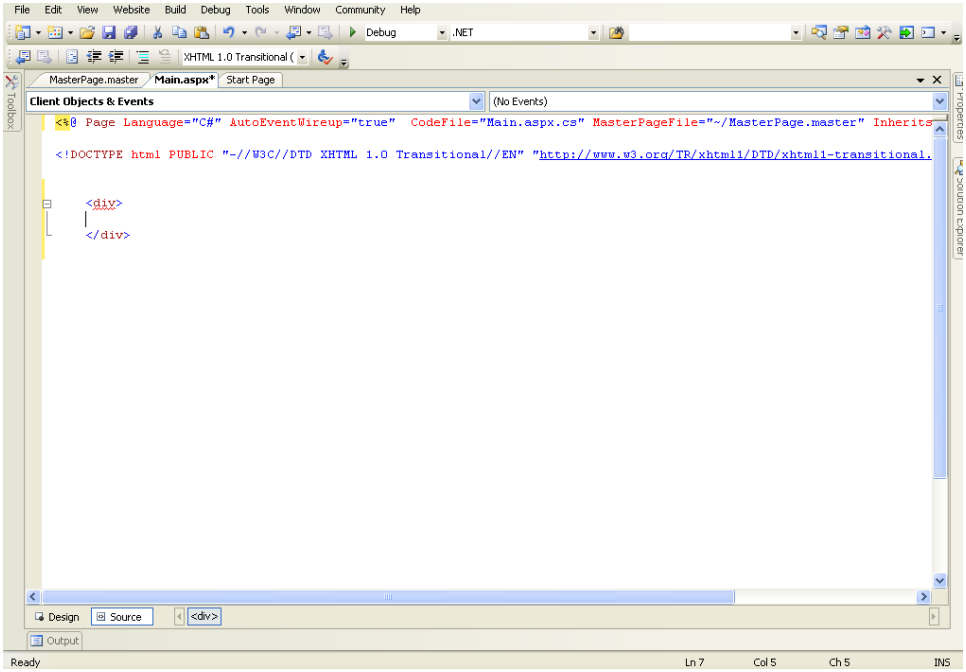


Figure 2.2-8: In Mainpage.aspx adding master page attribute and controls.

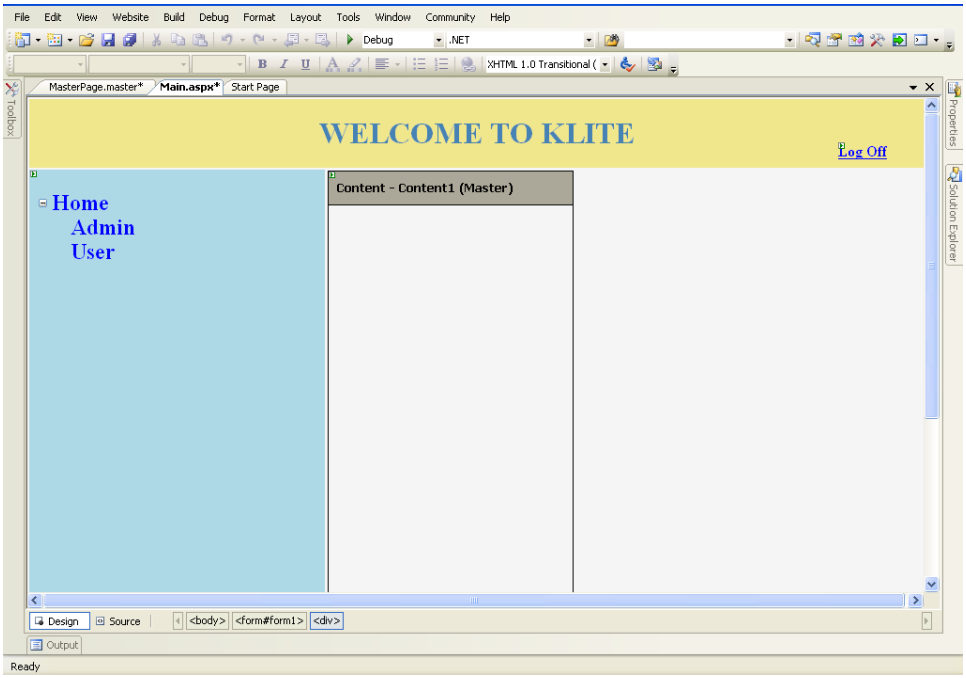


Figure 2.2-9: Mainpage.aspx inheriting master page.

Context :

- Use Master Pages for consistent look of group of pages in a Web site.
- To modify certain details in the layout or functionality of site, updating a master page has immediate effects on all the web forms that use the file.

Practice Session :

create Employee personal detail page inheriting the master page .Master page consist of header, footer and Menu bar consisting menu toolbox.

Check List :

- Importance of including Master Pages in a Web Application.
- Using @Master and @Page directives.

Common Errors :

- Non inclusion of **@Master** directive.
- Non inclusion of **MasterPageFile** attributes.

Exceptions :

- If we have 2 different master pages then we have a problem, because the Virtual Path attribute supports only a single master page. If we assign a MasterPageFile that does not match the Master Type, the runtime will throw an exception:

  Unable to cast object of type 'ASP.master2_master' to type 'ASP.master1_master'.

- A page's MasterPageFile property sets the master page for the content page to use. If we try to set this property from the Load event, we will create an exception:

  The 'MasterPageFile' property can only be set in or before the 'Page_PreInit' event.

Lessons Learnt :

- ☑ Inheriting MasterPage class.
- ☑ Importance of @Master and @Page directives.
- ☑ Understanding of Master.

Best Practices :

- Always provide header and footer used in all the pages in master page only.

Master page can define some default content for display inside the Content Placeholder on pages whose web forms don't provide a Content element for that placeholder.

Master Page can have more than one content Place holder.

If all the pages use the same Master Page then it's better to set it once in Web.config file as shown below
<pages masterpagefile="~/master.master"/>

Objectives : At the end of the activity, the participant should understand

- Html Server Controls
- Web Server Controls
- Validation Controls
- Custom Controls

Presentation :

- **Asp.net** server controls are different from usual windows controls because they work within the asp.net framework. These controls supports raising of events on user interaction
- In Asp.net each server control is the instance of a particular base class which resides in **System.Web.UI** namespace.
- There are 4 types of server controls
  - **HTML server controls:** 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:** 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:** These controls are used to validate. Validation is a set of rules that you apply to the data you collect.
  - **Custom controls:** You can define your own controls with few modifications, almost any Web Forms page can be reused in another page as a server control (note that a user control is of type **System.Web.UI.UserControl**, which inherits directly from **System.Web.UI. Control**). As a matter of convention, the ***.ascx*** extension is used to indicate such controls. This ensures that the user control's file cannot be executed as a standalone Web Forms page.

Html Server Control :

Scenario :

Mr. Brown, the head of oxford university wants to collect information of the students . To get better performance by reducing the server interaction with the application developer Mr. Wili suggest Mr.Brown with the features of HTML Controls in the website.

Demonstration/Code Snippet :

HtmlControl.aspx page design is shown below.

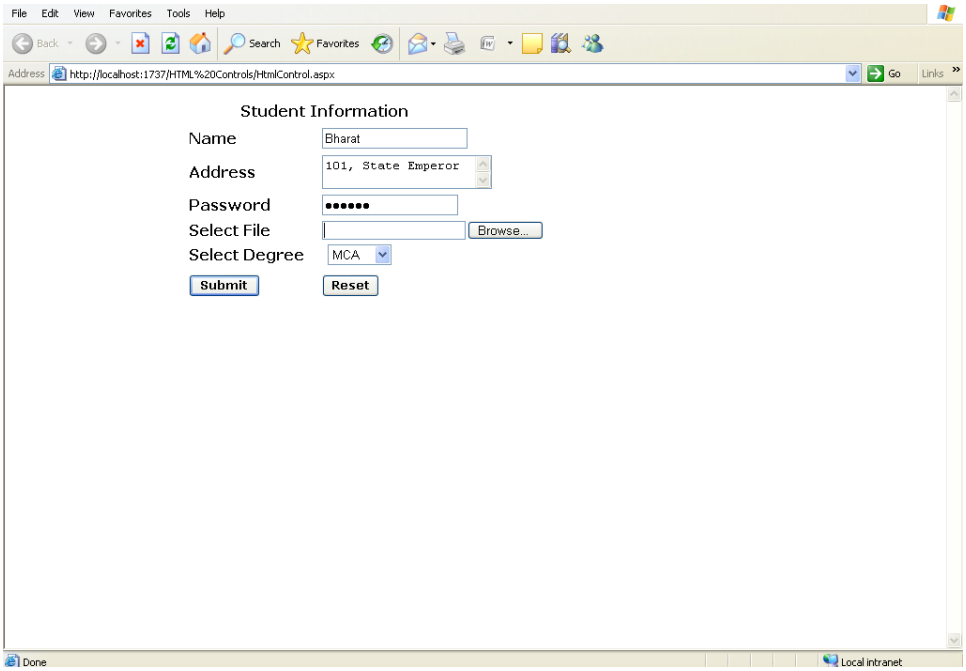


Figure 2.3-1:HtmlControl page design.

Step 1: Create a new Web application project called HtmlControlApplication using visual studio 2005. Rename Default.aspx to HtmlControl.aspx.

Step 2: Insert HTML Table from toolbox in HtmlControl.aspx. Drag and drop the HTML controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
input#	txtName	-
input#	txtPwd	-
textarea#	txtaAddress	-
input#file	txtFile	-
input#reset	btnReset	Value: Reset
input#Submit	btnSubmit	Value: Submit
asp:label	lblName	Text:Name
asp:label	lblAddress	Text:Address
asp:label	lblPwd	Text:Password
asp:label	lblFile	Text:Select File

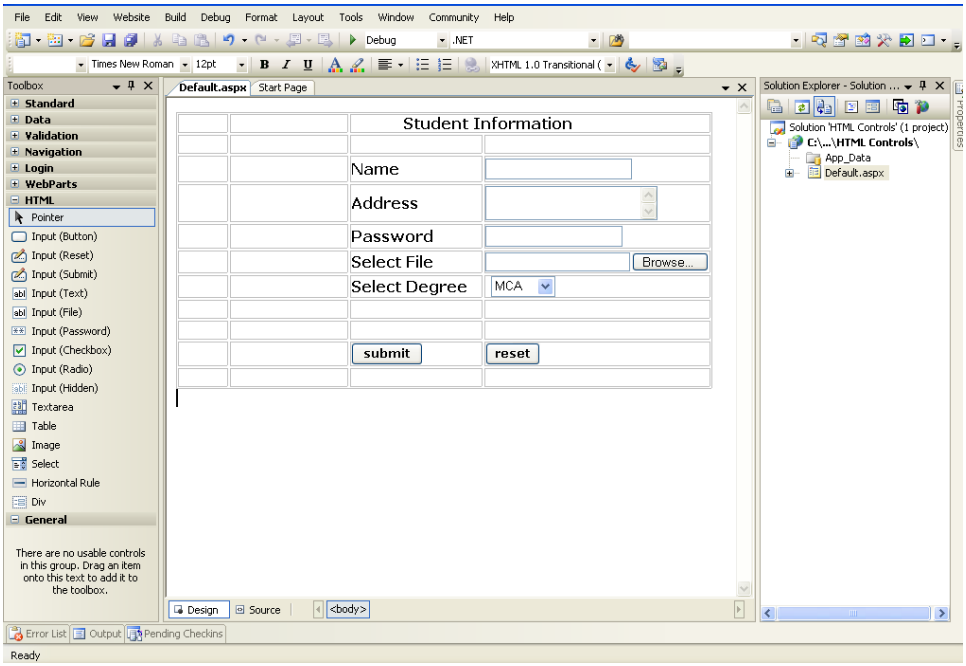


Figure 2.3-2: HtmlControl.aspx design.

User can also convert the HTML Controls to Server Controls by adding <input type=button runat=server> code snippet in the source file of the website or right click on HTML control and selecting Run as Server Control.

Web Server Control :

Scenario :

Mr. Brown, the head of the Oxford University wants to collect students information.Application needs to have an interaction between Server and Client. Mr. Wili, developer with the Oxford university, suggests the use of server controls in the application.

Demonstration/Code Snippet :

Page using web server control shown below.

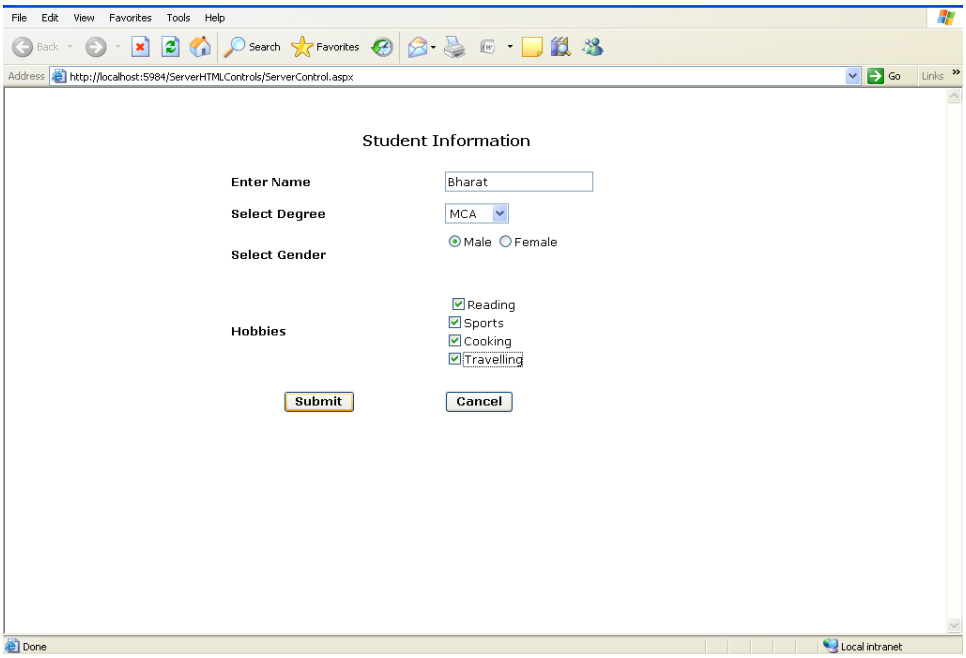


Figure 2.3-3:ServerControl.aspx.

Step 1: Create a new Web application project called ServerControlApplication using visual studio 2005. Rename Default.aspx to ServerControl.aspx.

Step 2: Insert HTML Table from toolbox in ServerControl.aspx. Drag and drop the Server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblName	Text:Enter Name
asp:label	lblDegree	Text:Select Degree
asp:label	lblGender	Text:Select Gender
asp:label	lblHobbies	Text:Hobbies
asp:label	lblStudentinfo	Text:Student information
asp:radiobutton	rbtnMale	Text:Male ValidationGroup:Gender
asp:radiobutton	rbtnFemale	Text:Female ValidationGroup:Gender
asp:checkbox	chkSports	Text:Sports
asp:checkbox	chkTravelling	Text:Travelling
asp:dropdownlist	ddlDegree	Items:Listitems Text:BCA
asp:button	btnSubmit	Text:Submit
asp:button	btnCancel	Text:Cancel

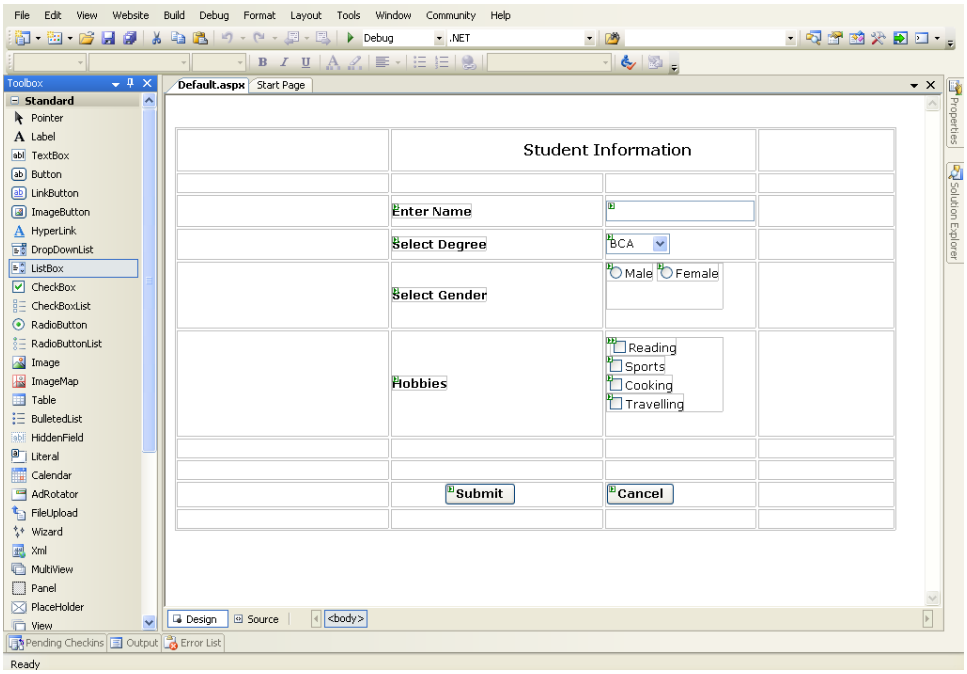


Figure 2.3-4: ServerControl.aspx design.

Context :

- Use html control where server side processing is not required.
- HTML controls provide attributes and events that can be used in scripts that run on the client.
- Use server control where interaction with server is required.

Practice Session :

Create a design for collecting Employee Information like name, address, age, phone no, gender and department. Use drop down list for department and radio button for gender.
Use html and server control both wherever necessary.

Check List :

- Importance of different Server Controls.

Common Errors :

- Using html controls try to run server control without making them run at server control.

Exceptions :

- No Specific exception.

Lessons Learnt :

- ☑ Importance of Html controls.
- ☑ Importance of server Controls.

Best Practices :

- Use server controls as server controls are superset of HTML controls.

HTML controls integrate well with data binding and the ASP.NET state maintenance, and they also provide full support for postback events and client scripting. For example, for a button that gets clicked, you can have some JavaScript code running on the client responding to the **onclick** event, as well as some codes that handle the event on the server if the page posts back as the result of that event.

HTML controls are defined in the **System.Web.UI.HtmlControls** namespace.

An instance of the **HtmlHead** control is automatically created if the page contains a **<head>** tag marked with the attribute **runat=server**. Note that this setting is the default when a new page is added to a Visual Studio .NET 2005 Web project, as shown in the following snippet:

```
<head runat="server">
   <title>Untitled Page</title>
</head>
```

Html Control

```
<a runat=server onclick="Run()" onserverchange="Server()">Click</a>
```

The **onclick** attribute defines the client-side event handler written using JavaScript; the **onserverclick** attribute refers to the server-side code that will run after the page posts back. Of course, if both event handlers are specified, the client-side handler executes first before the post back occurs.

Server control

```
  <script type="text/javascript" language="javascript">
  function demo()
  {
  }</script>
```

JavaScript can be called by writing OnClientClick="demo()" in the control tag in source file

Topic: Data Bound Controls

Estimated Time: 60 mins.

Objectives : At the end of the activity, the participant should understand

- Data Bound Controls.

Presentation :

- Data-Bound controls can be bound to a data source control to display and modify data in Web application. Types of Data Bound Controls are as follows.

GridView:

- The DataGrid's purpose is to display tabular data.
- Automatic Paging and Sorting with no coding needed.
- Complete command button functionality (such as the Select, Edit and Delete buttons) without any coding.
- Multiple records can be visible at the same and user can navigate through Paging.
- User can set the following properties of Grid View.
  - AllowPaging
  - AllowSorting
  - AutoGenerateDeleteButton
  - AutoGenerateSelectButton
  - AutoGenereateEditButton

DetailsView:

- Automatic Paging with no coding needed.
- Complete command button functionality (such as the Insert, Edit and Delete buttons) without coding.
- Only single record is visible at a time and user can navigate through Paging.
- User can set the following properties of Details View.
  - AllowPaging
  - AutoGenerateDeleteButton
  - AutoGenerateInsertButton
  - AutoGenereateEditButton

FormView:

- FormView is a data-bound user interface control that renders a single record at a time from its associated data source, optionally providing paging buttons to navigate between records.

- It is similar to the DetailsView control, except that it requires the user to define the rendering of each item using templates, instead of using data control fields.

DataList:

- DataList control is used to display repeated list of items that are bound to control.
- DataList control displays data from a data source in the form of list.

Data Source Control:

- Data Source Control acts as coordinator between underlying different physical data sources at data access layer and data bound controls like GridView, DataGrid at presentation layer.
- Data Source Control has features like Sorting, Paging, Editing, Updating, Filtering, Deleting, and Inserting data directly in data source.
- Following are the types of Data Source Controls
  - SqlDataSource
  - ObjectDataSource
  - AccessDataSource
  - SiteMapDataSource
  - XmlDataSource

Scenario:

Mr. Ponting, owner of Spensor Mall wants customer details to be displayed in a control.It should be displayed in tabular format with edit ,delete and select links allowing to perform respectivity activity. So Mr. Gilly, developer of website uses the concept of the GridView.

Demonstration/Code Snippet:

GridViewDemo.aspx page.

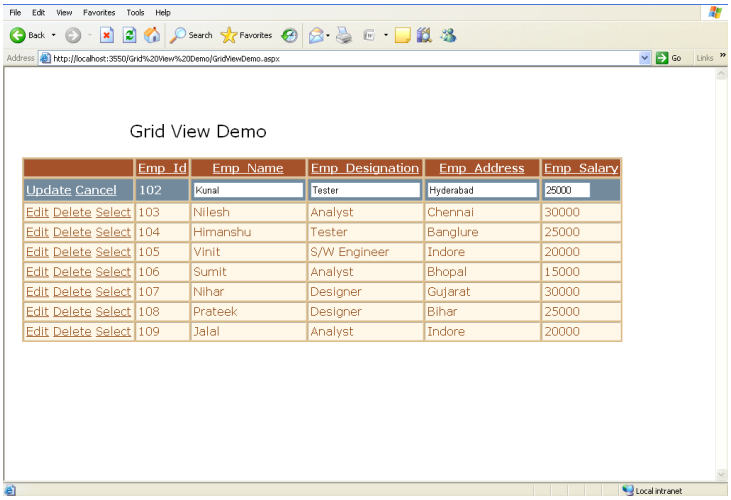


Figure 2.4-1:

GridViewDemo.aspx page showing details in grid view.

Step 1: : Create a new Web application project called DataboundWebApplication using visual studio 2005. Rename Default.aspx to GridViewDemo.aspx.

Step 2: Insert HTML Table from toolbox in Login.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:gridview	gdvEmpdetails	-
asp:label	lblHeading	Text:Grid View Demo

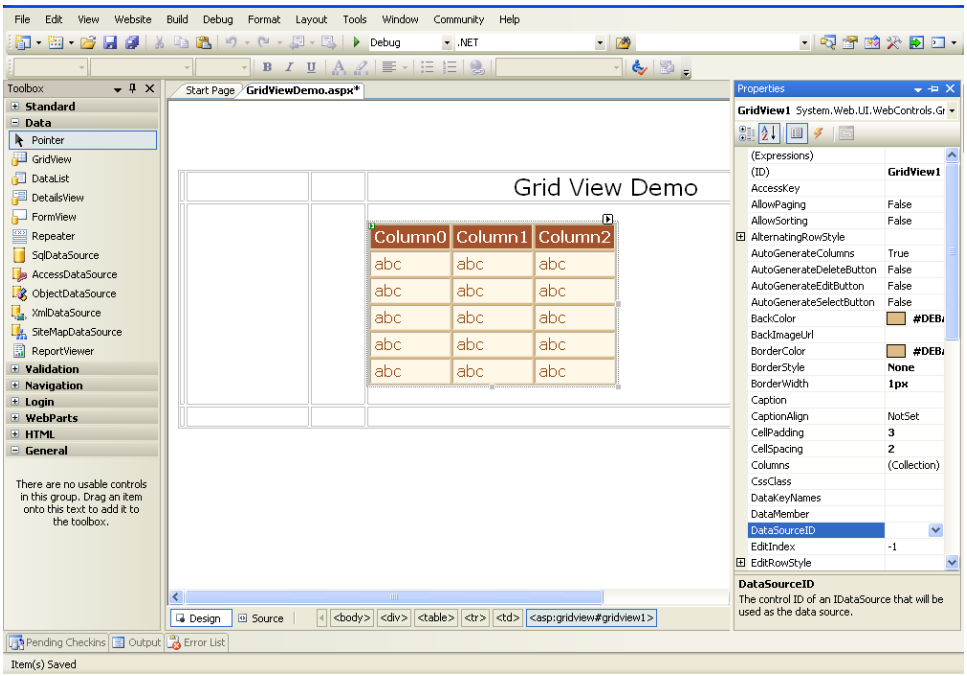


Figure 2.4-2:GridViewDemo.aspx design.

Step 3: To fetch data from the table select GridView and click the small arrow (smart tag) which is on the top of the GridView. Select New Data Source from Choose Data Source.

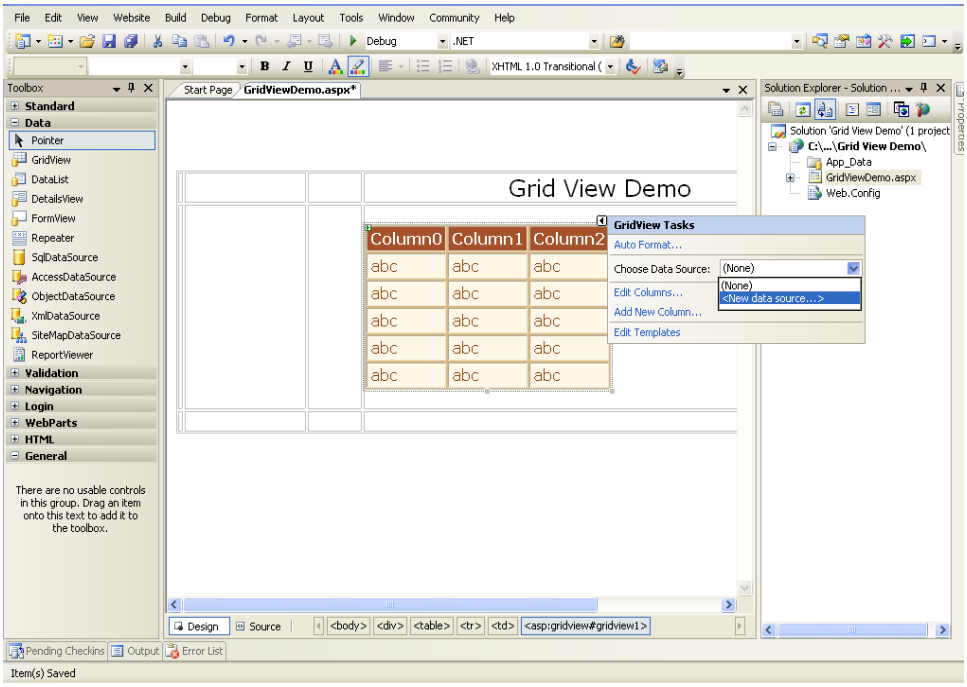


Figure 2.4-3: Selecting New data Source.

Step 4: Data Source Configuration Wizard" will be opened. Select the Database and specify the ID for the Data Source.

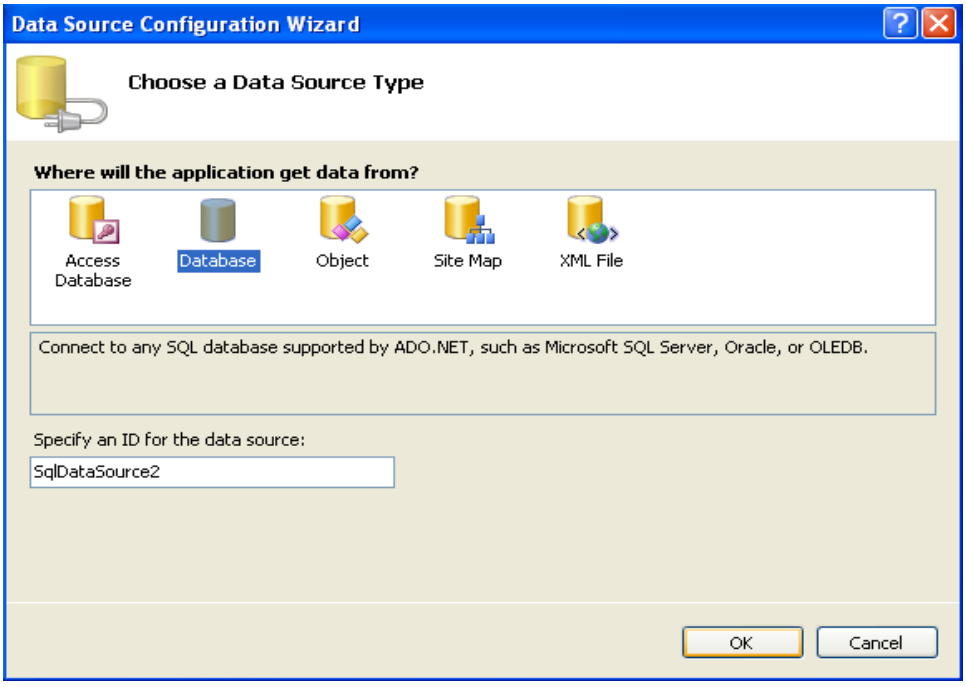


Figure 2.4-4: Selecting Database and Specifying its ID.

Step 5: Configuration Data Source" dialog box will open where user can choose a New Connection or use the existing one.

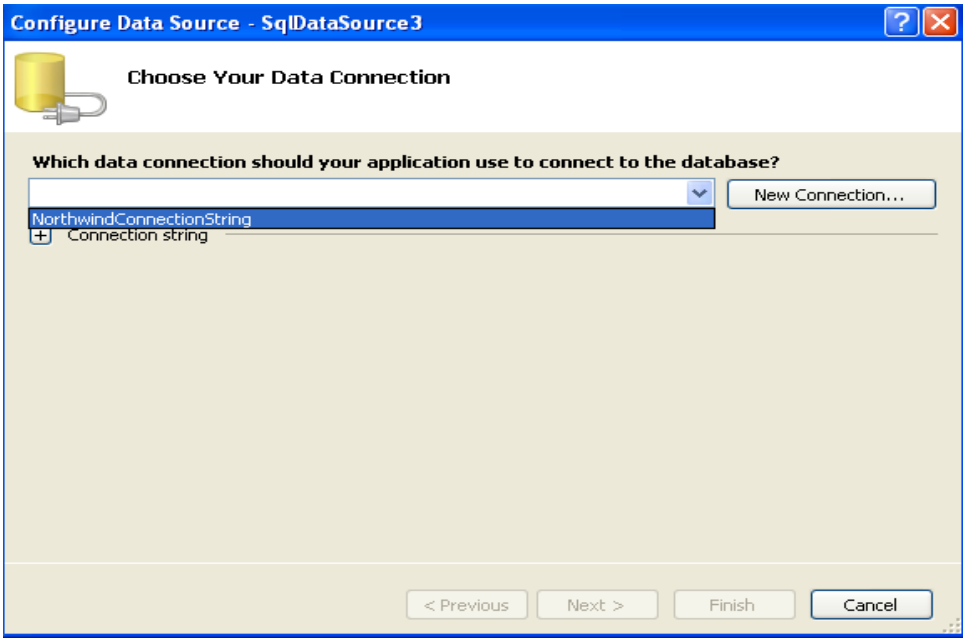


Figure 2.4-5: Configuration Data Source.

Step 6: If user selects New Connection, then "Add Connection" dialog box will be opened. Select the Server Name. If user selects the "Use Windows Authentication" then same User Name and Password will be applied which user has entered at the time of login into the system or else user has to enter User Name and Password for the server. Select the Database Name and test the connection. Data Connection string will be created.

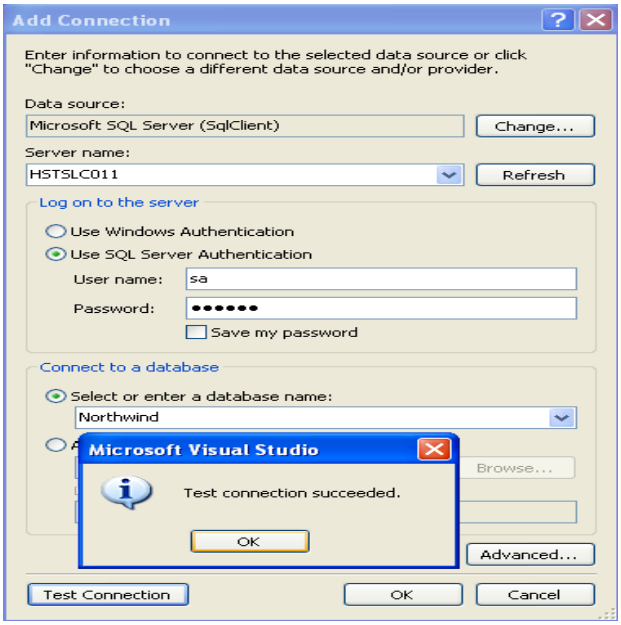


Figure 2.4-6: Creating connection with database and testing it.

Step 7: Save the Connection String to the Application Configuration File.

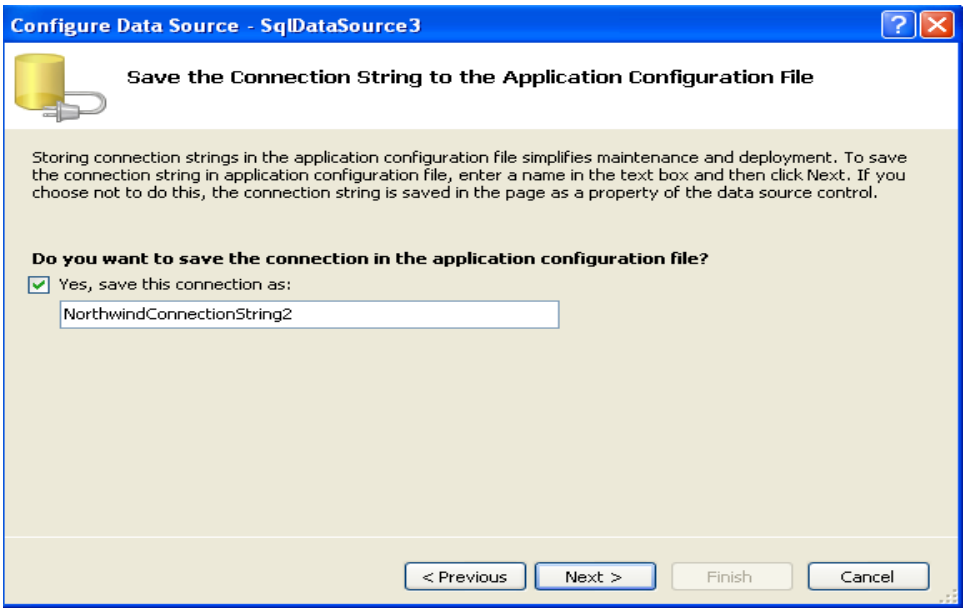


Figure 2.4-7: Saving Connection String.

Step 8: Click on next button and from the Configure Data Source dialog box user can select the table name and columns or also specify the Custom SQL statement or Stored Procedure.

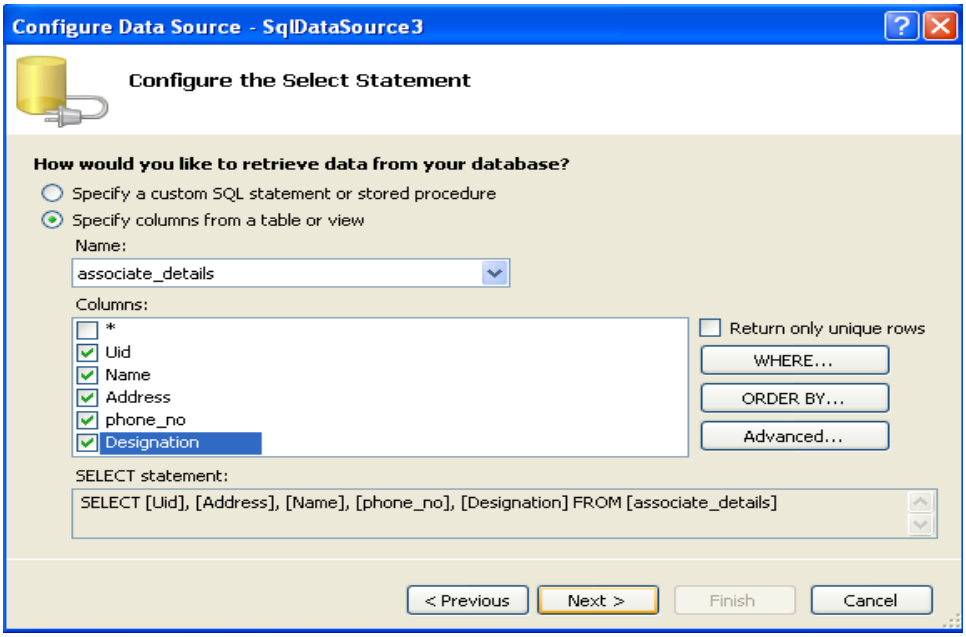


Figure 2.4-8: Selecting Columns to be displayed in Grid View.

Step 9: As GridView provides the facility to Select, Edit, Delete, Paging and Sorting, for this user has to click "Advanced…" button. "Advanced SQL Generation Option" dialog box will be opened.
Check the Generate Insert, Update and Delete statements.

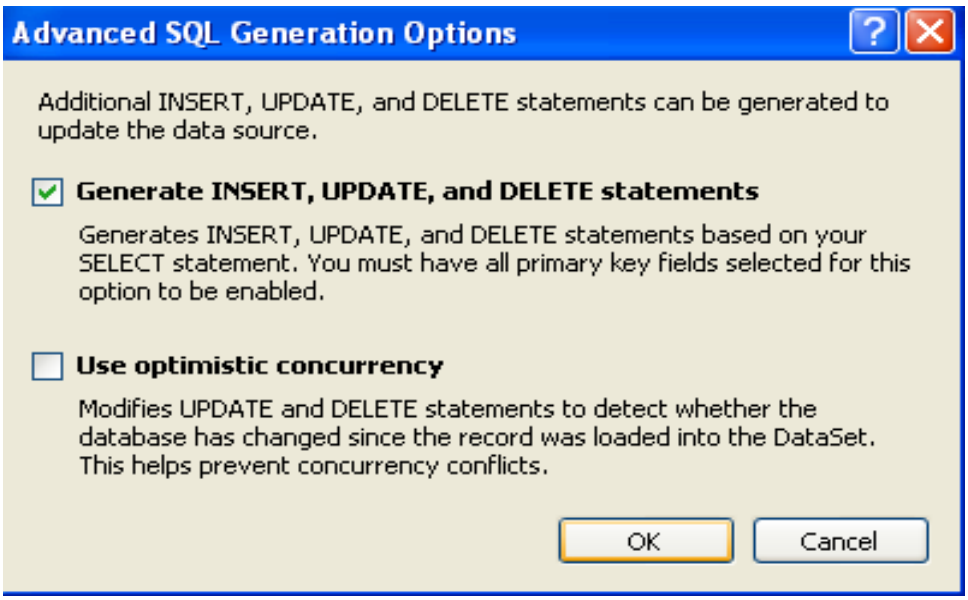


Figure 2.4-9: Enabling Insert, Update and Delete command.

Step 10: User can also test the query which user has fired and see the results.

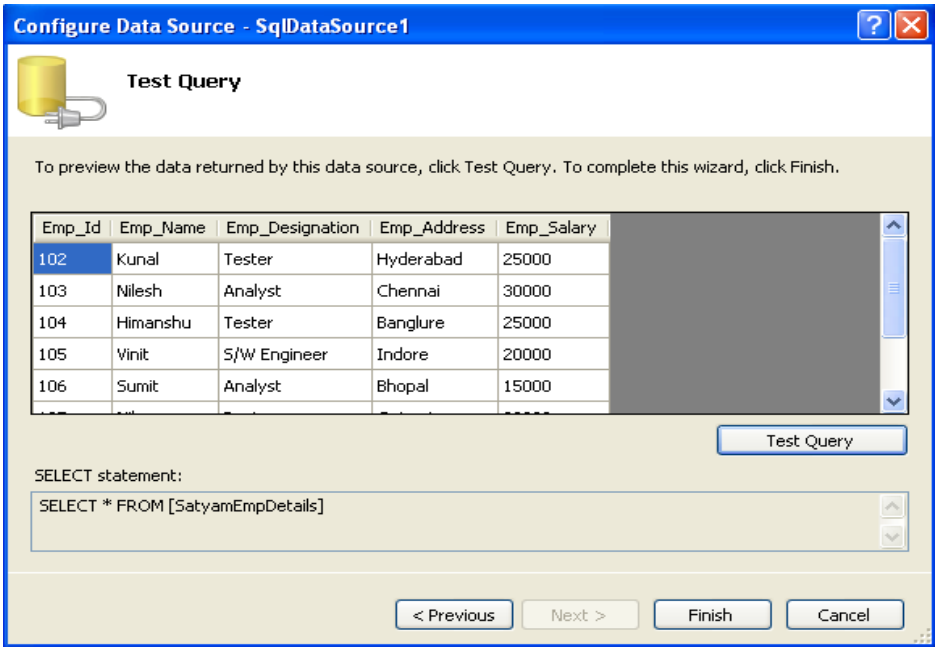


Figure 2.4-10: Testing the Query Specified after selecting columns.

Step 11: Actual columns of the table will be added to the GridView. Click on smart tag and check the Paging, Sorting, Editing, Deleting and Selection option.

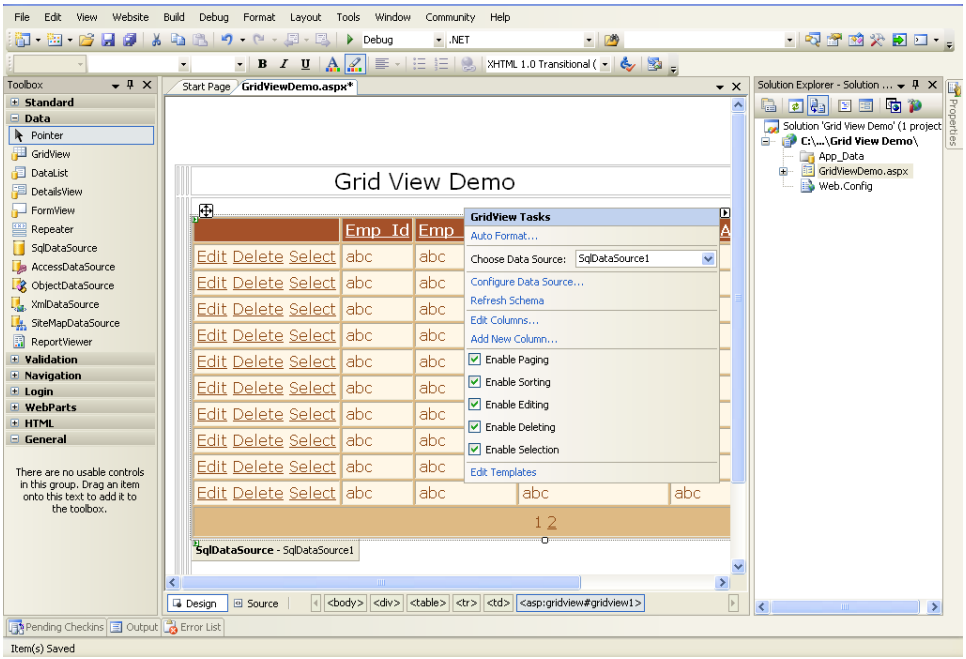


Figure 2.4-11: Enabling different properties of grid view.

Tables used: SatyamEmpDetails
Database: Northwind

Emp_Id	Emp_Name	Emp_Designation	Emp_Address	Emp_Salary
102	Kunal	Tester	Hyderabad	25000
103	Nilesh	Analyst	Chennai	30000
104	Himanshu	Tester	Banglure	25000
105	Vinit	S/W Engineer	Indore	20000
106	Sumit	Analyst	Bhopal	15000

Figure 2.4-12: SatyamEmpDetails table.

Context :

- Use Data Bound Controls where there is no need for coding.
- Use Data Bound Controls where large no of data to be populated.

Practice Session/s :

Create a webpage and populate all the details of Student from StudentMaster table from the Northwind database using form view & details view.

Check List :

- Importance of using Data Bound Controls in a Web Application.
- Features of Data Bound Controls.

Common Error/s :

- Connecting the database via SqlDataSource or via coding will lead to an error if not specified properly.
- Enabling insert, delete and update command in Advanced SQL generation option without specifying primary key in the database table.

Exception/s :

- Improper Connection String and Data Source while configuring the Data Bound control may lead to an exception.

Lesson/s Learnt :

- ☑ Understanding the Data Bound Controls.
- ☑ Establishing connection of Data Bound Controls with database.

Best Practice/s :

- Use smart tags where ever you want to minimize the code and to configure Data Bound controls with relatively ease.

Coding of Grid view is as follow

```
Using System.Data.Sqlclient;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox

protected void Page_Load(object sender, EventArgs e)
{
 if (!IsPostBack) {
String
Connectionstr="server=hstslc011;database=Northwind;uid=sa;pwd=satya
m"
SqlConnection con= new SqlConnection(Connectionstr);
String Query="select * from satyamempdetails";
 try {
        con.Open ();
        SqlCommand command = new SqlCommand(Query,con);
        SqlDataReader reader = command.ExecuteReader ();
         gdvEmpDetails.DataSource = reader;//Providing datasource
                    gdvEmpDetails.DataBind ();//Binding data
```

```
        }
        catch (Exception ex)
        {
        MessageBox(ex);
        }
        finally
        {
        con.Close();
        }

    }
```

ISPOSTBACK property of the page allows to check whether the page is being requested for the first time. A false value of the ISPOSTBACK property indicates that the page is displayed for the first time. A true value indicates that the page is run as result of a round trip.

Topic: Themes and Skins

Estimated Time:40 mins.

Objectives : At the end of the activity, the participant should understand

- Skins
- Cascading Style Sheets (CSS)

Presentation :

- Themes are set of elements like skins, Cascading Style Sheet, images and other resources like script files.
- Themes are defined in special directory called App_Themes in application hierarchy.
- ASP.NET themes can be applied at the application, page, or server control level.
- There are 2 types of Themes:
  - **Skin Files: S**kin file contains style definitions for the individual server controls. Skinned version do not include ID attribute for control.
  - **Cascading Style Sheet:** CSS usually deals with HTML code. CSS cannot be applied to certain ASP.NET specific server controls which are not present in HTML.

Scenario:

Manager Ryan of Wilson Company wants to have uniqueness in company's website by having same background color, size of controls like textbox, buttons etc. So Ryan suggests developers his view and ask them to implement it.

Demonstration/Code Snippet:

1) **Using Skin File**

Before Skin File is applied Skin.aspx page looks like as shown below.

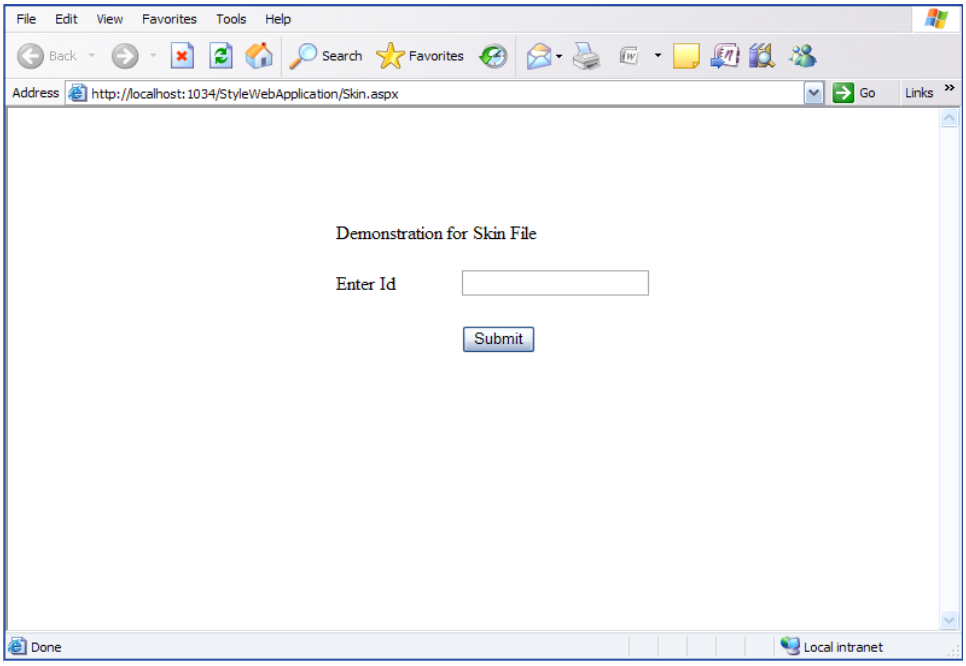


Figure 2.5-1: Skin.aspx page without applying Skin style.

After Skin File is applied Skin.aspx page looks like as shown below.

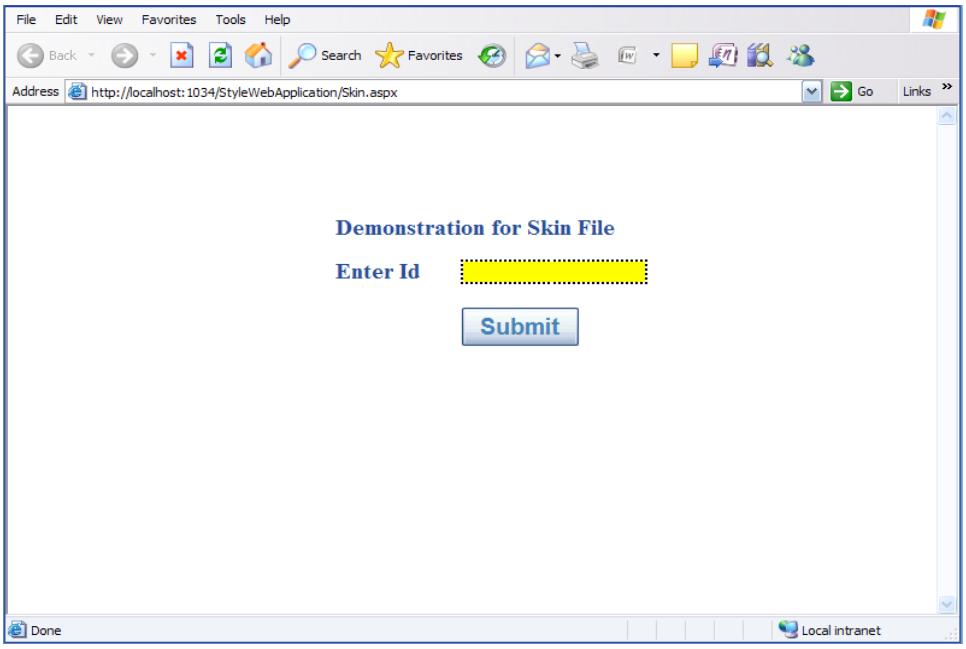


Figure 2.5-2: Skin.aspx after applying skin file.

Step 1: Create a new Web application project called StyleWebApplication using visual studio 2005. Rename Default.aspx to Skin.aspx.

Step 2: Insert HTML Table from toolbox in Skin.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Demonstration for skin File
asp:label	lblUserid	Text:Enter Id
asp:textbox	txtUserid	-
asp:button	btnSubmit	Text:Submit

Step 3: In the Solution Explore, right click on the name of your WebApplication (StyleWebApplication) and selecting asp.net folder add theme folder (here name given is skintheme). Right click on Skintheme folder and selecting add new items add skin file. To include Skin file in the website we need to write in source code **theme="skintheme".**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Theme="skintheme" Inherits="_Default" %>
```

Step 4: In Skin file create styles as shown below.

Style for the label specifying its position, font size and color of the font.

```
<asp:Label runat="server" Style="position: static;font-weight: bold;font-size:20px; color: #284e98;" ></asp:Label>
```

Style for the button specifying its position, font size, font type and color of the font.

```
<asp:Button runat="server" Font-Bold="True" ForeColor="SteelBlue" Font-size="20px" Style="position: static"/>
```

Style for the textbox specifying its Background color and Border Style.

```
<asp:TextBox BackColor="Yellow" BorderStyle="dotted" Runat="Server" />
```

Step 5: Run the application and observe the output.

2) Using Cascading Style Sheet (CSS)

Before CSS file is applied CSS.aspx page looks like as shown below.

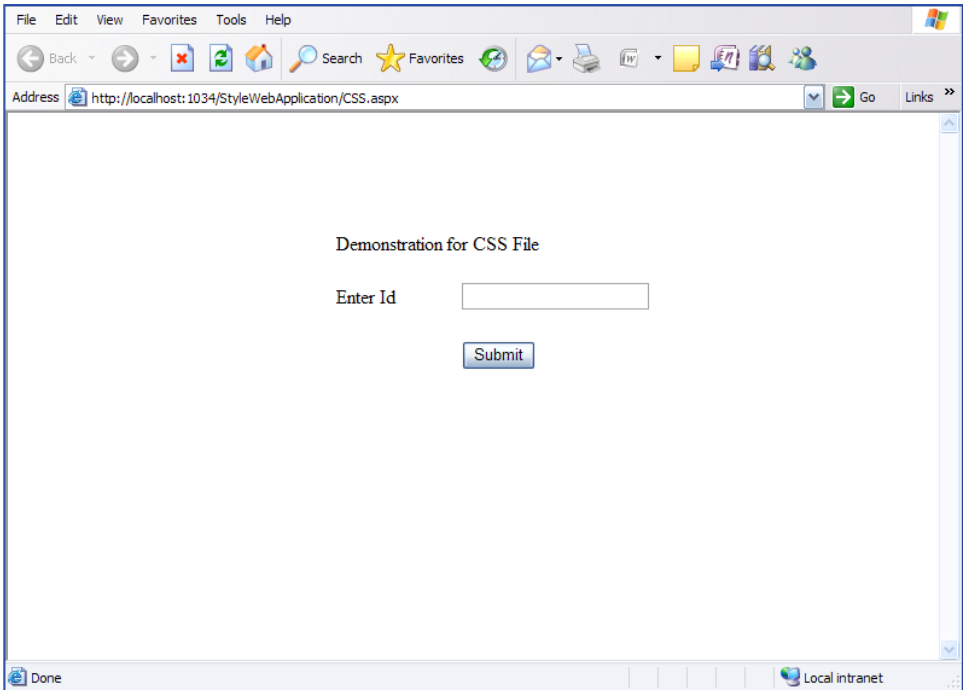


Figure 2.5-3: CSS.aspx before applying CSS style.

After CSS file Applied CSS.aspx Page Look like as shown below

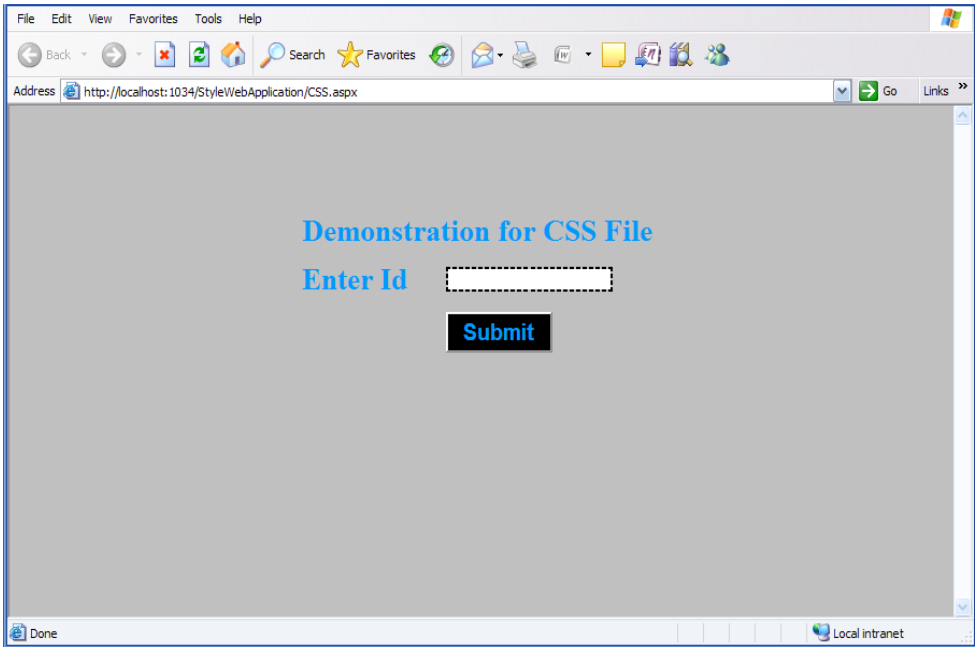


Figure 2.5-4: CSS.aspx after applying CSS style.

Step 1: Create a new Web application project called StyleWebApplication using visual studio 2005. Rename Default.aspx to CSS.aspx.

Step 2: Insert HTML Table from toolbox in CSS.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Demonstration for CSS File
asp:label	lblUserid	Text:Enter Id
asp:textbox	txtUserid	-
asp:button	btnSubmit	Text:Submit

Step 3: In the Solution Explore, right click on the name of your WebApplication (StyleWebApplication) and selecting asp.net folder add theme folder (here name given is CSStheme). Right click on CSStheme folder and selecting add new items add Style Sheet file. To include CSS file in the website we need to write in source code **theme="CSStheme".**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Theme="CSStheme" Inherits="_Default" %>
```

Step 4: The properties of the controls can be set dynamically using the cascading style sheets. Here we have created classes like content, label and button in CSS file.

Style for the html page.

```
html
{
      background-color: silver;
      font-weight: bold;
      font-size: 14px;
}
```

Style for its contents in the html page.

```
.content
{
      width :100px;
      border :solid 1px black;
      background-color: white;
      margin:auto;
      padding:10px;
}
```

Style for the label.

```
.label
{
      font-weight: bold;
      font-size: 20pt;
      color: #0099ff;
}
```

Style for the textbox.

```
.textbox
{
        border-left-color: #000000;
        border-bottom-color: #000000;
        border-top-style: dashed;
        border-top-color: #000000;
        border-right-style: dashed;
        border-left-style: dashed;
        border-right-color: #000000;
        border-bottom-style: dashed;
}
```

Style for the button.

```
.button
{
        font-weight: bold;
        font-size: 15pt;
        color: #0099ff;
        background-color: #000000;
}
```

Step 5: Include the following classes created for different controls like textbox, button and label. There is no need to include html class i.e. content class. Include classes are shown below

```
<asp:Label ID="lbl_heading" runat="server" Height="27px" Style="position: static" Text="Demonstration for Skin File" Width="345px" CssClass="label"></asp:Label>

<asp:Label ID="lbl_id" runat="server" Style="position: static" Text="Enter Id " Width="117px" CssClass="label"></asp:Label>

<asp:TextBox ID="txt_input" runat="server" Style="position: static" CssClass="textbox"></asp:TextBox>

<asp:Button ID="btn_submit" runat="server" Style="position: static" Text="submit" CssClass="button"/>
```

Step 6: Run the Application.

Context :

- Use for defining the formatting details for various controls.
- Use for giving aesthetic look to Website.

Practice Session/s :

Create a website for collecting employee personal details like employee name, address, department, experience which are supposed to be entered by employees and implement styles using skin file & web.config file.

Check List :

- Importance of Theme Files.
- Using Skin and cascading style sheet files.

Common Error/s :

- Adding skin and CSS theme without creating theme folder.

Exception/s :

- No specific exceptions.

Lesson/s Learnt:

☑ Types of Themes.
☑ Implementing Web Pages with good decent font and size and many more styles.

Best Practice/s:

- It is better to use skin files than CSS file.

A Skin can override various visual properties that where explicitly set on Server Controls within the page (a global CSS specification can never override a style set on a particular control).

CSS file can also be included by writing following code in head tag in source file as shown below

```
<link rel="stylesheet" type="text/css"
href="App_Themes/CSSTheme/StyleSheet.css" />
```

Crossword: Unit-2 **Estimated Time: 10 mins.**

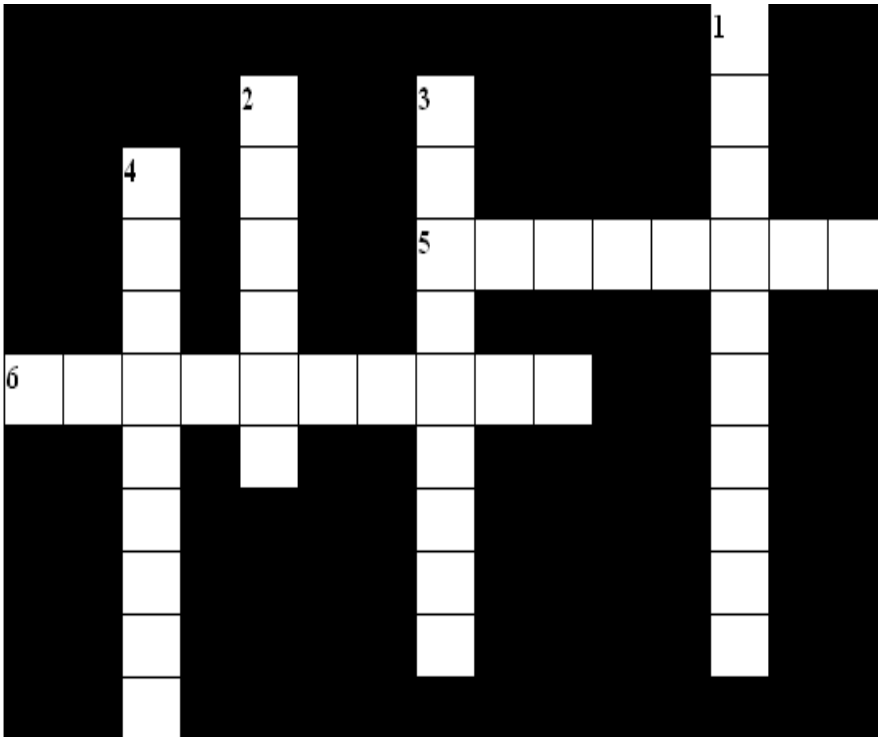


Fig. 2-1

Across:

5. Which file contains style definitions for the individual server controls.(8)

6. These pages create a consistent look and behavior for all the pages (or group of pages) in web application and provides a template for other pages, with shared layout and functionality.(10)

Down:

1. Which controls are used to validate a set of rules that you apply to the data you collect.(10)

2. Which directive, at the top of the page, is used to define that the page is Master Page.(6)

3. Which Style sheet language is used to describe the presentation of a document written in a markup language. Its most common application is to style web pages written in HTML and XHTML.(9)

4. Which are the Controls with more built-in features than HTML server controls, which 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.(9)

3.0 Adding code to ASP .Net Web Form

Topics

- 3.1 Inline Coding and Code Behind Model
- 3.2 Manipulating Pages and Server Controls
- 3.3 Event Procedure to Web Server Control
- 3.4 Crossword

Topic: Inline coding and Code behind model

Estimated Time: 20 mins.

Objectives : At the end of the activity, the participant should understand basics of

- Inline Coding
- Code Behind Model

Presentation :

Inline Coding:

- The code is in <script> and </script> blocks in the same **.aspx** file that contains the HTML and controls.
- ASP.NET calls this type of page programming **code-inline**, and it is very useful when code and presentation logic are to be maintained in a single file.

Code behind Model:

- Here the code for handling events is located in a physically separate file with .cs extension from the presentation content page that contains server controls and markup.

Scenario:

Network solution is a software company. Mr. Thomas is developing an application having a login page. Thomas wants to make code easy to understand either by separating application logic from presentation logic or by keeping application logic and presentation logic in single file. So for this purpose Mr. Thomas uses either code behind model or code-inline model respectively.

Demonstration/Code Snippet:

Login page for users:

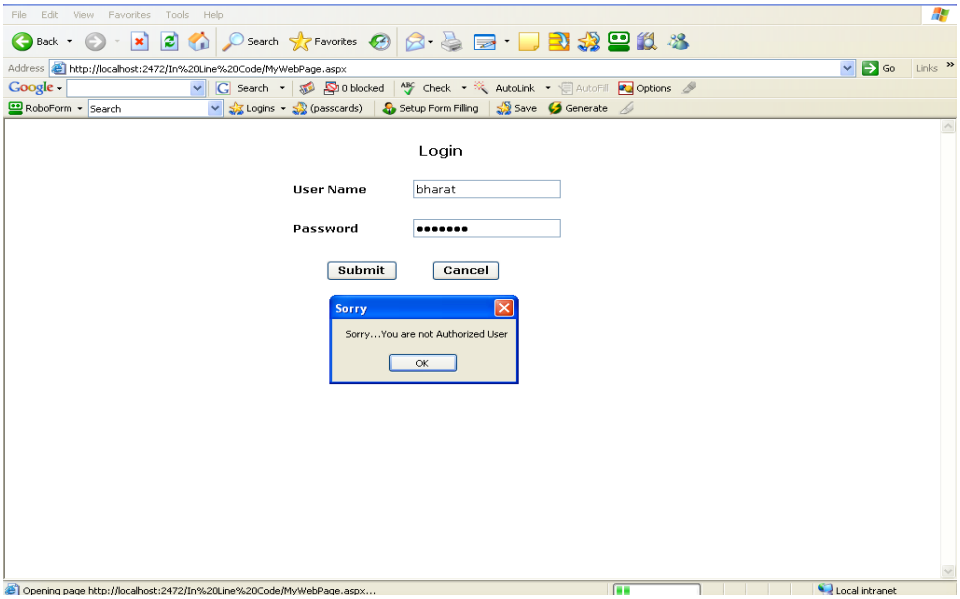


Figure 3.1-1:Loginpage design shown.

Using code behind model:

Step 1: Create a new Website using Visual Studio. Add the New Web Form by right clicking on the project in Solution Explore. "Add New Item" dialog box will be opened select Web Form and check the "Place code in separate file" option.

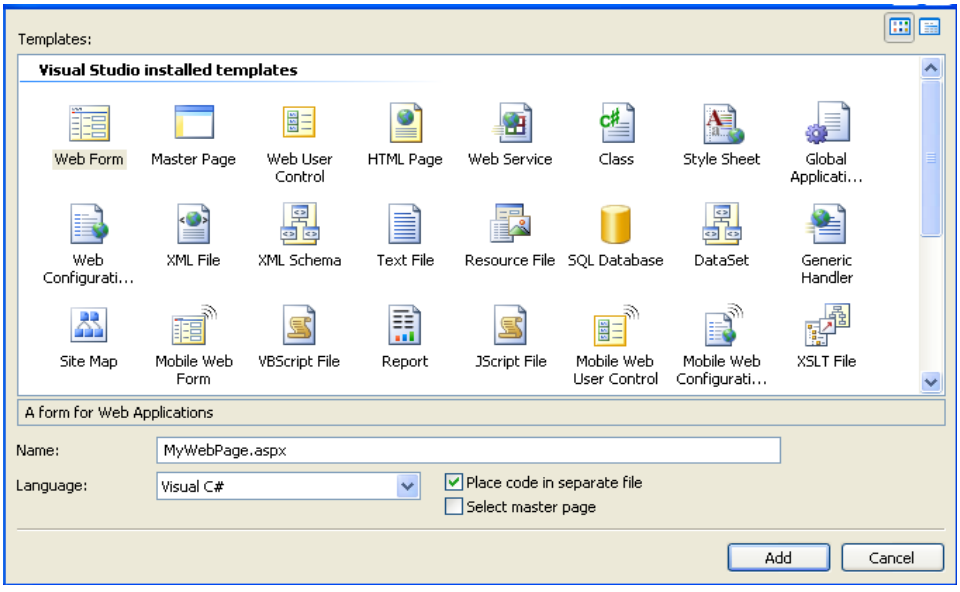


Figure 3.1-2: Selecting Place Code in separate file.

Step 2: In the Design view make the design of your page i.e. add controls to it. Suppose some action needs to be performed, like User must be authenticated when user clicks the Submit button, code must be written in the Click event of the Submit button.

Double click on Submit button one separate file will be opened e.g. “MyWebPage.aspx.cs”, and code is placed here. So the Presentation Logic is different from Business Logic. Presentation Logic will be in the source file.

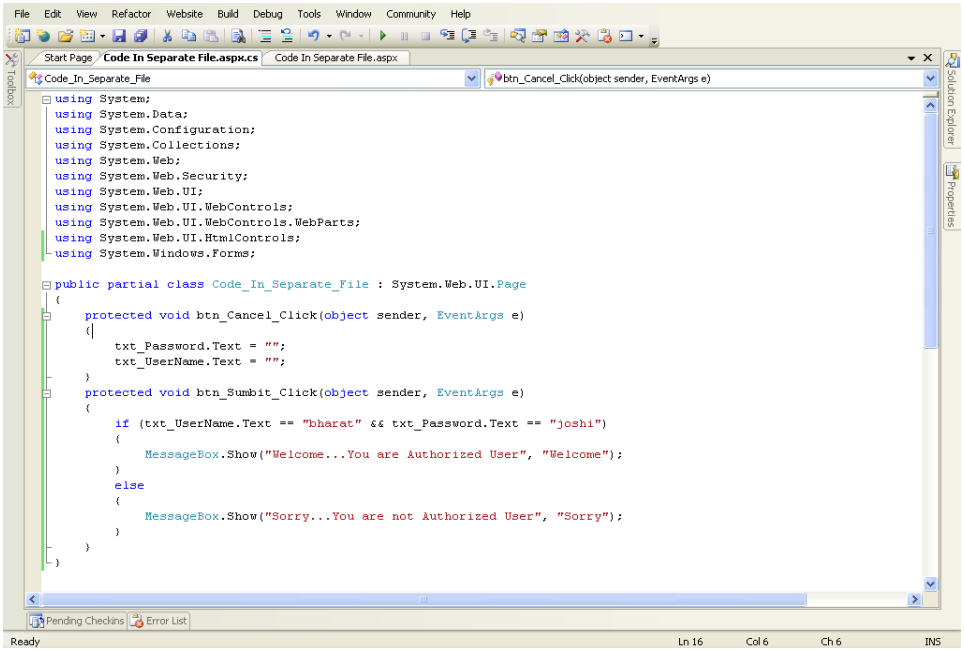


Figure 3.1-3: MywebPage.aspx.cs showing code behind model.

Using code-inline model

Step 1: Create a new Website using Visual Studio. Add the New Web Form by right clicking on the project in Solution Explore. “Add New Item” dialog box will be opened select Web Form and uncheck the “Place code in separate file” option.

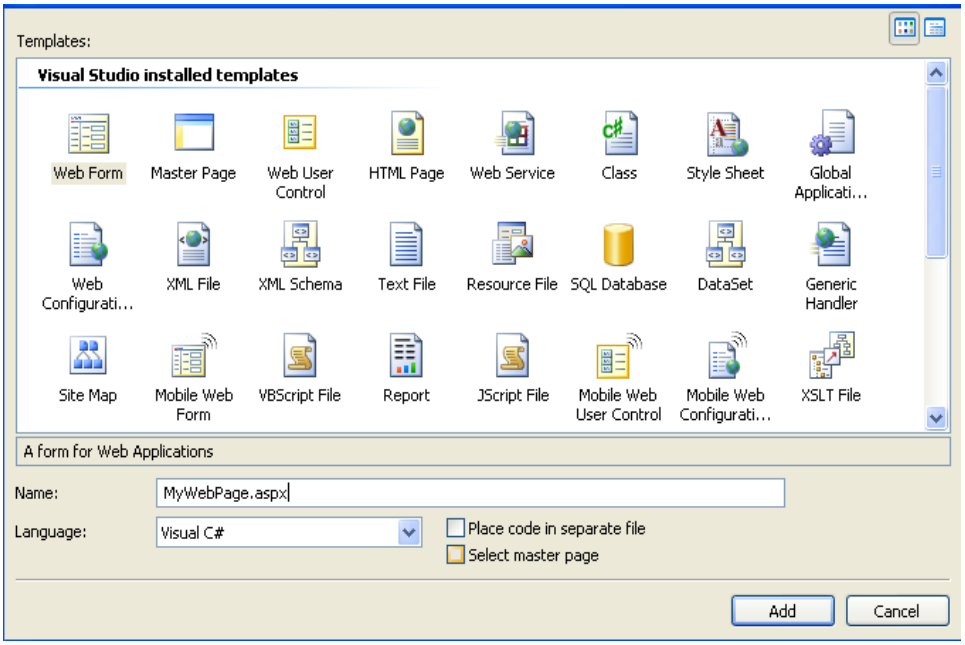


Figure 3.1-4: Adding Web form without checking Place code in separate file.

Step 2: In the Design view make the design of your page i.e. add controls to it. Suppose some action needs to be performed, like User must be authenticated when user clicks the Submit button, code must be written.

Code will be written in the source file i.e. MyWebPage.aspx, where the Presentation Logic and the Business Logic is in the same file.

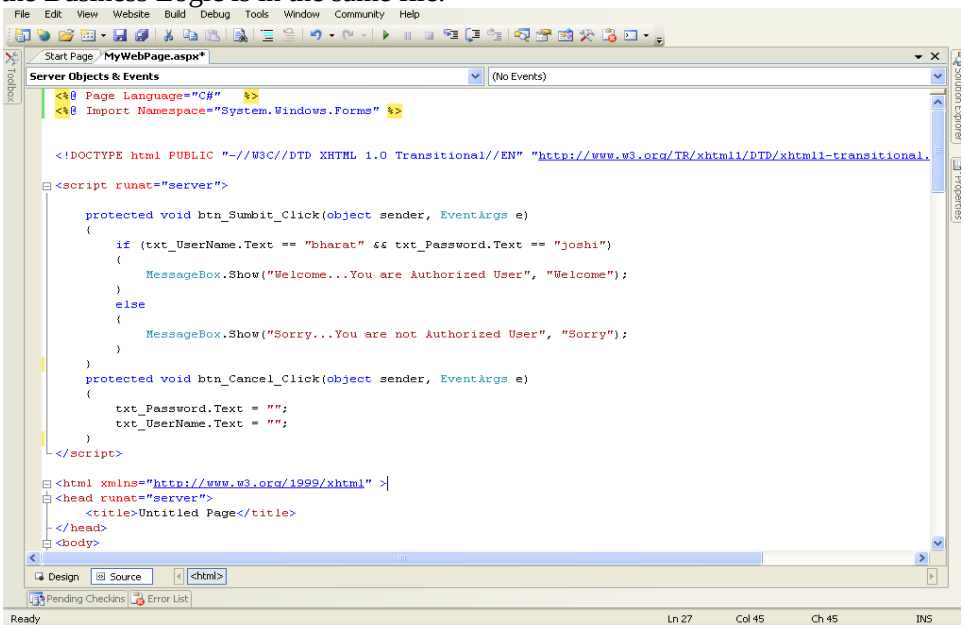


Figure 3.1-5: MyWebPage.aspx showing Code-inline model.

Context :

- **Code-inline** is very useful for maintaining code and presentation logic in a single file.
- **Code-behind** model is used in environment where designers working on the UI portions of an application while developers work on the behavior or code. It makes code easy to understand and debug by separating application logic from html tags.

Practice Session/s :

- Identify code behind model and code-inline model.
- Identify the attribute necessary for Code-Behind model.

Check List :

- Importance of Code-inline.
- Importance of Code behind model.

Common Error/s :

- If not specifying code file name in inherits part, code file will not be accessible.

Exception/s :

- No Specific Exception.

Lesson/s Learnt:

- ☑ Use of inline coding and code behind model.
- ☑ Use of Code Behind attribute in @Page directive.

Best Practice/s :

- Always use code behind model rather than inline coding.

Directives: This statement enclosed within <%@ and %> is a directive. Page directives are enclosed in <%@Page ... %> and are used to declare a number of attributes about the page.

Attribute	Possible Values	Comments
Language	A valid .NET language	Language used by all source codes on the page.
AutoEventWireup	true or false	Indicates if the page events are automatically wired up. If false, events (such as Page_Load) must be enabled by the developer.
Codebehind	A code-behind file name	Informs the compiler which file is the code-behind file for this aspx page.
EnableSessionState	true, ReadOnly, false	If true, the page has access to the Session object. If ReadOnly, the page can read but not change session variables.
EnableViewState	Valid class name	Page's ViewState is maintained for server controls.
Inherits	Valid class name	Page's ViewState is maintained forserver controls.
ErrorPage	Valid URL	Page to be redirected to if an unhandled error occurs

Topic: Manipulating Pages and Server Controls

Estimated Time: 30 mins.

Objectives : At the end of the activity, the participant should understand

- The concept of Server controls
- Manipulation of Server controls

Presentation :

- ASP.NET server controls are identified within a page using declarative tags that contain a runat="server" attribute.
- ASP.NET server control can be identified within a page by providing it with an **id** attribute. You can use this **id** reference to programmatically manipulate the server control's object model at run time.
- ASP.NET server controls can optionally expose and raise server events, which can be handled by page developers. A page developer may accomplish this by declaratively wiring an event to a control (where the attribute name of an event wireup indicates the event name and the attribute value indicates the name of a method to call).
- Each server control is an instance of a particular class. For example, when you add a button control to web page, actually instance of button class is created. The base class for all server control resides in **System.Web.UI.**

Server Controls:

Scenario:

Mr. Michael of Xmart Company wants to create an application where server controls are declared and its text and style properties customized individually.

Demonstration/Code Snippet:

Step 1: Create a new Web application project called Manipulatedemo using visual studio 2005.

Step 2: Drag and drop the server controls in default.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblMessage1	Text:This is Message One
asp:label	lblMessage2	Text:This is Message Two
asp:label	lblMessage3	Text:This is Message Three

Step 3: The following code declares three <asp: label runat="server"> server controls and customizes the text and style properties of each one individually.

```
<%@ Page Language="C#"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
<title>Server Control</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h3><font face="Verdana">Declaring Server Controls</font></h3>
```

This sample demonstrates how to declare the <asp:label> server control and manipulate its properties within a page.

```
 <p>
<hr>

<asp:label    id="lblMessage1"    font-size="16"    font-bold="true"
forecolor="red" runat=server>This is Message One</asp:label>

<br>  <asp:label  id="lblMessage2"  font-size="20"  font-italic="true"
forecolor="blue" runat=server>This is Message Two</asp:label>

<br><asp:label  id="lblMessage3"  font-size="24"  font-underline="true"
forecolor="green" runat=server>This is Message Three</asp:label>

</div>
</form>
</body>
</html>
```

Step 4: Following is the output after executing above code.

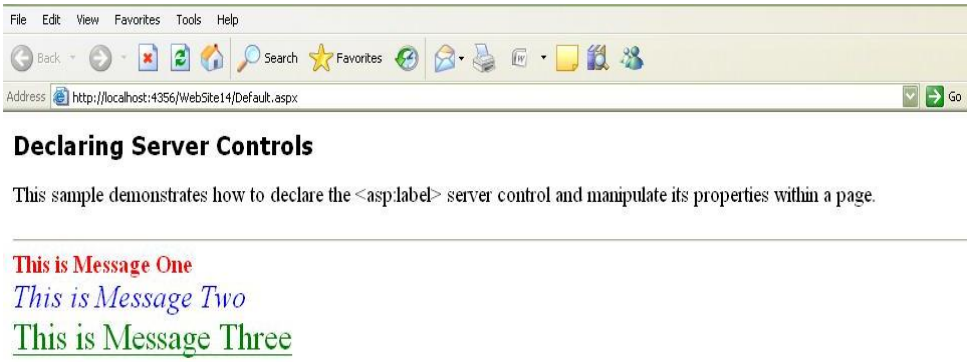


Figure 3.3-1: default.aspx showing manipulating three labels.

Manipulating Server Controls:

Scenario :

Mr. Michael of Xmart Company wants to create an application having server controls and its property manipulated in front-end logic to output the current time.

Demonstration/Code Snippet :

Step 1: Create a new Web application project called Manipulateserverdemo using visual studio 2005.

Step 2: Drag and drop the server controls in default.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblMessage	-

Step 3: The following code demonstrates how a page developer could programmatically set an **<asp: label runat="server">** control's **Text** property within the **Page_Load** event.

```
<%@ Page Language="C#"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
<script language="C#" runat="server">
void Page_Load(Object Src, EventArgs E)
  {
   Message.Text = "You last accessed this page at: " + DateTime.Now;
  }
</script>
<body>
 <h3><font face="Verdana">Manipulating Server Controls</font></h3>
```

It demonstrates how to manipulate the <asp:label> server control within the Page_Load event to output the current time.

```
<p>
 <hr>
  <asp:label     id="Message"     font-size="24"     font-bold="true"
runat=server/>
  </body>
</html>
```

Step 4: Following is the output after executing the above code:

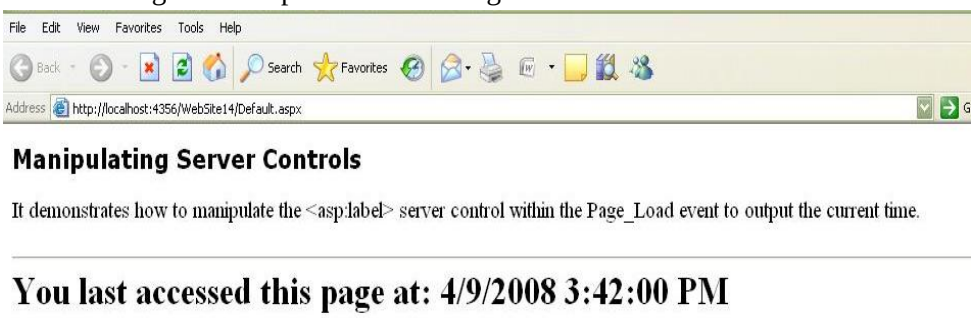


Figure 3.3-2: Default.aspx showing manipulation of label control.

Context :

- Use server controls to post any information to server.

Practice Session/s :

Design an application having server controls having "SATYAM" written in different styles and properties.

Check List :

- Importance of manipulating Server Controls.

Common Error/s :

- While retrieving data from the server, AutoPostBack property of the control must be true.
- If incorrect function name is given in button's Onclick event.

Exception/s :

- No Specific Exception.

Lesson/s Learnt :

☑ Manipulating server controls.
☑ Applying styles to server controls.

Best Practice/s :

- By directly using server controls development times minimizes.

Topic: Event Procedure to Web Server Control

Estimated Time: 60 mins.

Objectives :

At the end of the activity, the participant should understand

- Event handling Procedure

Presentation :

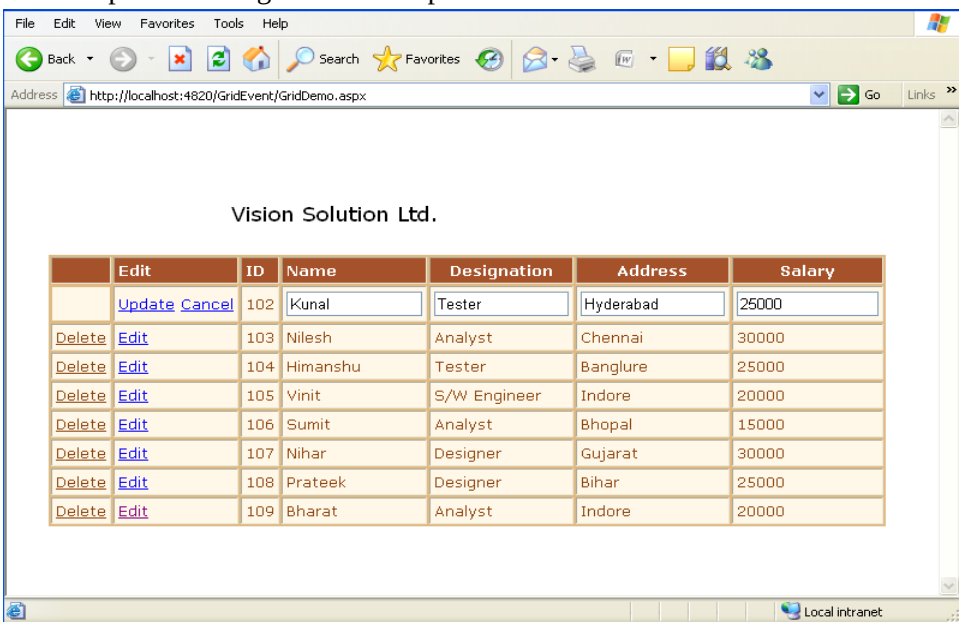
- Event Procedure can be added to Web Server Control like Grid View, Details View and DataList etc.
- For each Server Control Event Procedures are available and that can be used to write the specific code programmatically to achieve the specific task.

Scenario :

Mr. Adam working with Vision Solutions as manager wants to updates the details of employees in control. Details should be displayed in tabular format with Edit, Update, Cancel and Delete links. And allows to perform respectivity task. So Mr. Gilly, developer of website uses the concept of the GridView.

Demonstration/Code Snippet :

GridDemo.aspx is showing use of event procedure.



	Edit	ID	Name	Designation	Address	Salary
	Update Cancel	102	Kunal	Tester	Hyderabad	25000
Delete	Edit	103	Nilesh	Analyst	Chennai	30000
Delete	Edit	104	Himanshu	Tester	Banglure	25000
Delete	Edit	105	Vinit	S/W Engineer	Indore	20000
Delete	Edit	106	Sumit	Analyst	Bhopal	15000
Delete	Edit	107	Nihar	Designer	Gujarat	30000
Delete	Edit	108	Prateek	Designer	Bihar	25000
Delete	Edit	109	Bharat	Analyst	Indore	20000

Figure 3.4-1: Final Output of Gridview Event Handler.

Step 1: : Create a new Web application called Gridview Event Handler project using visual studio 2005. Rename Default.aspx to GridDemo.aspx.

Step 2: Drag and drop the GridView controls on GridView.aspx Change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:gridview	gdvMyGrid	-
asp:gridview	gdvMyGrid	AutoGenerateColumns: False
asp:label	lblHeading	Text:Vision Solution Ltd.

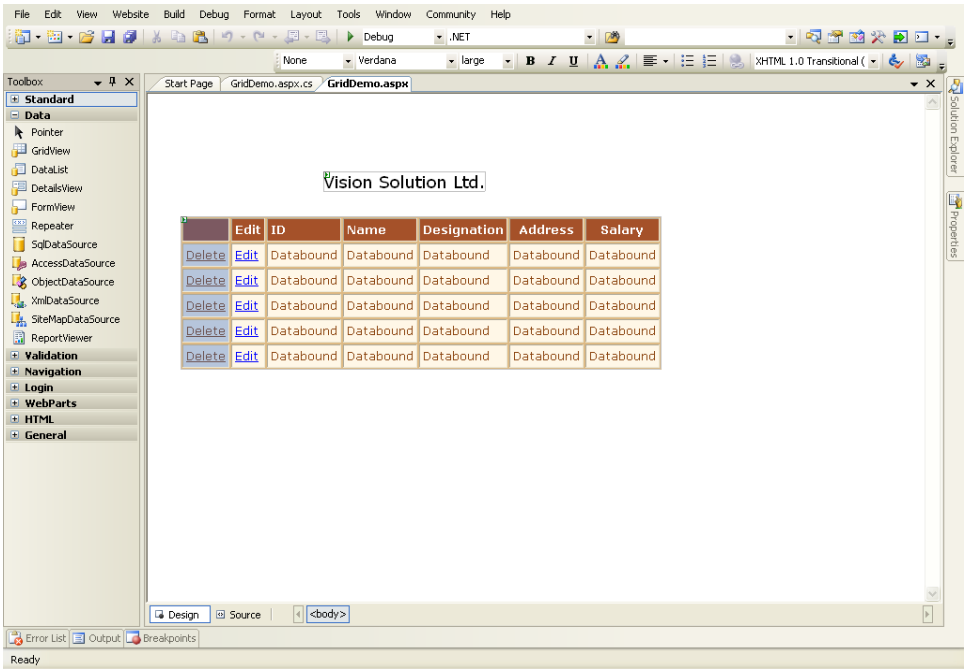


Figure 3.4-2: Design View of GridDemo.aspx

Step 3: Click the smart tag of gdvMyGrid and choose Edit Column. Select TemplateField and specify the HeaderText for all the columns to be included in the gdvMyGrid.

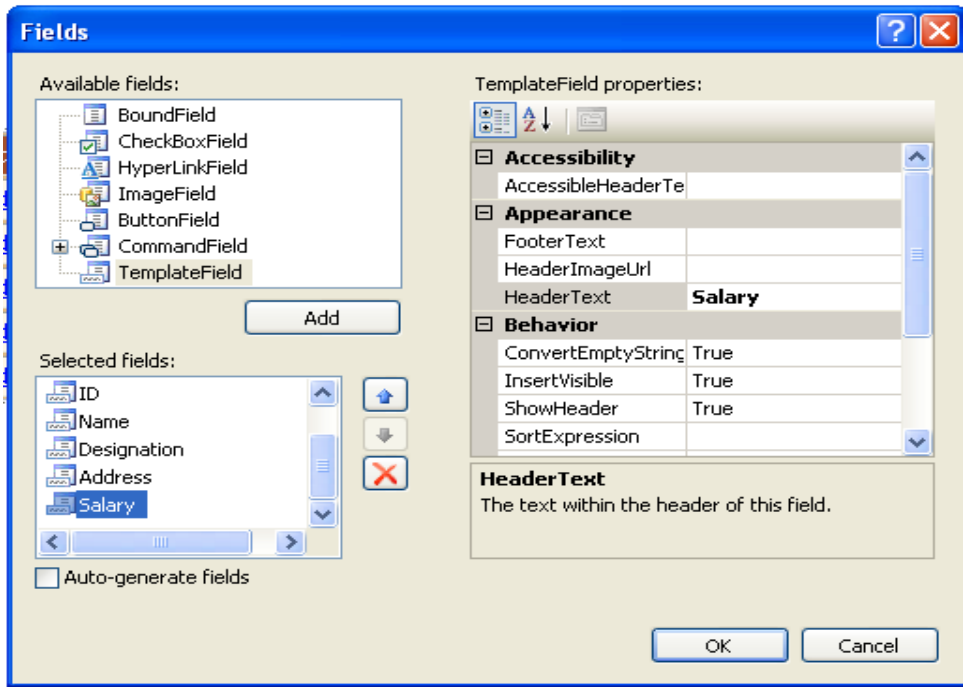


Figure 3.4-3: Fields dialogbox after clicking Edit Colums.

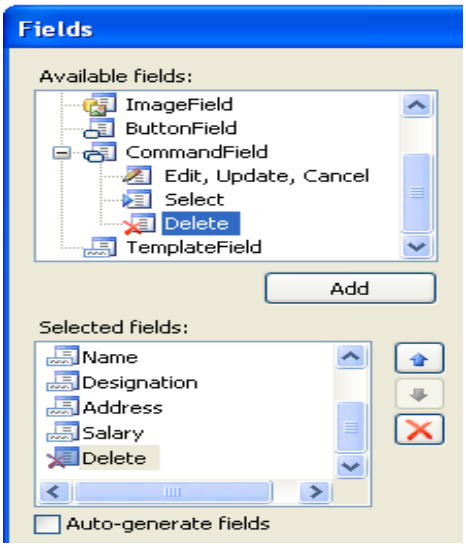


Figure 3.4-4: Adding Delete link through CommandField in Field Colums.

Edit Link can be added by Clicking TemplateField. For adding Delete button select CommandField / Delete`.

Step 4: Click the SmartTag of gdvMyGrid and click EditTemplate. Again click the SmartTag of EditTemplate. Select any of the Column from the Display drop down list. In this case Edit is selected.

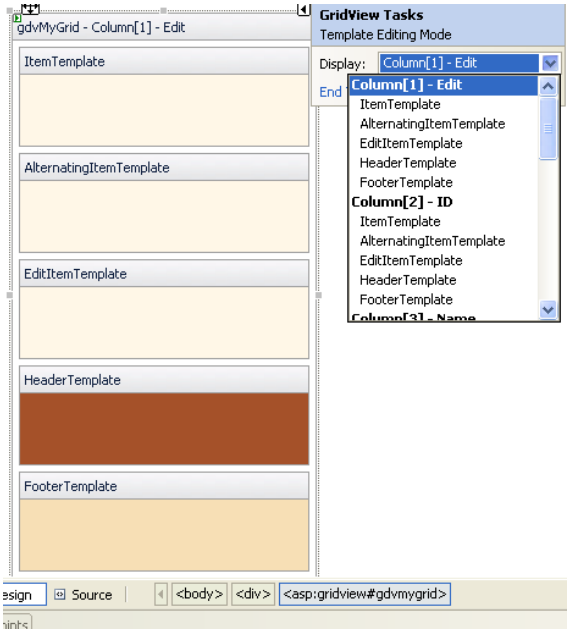


Figure 3.4-5: EditTemplate looks like this.

ItemTemplate: Gets or sets the template for the items in the GridView control.

EditItemTemplate: Gets or sets the template for the item selected for editing in the GridView control.

Step 5: : Drag and drop the one LinkButton in ItemTemplate and two LinkButtons in EditItemTemplate. Change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:linkbutton	lnkEdit	Text: Edit
asp:linkbutton	lnkUpdate	Text: Update
asp:linkbutton	lnkCancel	Text: Cancel

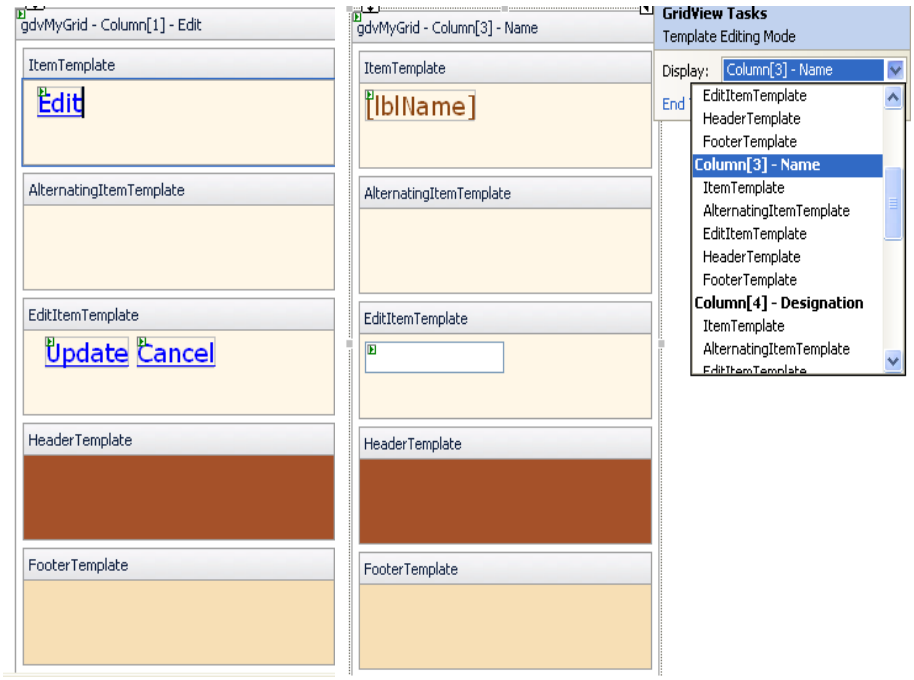


Figure 3.4-6: Adding LinkButtons and Textbox in ItemTemplate and EditItemTemplate.

Similarly add the controls to Item Template and EditItemTemplate for rest of the columns.

Step 6:

1. Click the Smart Tag of lblName label. For binding data from table to columns of GridView. Click Edit Data Bindings. lblName Data Bindings dialog box will be opened.

2. Add the statement in Code Expression "DataBinder.Eval(Container.DataItem, "Emp_Name")"

3. Repeat the above steps 1 and 2 for textbox in EditItemTemplate.

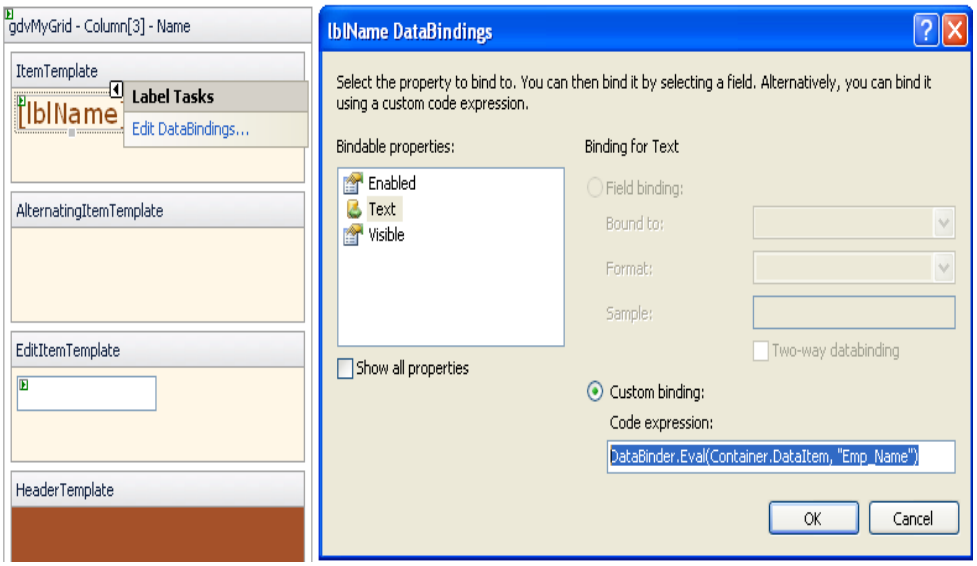


Figure 3.4-7: lblName DataBindings for binding data.

Repeat Step 6 for all the columns except Edit and Delete.

Column Name	Code Expression for Controls
Id	DataBinder.Eval(Container.DataItem, "Emp_Id")
Name	DataBinder.Eval(Container.DataItem, "Emp_ Name")
Designation	DataBinder.Eval(Container.DataItem, "Emp_ Designation")
Address	DataBinder.Eval(Container.DataItem, "Emp_ Address")
Salary	DataBinder.Eval(Container.DataItem, "Emp_ Salary")

At the end of this procedure click End Template Editing from smart tag to switch to normal mode.

All the events of GridView control.

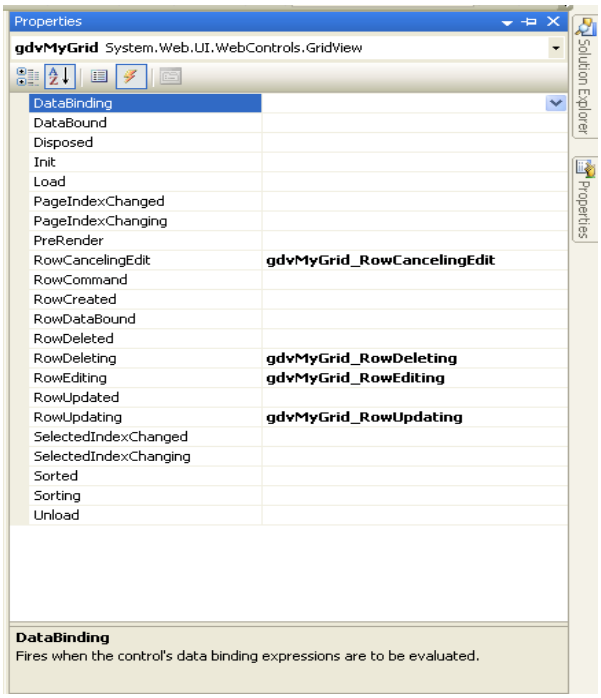


Figure 3.4-8: Properites Window for gdvMyGrid.

GridView Events	GridView Event Handlers
RowCancelingEdit	gdvMyGrid_RowCancelingEdit
RowDeleting	gdvMyGrid_RowDeleting
RowEditing	gdvMyGrid_RowEditing
RowUpdating	gdvMyGrid_RowUpdating

Step 7: Import the following Namespaces.

// Defined Namespace

```
using System.Data;
using System.Data.SqlClient;
using  System.Windows.Forms;//Add  system.windows.forms  reference  for
messagebox
```

Step 8: Global Declaration**.**

```
string query;
// Global Connection object, so that can be used to everywhere in
application.
SqlConnection con = new
SqlConnection("server=hstslc011;database=northwind;uid=sa;pwd=satyam
");
```

Step 9: Write the event handler for the Page Load.

```
protected void Page_Load(object sender, EventArgs e)
{
try
{
 if (!IsPostBack)
   {
       query = "select * from SatyamEmpDetails";
       RefereshGrid(query);
   }
}
Catch(Exception ex)
{
       MessageBox(ex);
}
}
```

Step 10: Write the User Defined Method.

```
// This User defined method contains other database objects
void RefereshGrid(string query)
{

try
{
    SqlDataAdapter adp = new SqlDataAdapter(query, con);

    // The result of query is being stored in the dataset.
    DataSet ds = new DataSet();
    adp.Fill(ds);

    // Data is being bind to GridView
    gdvMyGrid.DataSource = ds;
    gdvMyGrid.DataBind();
}
Catch(Exception ex)
{
       MessageBox(ex);
}
}
```

Step 11: Write the event handler for the _RowEditing.

```
// RowEditing handler for editing the data in the GridView
 protected void gdvMyGrid_RowEditing(object sender,
 GridViewEditEventArgs e)
 {
    gdvMyGrid.EditIndex = e.NewEditIndex;
    try
    {
          query = "select * from SatyamEmpDetails";
          // Calling RefreshGrid method
```

```
            RefereshGrid(query);
        }
        Catch(Exception ex)
        {
            MessageBox(ex);
        }
}
```

Step 12: Write the event handler for the _RowUpdating.

```
// RowUpdating handler is used for comminting the changes made by
user in RowEditing event of GridView.
protected void gdvMyGrid_RowUpdating(object sender,
GridViewUpdateEventArgs e)
{
    // FindControl method is used for finding the control being
    accessed.
    Label lblId =
    (Label)gdvMyGrid.Rows[e.RowIndex].FindControl("lblId");
    TextBox txtName =
    (TextBox)gdvMyGrid.Rows[e.RowIndex].FindControl("txtName");
    TextBox txtDesignation
    (TextBox)gdvMyGrid.Rows[e.RowIndex].FindControl("txtDesg");
    TextBox txtAddress =
    (TextBox)gdvMyGrid.Rows[e.RowIndex].FindControl("txtAddress");
    TextBox txtSalary =
    (TextBox)gdvMyGrid.Rows[e.RowIndex].FindControl("txtSalary");

    query = "update SatyamEmpDetails set emp_name='" +
    txtName.Text + "',emp_designation= '" + txtDesignation.Text+
    "',emp_address= '" +txtAddress.Text +"',emp_salary= '" +
    txtSalary.Text+ "' where emp_id='" + lblId.Text +"'";
    try
    {
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Connection.Open();
    cmd.ExecuteNonQuery();
    gdvMyGrid.EditIndex = -1;
     query = "select * from SatyamEmpDetails";// Calling
    RefreshGrid method
    RefereshGrid(query);
    }
    Catch(Exception ex)
    {
        MessageBox(ex);
    }
}
```

Step 13: Add the following code in _RowDeleting handler.

```
// RowDeleting handler is used for deleting the data from GridView.
protected void gdvMyGrid_RowDeleting(object
sender,GridViewDeleteEventArgs e)
{
   Try
   {
   Label lblId =
   (Label)gdvMyGrid.Rows[e.RowIndex].FindControl("lblId");
   query = "delete from SatyamEmpDetails where emp_id='" +
   lblId.Text + "'";
   SqlConnection con = new
   SqlConnection("server=hstslc011;database=northwind;uid=sa;pwd=sa
   tyam");
   SqlCommand cmd = new SqlCommand(query, con);
   cmd.Connection.Open();
   cmd.ExecuteNonQuery();
   gdvMyGrid.EditIndex = -1;
    try
    {
          query = "select * from SatyamEmpDetails";
          // Calling RefreshGrid method
          RefereshGrid(query);
    }
    Catch(Exception ex)
    {
          MessageBox(ex);
    }
}
```

Step 14: Add the following code in _RowCancelling handler.

```
// RowDeleting handler is used for deleting the data from GridView.
protected void gdvMyGrid_RowCancelingEdit(object sender,
GridViewCancelEditEventArgs e)
{
    gdvMyGrid.EditIndex = -1;


     try
    {
          query = "select * from SatyamEmpDetails";
          // Calling RefreshGrid method
          RefereshGrid(query);
    }
    Catch(Exception ex)
    {
          MessageBox(ex);
    }
}
```

Table Used: SatyamEmpDetails

Database Used: Northwind

	Emp_Id	Emp_Name	Emp_Designation	Emp_Address	Emp_Salary
1	102	Kunal	Tester	Hyderabad	25000
2	103	Nilesh	Analyst	Chennai	30000
3	104	Himanshu	Tester	Banglure	25000
4	105	Vinit	S/W Engineer	Indore	20000
5	106	Sumit	Analyst	Bhopal	15000
6	107	Nihar	Designer	Gujarat	30000
7	108	Prateek	Designer	Bihar	25000
8	109	Bharat	Analyst	Indore	20000

Figure 3.4-9: SatyamEmpDetails data.

Context :

- Use Event Handler for DataBound controls for manipulating the data.

Practice Session/s :

Create the webpage which allows user to increase the price of the products by 10%. And delete the product details which are no more in demand by the customers. Use Grid View and Event Handler to achieve the task.

Check List :

- Importance of Event Handler for handling events manually.

Common Error/s :

- Not binding the controls with the field of table in Item Template.
- Ensure connection is closed

Exception/s :

- Might get SQL Exception.

Lesson/s Learnt :

☑ Use of Event Handler for the controls.

Best Practice/s :

- Always use event handler to perform specific task programmatically.
- Always bind the controls to the respective fields of tables.

Crossword: Unit-3 **Estimated Time: 10 mins.**

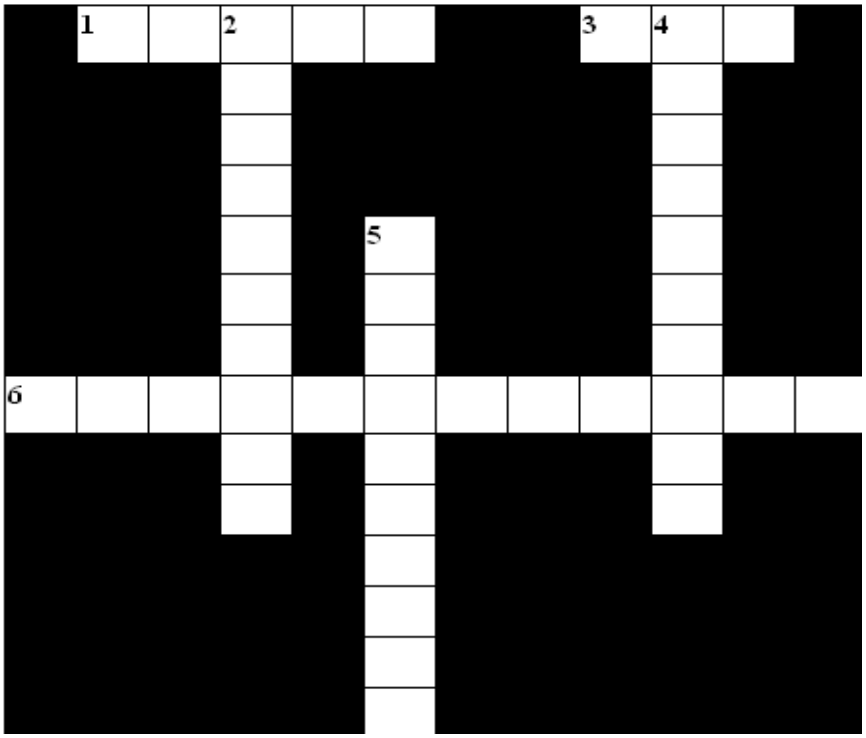


Fig. 3-1

Across:

1. When the code is in 'script' blocks in the same file (Inline) that contains the HTML and controls, the file is saved with the extension.(5)

3. When the code for handling the events is written in a separate file, then the file is saved with the extension of.(3)

6. When the code is in 'script' blocks in the same .aspx file that contains the HTML and controls, then it is called as.(12)

Down:

2. What type of code (server or client) is found in a Code-Behind class.(10)

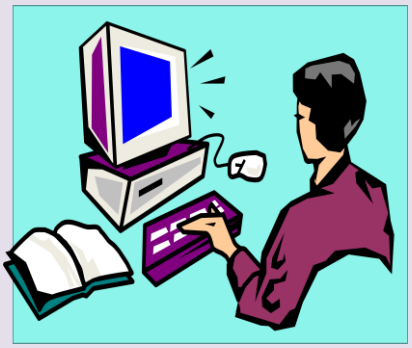
4. When the code for handling events is located in a physically separate file with .cs extension from the presentation content page that contains server controls and markup, then it is called as.(10)

5. The code which runs on the client's browser is called as.(10)

4.0 Tracing in ASP.Net Web Application

Topics

- 4.1 Tracing
- 4.2 Debugging
- 4.3 Crossword



Topic: Tracing

Estimated Time: 30 mins.

Objectives : At the end of the activity the participant should understand

- Basic of Tracing

Presentation :

- Tracing is a way to monitor the execution of ASP.NET application.
- Exception Details and Program Flow can be recorded in a way that doesn't affect the program's output.
- Tracing records diagnostic information about a particular web page.
- Tracing can be enabled for an individual page or an entire application.
- There are 2 types of tracing
  - Page Level Tracing
  - Application Level Tracing

Page Level Tracing:

- Page level tracing allows you to write debugging messages directly to the web form.
- The System.Web.TraceContext class, which is provided with .NET Framework, supports the trace functionality for asp.net pages.
- When a web page is requested, the execution details of the requested page, such as URL path and time of request for the requested page, are store in and displayed to user by the TraceContext class.
- Page level tracing can be enabled by writing **Trace="true"** attribute to the source code in design in @Page directive or in page load event of web form write **Trace.IsEnabled=true**

Application Level Tracing:

- On enabling application-level tracing, ASP.NET collects trace information for each request to the application, up to the maximum number of requests specified. The default number of requests is 10.
- Configure the application's Web.Config file so that all pages display trace information unless the page explicitly disables tracing.
- After trace feature is enabled in application's Web.Config file, trace information can be viewed by navigating to trace.axd file which is the special URL intercepted by asp.net, after executing application at least once.
- The number of requests displayed in trace.axd file will be equal to the number of times pages of the web application are accessed before navigating to the trace.axd file. Application level tracing can be enabled as shown below

```
<!--    Trace enable is used to enable the trace feature for the entire application
        RequestLimit attribute is used to specify the maximum number of HTTP
        requests for which trace information is stored in trace log
        PageOutput to display trace information for each form
        TraceMode to sort the trace information by category or by time
        localOnly to available the trace information to only local users or
        both local and remote users.-->
<trace enabled ="true" requestLimit ="100" pageOutput ="false" traceMode ="SortByTime"
localOnly ="false"/>
```

Scenario :

WilSet Solution has an online web application to collect Employee Personal Information, which is not working properly. When an employee logins for the first time employee is asked to fill the personal details and then redirected to home page. But after entering the details employee is not redirected to homepage. The application does not give any error. Mr. John, the developer with Wilset, suggests to use the feature of Trace to accomplish the task.

Demonstration/Code Snippet :

Tracing can be done in two ways

1. **Page-Level Tracing:**

On making **Trace.IsEnabled=true** in page load event Trace message is written in the Load event of the form is displayed in the **Trace information** section. The category specified in the **Trace. Write ()** method is displayed under the category column and the message string is displayed under the Message column.

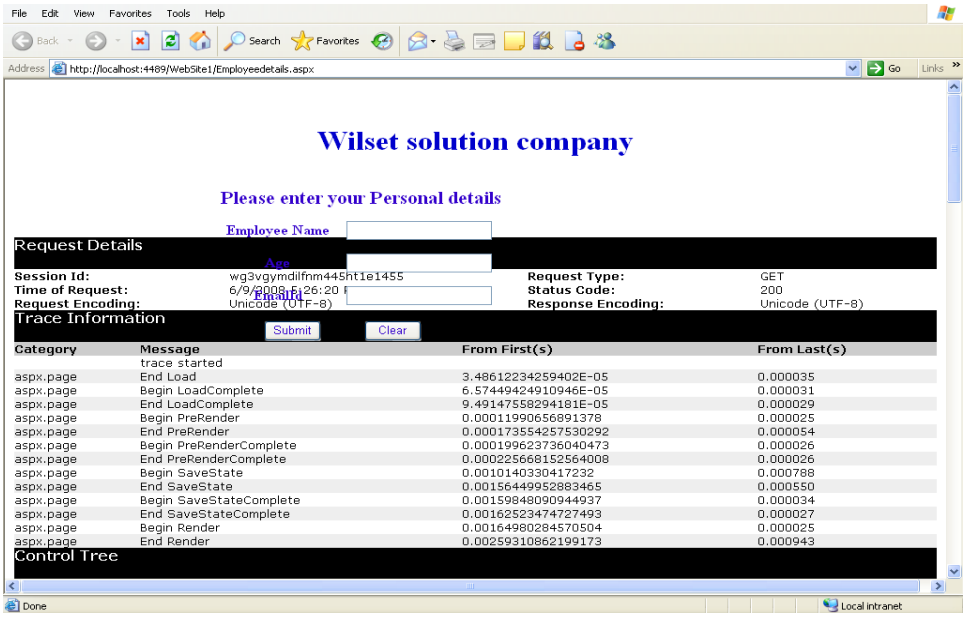


Figure 4.1-1: Page Level Tracing.

After entering details and clicking submit button page is viewed as shown below. When a user clicks the Submit Button control present in the form, the trace message is written using Trace.warn is displayed in red.

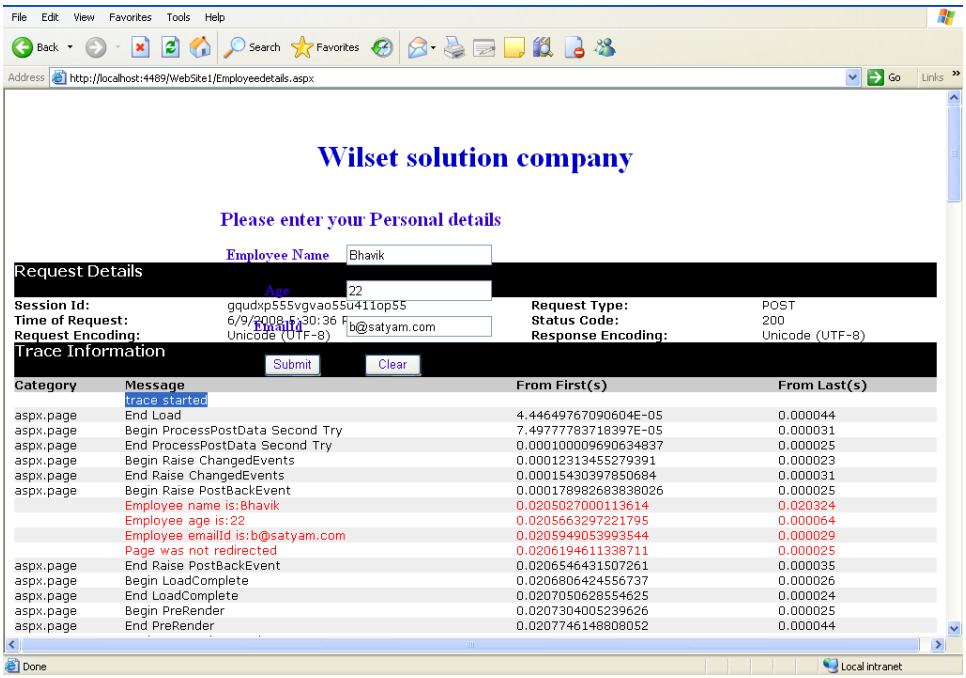


Figure 4.1-2: Page Level Tracing.

Step 1: Following code is written in the Employeedetails.aspx

```
Using System.Data;//Defining namespaces
Using System.Data.SqlClient;//Namespace for Sqlclient objects
Using  System.Windows.Forms;//Add  system.windows.forms  reference  for
messagebox


protected void Page_Load(object sender, EventArgs e)
{
     Trace.IsEnabled = true; ");// To Enable page level tracing
     Trace.Write("trace started"); // To diplay the trace message in
the Trace Information section.
 }

protected void btnSubmit_Click(object sender, EventArgs e)
{
     SqlConnection con = new SqlConnection
     ("server=hstslc011;uid=sa;password=satyam;database=northwind");
     //Defining Connection
     SqlCommand cmd = new SqlCommand("insertempdetails",
     con);//calling stored procedure
     cmd.CommandType = CommandType.StoredProcedure;
```

```
    cmd.Parameters.Add("@empname", SqlDbType.VarChar, 30).Value =
    txtemployeename.Text.ToString();
    cmd.Parameters.Add("@empage", SqlDbType.Int).Value =
    int.Parse(txtage.Text.ToString());
    cmd.Parameters.Add("@empemailid", SqlDbType.VarChar, 30).Value
    = txtEmailId.Text.ToString();
    try
    {
        con.Open();
        cmd.ExecuteNonQuery();
     }
     Catch(Exception ex)
     {
        MessageBox(ex);
     }
     finally
     {
        con.Close();
     }

     //Response.Redirect("home.aspx");
     Trace.Warn("Employee name is:" +
     txtemployeename.Text.ToString());//Display trace message in
     the trace Information section in red.
     Trace.Warn("Employee age is:" + txtage.Text.ToString());
     Trace.Warn("Employee emailId is:" +
     txtEmailId.Text.ToString());
     Trace.Warn("Page was not redirected");
}
```

Step 2: Writing **Response.Write** after close command it will redirect to home page and **Trace.Warn** message won't work.

Tables used: emp_details
Database: Northwind

	empid	empname	empage	empemailid
1	1	kunal parikh	22	k@satyam.com
2	2	Nilesh jain	22	n@satyam.com
3	3	Bhavik	22	b@satyam.com

Figure 4.1-3: emp_details Table.

Procedure:

```
create procedure insertempdetails @empname varchar(30),@empage
int,@empemailid varchar(30)
as
begin
insert into emp_details(empname,empage,empemailid) values
(@empname,@empage,@empemailid)
end
```

2. Application Tracing:

Tracing can be enabled by modifying the trace element in the web.config file. View the trace information by navigating to the **Trace.axd** file,after executing the application atleast once. Browsing to the **Trace.axd** file,the result shown as:

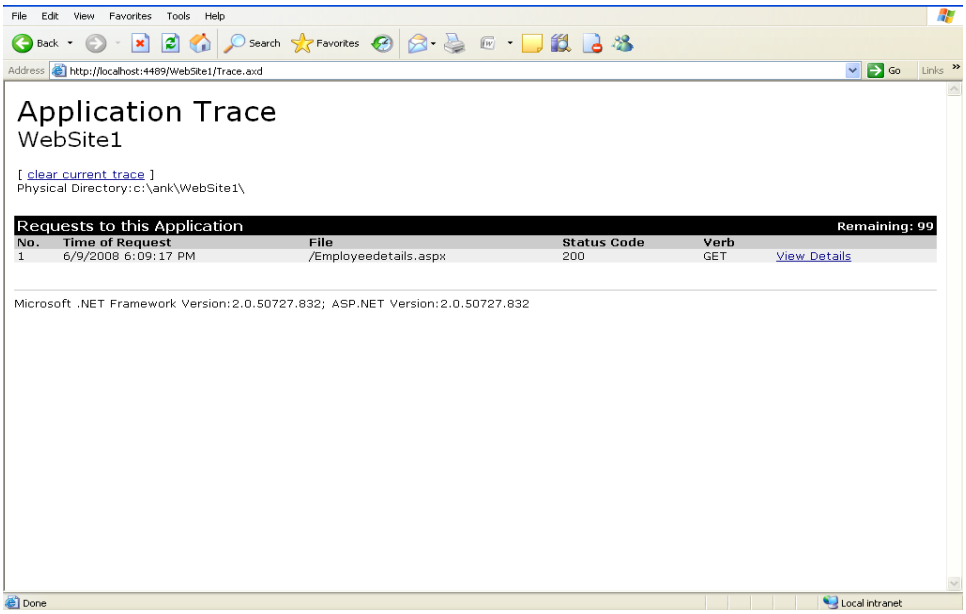


Figure 4.1-4: Application Level Tracing.

On clicking the **View Details** link, result shown as:

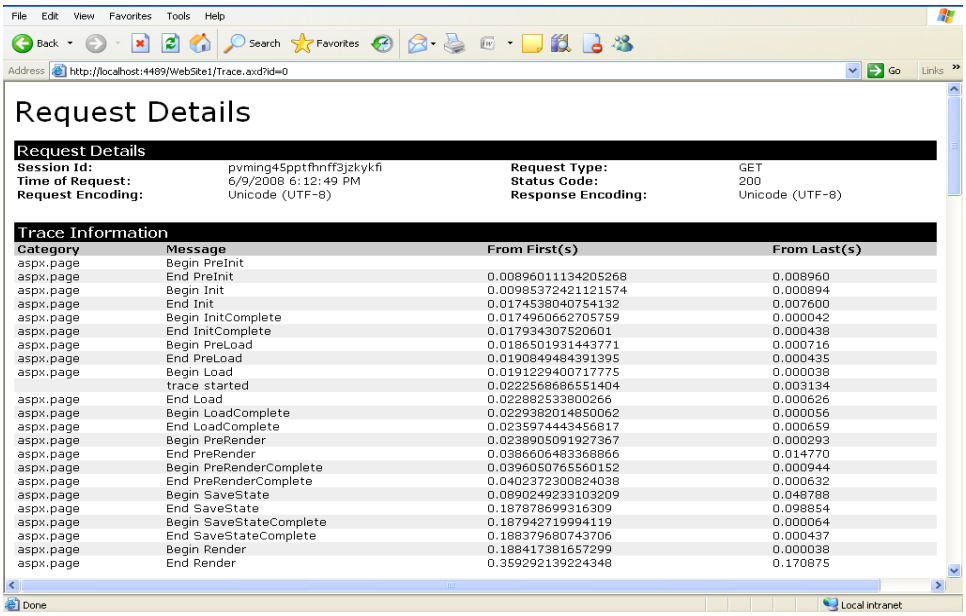


Figure 4.1-5: Request Details using Application Level Tracing.

Step 1: Follow the step 1 of Page Level Tracing.

Step 2: Right click on the website in solution explorer. Selecting add new item add web configuration file. Add following code in web.config file:

```
<!-- Enables the trace feature for the entire application-->
<trace enabled ="true" pageOutput ="false"/>
```

On making pageoutput=”true” it work same as page level tracing .The only difference is no need to specify **Trace.IsEnable=true** or **trace=”true”** attribute in @page directive of each page.

Sorting of the trace messages can be done by using **Trace mode** property set to **SortByCategory** or **SortByTime** in the **@Page directive.**

Context :

- Page level tracing is used to debug a single page.
- Application level tracing is used to debug the whole application.

Practice Session/s :

- Identify the mechanism to trace an application.

Check List :

- Importance of tracing for developers to trace the application and rectify logical errors.

Common Error/s :

- Tracing an application without enabling trace.

Exception/s :

- No specific exception for Tracing.

Lesson/s Learnt :

☑ Use of Page level tracing.
☑ Use of application level tracing.

Best Practice/s :

- Developer can use page-level or application-level tracing depending upon the feasibility.

Trace is used to see the Session and Application state variables.

Topic: Debugging

Estimated Time: 30 mins.

Objectives : At the end of the activity, the participant should understand

- Basics of debugging.

Presentation :

- Debugging is the process of finding and fixing errors in application. Without good tools this process can be time consuming.
- The VS.NET debugger does offer additional features however, as well as offering the advantage of being integrated with the rest of the IDE toolset.
- The steps of the stepping process are definable by the developer. The debug menu of VS.NET provides three options for step execution:
  - **Step Into**: Executes the code in step mode – if a method call is encountered the program execution steps into the code in step mode.
  - **Step Over**: Will not execute the next method call in step mode, continuing to the next statement after the method call.
  - **Step Out**: If already in a method call will continue through the rest of the method without stepping, returning to step mode when control returns to the calling statement.
- There are 2 type of errors
  - **Compile time errors**: Compile time errors are largely syntactic errors which will be captured by the compiler. Note that the use of Option Explicit and Option Strict can reduce the likelihood of runtime errors via compile time identification of likely issues.
  - **Runtime errors**: Runtime errors are bugs – programs that compile successfully but do not run as expected.

Scenario:

WilSet Solution owns the application which allows their user to enter Personal Information. After submitting all the details to database, database is not updated with correct values. So application must be debugged to find out the problem.

Demonstration/Code Snippet:

Step 1: Create a new Web application project called Debugdemo using visual studio 2005. Rename the default.aspx to debugpage.aspx.

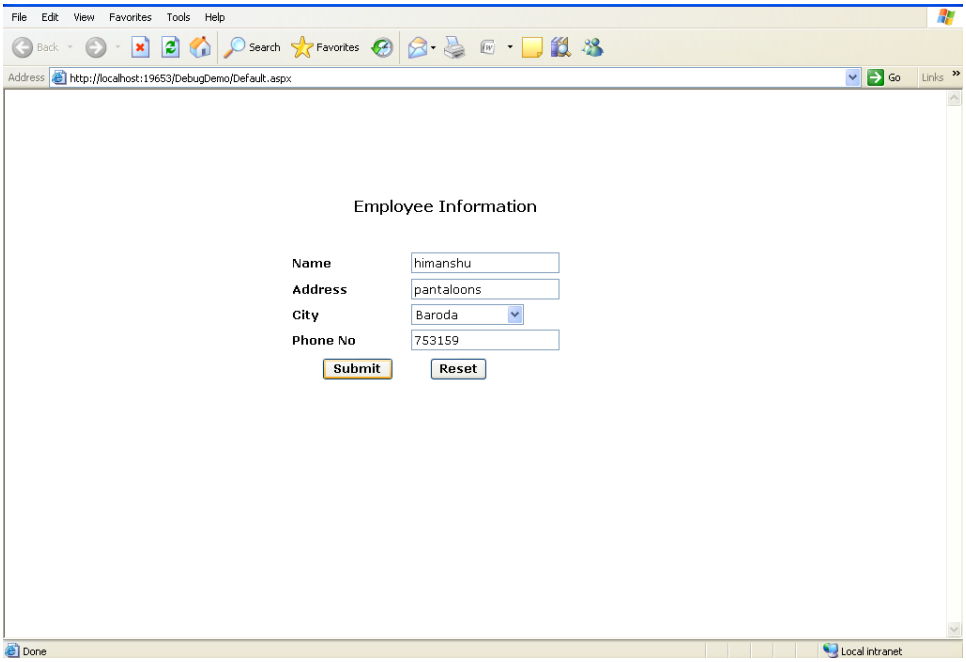


Figure 4.2-1: Employee Information Form.

Step 2: Following code is written in debugpage.aspx.

```
Using System.Data.Sqlclient;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox

public partial class _Default : System.Web.UI.Page
{
    SqlConnection conn;
    SqlCommand cmd;

protected void btnReset_Click(object sender, EventArgs e)
{
    txtAddress.Text = "";
    txtName.Text = "";
    txtPhone_No.Text = "";
    ddlCity.Text = "Select";
}
protected void Page_Load(object sender, EventArgs e)
{
    String connstr = "server=hstslc011; uid=sa; password=satyam;
    database=northwind";
    conn = new SqlConnection(connstr);
}
```

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
try
{
     cmd = new SqlCommand("InsertEmpDetailsProc", conn);
     cmd.CommandType = CommandType.StoredProcedure;

     string city= ddlCity.SelectedItem.ToString ();
     city= "Hyderabad";

     cmd.Parameters.Add("@name", SqlDbType.VarChar, 50).Value =
     txtName.Text.ToString();
     cmd.Parameters.Add("@address", SqlDbType.VarChar, 50).Value =
     txtAddress.Text.ToString();
     cmd.Parameters.Add("@city", SqlDbType.VarChar, 50).Value =
     city.ToString ();
     cmd.Parameters.Add("@phone", SqlDbType.Int).Value =
     txtPhone_No.Text;

     conn.Open();
     cmd.ExecuteNonQuery();
}
catch (Exception ex)
{
     MessageBox(ex);
}
finally
{
     con.Close();
}
}
}
```

Step 3: Insert a **Debug** / **Breakpoint** in the code. In this case enter a breakpoint at click event of Submit button.

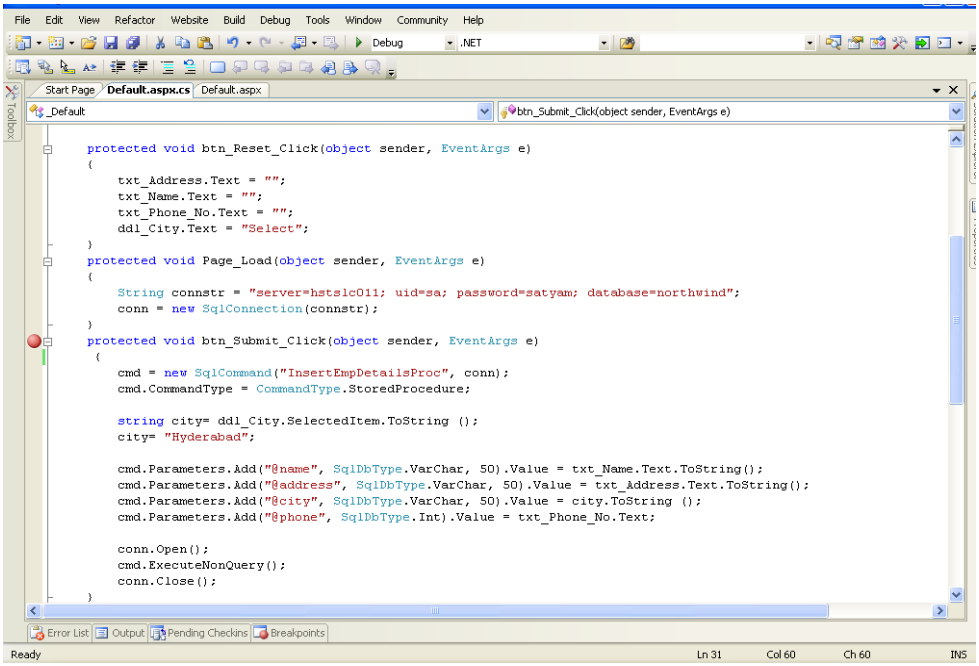


Figure 4.2-2: Break Point is inserted in EmpDetails.aspx.

Step 4: Build and Run the application. After Submitting the details application will enter in break mode.

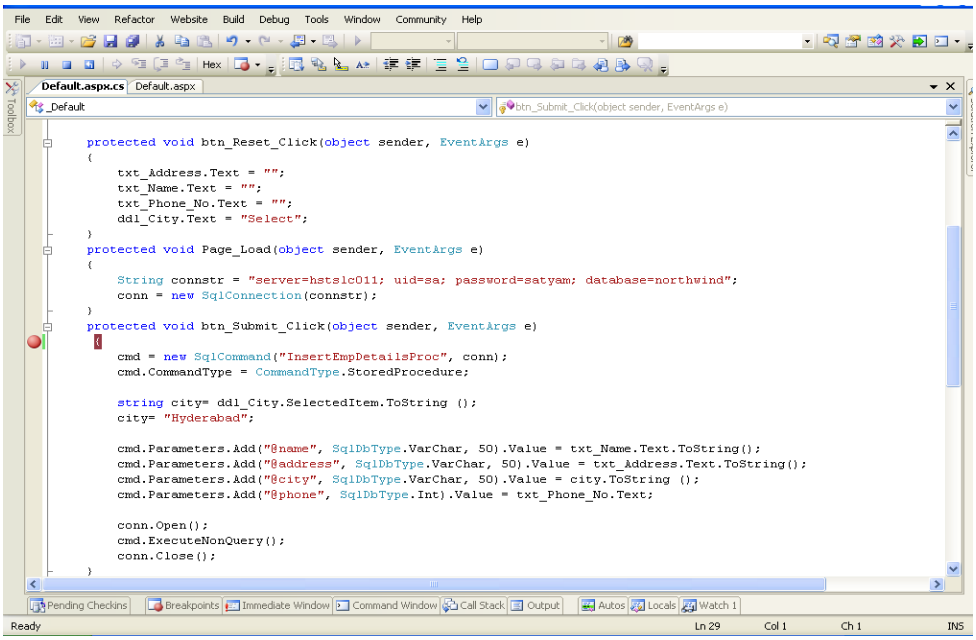


Figure 4.2-3: EmpDetails.aspx is entered in Break Point.

Step 5: Select **Step Into** from the **Debug** menu or press F11.

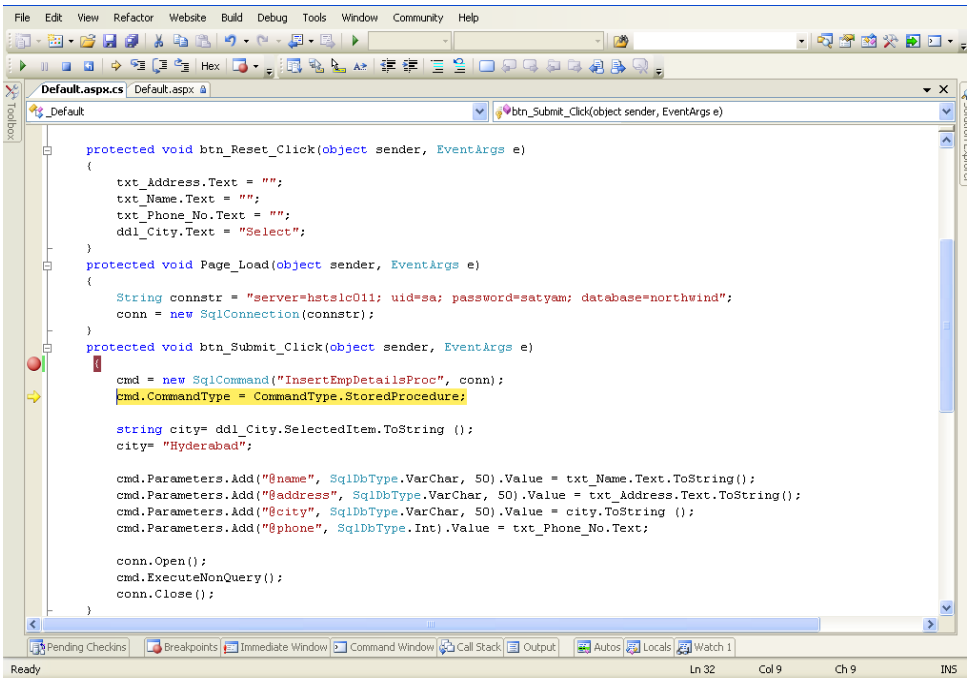


Figure 4.2-4: EmpDetails is in Debug mode.

Step 6: Find the error in the code if any.

Step 7: Stop the debugging process and replace the line containing error.

Step 8: Remove the breakpoint. In this case after assigning value to city variable, city variable is again reinitialized to Hyderabad. So removing that code snippet, Build and Run the application again.

Table used: EmpDetailsDebug
Database: Northwind

	EmpId	EmpName	EmpAddress	EmpCity	EmpPhoneNo
1	14	bharat	begampet	Hyderabad	123456
2	15	nilesh	pantaloons	Hyderabad	753159
3	19	niroja	aamirpet	Hyderabad	951357
4	20	vinit	phokatpally	Hyderabad	5555555
5	21	kunal	begampet	Hyderabad	753159

Figure 4.2-5: EmpDetailsDebug table.

Procedure:

```
create proc InsertEmpDetailsProc @name varchar(50), @address
varchar(50), @city varchar(50), @phone int
as
begin
insert into EmpDetailsDebug values(@name,@address,@city,@phone)
end
```

Context :

- Debugging is used to solve the logical errors.

Practice Session/s :

- Identify the mechanism to debug an application.

Check List :

- Importance of debugging in solving logical errors.

Common Error/s :

- Without breakpoints trying to debug application during runtime.

Exception/s :

- No specific exception.

Lesson/s Learnt :

☑ Use of debugging.

Best Practice/s :

- Always use debugging for finding the logical errors.

Crossword: Unit-4 **Estimated Time: 10 mins.**

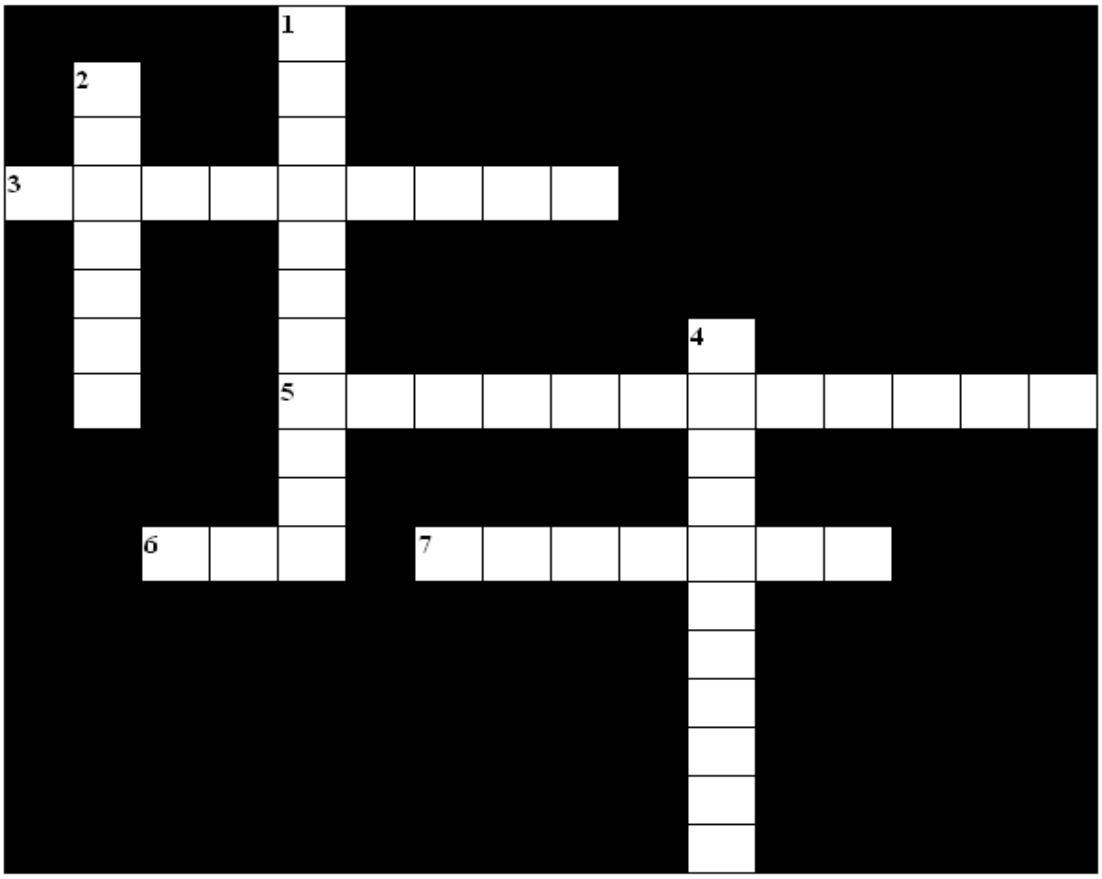


Fig. 4-1

Across:

3. One of the level of tracing which allows you to write the debugging messages directly to the web form.(9)

5. Which System. Web class is provided with .NET Framework, which supports the trace functionality for asp.net pages.(12)

6. what is the default number of requests in application level tracing.(3)

7. This is a type of error. Programs that compile successfully but do not behave as expected.(7)

Down:

1. By enabling this type of tracing, ASP.NET collects trace information for each request to the application, up to the maximum number of requests you specify.(11)

2. What is the way to monitor the execution of your ASP.NET application by which you can record exception details and program flow in a way that doesn't affect the program's output.(7)

4. One of the type of error, which is captured by the compiler.(11)

5.0 Validation User Input

Topics

- 5.1 Validation Controls
- 5.2 Crossword

Topic: Validation Controls **Estimated Time: 90 mins.**

Objectives : At the end of the activity, the participant should understand

- Required Field Validator
- Range Validator
- Compare Validator
- Regular Expression Validator
- Custom Validator
- Validation Summary

Presentation :

- Validation server controls are set of controls that work with the information provided by end users as inputs.
- Validation is a set of rules that can be applied to the data collected and record the valid data.
- Validation controls generally do not appear on a form when the user runs the application. Validation occurs when the form is posted to the server. The validation controls test the user's input and, if the data fails any of the validations, the server sends the page back to the client device. When this occurs, the validation controls that detected errors display error messages.
- Validation controls include five controls that make comparisons and one control that summarizes any errors that occur.

  - **Required Field Validator** (Requiring an Entry)
    - ✓ Required Field Validator ensures that the user does not skip an entry.
    - ✓ The control fails validation if the value it contains does not change from its initial value when validation is performed.

  - **Range Validator** (Checking the Range)
    - ✓ The RangeValidator control tests whether an input value falls within a given range.

  - **Compare Validator** (Comparing with a Value)
    - ✓ The CompareValidator control compares the value of one control to another, or to an explicit value in the control's ValueToCompare property.

  - **Regular Expression Validator** (Matching a Pattern)
    - ✓ The RegularExpressionValidator control confirms that the entry matches a pattern defined by a regular expression.
    - ✓ This type of validation allows user to check for predictable sequences of characters, such as those in social security numbers, e-mail addresses, telephone numbers etc.

  - **Custom Validator** (Defining your own Exception)
    - ✓ The CustomValidator control calls a user-defined function (custom function) to perform validations that the other standard validators can't handle.

- ✓ The custom function can execute on the server or in client-side script, such as JavaScript or VBScript.

➢ **Validation Summary** (Summarizing Errors)

- ✓ Use the ValidationSummary control to present the user with a list of errors that occurred when a form was posted to the server.

Validation Summary does not contain a **ControlToValidate** property.

Required Field Validator:

Scenario:

Johnson owner of J&J Ltd. Owns one website called J.J.Management and wants to create a webpage that maintains employee details. Some of the fields in the page are mandatory which cannot be left blank. So Steve, developer with the company, gives an idea to use RequiredFieldValidator control to fulfill the requirement.

Demonstration/Code Snippet:

"Required" message will be displayed in case Name field is kept blank.

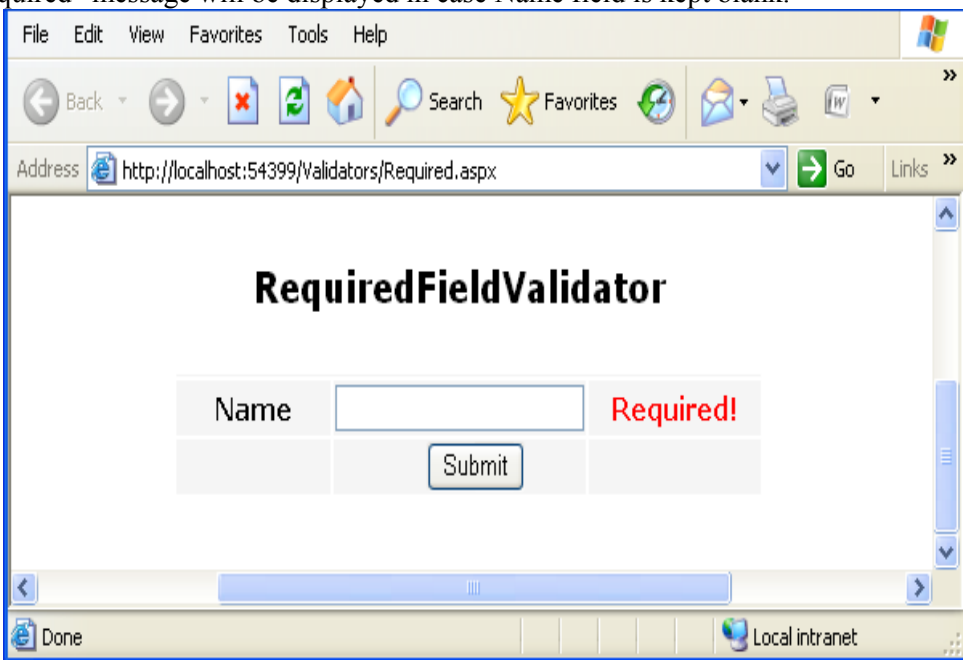


Figure 5.1-1: Required.aspx showing Required field Validator for name field.

Step 1: Create a new Web application project called Validators using visual studio 2005. Rename your default.aspx to Required.aspx.

Step 2: Insert HTML Table from toolbox in Required.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:RequiredFieldValidator
asp:label	lblName	Text:Name
asp:textbox	txtName	-
asp:button	btnSubmit	Text:Submit
asp:RequiredFieldValidator	rfvName	ControlToValidate: txtName
		ErrorMessage: Required!

Step 3: Run the application.

Range Validator:

Scenario:

Johnson owner of J&J Ltd. Wants to create a webpage that maintains employee details. Johnson wants that age of employee should have specific range. So Steve developer with the website gives an idea to use RangeValidator control to fulfill the requirement.

Demonstration/Code Snippet:

RangeValidator.aspx show that age of an employee must be between 30 and 40.

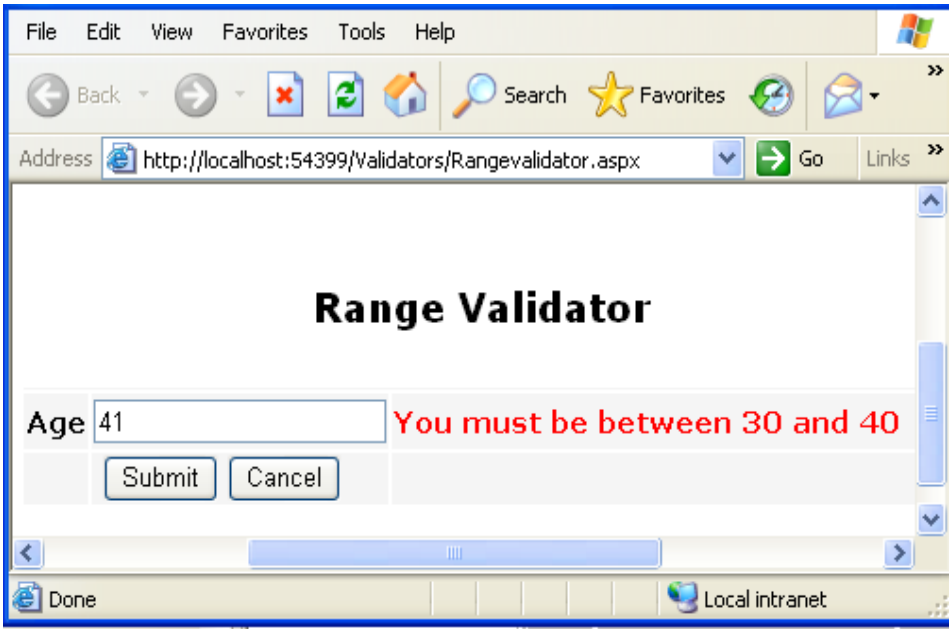


Figure 5.1-2: RangeValidator.aspx Showing Range Validator implementation.

Step 1: Create a new Web form in the same project called RangeValidator.aspx. In the Solution Explorer, right click on the name of your Web Application (Validators) and select Add New Item/Web Form.

Step 2: Insert HTML Table from toolbox in Required.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Range Validator
asp:label	lblName	Text:Name
asp:textbox	txtName	-
asp:button	btnSubmit	Text:Submit
asp:cancel	btnCancel	Text:Cancel
asp:RangeValidator	rvAge	ControlToValidate: txtAge
		ErrorMessage: You must be between 30 and 40.
		Minimum Value: 30
		Maximum Value: 40
		Type: Integer

Step 3: Run the application making RangeValidator.aspx as startup page and click on submit button.

Regular Expression Validator:

Scenario:

Johnson owner of J&J Ltd. wants to create a webpage that maintains employee details. Email Address of each employee must be in Regular Email pattern i.e. johnson@jj.com. Steve developer of the website uses RegularExpressionValidator control to fulfill the requirement.

Demonstration/Code Snippet:

RegularExpressionValidator.aspx show that email address entered is incomplete.

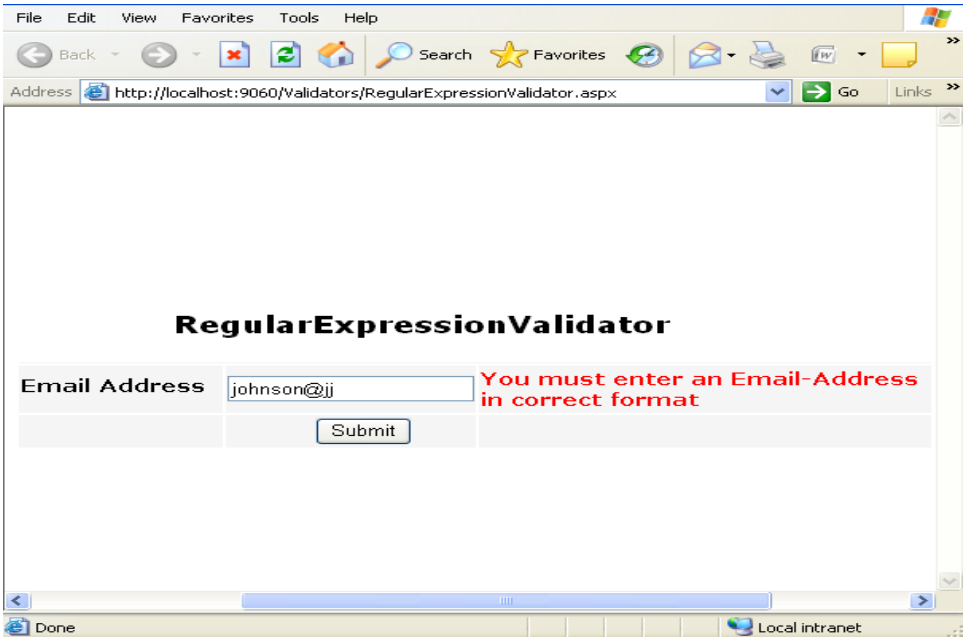


Figure 5.1-3: RegularExpressionValidator.aspx showing regular expression Validator.

Step 1: Create a new Web form in the same project called RegularExpressionValidator.aspx. In the Solution Explorer, right click on the name of your Web Application (Validators) and select Add New Item/Web Form.

Step 2: Insert HTML Table from toolbox in RegularExpressionValidator.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:RegularExpressionValidator
asp:label	lblEmail	Text:Email Address
asp:textbox	txtEmail	-
asp:button	btnSubmit	Text:Submit
asp:RegularExpressionValidator	revAge	ControlToValidate: txtEmail
		ErrorMessage: You must enter a valid email Address in Correct format
		ValidationExpression: Internet E-mail Address

Step 3: Run the application by making RegularExpressionValidator.aspx as startup page.

Compare Validator:

Scenario:

Johnson owner of J&J Ltd. wants to create a webpage that maintains employee details. Johnson wants to provide Change Password facility and also wants to ensure that newly entered password is correct. So Steve uses CompareValidator to fulfill the requirement.

Demonstration/Code Snippet:

CompareValidator.aspx show that Password and Confirm Password does not match.

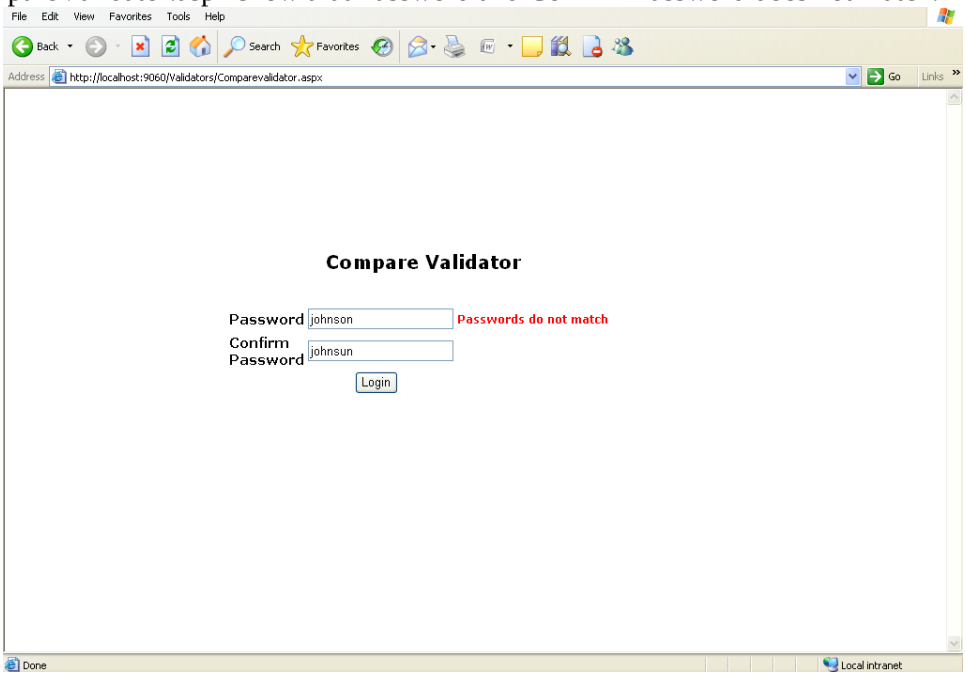


Figure 5.1-4: CompareValidator.aspx showing Compare validator implementation.

Step 1: Create a new Web form in the same project called CompareValidator.aspx. In the Solution Explorer, right click on the name of your Web Application (Validators) and select Add New Item/Web Form.

Step 2: Insert HTML Table from toolbox in CompareValidator.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Compare Validator
asp:label	lblPwd	Text:Password
asp:label	lblCfrmPwd	Text:Confirm Password
asp:textbox	txtPwd	-
asp:textbox	txtCfrmPwd	-
asp:button	btnLogin	Text: Login
asp:CompareValidator	cvPwd	ControlToValidate: txtCfrmPwd

		ErrorMessage: Passwords do not match
		ControlToCompare: txtPwd

Step 3: Run the application by making CompareValidator.aspx as startup page.

Custom Validator:

Scenario:

Johnson owner of J&J wants to add web page that maintains the records of all the products. Johnson wants to add some rules to the products which are to be ordered. Quantity should be in multiple of 10. So Steve uses CustomValidator to fulfill the requirement.

Demonstration/Code Snippet:

Quantity Number is not multiple of 10.

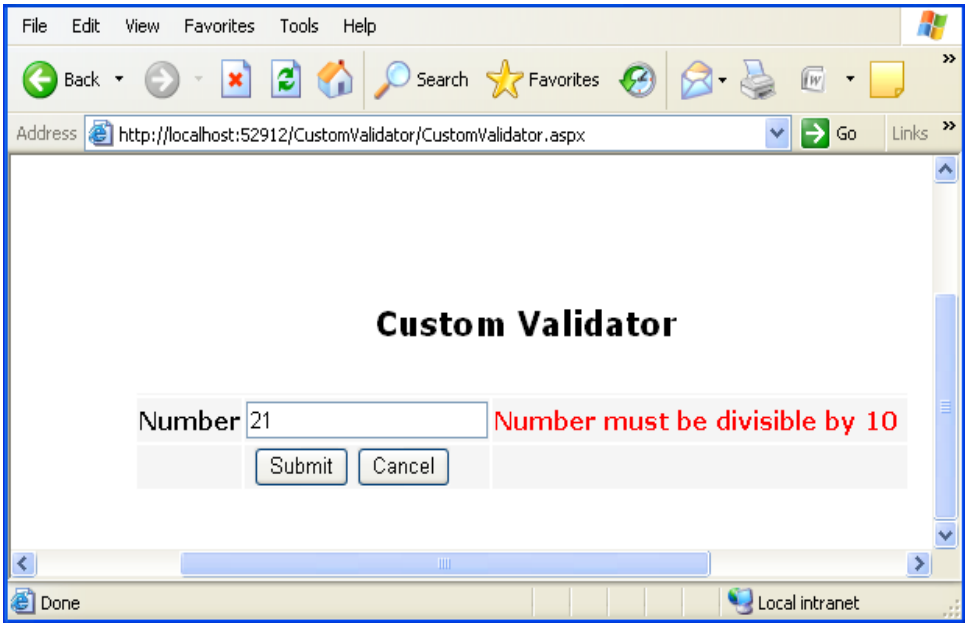


Figure 5.1-5: CustomValidator.aspx showing custom validation.

Step 1: Create a new Web application project called CustomValidator using visual studio 2005. Rename Default.aspx to CustomValidator.aspx

Step 2: Insert HTML Table from toolbox in CustomValidator.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Custom Validator

asp:label	lblNumber	Text:Number
asp:textbox	txtNumber	-
asp:button	btnSubmit	Text:Submit
asp:CustomValidator	cvPwd	ControlToValidate: txtNumber
		ErrorMessage: Number must be divisible by 10
		ClientValidationFunction: ClientValidate(use only for Client side validation)

Step 3: Write the event Handler for Submit button.

```
void btnSubmit_click(object sender,EventArgs e)
    {
        if (Page.IsValid)
            lblValid.Text = "Number is divisible by 10";
        else
            lblValid.Text = "Number is not divisible by 10";
    }
```

Using Server-side Validation:

Step 3: Write the following code in the source code file of CustomValidator.aspx below @page directive.

```
<script runat ="server">
    void validatenumber(object source, ServerValidateEventArgs args)
    {
    try
    {
    // Test whether the value entered into the text box is even.
    int i = int.Parse(args.Value);
    args.IsValid = ((i%10)==0);
    }
    catch(Exception ex)
    {
    args.IsValid = false;
    }
    }
</script>
```

Step 4: In Custom Validator tag write following to call Server validation function "validatenumber" for server validation.

```
<asp:CustomValidator ID="cvPwd" runat="server"
ControlToValidate="txtNumber"
```

```
ErrorMessage="Number must be divisible by 10" OnServerValidate
="validatenumber"></asp:CustomValidator></td>
```

Using Client-side Validation:

Step 3: Write the following code in the source code file of CustomValidator.aspx below Html closing tag </html>

```
<script type="text/javascript" >
   function ClientValidate(ctl, args)
     {
               args.IsValid=(args.Value%10 == 0);
     }
</script>
```

- Client side validation Function is written below Html closing tag because it is written in different scripting language
- The vbscript function employed uses args.IsValid property and set this property to either true or false depending on the outcome of the validation check. The Boolean value returned is then assigned to the args.IsValid property, which is then used by CustomValidator control.
- To make the association between a CustomValidator control and a function, use the OnServerValidate property. The value assigned to this property is the name of the function "validatenumber".

Step 4: Check whether ClientValidationFunction property is set or not.

Step 5: Run the Application.

Validation Summary:

Scenario:

Johnson owner of J&J wants to add web page that maintains the Associate Details. Johnson wants to provide summary of validation errors that occur during data entry. So Steve, the developer with the company, uses Validation Summary to fulfill the requirement.

Demonstration/Code Snippet:

Validators.aspx shows that on entering wrong details validation summary shown in form of message box.

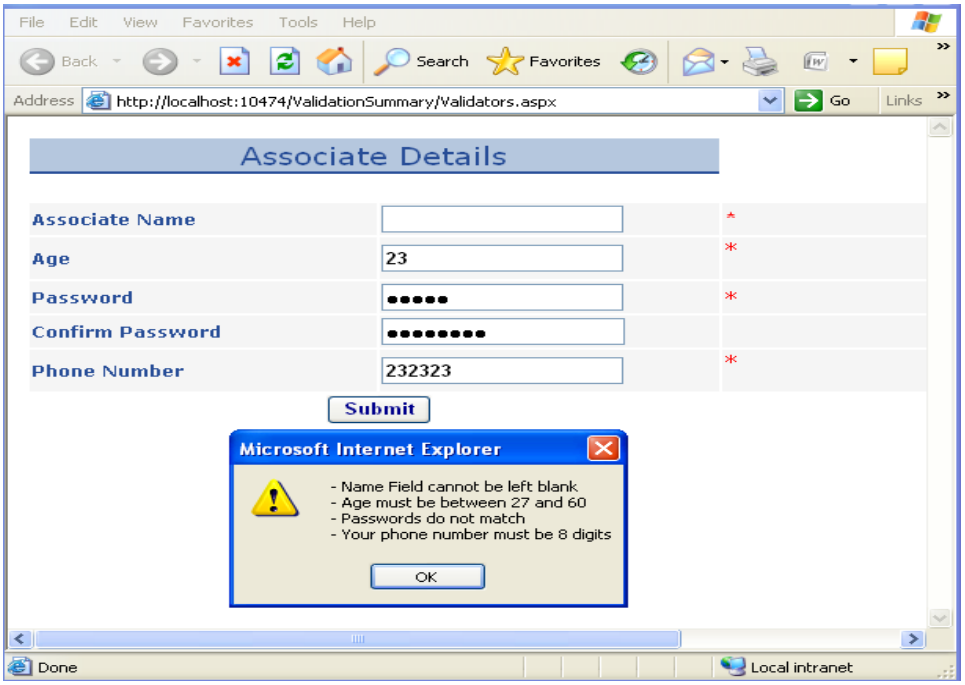


Figure 5.1-6: Validators.aspx showing validation summary in form of messagebox.

Step 1: Create a new Web application project called ValidationSummary using visual studio 2005. Rename Default.aspx to Validators.aspx

Step 2: Insert HTML Table from toolbox in CompareValidator.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblName	Text: Associate Name
asp:label	lblPassword	Text: Password
asp:label	lblCfrmPwd	Text: Confirm Password
asp:label	lblAge	Text: Age
asp:label	lblPhoneno	Text: Phone Number
asp:textbox	txtName	-
asp:textbox	txtPassword	-
asp:textbox	txtCfrmPwd	-
asp:textbox	txtAge	-
asp:textbox	txtPhoneno	-
asp:button	btnLogin	Text: Login
asp:CompareValidator	txtPwd	ControlToValidate: txtCfrmPwd
		ErrorMessage: Passwords do not match
		ControlToCompare: txtPwd
		Text: *
asp:RegularExpression Validator	revAge	ControlToValidate: txtPhoneno
		ErrorMessage: Your Phone number

		must be 8 digits.
		ValidationExpression: \d{8}
		Text: *
asp:RangeValidator	rvAge	ControlToValidate: txtAge
		ErrorMessage: You must be between 30 and 40.
		Minimum Value: 27
		Maximum Value: 60
		Type: Integer
		Text: *
asp:RequiredFieldValidator	rfvName	ControlToValidate: txtName
		ErrorMessage: Name Field cannot be left blank.
		Text: *
asp: ValidationSummary	vsSummary	ShowMessageBox: True
		ShowSummary: False

In this exercise, set the text property of all validator controls to a red asterisk, and the ErrorMessage to the actual message which the user sees in the validation summary. This may be the preferred method especially if the form is a lengthy one and the user has to scroll down to see the whole form.

Context :

- Use Validation Server Controls to validate user input.
- Use Validation Server Controls to check the user's input without a round trip back to the server.
- Use Required Field Validator for mandatory fields.
- Use Range Validator for having inputs in specified range.
- Use Compare Validator to compare 2 fields.
- Use Regular Expression Validator for email, phone no etc validation.
- Use Custom Validator to create your own validation rules.
- Validation Summary summarizes the validation errors which the user has made on a form.

Practice Session/s :

Create a web page for job application. Here applicant can apply for the job by filling certain details. Details which need to be filled are job applied from the list of job available, name,

Date of birth, Qualification using dropdown list, Gender using radio button, Email id, Phone no, Address, City, Zip code. Use Validators wherever necessary.

Check List :

- Inclusion of ClientValidation and ServerValidation function.
- Understanding different properties of Validation controls like ControlToValidate, ControlToCompare etc.
- Use of different types of Validators.

Common Error/s :

- Not assigning the values to any properties of Validator.
- Not specifying proper validation rules.
- Non inclusion of Server Validation while using Custom Validation.
- Including Client Script in Html Tag.

Exception/s :

- **(Unable to find script library'/aspnet_client/system_web/1_1_4322/WebUIValidation.js'. Try placing this file manually, or reinstall by running 'aspnet_regiis -c'.)**

It means that someone has removed the WebUIValidation.js file (or probably the whole aspnet_client folder in c:\Inetpub\wwwroot of the server). This JavaScript file contains important JavaScript functions and need to be sent to the client together with the Web form. So rectify this problem before continuing, or else all your validators will not work (since they depend on this JavaScript file). To replace WebUIValidation.js at the server, go to VS.NET's command prompt (not the usual DOS command prompt) by clicking on START/All Programs/Microsoft Visual Studio.NET 2005/Visual Studio.NET Tools/Visual Studio.NET 2005 Command Prompt.
Type aspnet_regiis -c at the VS.NET command prompt, and this command-line tool will download from the Internet and reinstall the missing JavaScript file and folder.

Lesson/s Learnt :

- ☑ Understand the importance of Client side and Server side Validation.
- ☑ Understanding the usages of various symbols in CustomValidation.
- ☑ Usages of different properties of various validation controls.

Best Practice/s :

- Always use Client side validation using JavaScript rather than validation controls.

- User can prevent validation from being performed when a button control is clicked by setting the button control's **CausesValidation** property to **false**. This property is normally set to false for a Cancel or Clear button to prevent validation from being performed when the button is clicked.
- Validation control performs automatic client-side *and* server-side validation. If the corresponding control is empty, doesn't contain the correct data type, or doesn't adhere to the specified rules, the validator will prevent the page from being posted back altogether.
- **The Difference between Error Message and Text:** Text will be the message that appears at the position of the Validator on the form when validation fails. Error Message appears in the ValidationSummary on the form. If the Text property is left blank, the value of Error Message will be used as the text to display.

Crossword: Unit-5 **Estimated Time: 10 mins.**

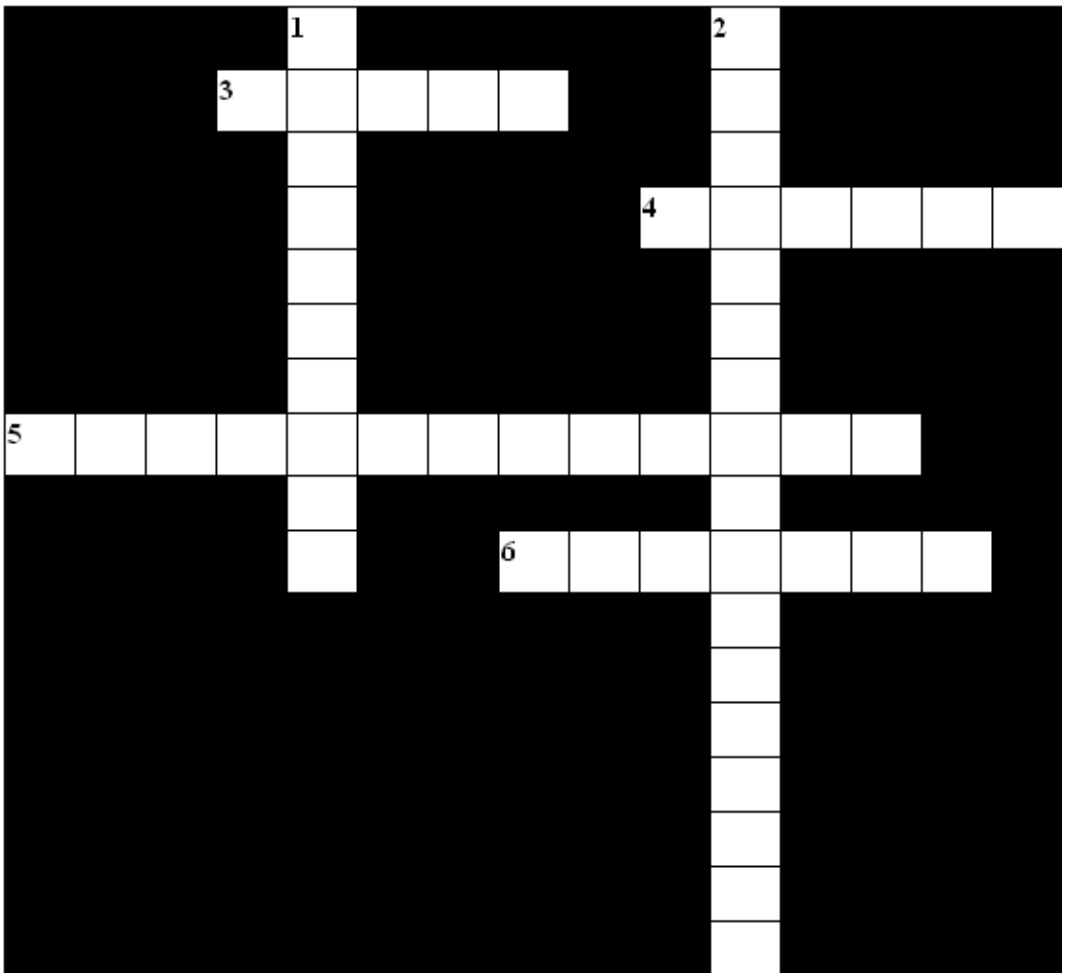


Fig. 5-1

Across:

3. One of the validator, which tests whether an input value falls within a given range or not.(5)

4. A validator control, which calls a user-defined function (custom function) to perform validations that the other standard validators can't handle.(6)

5. One of the type of validator, Which ensures that the user does not skip an entry is called as.(13)

6. Which control validator is used to compare the value of one control to another.(7)

Down:

1. A set of rules that is applied to the data you collect and records valid data is called as.(10)

2. A type of validator control, which confirms that the entry matches a pattern defined by a particular expression.(17)

6.0 Creating User Controls

Topics

- 6.1 User Controls
- 6.2 Custom Controls
- 6.3 Crossword

Topic: User Controls

Estimated Time: 30 mins.

Objectives :

At the end of this module participant should understand:

- Importance of the User Control.

Presentation :

- User controls are custom, reusable controls, and they use the same techniques that are employed by HTML and Web server controls.
- User Controls offer an easy way to partition and reuse common user interfaces across ASP.NET Web applications.
- User Controls are good choice when user needs to build and reuse site headers, footers, and navigational aids.
- Creation is similar to the way Web Forms are created.
- An ASP.NET user control is a group of one or more server controls or static HTML elements that encapsulate a piece of functionality.
- A user control could simply be an extension of the functionality of an existing server control(s). Or, it could consist of several elements that work and interact together to get a job done.
- ***.ascx*** extension is used to indicate Web User Controls.
- User controls are included in a Web Forms page using a **Register** directive:
  <%@ Register Src="timezone.ascx" TagName="timezone" TagPrefix="uc1" %>
  - The **TagPrefix** determines a unique namespace for the user control.
  - The **TagName** is the unique name for the user control.

Scenario :

Steve owner of Johnson and Johnson Ltd. wants to show different timezones on the company's website. Steve also wants to provide the facility for adding more timezones for different cities in future and it should not affect the website. So the developer dealing with this website uses User Control.

Demonstration/Code Snippet :

Page containing web user control is displayed as shown below

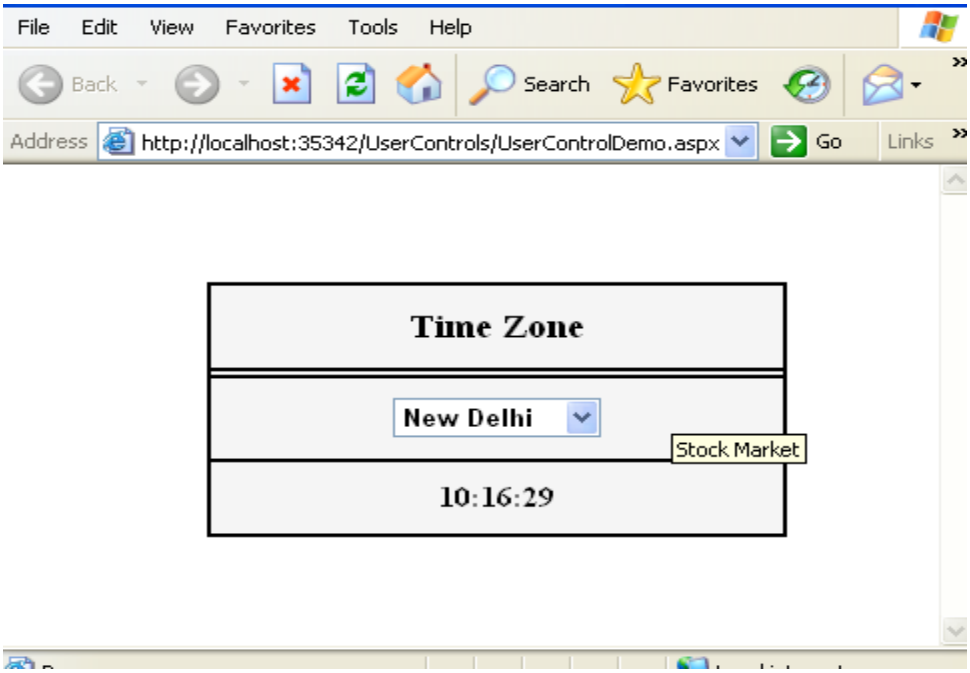


Figure 6.1-1: UserControlDemo.aspx showing user control.

Step 1: Create a new Web application project called User Control Demo using visual studio 2005. Create a new Web User Control form in the same project called TimeZone.ascx. In the Solution Explorer, right click on the Web Application (User Control Demo) and select Add New Item/Web User Control.

Step 2: Insert HTML Table from toolbox in TimeZone.ascx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblTimeZone	Text: Time Zone
asp:dropdownlist	ddlLocation	AutoPostBack: True
		Items/Add/Text: New Delhi Add/Text: Sydney Add/Text: Singapore Add/Text: London
asp:label	lbltime	-

Step 3: Write the event handler for Page Load of TimeZone.ascx.

```
protected void Page_Load(object sender, EventArgs e)
{

TimeSpan ts;
switch (ddlLocation.SelectedValue)
{
            case "New Delhi":
                 lbltime.Text =
                 System.DateTime.Now.TimeOfDay.ToString()
                 .Substring(0, 8);
                 break;
            case "Sydney":
                 ts = new TimeSpan(4, 30, 0);
                 lbltime.Text =
                 System.DateTime.Now.TimeOfDay.Add(ts).ToString()
                 .Substring(0, 8);
                 break;
            case "Singapore":
                 ts = new TimeSpan(2, 30, 0);
                 lbltime.Text =
                 System.DateTime.Now.TimeOfDay.Add(ts).ToString()
                 .Substring(0, 8);
                  break;
            case "London":
                 ts = new TimeSpan(4, 30, 0);
                 lbltime.Text =
                 System.DateTime.Now.TimeOfDay.Add(ts).ToString()
                 .Substring(0, 8);
                  break;
  }
}
```

Step 4: Create a new Web form in the same project called UserControlDemo.aspx. In the Solution Explorer, right click on the name of your Web Application (User Control Demo) and select Add New Item/Web Form.

Step 5: Drag and drop TimeZone.ascx onto UserControlDemo.aspx.

Context :

- Using UserControls for breaking down a large application into smaller, more manageable chunks.
- Importance of UserControls in encapsulating a piece of functionality.
- Modification can be done easily.
- To reduce redundancy and maintenance problems.

Practice Session/s :

Create a user control that contains two textboxes for accepting the user's name and email address. Insert this user control in an ASP.NET page. The ASP.NET page also contains a label control and a submit button. When the user clicks the submit button, the message "Thank You,"&<username>&"for registering. You will receive our news letter at "&<email address> must be shown as the text of the label control.

Check List :

- Importance of UserControl in an application.
- Understanding different properties of UserControl.
- Use to divide the parts of the pages which are being used on many pages.
- User controls are substantially easier to create than custom controls, because one can reuse existing controls.

Common Error/s :

- Debugging ***.ascx*** file.
- Running UserControls without dragging it on a page.

Exception/s :

- Can define custom exceptions as per usage of the UserControl.

Lesson/s Learnt :

- ☑ Understand the importance of UserControls.
- ☑ Using UserControls in applications.
- ☑ User Controls are easy to create.

Best Practice/s :

- Always use web user control for encouraging modularity in the web application.

Topic: Custom Controls

Estimated Time: 30 mins.

Objectives :

At the end of this module participant should understand:

- Importance of custom controls

Presentation :

- Custom controls are compiled code components that execute on the server, expose the object model, and render markup text, such as HTML or XML.
- It can be used by more than one application.
- Distributed easily and without problems associated with redundancy and maintenance.
- Creating custom control means writing from scratch which requires a good understanding of the control's life cycle and the order in which events execute.
- More suited when an application requires dynamic content to be displayed for example, for a data bound table control with dynamic rows.

Scenario :

Administrator Steve of Texas wants to have same Login page for all websites of company. In order to reduce time Steve decided to create one login page and used that page in all the websites. For this Steve decides to make custom control named Login so that Steve can drag and drop this control from toolbox directly.

Demonstration/Code Snippet :

CustomControlDemo.aspx showing use of Custom control.

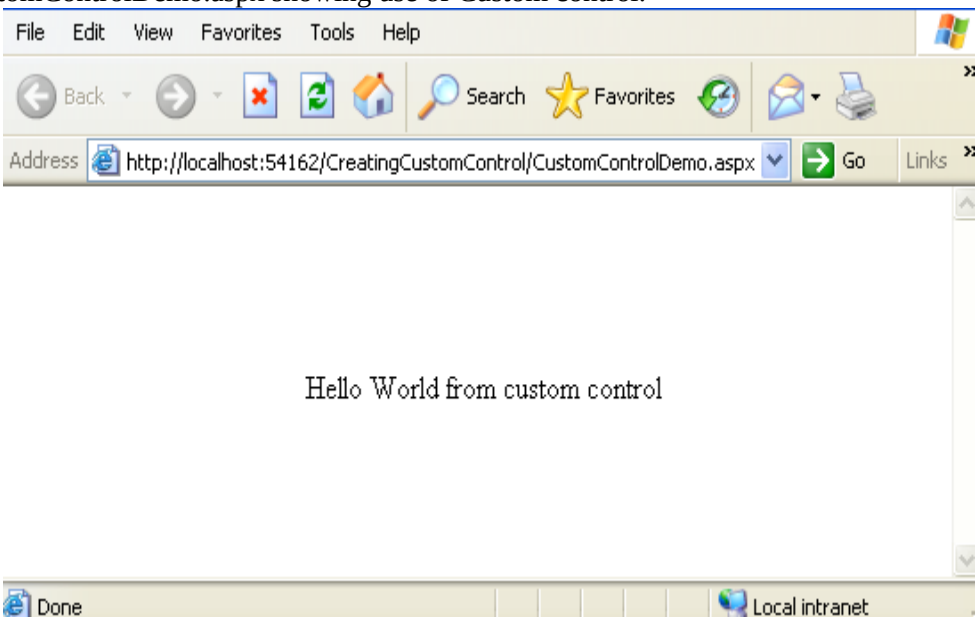


Figure 6.1-1: CustomControlDemo.aspx showing use of CustomControls.

Step 1: Create a new ClassLibrary project called CustomControls using visual studio 2005.

Step 2: Include the reference of the System. Web namespace in the references section. In the Solution Explorer, right click on the name of your Project (CustomControls) and select Add Reference.

Step 3: Check whether the following namespaces are included in SimpleCustomControl.cs file.

```
using System.Collections.Generic;
using System.Text;
using System.Collections;
using System .ComponentModel ;
using System .Data ;
using System .Web;
using System .Web.SessionState ;
using System .Web .UI ;
using System .Web .UI .WebControls ;
```

Step 4: Inherit the SimpleCustomControl class with the Control base class.

public class SimpleServerControl: Control

Step 5: Override the Render method to write the output to the output stream.

```
namespace CustomControls
{
    public class SimpleCustomControl: Control
    {
        protected override void Render(HtmlTextWriter writer)
        {
            writer.Write("Hello World from custom control");
        }
    }
}
```

Step 6: Compile the class library project. It will generate the DLL output.

Step 7: Create a new Web application project called CreatingCustomControl using visual studio 2005. Rename the default .aspx Web Form to CustomControlDemo.aspx.

Step 8: Add a reference to the class library in the references section of the ASP.NET project. In the Solution Explorer, right click on the name of your Application (CreatingCustomControl) and select Add Reference.

Step 9: Register the custom control on the Web Forms page by writing the following code in source file below @page directive.

```
<%@ Register TagPrefix ="CC" Namespace ="CustomControls" Assembly ="CustomControls"%>
```

Step 10: To instantiate or use the custom control on the Web Forms page, add the following line of code in the <form> tags of source file.

```
<form id="form1" method ="post" runat="server">
 <CC:SimpleCustomControl ID="Class1" runat="server">
 </CC:SimpleCustomControl><%--In this code, SimpleCustomControl
      is the control class name inside the class library.--%>
 </form>
```

Step 11: Run the Web Forms page, and output will be seen from the custom control.

Context :

- Using Custom Controls for breaking down a large application into smaller, more manageable chunks.
- Importance of Custom Controls in encapsulating a piece of functionality.
- Modification can be done easily.
- To reduce redundancy and maintenance problems.
- Use custom control for dynamic content to be displayed in application .

Practice Session/s :

Create a custom control that contains two textboxes for accepting the user's name and email address. Insert this user control in an ASP.NET page. The ASP.NET page also contains a label control and a submit button. When the user clicks the submit button, the message "Thank You,"&<username>&"for registering. You will receive our news letter at "&<email address> must be shown as the text of the label control.

Check List :

- Importance of UserControl in an application.
- Understanding different properties of UserControl.
- Use to divide the parts of the pages which are being used on many pages.
- User controls are substantially easier to create than custom controls, because you can reuse existing controls.

Common Error/s :

- Debugging ***.ascx*** file.
- Running UserControls without dragging it on a page.
- In using Custom Controls if the control is not registered rightly error creating custom control will occur.

Exception/s :

- If the Web Forms Designer cannot render a Web server control correctly, it displays a grey box with the text "Error Creating Control." This often means that the ASP.NET syntax of the control is incorrect - for example, if the runat="server" attribute is missing in a Web server control element, you will see this error. Point to the information icon (); a ToolTip is displayed with details about the error.

Lesson/s Learnt :

- ☑ Understand the importance of UserControls.
- ☑ Using UserControls in applications.
- ☑ User Controls are easy to create.

Best Practice/s :

- Always use custom control for encouraging modularity in the web application.

Crossword: Unit-6 **Estimated Time: 10 mins.**

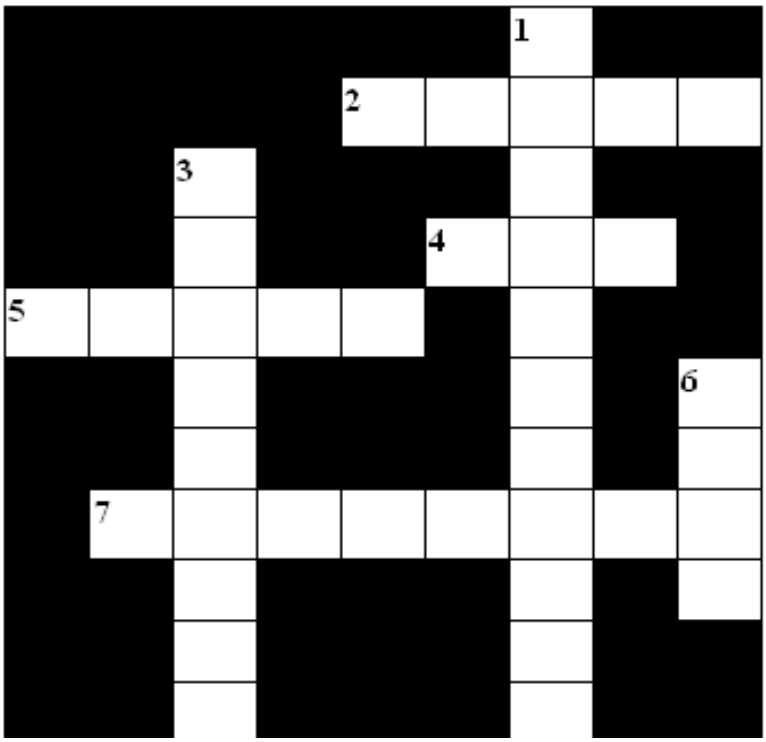


Fig. 6-1

Across:

2. What extension is used to indicate such user controls.(5)

4. Which attribute defines the virtual path to the user control file that is included.(3)

5. Which control is an easy way to implement Forms authentication without having to write any code.(5)

7. User controls are included in a Web Forms page using which directive.(8)

Down:

1. This tool can be very useful for breaking down a large application into smaller, more manageable chunks, this tool in ASP.NET is a group of one or more server controls or static HTML elements that encapsulate a piece of functionality.(11)

3. In register directive which Tag determines a unique namespace for the user control.(9)

6. In the body of the Web page, in which element the user control is declare d.(4)

7.0 Accessing Relational Data using Visual Studio.NET

Topics

- 7.1 ADO.Net Overview
- 7.2 Accessing Data using ADO.Net
- 7.3 Stored Procedures
- 7.4 Reading and Writing XML Data
- 7.5 Crossword

Topic: ADO.NET Overview

Estimated Time: 20 mins.

Objectives : This module will help participants to understand

- ADO .Net Architecture
- Content components and Managed-provider components
- Connected Environment and Disconnected Environment

Presentation :

- ADO.NET is an integral part of the .NET Compact Framework, providing access to relational data, XML documents, and application data.
- ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results.
- ADO.NET is a set of classes that is used to connect to and manipulate data sources.
- Data-sharing consumer applications can use ADO.NET to connect to data sources and retrieve, handle and update the data that they contain.
- ADO.NET separates data access from data manipulation into discrete components that can be used separately.
- ADO .Net Architecture uses a multilayered architecture that revolves around a few key concepts, such as Connection, Command, and DataSet objects.

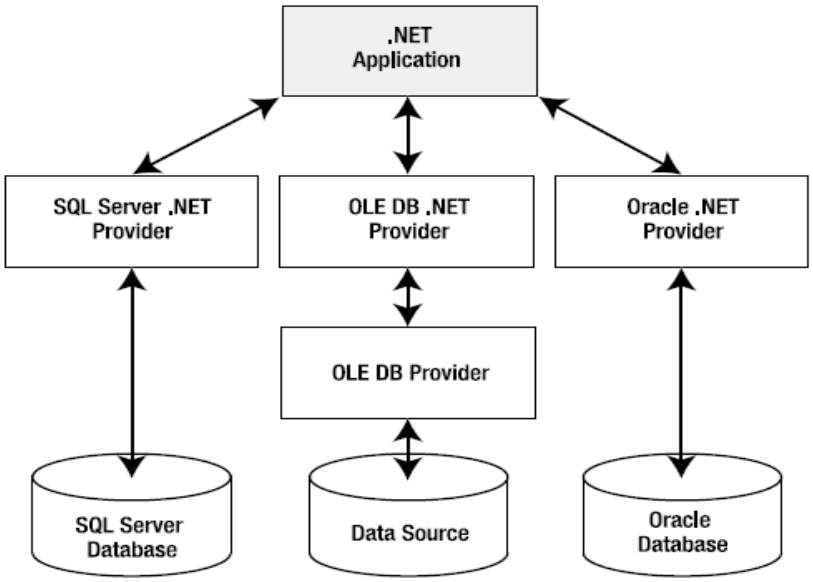


Figure 7.1-1: ADO .Net Architecture.

- In ADO.NET, functionality is split into two key class groups
  - Content components
  - Managed-provider components

- **Content Components**: The content components essentially hold actual data and include.

Class	Description
DataSet	The DataSet is a local buffer of tables or a collection of disconnected record sets
DataTable	DataTable is used to contain the data in tabular form using rows and columns. DataRow Represents a single record or row in DataTable
DataRow	Represents a single record or row in DataTable
DataColumn	Represents a column or field of DataTable
DataRelation	Represents the relationship between different tables in a data set.
Constraint	Represents the constraints or limitations that apply to a particular field or column.

- **Managed-provider components**: They actually talk to the database to assist in data retrievals and updates. Such objects include:

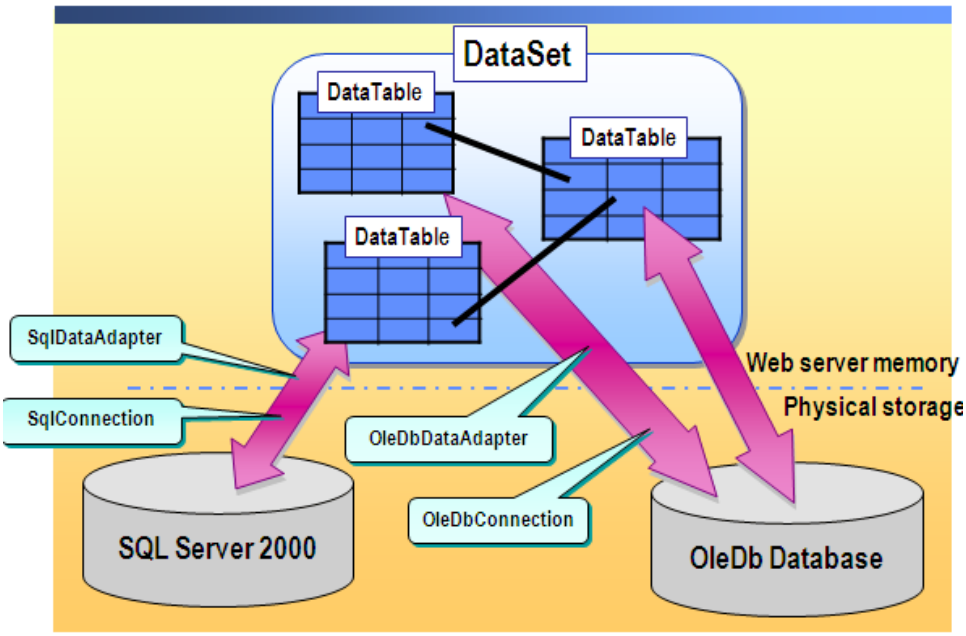
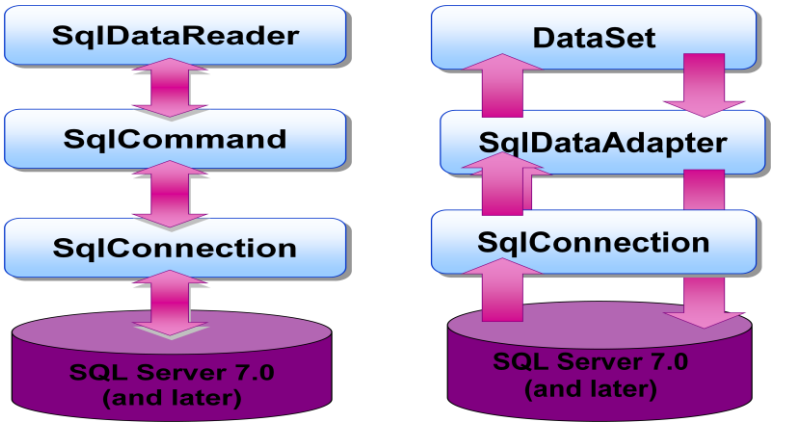


Figure 7.1-2: Connection to Dataset.

Class	Description
Connection	Represents a connection to the database system
Command	Represents SQL query or command to be executed at the database management system
DataAdapter	A class that connects to the database system, fetch the record and fill the dataset. It contains four different commands to perform database operations; Select, Update, Insert, Delete.
DataReader	A stream that reads data from the database in connected design
Parameter	Represents a parameter to a stored procedure

- **Connected Environment**: A connected environment is one in which users are constantly connected to a data source.



Connected Environment **Disconnected Environment**

- **Disconnected Environment**: In a disconnected environment, a subset of data from a central data store can be copied and modified independently, and the changes merged back into the central data store.

Context :

- ADO.NET promotes the use of disconnected datasets, with automatic connection pooling bundled as part of the package.
- All data in ADO.NET is transported in XML format, meaning it's simply a structured text document that can be read by anyone on any platform.

Practice Session/s :

- Identify ADO .Net Architecture Components
- ADO .Net Functionality
- Identify the Connection Providers
- Identify Content Components

Check List :

- Types of provider
- Types of Content Components
- Connected and Disconnected Environment

Common Error/s :

- Check whether Connection string is defined properly.
- Always use try and catch to any part of the code that is having database connectivity.
- Check if System.Data.SqlClient namespace has been used.

Exception/s :

- If Connection String is not defined properly.
- If Connection String is define properly but server is not responding.
- Might get SQLException.

Lesson/s Learnt :

- ☑ Basics of ADO.Net
- ☑ Efficiently managing data from data source.

Best Practice/s :

- Always use ADO .Net to store data in the database on the server.

Topic: Accessing Data using ADO.NET

Estimated Time: 30 mins.

Objectives : At the end of the activity, the participant should understand

- Connecting to the DataBase.
- Assessing Data with DataSets.
- Assessing Data with DataReader.

Presentation :

- ADO.NET is a set of classes that expose data access services.
- ADO.NET is an evolutionary improvement to Microsoft ActiveX Data Objects (ADO).
- ADO.NET is an integral part of the .NET Compact Framework, providing access to relational data, XML documents, and application data.
- The .NET Framework data providers of ADO.NET serve as a bridge between an application (presentation layer) and a data source (data layer) that execute commands as well as retrieve data by using a **DataReader** or a **DataAdapter**.

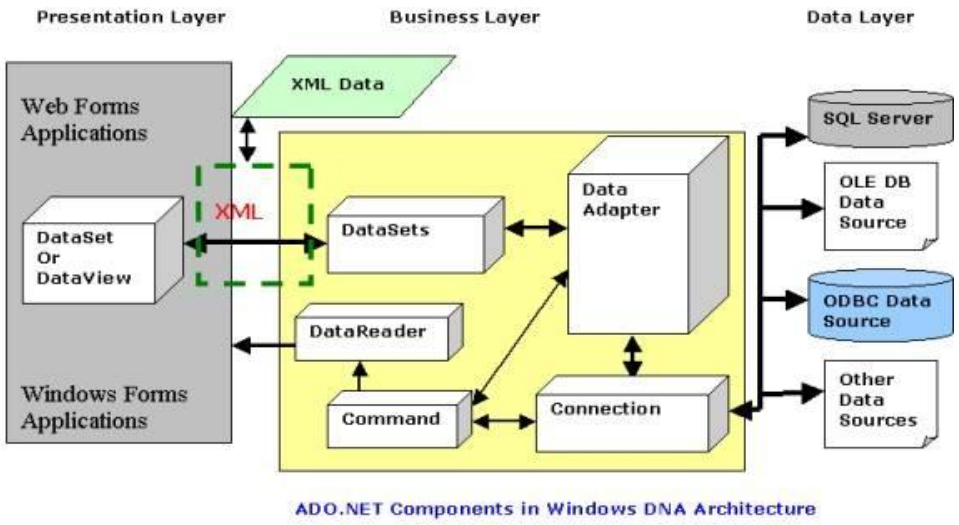


Figure 7.2-1:ADO.Net Components.

Connecting to DataBase:

- The Connection object creates a link to the DataSource. This object needs the necessary information to discover the DataSource and to log in to it properly like Datasource, Database, and Connection String.

- Each .NET Framework data provider included with the .NET Framework has a **Connection** object:

DataProvider	Object
OLE DB	OleDbConnection object
SQL Server	SqlConnection object
ODBC	OdbcConnection object
Oracle	OracleConnection object

DataSets:

- Datasets are containers — caches — in which you can store data to use in your application. Datasets are a fundamental part of the ADO.NET architecture, providing both high-performance data access as well as scalability.
- Datasets store data in a disconnected cache. The structure of a dataset is similar to that of a relational database; it exposes a hierarchical object model of tables, rows, and columns. In addition, it contains constraints and relationships defined for the dataset.

DataReader:

- The DataReader object is a simple forward-only and read-only cursor. It requires a live connection with the data source and provides a very efficient way of looping and consuming all or part of the result set.
- Results are returned as the query executes, and are stored in the network buffer on the client until one requests them using the **Read** method of the **DataReader**.
- DataReader object cannot be directly instantiated. Instead, it must be called by ExecuteReader method of the Command object.
- Using the **DataReader** can increase application performance both by retrieving data as soon as it is available, and (by default) storing only one row at a time in memory, reducing system overhead.
- Be sure to close the connection when done using data reader. Otherwise, the connection stays alive until it is explicitly closed.

Scenario :

Mr. Bill Smith, principal of St. Mark's School, wants percentage details of all the students who gave entrance exam in 12th class in current year and scored above 70%. St. Mark's School is already having all data stored in their database. So Mr. Bill Smith asks Robin to develop this application. Robin uses DataSets and DataReader to develop this application.

Demonstration/Code Snippet :

Using DataSet:

Stud_info.aspx page showing students percentage and student having percentage > 70.

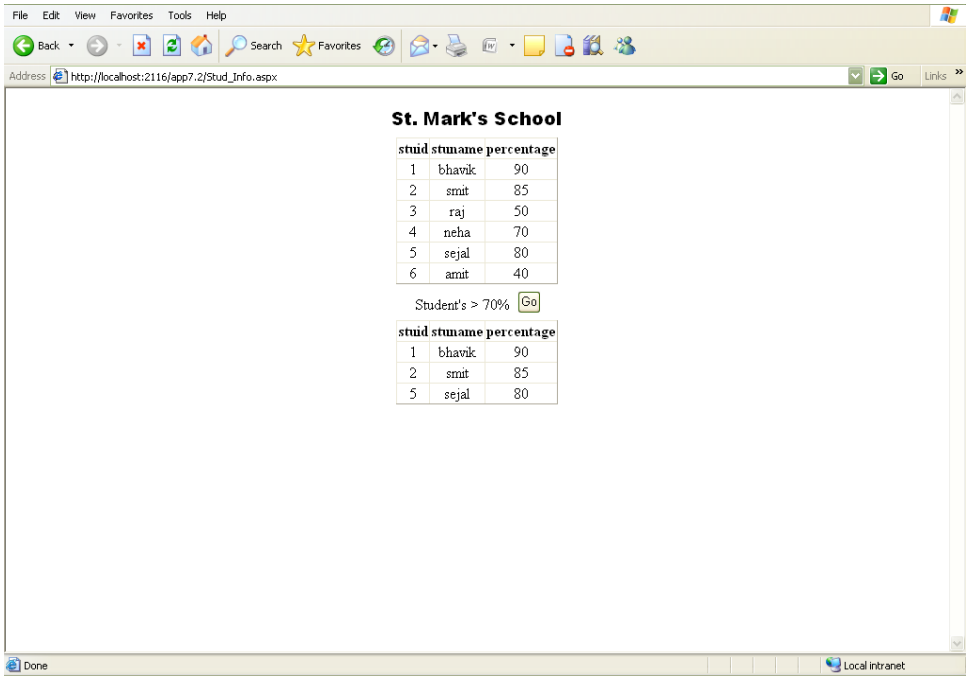


Figure 7.2-2: Stud_Info.aspx page showing percentage of student.

Step 1: Create a new Web application project called ADO.Net_Sample using visual studio 2005. Rename Default.aspx to Stud_Info.aspx

Step 2: Insert HTML Table from toolbox in Stud_Info.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:St. Mark's School
asp:label	lblStuinfo	Text:Student's > 70%
asp:button	btnGo	Text:Go
asp:Gridview	gdvAllstud	-
asp:Gridview	gdvSelstud	-

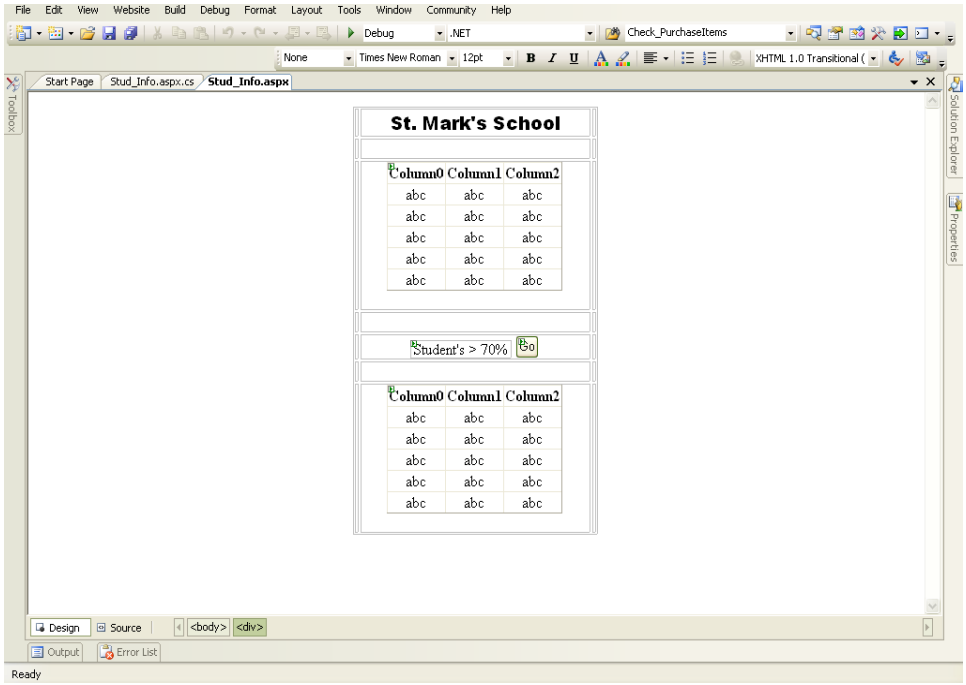


Figure 7.2-3: Design View of Stud_Info.aspx.

Step 3: Write the event handler for the Page Load by double clicking the webform.

```
Using System.Data.Sqlclient;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox


protected void Page_Load(object sender, EventArgs e)
   {
if (!IsPostBack) {

String connect =
"server=hstslc011;uid=sa;password=satyam;database=northwind";

       SqlConnection con = new SqlConnection(connect);
       DataSet ds;
       SqlDataAdapter adap;
       string str;

       str = "select * from student_12";
      try {

            con.Open ();
            adap = new SqlDataAdapter(str, con);
            ds = new DataSet();
       adap.Fill(ds);
       gdvAllstud.DataSource = ds.Tables[0];
```

```
        gdvAllstud.DataBind();
      }

  catch (Exception ex)
  {
   MessageBox(ex);
  }
  finally
  {
 con.Close();
  }
 }
}
```

Step 4: Write the event handler for Go button by double clicking it.

```
Using System.Data.Sqlclient;
Using  System.Windows.Forms;//Add  system.windows.forms  reference  for
messagebox

 Protected void  btnGo_Click(object sender, EventArgs e)
   {
     SqlConnection con = new SqlConnection(connect);
     DataSet ds;
     SqlDataAdapter adap;
     string str;

     str = "select * from student_12 where percentage > 70";
     try {

             con.Open ();
             adap = new SqlDataAdapter(str, con);
             ds = new DataSet();
             adap.Fill(ds);
             gdvSelstud.DataSource = ds.Tables[0];
             gdvSelstud.DataBind();
          }
    catch (Exception ex)
          {
            MessageBox(ex);
          }
   finally
         {
           con.Close();
         }
  }
```

Step 5: Run the application. On Go button click, Student's having percentage > 70 will be displayed.

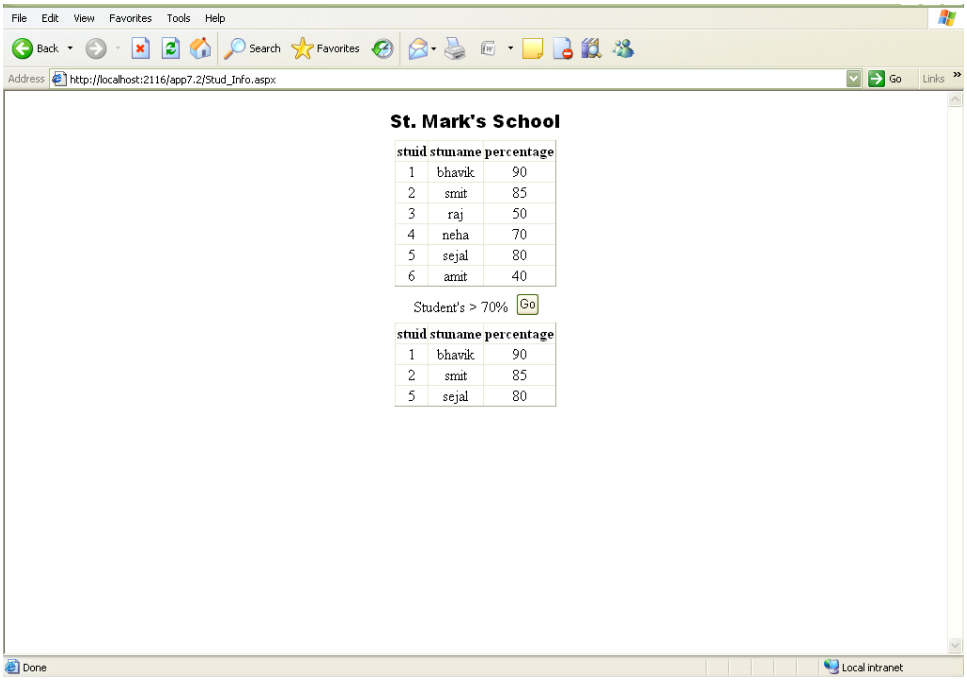


Figure 7.2-4: Final Output of application.

Using Data Reader:

Design view of application page develop by Robin using DataReader has been viewed

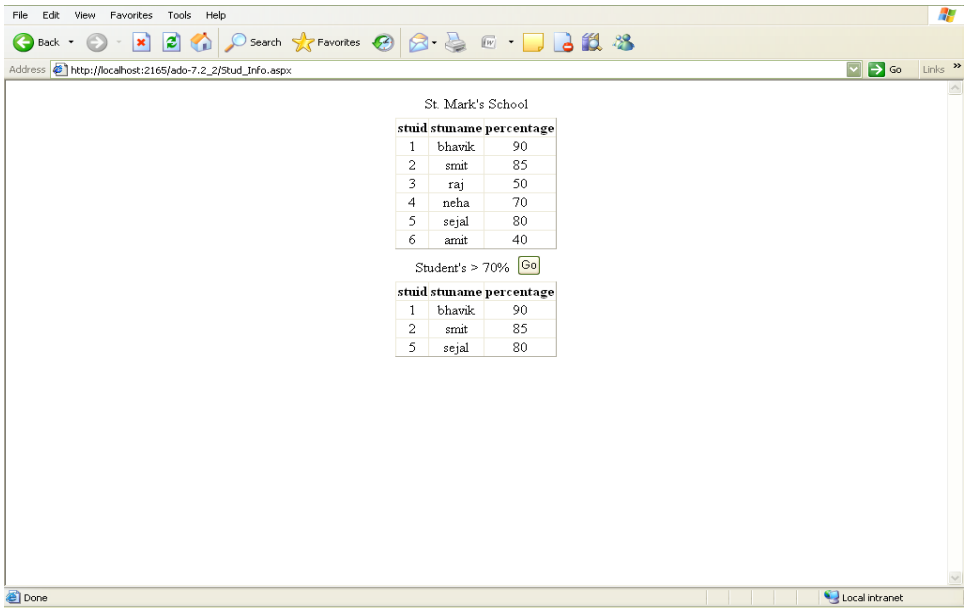


Figure 7.2-5: Stud_info.aspx. showing student percentage .

Step 1: Create a new Web form in the same project called Stud_info1.aspx. In the Solution Explorer, right click on the name of your Web Application (ADO.Net_Sample) and select Add New Item/Web Form.

Step 2: Insert HTML Table from toolbox in StudentInfo.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:St. Mark's School
asp:label	lblStuinfo	Text:Student's > 70%
asp:button	btnGo	Text:Go
asp:Gridview	gdvAllstud	-
asp:Gridview	gdvSelstud	-

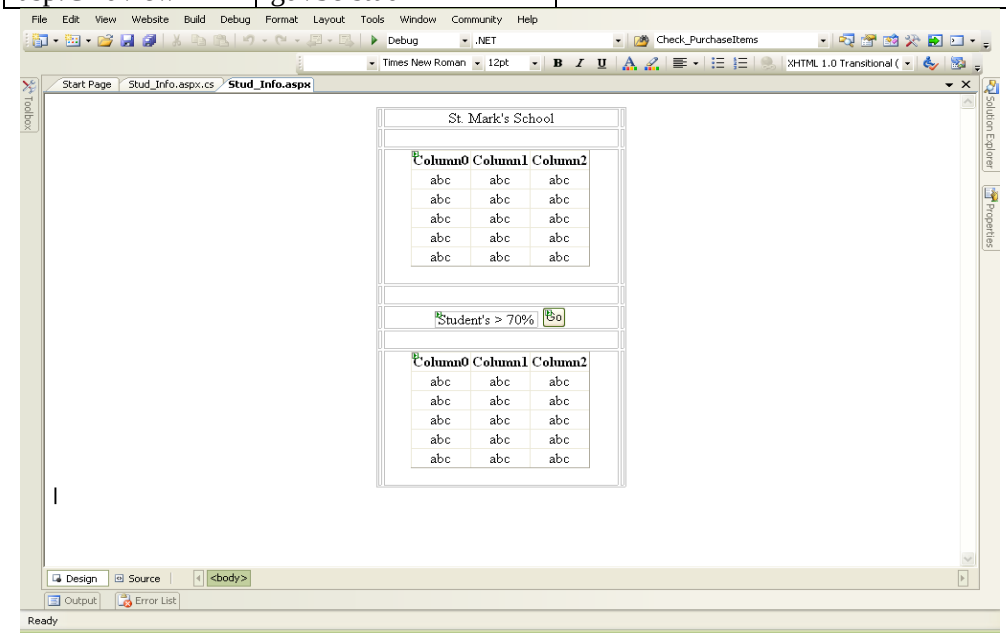


Figure 7.2-6: Design view of Stud_Info1.aspx.

Step 3: Write the event handler for the Page Load by double clicking it.

```
Using System.Data.Sqlclient;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox

    protected void Page_Load(object sender, EventArgs e)
    {
    string connect =
    "server=hstslc011;uid=sa;password=satyam;database=northwind";
            SqlCommand mycommand;
            SqlDataReader myreader;
            SqlConnection con;
            con = new SqlConnection(connect);
            try {
```

```
                mycommand = new SqlCommand();

                mycommand.CommandText = "select * from student_12";
                mycommand.CommandType = CommandType.Text;
                mycommand.Connection = con;
                mycommand.Connection.Open();
                myreader =
            mycommand.ExecuteReader(CommandBehavior.CloseConnection);
                gdvAllstud.DataSource = myreader;
                gdvAllstud.DataBind();
                mycommand.Dispose();
                }

    catch (Exception ex)

{
 MessageBox(ex);
}
 finally
{
con.Dispose()
}
  }
```

Step 4: Write the event handler for Go button by double clicking it.

```
Using System.Data.Sqlclient;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox

      protected void  btnGo_Click(object sender, EventArgs e)
      {
        SqlCommand mycommand;
        SqlDataReader myreader;
        SqlConnection con;
        con = new SqlConnection(connect);
  try {
        mycommand = new SqlCommand();
        mycommand.CommandText = "select * from student_12 where
        percentage > 70";
        mycommand.CommandType = CommandType.Text;
        mycommand.Connection = con;

        mycommand.Connection.Open();
        myreader =
        mycommand.ExecuteReader(CommandBehavior.CloseConnection);

        gdvSelstud.DataSource = myreader;
        gdvSelstud.DataBind();

        mycommand.Dispose();
       }
```

```
    catch (Exception ex)
   {
    MessageBox(ex);
   }
   finally
  {
    con.Dispose()
  }
 }
}
```

Step 5: Run the application, on Go button click, Student's having percentage > 70 will be displayed.

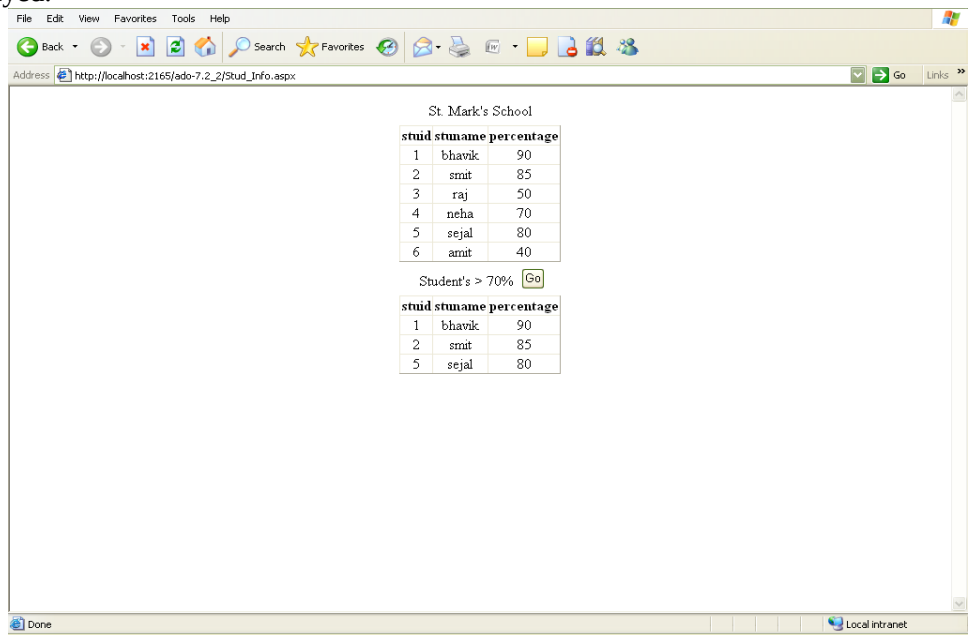


Figure 7.2-7: Final output of application.

Tables Used: student_12
Database: Northwind

	stuid	stuname	percentage
1	1	bhavik	90
2	2	smit	85
3	3	raj	50
4	4	neha	70
5	5	sejal	80
6	6	amit	40

Figure 7.2-8: Table contents of student_12 table.

Context :

- Use DataReader for connected architecture.
- Use DataAdapter for disconnected architecture.

Practice Session/s :

Create a student homepage where students mark in three subjects is viewed in grid view. Display marks > 70 in green, marks < 50 in red and rest in blue. Implement using data reader, data adapter and grid view.

Check List :

- Connecting to database.
- Importance of datareader and dataset.

Common Error/s :

- Without opening connection trying to use datareader leads to an exception.
- Mentioning connection string wrongly.
- Filling dataset without adapter.
- Use try...catch for any code that is having DataBase Connectivity.

Exception/s :

- Might get SQLException.

Lesson/s Learnt :

- ☑ Database connection.
- ☑ Use of datareader and dataadapter.

Best Practice/s :

- Always use dataadapter instead of data reader to reduce server load.

Multiple Active Result Sets (MARS) enables us to reuse a single open connection for multiple accesses to the database, even if the connection is currently processing a result set. This feature becomes even more powerful when it is used in conjunction with the asynchronous command processing.

Topic: Stored Procedures

Estimated Time: 20 mins.

Objectives : At the end of the activity participant should understand

- Concept of Stored Procedure
- Calling a Stored Procedure

Presentation :

- A stored procedure is an executable object stored in a database.
- A stored procedure can have any number of (including zero) input or output parameters and can pass a return value. In OLEDB, parameters to a stored procedure can be passed by:
  - Hard-coding the data value.
  - Using a parameter marker (?) to specify parameters, bind a program variable to the parameter marker, and then place the data value in the program variable.
- Stored procedures are used for database INSERT, UPDATE, and DELETE operations and for retrieving single values and result sets.
- Stored procedures give performance benefits and restrict data access to the predefined interfaces that they expose.
- The OLE DB provider for SQL Server (SQLOLEDB) supports the following mechanisms that stored procedures use to return data:
  - Every SELECT statement in the procedure generates a result set.
  - The procedure can return data through output parameters.
  - The procedure can have an integer return code.

Scenario :

Administrator John of Johnson and Johnson Company wants to create a web page which will give the details of all signed in users on the website. And also wants good performance for the site by not keeping the code in the application logic. So Mr. Williams, developer of website uses stored procedure.

Demonstration/Code Snippet :

StoredProcedure.aspx showing retrieval of data using stored procedure.

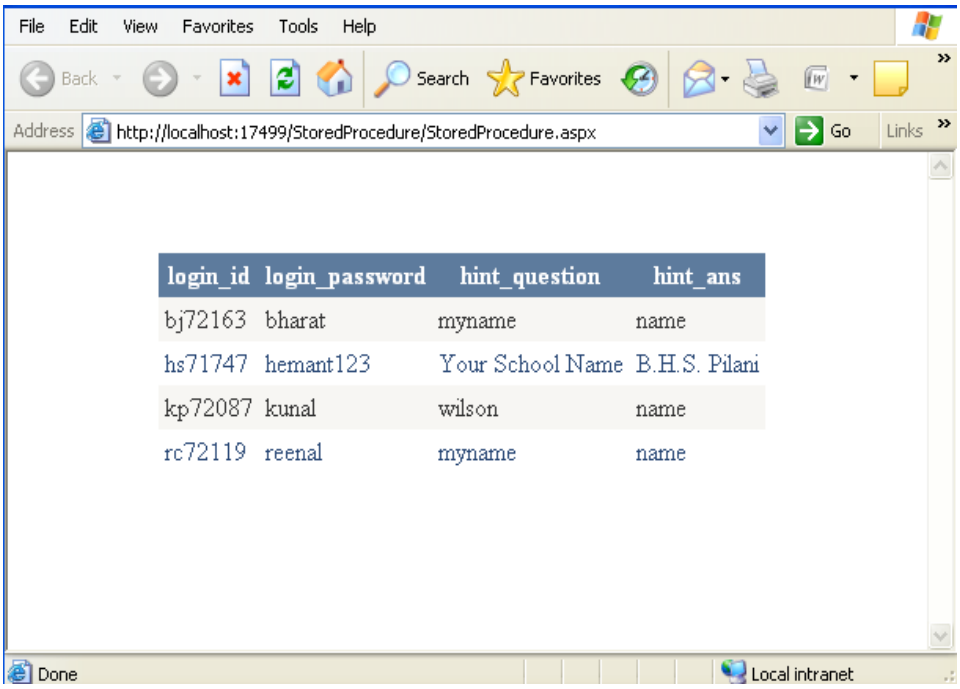


Figure 7.3-1: StoredProcedure.aspx showing retrieval of data using stored procedure.

Step 1: Create a new Web application project called StoredProcedure using visual studio 2005. Rename Default.aspx to StoredProcedure.aspx

Step 2: Drag and drop the server controls in StoredProcedure.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:Gridview	gdvGetProcedure	-

Step 3: Write in the Page Load event handler of StoredProcedure.aspx by double clicking webform.

```
Using System.Data.Sqlclient;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox


protected void Page_Load(object sender, EventArgs e)
{

```

```
if (!IsPostBack) {

String Connectionstr =
"server=hstslc011;database=Northwind;uid=sa;pwd=satyam"
SqlConnection con= new SqlConnection(Connectionstr);
//Establish connection with the database using stored procedures
try {
   con.Open ();
  SqlCommand cmd = new SqlCommand("GetProcedure",con);
//GetProcedure is the name of the procedure created in sqlserver
  cmd.CommandType = CommandType.StoredProcedure;
//To run a stored procedure, we need to set the command type
property
  SqlDataAdapter adap = new SqlDataAdapter(cmd);
  DataSet ds = new DataSet();
  adap.Fill(ds);
  GdvGetProcedure.DataSource = ds.Tables[0];
  GdvGetProcedure.DataBind();
  }
  catch (Exception ex)
  {
  MessageBox(ex);
  }
  finally
  {
  con.Close();
  }
}
```

To call a stored procedure, user first need to identify the stored procedure, create a **DataAdapter** object, and point the **DataAdapter** object to the database connection. Set the **CommandText** property to the name of the identified stored procedure, and finally, set the **CommandType** property to **CommandType.StoredProcedure**.

Tables used: login
Database Used: northwind

	login_id	login_password	hint_question	hint_ans
▶	bj72163	bharat	myname	name
	hs71747	hemant123	Your School Name	B.H.S. Pilani
	kp72087	kunal	wilson	name
	rc72119	reenal	myname	name

Figure 7.3-2: Login table

Procedures:

```
create procedure GetProcedure
as
begin
```

```
Select * from login
end
```

Context :

- Stored procedures allow a lot more flexibility offering capabilities such as conditional logic.
- SQL Server pre-compiles stored procedures such that they execute optimally.
- To abstract complex designs from Client developers.

Practice Session/s :

Create a stored procedure for successful login. If password and id does not match then return login failed message from stored procedure else return login successful message. Use login table in NorthWind DataBase.

Check List :

- Stored procedures are stored within the DBMS, and bandwidth and execution time are reduced.
- It provides more security and moreover the code is reusable.

Common Error/s :

- Connection not opened before executing the query.
- Incorrect information being provided in the connection string.
- If using **ExecuteNonQuery** command not specifying input and output parameters properly.

Exception/s :

- Exception might occur if the stored procedure was not found
- Argument exception might occur if the details specified while passing parameters to the data source are not matching with its columns.

Lesson/s Learnt :

☑ Understanding the basics of Stored Procedure.
☑ Calling Stored Procedure from front end.

Best Practice/s :

- To use stored procedures in programming so that the server suffers less load and also to restrict the expose of database design.

When using parameters in an OLE DB database, the order of the parameters in the **Parameters** collection must match the order of the parameters that are defined in the stored procedure.

Topic: Reading and Writing XML data

Estimated Time: 45 mins.

Objectives : At the end of the activity, the participant should understand

- The architecture of XML
- The objects of XML and Dataset
- The concept behind XML web server controls

Presentation :

- XML is a cross-platform, hardware and software-independent, text-based markup language that enable to store data in a structured format by using meaningful tags.
- In .NET Framework, the support for XML documents includes:
  - XML namespace : In .NET Framework, the System.Xml namespace provides a rich set of classes for processing XML data. The commonly used classes for working with XML data are shown below:
    - ✓ Xml TextReader
    - ✓ Xml TextWriter
    - ✓ Xml Document
    - ✓ Xml DataDocument
    - ✓ XPathDocument
    - ✓ Xml NodeReader
    - ✓ Xsl Transform
  - XML designer: To create and edit XML documents.
  - XML Web Server control: To display the contents of an XML document without formatting the contents.
- XML Document Object Model (DOM) is an in-memory representation of an XML document and represents data as a hierarchy of object nodes.
- The XML Web server control has the following properties:
  - DocumentSource
  - TransformSource
  - Document
  - Transform

Scenario:

Mr. Parker of Xmart wants to store the details of the products available for sale. The details of the products should be displayed in a tabular format in a browser. The details of the product include Product Id, Product Name and Price. Parker wants that information should be retrieved from an XML file as it can be updated in future.

Demonstration/Code Snippet :

The output of the web application is shown:

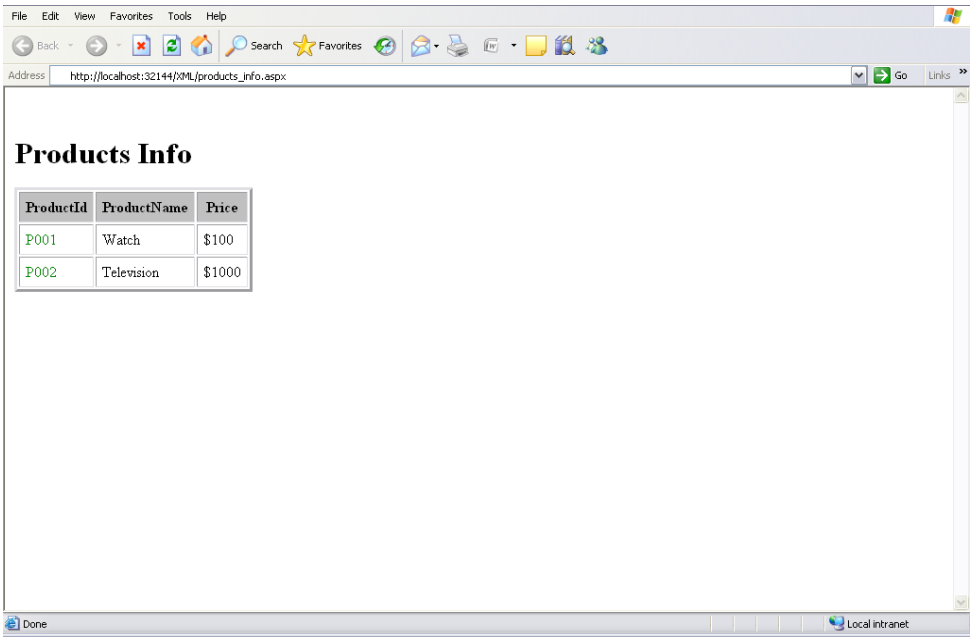


Figure 7.4-1: products_info.aspx showing Products Information.

Step 1: Create a new Web application project using visual studio 2005. Rename Default.aspx Web form to products_info.aspx.

Step 2: Select **Add New Item** option from the project menu and select **XML File** as a template from the dialog box. Specify the name as **Products.xml** and click open to display the **XML designer**.

Step 3: The XML designer automatically generates a line of code. This line of code states the version number of the XML code used. Enter the following lines of code after this line in the products.xml file:

```
<?xml version="1.0" encoding="utf-8" ?>
<Products>
       <Product>
             <ProductId>P001</ProductId>
             <ProductName>Watch</ProductName>
             <Price>$100</Price>
      </Product>
     <Product>
           <ProductId>P002</ProductId>
          <ProductName>Television</ProductName>
          <Price>$1000</Price>
    </Product>
</Products>
```

It is an XML declaration statement that notifies the browser that the document being processed is an XML document. ProductId, ProductName, Price are the elements represented by tags i.e defines the structure of an XML document that will store data about products.

Step 4: Select the Add New Item option from the project menu and select XSLT File as a template from the dialog box. Specify the name as Style_sheet.xsl.

XSLT is an XML-based language that performs transformations of XML documents into arbitrary text-based formats, which may or may not be XML.

Step 5: Add the following lines of code in the <BODY>…..</BODY> tag to display data in tabular format:

```
<BODY>
      <h1>Products Info</h1>
      <TABLE BORDER ="3" CELLSPACING="2" CELLPADDING="6">
        <THEAD ALIGN="CENTER" BGCOLOR="SILVER">
          <TH>ProductId</TH>
          <TH>ProductName</TH>
          <TH>Price</TH>
        </THEAD>
        <TBODY>
          <xsl:for-each select="Products/Product">
          <TR>
            <TD>
              <FONT COLOR="green">
                <xsl:value-of select="ProductId"/>
              </FONT>
            </TD>
            <TD>
              <xsl:value-of select ="ProductName" />
            </TD>
            <TD>
              <xsl:value-of select="Price" />
            </TD>
            </TR>
            </xsl:for-each>
        </TBODY>
      </TABLE>
</BODY>
```

Step 6: Right click the XML server control added to the products_info.aspx and select the properties option to display the property window. Select the DocumentSource property to open the XML File dialog box. Select the Products.xml from the contents and click OK button.

The **Document Source** property provides the input for the transformation from XML into other arbitrary format

Step 7: Select the TransformSource property to open the XSL Transform File dialog box. Select the Style_sheet.xsl from the contents of <project Name> and click OK button.

The **Transform Source** property transforms the contents of XML file into other formats, such as XML or HTML

Step 8: Drag and drop the XML control and change the control property values as indicated in the following table:

Control type	Property values to be modified
asp: Xml	DocumentSource:Products.xml
asp: Xml	TransformSource:Style_sheet.xsl

Step 9: Add an XML control from the Web Forms tab in the Toolbox to the products_info.aspx file.

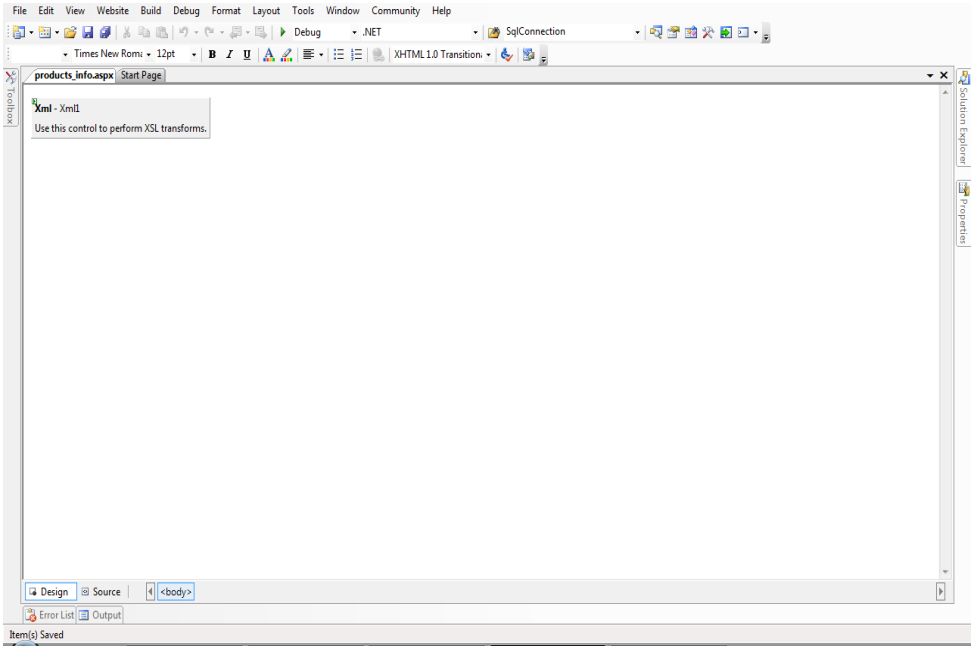


Figure 7.4-2: XML control added to perform Xsl transform.

Step 10: Execute the application by selecting Start from the Debug menu.

Adrotator

Scenario:

Virgin airline is having a website where booking can be done. Different tourism agencies want to give advertisement in the virgin airlines website so that travellers could go to different tourism site for further information. So tourism agencies suggest about the idea of advertisement to virgin airlines. To implement the same in the website, the developers of virgin airlines, uses ad rotator.

Demonstration/Code Snippet :

After Executing adrotator.aspx satyam Advertisement is viewed

Figure 7.4-3: Output after executing adrotator.aspx.

After clicking satyam logo www.satyam.com site will get open.

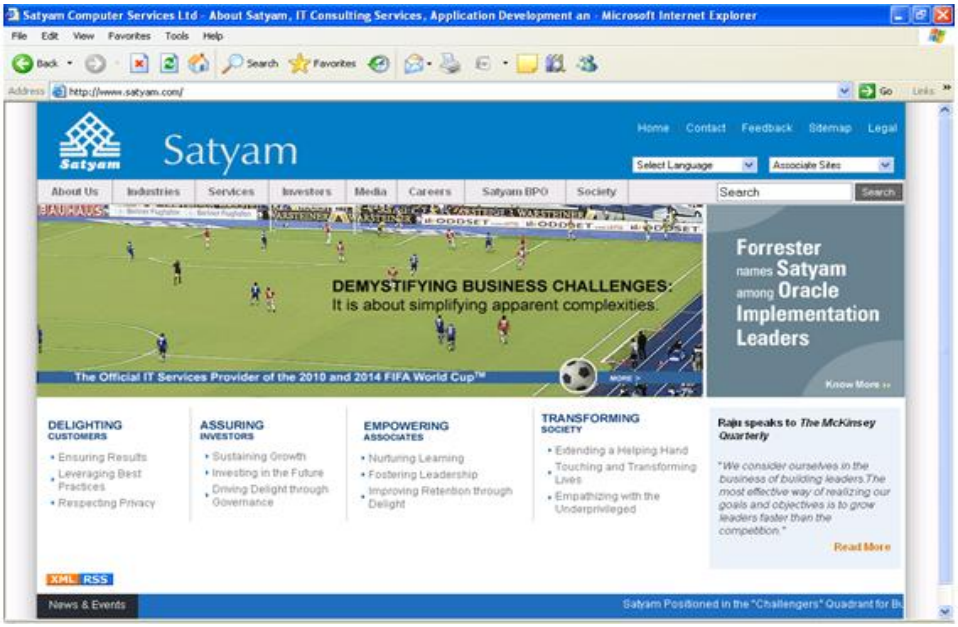


Figure 7.4-4: On Clicking satyam image it takes to satyam website.

Step 1: Create a new Web application project called Ad rotator using visual studio 2005. Rename Default.aspx to adrotator.aspx .

Step 2: Create a new folder name Image. In the Solution Explorer, right click on App_Data select New Folder. Add some images to image folder.

Step 3: Add a new XML File in the same project called Advertisement.xml. In the Solution Explorer, right click on the name of your Web Application (Ad rotator) and select Add New Item/XML File.

Step 4: Add the following in the XML editor to edit Advertisement.xml:

```
<Advertisements>
<Ad>
<!-- The URL for the ad image -->
<ImageUrl>C:\reenal\Adrotator\App_Data\Image\satyam.jpg</ImageUrl>
<!-- The URL the ad redirects the user to -->
<NavigateUrl>http://www.satyam.com/ </NavigateUrl>
<!--<Keyword>Satyam</Keyword>-->
</Ad>

<Ad>
<ImageUrl>C:\reenal\Adrotator\App_Data\Image\image.jpg</ImageUrl>
<NavigateUrl>https://imailhyd.satyam.com/exchange</NavigateUrl>
</Ad>
</Advertisements>
```

Step 5: Drag and drop the server controls onto Adrotator.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:adrotator	adr_add	Advertisement File: Advertisement.xml

Step 6: Execute the application. Refresh the page several times to confirm that the advertisements appear.

Step 7: Click the advertisement, and verify that you are redirected to the appropriate URL.

Context :

- Defines the formatting details for various controls.
- Gives an aesthetic look to the Website.
- Use Xml for updating new things in future and for site navigation.

Practice Session/s :

Create a web application to store the details of the Books available in Library in the XML format. The details should be displayed in tabular format in a browser. The details of the books should include BookId, Title and Author's first and last names.

Check List :

- Use of XML Web Server control to display, load and save XML data.
- Use XSLT file to display data in specific format.
- Use ad rotator for advertisement.

Common Error/s :

- Syntax not properly defined.
- Properties for Xml file and Xsl transform file are not defined properly.

Exception/s :

- Might get XML parser exceptions.

Lesson/s Learnt :

- ☑ XML Web Server control.
- ☑ XML architecture.

Best Practice/s :

- Always use the Xsl transform file to load a .xsl style sheet file and transform Xml file into HTML file.

You can read an XML document or stream it into a DataSet by using the **ReadXML** method of the DataSet and then write a DataSet in XML by using the **WriteXML** method of the DataSet

The **Transform** method can only output to an **XmlReader**, a **TextReader**, or to **XmlWriter** objects.

Crossword: Unit-7 **Estimated Time: 10 mins.**

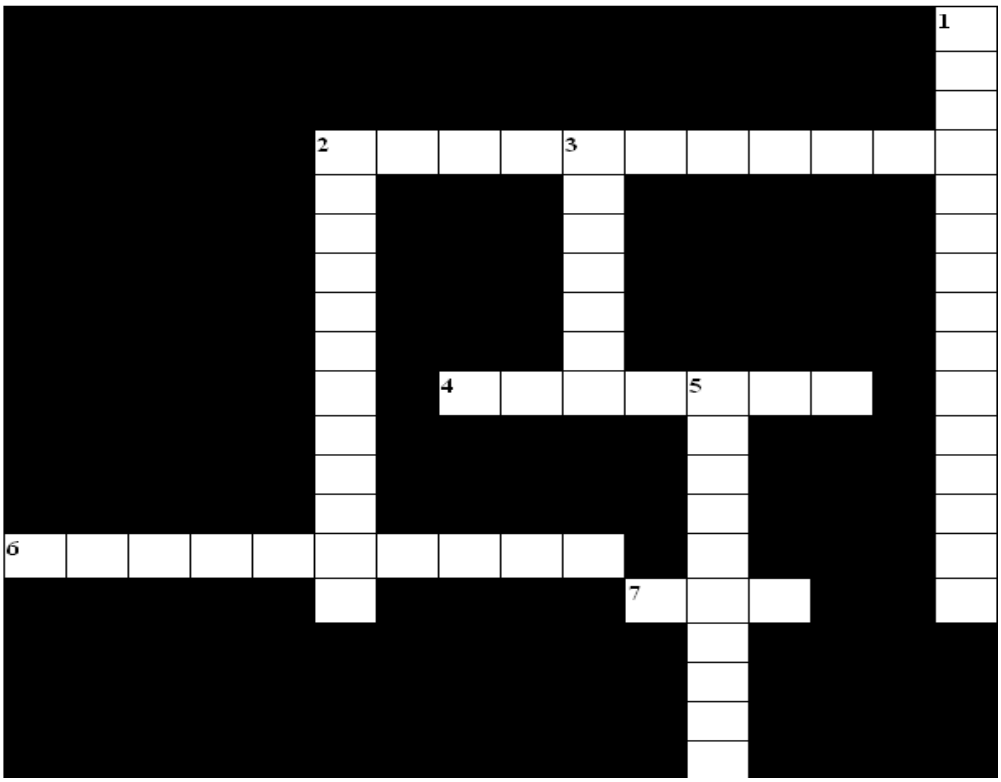


Fig. 7-1

Across:

2. Which object in ADO.NET provides an all-in-one stop for the data and It essentially serves as a middle man, going through the connection to retrieve data, and then passing that into a Dataset.(11)

4. One object of ADO.NET, which is like disconnected cache in which the data can be stored to use in the application.(7)

6. One of the objects provided by ADO.NET, which retrieves a read-only, forward-only stream of data from a database.(10)

7. Which markup language is a cross-platform, hardware and software-independent, text-based that enables us to store data in a structured format by using meaningful tags.(3)

Down:

1. It is an executable object stored in a database, which can have any number of input or output parameters and can pass a return value.(15)

2. In which type of environment, a subset of data from a central data store can be copied and modified independently, and the changes merged back into the central data store.(12)

3. A component of .NET framework. It is a set of classes used to connect to and manipulate data sources and which provides consistent access to data sources exposed through OLE DB and ODBC.(7)

5. In .NET Framework, which namespace provides a rich set of classes for processing XML data.(10)

8.0 Managing State

Topics

- 8.1 State Management
- 8.2 Client-Side State Management
- 8.3 Server-Side State Management
- 8.4 Global.asax
- 8.5 Crossword

Topic: State Management

Estimated Time: 15 mins.

Objectives : At the end of the activity, the participant should understand

- State management

Presentation :

- State management is a process by which you maintain application and session related information when multiple users request for the same or different web pages of an ASP.net application.
- A new instance of the Web page class is created each time the page is posted to the server.
- State management is used when the same users log on to website for multiple times.
- Web form pages are HTTP-Based, they are stateless, which means they don't know whether the requests are all from the same client, and pages are destroyed and recreated with each round trip to the server, therefore information will be lost, therefore state management is really an issue in developing web applications.
- There are 2 types of state management
  - Client-side State Management
  - Server-side State Management

Context :

- Use of state management for security purpose.
- Client side state management is used for client side.
- Server side state management is used for server side.

Practice Session/s :

- Identify the use of state management.

Check List :

- Importance of state management.
- To preserve data on both per-page basis and an application-wide basis.

Common Error/s :

- No Specific errors.

Exception/s :

- No Specific Exception.

Lesson/s Learnt :

- ☑ Understanding state management.
- ☑ State management includes maintaining page-level information during the round trip of a Web Form page.
- ☑ It also maintains application and session-related information when multiple users request for the same or different pages.

Best Practice/s :

- Understand the concepts regarding state management, it's types and when it is used.

One of the primary goals of the Web and its underlying protocol, HTTP, is to provide a scalable medium for sharing information. Adding user state inherently reduces scalability because the pages shown to a particular user will be different from those shown to another user and thus cannot be reused or cached.

Topic: Client-Side State Management

Estimated Time: 90 mins.

Objectives : At the end of the activity, the participant should understand

- Client side state management
- View State
- Hidden Form Field
- Cookies
- Query String

Presentation :

- Client-side state management involves storing information between calls to the server in the final HTML page, in an HTTP request, or on the disk cache of the client computer.
- There are 4 type
  - View state
  - Hidden Form Field
  - Cookies
  - Query String

View State:

- View state enables to retain page and control specific values between round trips. It is implemented with a hidden field called **_VIEWSTATE,** which is automatically created in all the Web Form Pages.
- Asp.net framework uses **ViewState** property, inherited from the base Control class, to automatically save the values of the page and of each control prior to rendering the page.

Hidden Form Field:

- A hidden field does not render visibly in the browser, but its properties can be set like a standard control.
- When a page is submitted to the server, the content of a hidden field is sent in the HTTP Form collection along with the values of other controls.
- Hidden field stores a single variable in its **value** property and must be explicitly added to the page.
- In ASP.NET, the **HtmlInputHidden** control provides the hidden field functionality.

Cookies:

- A cookie is a small data structure used by the web server to deliver data to a web client.
- A cookie contains page-specific information that a web server sends to a client along with the page output.

- Cookies are used for page-specific information because HTTP is a stateless protocol and cannot indicate whether page requests are coming from the same or from different clients.
- Cookie is used to keep track of each individual user who accesses a web page across HTTP connection. Cookies are saved on client computer.
- Cookies can be **temporary cookie** (session cookie) or **persistent**.
- Persistent cookie remains till expiry date while temporary cookies are deleted as soon as browser is closed.
- The class used to work on cookies is **HttpCookie**.

Query String:

- Query string passes information from one page to another, But most browsers and client devices impose a character limit on the length of the URL.
- In addition, the query values are exposed to the Internet via the URL so in most of the cases security may be an issue. For example, in a logon screen, the user name can be passed to next page in the application by using query string.
- A query string is contained in the HTTP request for a specific URL. Therefore, no server resource is involved to process a query string.

View State:

Scenario:

Manager Wilson of Mahindra and Mahindra has a website .Wilson wants to retain page and control-specific values specified in each web form between round trips. So Wilson suggested his views to the developers. Developers suggest to use View State to implement it.

Demonstration/Code Snippet :

When the page is loaded for the first time in the memory of client computer, the message **View State is Enabled** is assigned to text property of label control. If Click Here button is clicked, page is reloaded and label continues to display the **View State is Enabled** message. It is because the Label control state is automatically preserved through view state.

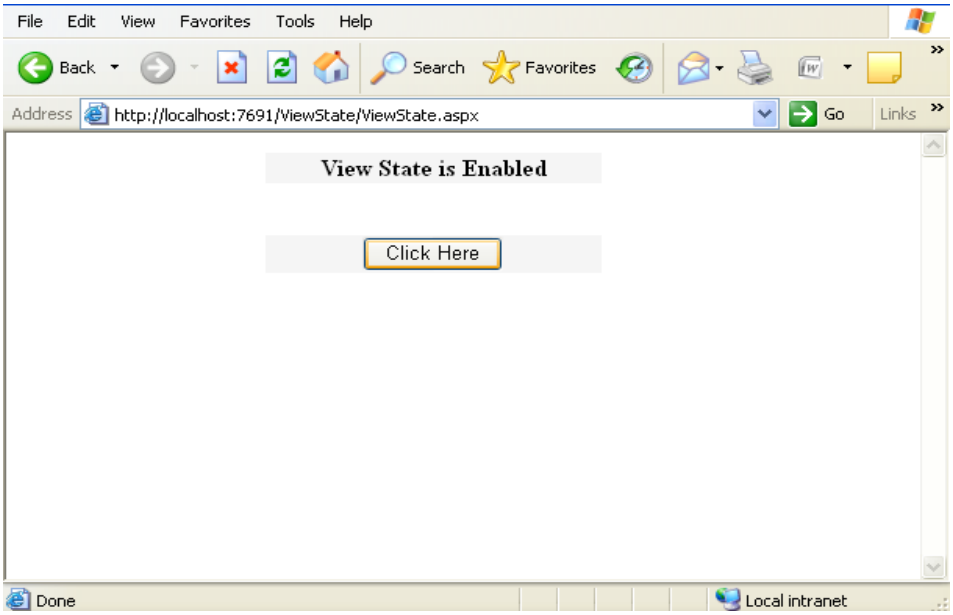


Figure 8.2-1: ViewState.aspx showing view state feature.

After setting the **EnableViewState** property of label control to **false** and execute the application. On Clicking Click Here Button following output is displayed.

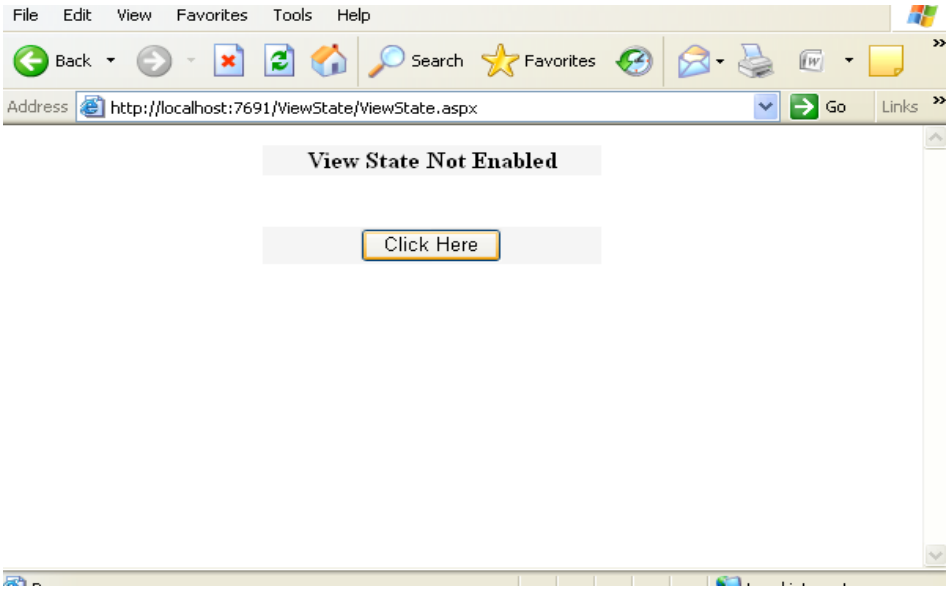


Figure 8.1-2: ViewState.aspx Showing View State not being Maintained.

Step 1: Create a new Web application project called ViewState using visual studio 2005. Rename Default.aspx to ViewState.aspx.

Step 2: Insert HTML Table from toolbox in ViewState.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblViewState	Text:View State is not Enabled EnableViewState:True
asp:button	btnClickHere	Text:Click Here EnableViewState:True

Step 3: Set **EnableSessionState** property of Page to true or Write in @page directive EnableSessionState=”true”.

Step 4: Write in the Page Load event Handler of ViewState.aspx page.

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            lblViewState.Text = "View State is Enabled";
        }
    }
```

The view state of a Web Page or control consists of values of page or the control. To preserve these values across stateless HTTP requests, Web pages and controls use an instance of StateBag class. StateBag class is primary storage mechanism for HTML and server controls.

Hidden Form Fields:

Scenario:

Administrator Steve of Tata Motors already having website for company. Steve wants to store page-specific information in some of the Web Form. So Steve suggests his ideas to the developers. Developers use Hidden Form Fields feature to implement it.

Demonstration/Code Snippet :

Enter some text in textbox and then click Submit button. Following is the output after execution. It stores the value entered in text in hidden field.

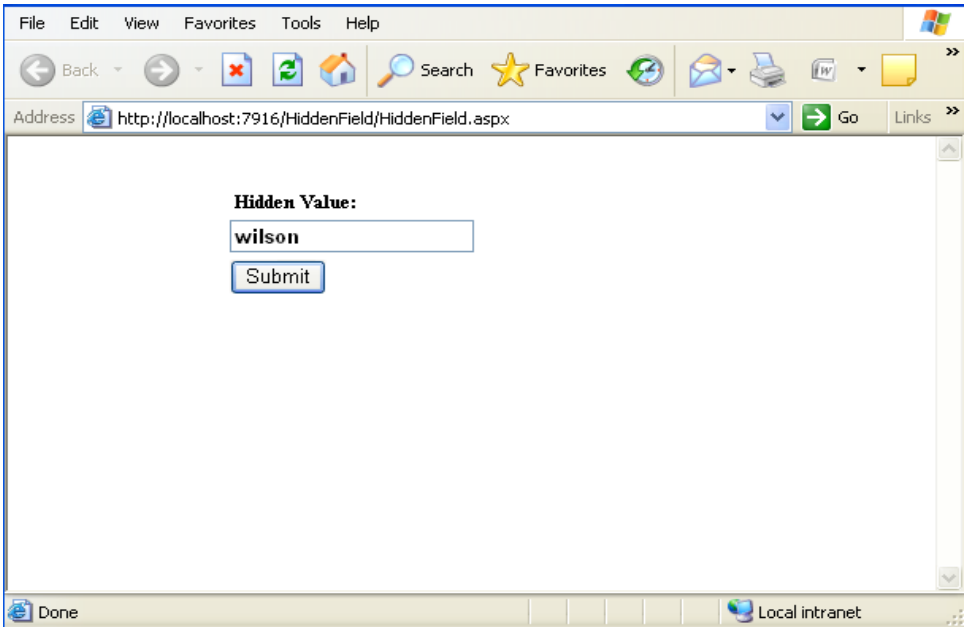


Figure 8.1-3: HiddenField.aspx on submit storing textbox value in hidden field.

Make the textbox field blank and click Submit button, hidden field is displayed.

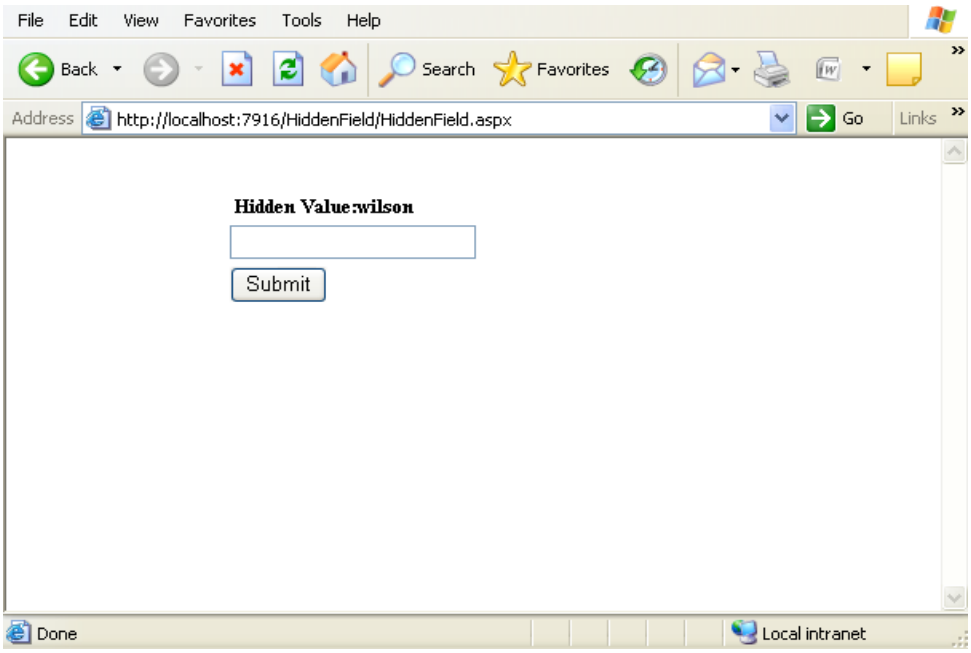


Figure 8.1-4: HiddenField.aspx on submit button showing hidden field value.

Step 1: Create a new Web application project called HiddenField using visual studio 2005. Rename Default.aspx to HiddenField.aspx.

Step 2: Insert HTML Table from toolbox in HiddenField.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHiddenField	-
asp:button	btnSubmit	Text:Submit
asp:textbox	txtName	-
Htmlinput#	hdnName	Run as server control

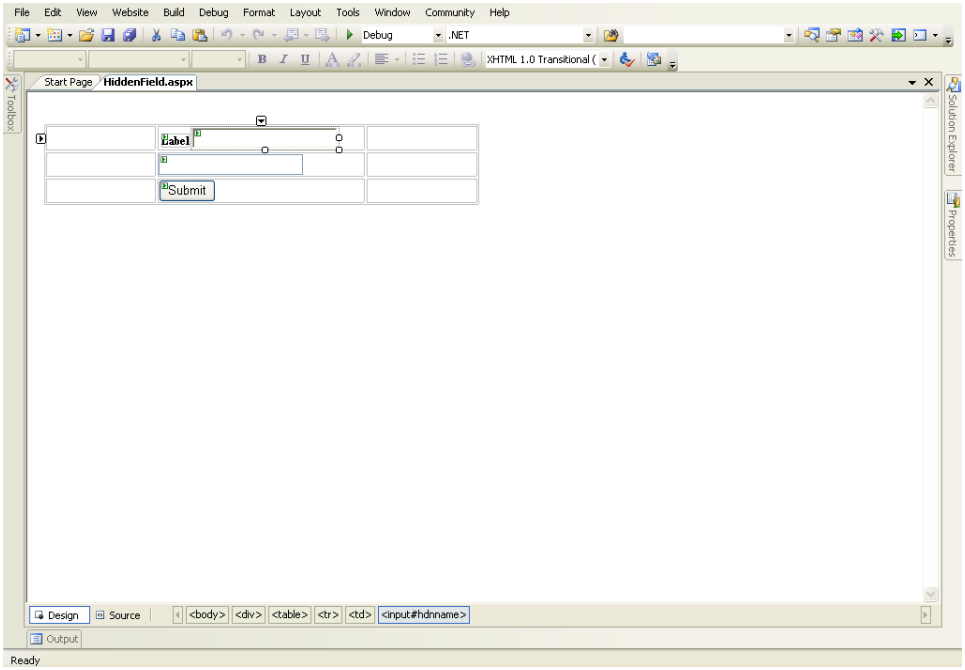


Figure 8.1-5: HiddenField.aspx design.

Step 3: Write the event handler for Page Load and Submit button.

```
protected void Page_Load(object sender, EventArgs e)
    {
        lblHiddenfield.Text = "Hidden Value:" + hdnName.Value;
    }
protected void btnSubmit_Click(object sender, EventArgs e)
    {
        hdnName.Value = txtName.Text;

    }
```

Step 4: Run the application.

Cookies:

Scenario :

Administrator James of Texas Company has created a website. James wants to keep track of each individual user who accesses Web Pages of the website. For this purpose James uses Cookies.

Demonstration/Code Snippet :

The name and the expiry date of the cookie are displayed in the client browser as shown below:

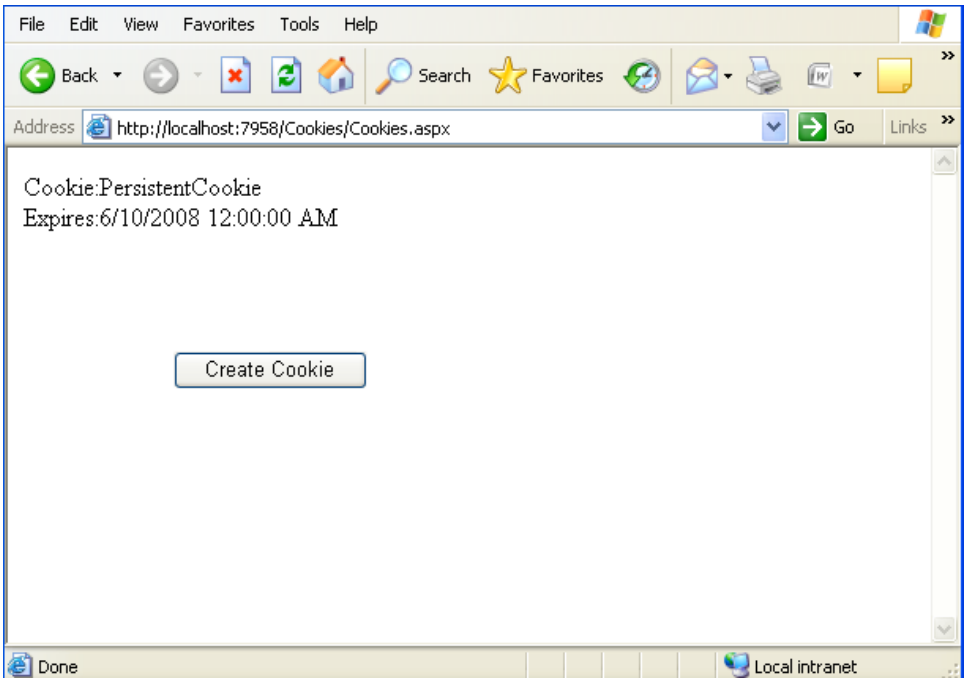


Figure 8.1-6: Cookies.aspx showing cookie name and expiry date.

Step 1: Create a new Web application project called Cookies using visual studio 2005. Rename Default.aspx to Cookies.aspx.

Step 2: Drag and drop the server controls on Cookies.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:button	btnCreateCookie	Text:Create Cookie

Step 3: Write the event handler for Create Cookie button.

```
protected void btnCreateCookie_Click(object sender, EventArgs e)
    {
        HttpCookie mycookie = new HttpCookie("PersistentCookie",
       "Welcome");
        //Instantiate the HttpCookie class object and assign the name
        mycookie.Expires = System.Convert.ToDateTime("06/10/2008");
        //specify the expiry date of the cookie.
        Response.Cookies.Add(mycookie);
        //Add the cookie to HttpCookieCollection
        HttpCookie Mycookie;
        //Declare object of cookie class
        Mycookie = Request.Cookies.Get("PersistentCookie");
        //Retrieve the dtails of cookie name persistentCookie
        Response.Write("Cookie:" + Mycookie.Name + "<br>");
        //Display the name of the cookie
        Response.Write("Expires:" + Mycookie.Expires + "<br>");
        //Display the expiry date of the cookie
    }
```

Step 4: Run the application.

Expired persistent cookies: Once a persistent cookie expires, it is no longer valid. You can verify this by setting a persistent cookie with a lifespan of 1 minute (use Now.AddMinutes(1)), refresh your page a few times, and then go for 1-minute break. Once you come back, refresh the same page and see what happens.

Query string:

Scenario:

Administrator Steve of Texas Company has a website. Steve wants that, when employees logon to the company's website, the user name of the employee should be passed on to the URL of next page. For this Steve uses Query String.

Demonstration/Code Snippet :

QueryString.aspx page showing name entered in Textbox.

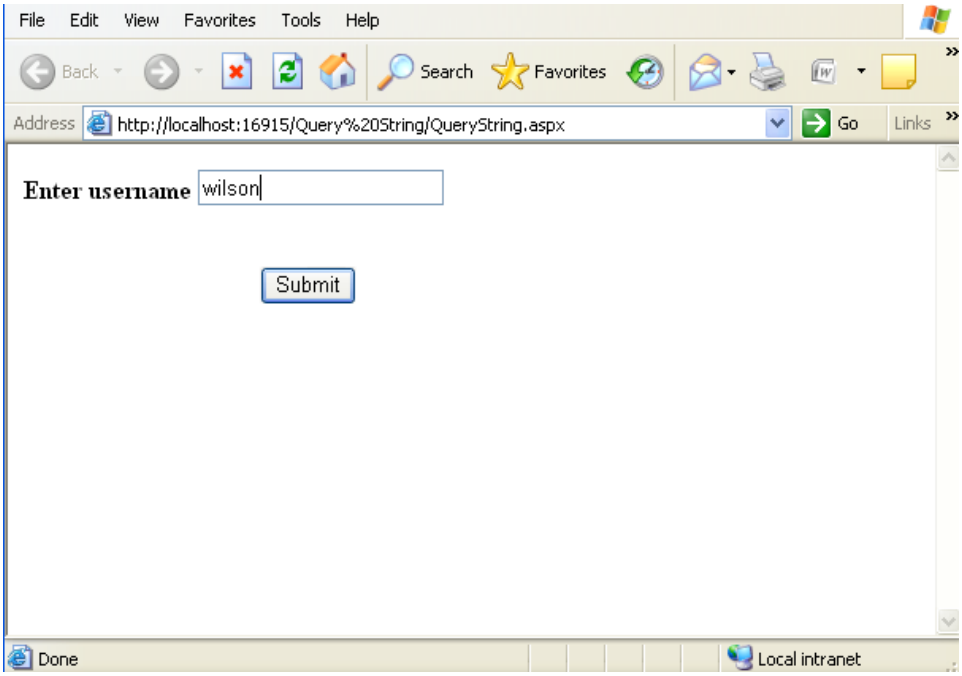


Figure 8.1-7:QueryString.aspx page .

After entering the text (here wilson) in the textbox control , click Submit and see the URL as shown below.

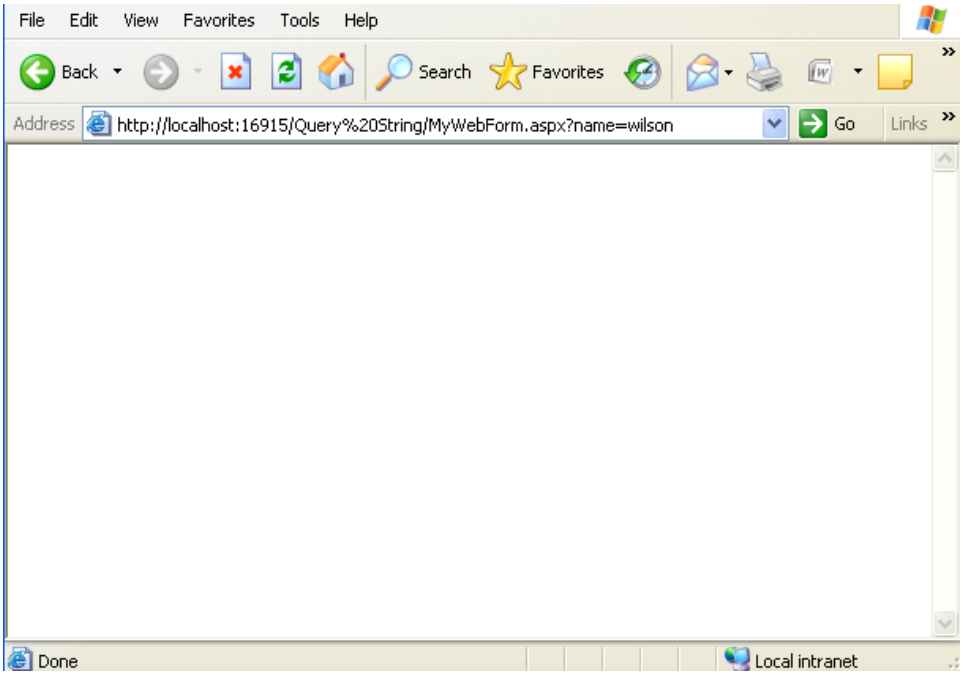


Figure 8.1-8:MywebForm.aspx showing name entered in textbox in URL.

Step 1: Create a new Web application project called QueryString using visual studio 2005. Rename Default.aspx to QueryString.aspx.

Step 2: Drag and drop the server controls on QueryString.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblUsername	Text: Enter Username
asp:textbox	Txtusername	-
asp:button	btnSubmit	Text:Submit

Step 3: Create a new Web form in the same project called MyWebForm.aspx. In the Solution Explorer, right click on the name of your Web Application (QueryString) and select Add New Item/Web Form.

Step 4: Write the event handler for submit button by double clicking it.

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
Response.Redirect("MyWebForm.aspx?name=" +System .Web .HttpUtility
.UrlEncode (txt_username .Text));//the URLEncode method of
HttpUtility class is used to pass the string entered in the textbox
control.
}
```

HttpUtility.UrlEncode: The URLEncode method of HttpUtility class is used to pass the string entered in the textbox control.

Step 5: Execute QueryString.aspx. Enter text in the textbox and then click Submit button.

Context :

- ViewState is used to store small amount of information for a page that will post back to itself.
- Hidden form fields are used to store small amount of information for a page that will post back to itself or to another page, when security is not an issue.
- Cookies are used to store small amounts of information on the client and security is not an issue.
- QueryStrings are used to transfer small amount of information from one page to another and security is not an issue.

Practice Session/s :

Create a website having a login page. Store the details in cookie on successful login and also maintain how many times user has login in the site in cookies. When a user login from that particular machine for 5th time a message like "you have won a gift" should be displayed.

Check List :

- Query string provides a simple way to pass information from one page to another.
- Cookies are saved on the client computer and can be either temporary or persistent.
- View State automatically saves the value of the page and of each control before the page is rendered.
- Values in view state are more secure than values stored in hidden form fields.

Common Error/s :

- By not setting the cookie's expiration time, the cookie is created but it is not stored on the user's hard disk and so is maintained as part of the user's session information.

Exception/s :

- As ViewState is saved in the user's session, it is possible for view state to expire if a page is not posted back within the session expiration time.

Lesson/s Learnt :

- ☑ State management includes maintaining page-level information during the round trip of a Web Form page.
- ☑ It also maintains application and session-related information when multiple users request for the same or different pages.

Best Practice/s :

- Values in view state are hashed, compressed, and encoded for Unicode implementations, so values are more secured than values stored in general hidden form fields.
- As http is a stateless protocol, it is recommended to manage the state by the programmer explicitly using ViewState, HiddenFormFields, Cookies and QueryString.

Topic: Server-Side State Management

Estimated Time: 60 mins.

Objectives : At the end of the activity, the participant should understand the use of

- Session and Application objects to store state variables.
- Application.Lock () and Application.UnLock () methods.

Presentation :

- Web applications need to be able to maintain some unique state information for each client which visits the Web site.
- Server-side state management is used to manage application and session-related information on web server.
- ASP.NET provides 2 useful universal objects which the Web application developer can use to store state variables at the server.
  - **Application State**
  - **Session State**

Application State:

- Application state means storing global application- specific information.
- The information in the application state is stored in a key value pair and is used to maintain data consistency between server round trips and between pages.
- Application state is created when each browser is made for a specific URL.
- All information stored in the application state is shared among all the pages of web application by using **HttpApplicationState** class.

Session State:

- Session state stores information in server memory, but unlike application state, the server stores a different copy of the session variables for each browser session.
- The server differentiates between sessions by assigning an internal Session ID which is unique across time.
- Session object is actually an instance of the **HttpSessionState** class.
- A user session starts when a user requests the first page from a Web Site. When the first page is requested, the Web server adds the ASP.NET Session ID cookie to the client computer.

Scenario:

Mr. Johnson, the owner of the Book World, wants to know that how many times his site is being visited and wants to welcome the user with his/her ID in homepage on successful login.

So Johnson ask Lui, the developer of this website to add this functionlity. Lui uses the concept of Application State and Session State.

Demonstration/Code Snippet :

Login.aspx page showing no of times site has been viewed.

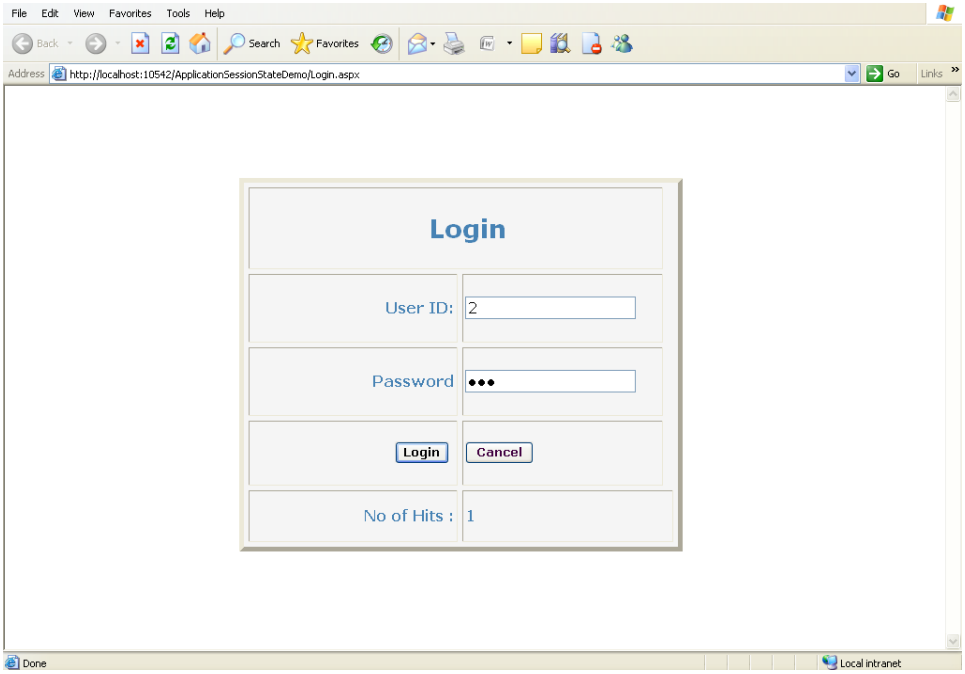


Figure 8.2-1: Login.aspx page Showing no of times web page viewed.

Step 1: Create a new Web application project called StateWebApplication using visual studio 2005. Rename Default.aspx to Login.aspx.

Step 2: Insert HTML Table from toolbox in Login.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Login
asp:label	lblUserid	Text:User ID
asp:label	lblPwd	Text:Password
asp:label	lblHitcnt	-
asp:label	lblHitcntheading	No of Hits
asp:textbox	txtUserid	-
asp:textbox	txtPwd	-
asp:button	btnLogin	Text:Login
asp:button	btnCancel	Text:Cancel

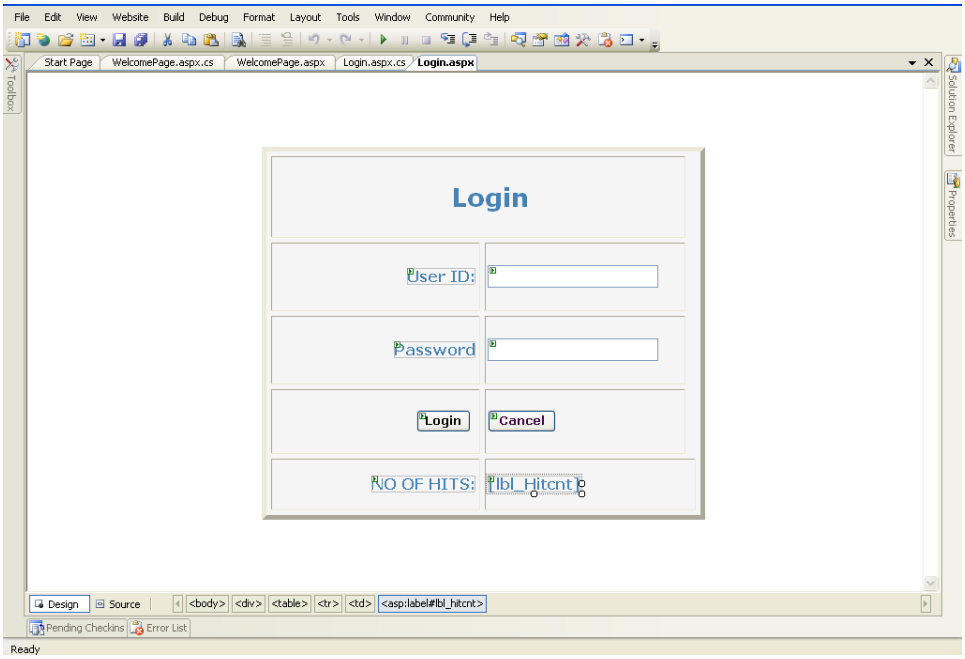


Figure 8.2-2: Login.aspx page design.

Step 3: Write the event handler for the Page Load. Fill in the method as follows.

```
    Using System.Windows.Forms;//Add system.windows.forms reference
    for messagebox
    protected void Page_Load(object sender, EventArgs e)
    {
      // Connection object to the database is being created.
      String Connectionstr= "server=hstslc011; uid=sa;
      password=satyam;database=northwind" ;
      conn = new SqlConnection(Connectionstr);
      Application.Lock();
      // Locks the application state object as a whole.
      // You can not selectively lock items in an application state.
      if (Application["HitCnt"] == null)
      {
      Application["HitCnt"] = 0;
   }
      else
      {
      Application["HitCnt"] = (int)Application["HitCnt"] + 1;
      }
      lblHitcnt.Text = Application["HitCnt"].ToString();
      Application.UnLock();
      // Unlocks the application state object.
}
```

Calling Application.Lock() and UnLock():
In a multi-user environment, there may be many concurrent visits to Login.aspx from different clients. And because ASP.NET is multi-threaded (Of course server applications have to be multi-threaded!), inconsistencies may occur when the same statement attempting to update a variable into Application executes simultaneously by more than one client. To prevent other clients from executing such statements at the same time, the **Application.Lock ()** method is invoked first. Only one of many Concurrently running processes can execute statements within the 'critical section' (between the Lock() and UnLock() statements). Only when that process which is executing statements within the critical section calls **Application.UnLock ()** can other processes go into the critical section.

Step 4: Whenever user runs the application the HitCnt,application varible will get incremented and that will store the no of hit count. Initially HitCnt will be null.

Step 5: Session State can also be used in the application. Write the following code in login button event handler by double clicking it.

```
protected void btnLogin_Click(object sender, EventArgs e)
 {
try
{
     Session["eid"] = txtUserid.Text; // Session State is used and
     "eid"
     session variable.
     cmd = new SqlCommand("loginprocess", conn);
     cmd.CommandType = CommandType.StoredProcedure;
     cmd.Parameters.Add("@eid", SqlDbType.Int).Value =
     txtUserid.Text;
     cmd.Parameters.Add("@pwd", SqlDbType.VarChar, 30).Value
     =txtPwd.Text;
     cmd.Parameters.Add("@reply", SqlDbType.VarChar,
     30).Direction =
     ParameterDirection.Output;
     conn.Open();
     cmd.ExecuteNonQuery();
    if (int.Parse(cmd.Parameters["@reply"].Value.ToString()) > 0)
    {
      Response.Redirect("WelcomePage.aspx");
    }
  else
  {
      Response.Write("Sorry... Not Authorized... ");
      txtUserid.Text = "";
      txtPwd.Text = "";
  }
  }
  catch (Exception ex)
  {
      MessageBox(ex);
```

```
    }
    finally
    {
        conn.Close();
    }
}
```

The expression Session ("eid") returns the value stored in the variable eid in the Session object.

Step 6: Create a new Web form in the same project called WelcomePage.aspx. In the Solution Explorer, right click on the name of your Web Application (StateWebApplication) and select Add New Item/Web Form.

Step 7: Drag and drop the server controls on WelcomePage.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblMessage	Text:Welcome User,
asp:label	lblUserid	-

Step 8: Write in the Page Load event handler of WelcomPage.aspx by double clicking webform.

```
public partial class WelcomePage : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblUserid.Text = Session["eid"].ToString();
    }
}
```

The expression Session ("eid") assigns the value stored in the Session object to lblUserid.

Step 9: Run the application. On successful login, user will be redirected to WelcomePage.

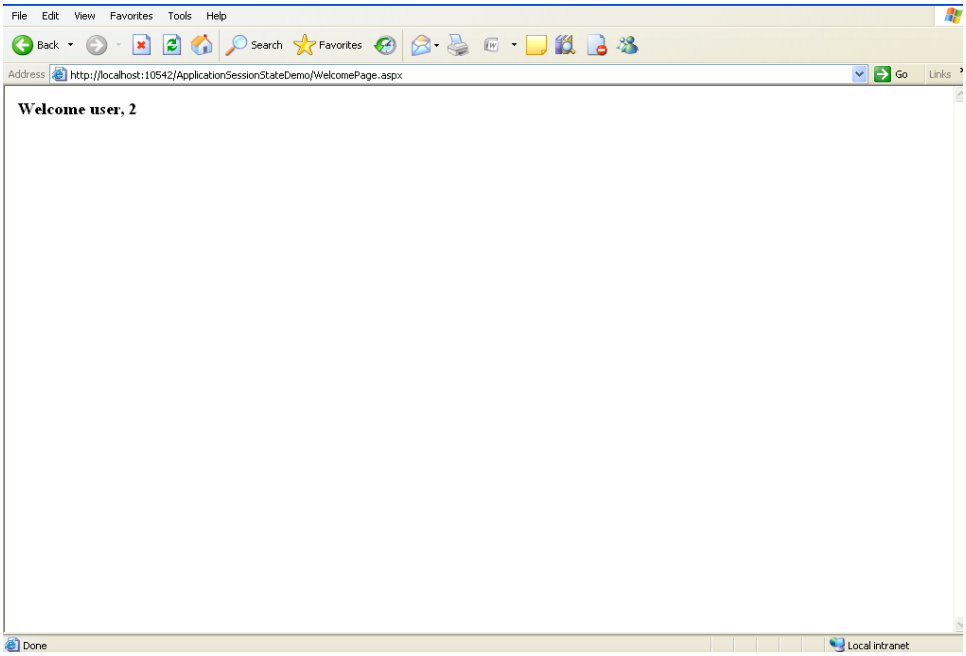


Figure 8.2-3: WelcomePage.aspx showing UserId.

Tables used: emplogindetails
Database: northwind

	eid	pwd
1	1	aaa
2	2	bbb
3	3	ccc

Figure 8.2-4: Emplogindetails table

Process:

```
create proc loginprocess @eid int,@pwd varchar(30),@reply int output
as
begin
select @reply=count(*) from emplogindetails where eid=@eid and
pwd=@pwd
end
```

Context :

- Use Application state throughout application when some process needs to be carried out for all users.
- Use session state for a particular user for tracking.

Practice Session/s :

Create a website having a login page. Calculate how many times a particular user has logged in (Use Application State). On successful login redirect user to homepage which display user name (Use Session State) retrieved from the database on the basis of login id. Store Session id and User id into database by creating a session_details table having fields Sessionid and userid.

Check List :

- Use of application state and session state.
- Use of Application.Lock () and Application.Unlock ().

Common Error/s :

- Syntax not properly defined.
- Trying to use session as global variable.

Exception/s :

- Trying to access an application once the session gets expired, will lead to an exception.

Lesson/s Learnt :

- ☑ Application state and session state.
- ☑ Application.Lock() and Application.Unlock().

Best Practice/s :

- Always use Application.Lock () and Application.Unlock () for common application state variable.
- **Wasting memory**: Do not place unnecessary variables (especially large objects)in Session or Application! The reason is because it takes up server resources to store these variables for persistence across multiple pages.

Accessing Session: Access Session Id using Session.SessionId.

Killing a session programmatically: You can programmatically invalidate a user session by calling the Session.Abandon () method.

Topic: Global.asax

Estimated Time: 30 mins.

Objectives :

At the end of the activity, the participant should understand the use of

- Basics of Global.asax
- Associated events of Global.asax.

Presentation :

- Global.asax is a file that resides in the root directory of your application.
- It is an optional file that responds to application level and session level events raised by asp.net
- The Global.asax is a code file where developers place application and session level event handler code
- In Global.asax there are many events to work with, such as **Application_Start**, **Application_End**, **Session_Start**, **Session_End**, and so on.

Scenario :

In a Klite Solution Company the manager Wilson wants to see that total no of users who has accessed the sites and a robust website which can handle generic error too. So for this Wilson suggests developers to use Global.asax.

Demonstration/Code Snippet :

Here number of hits is displayed and it will be increased every time who views the website.

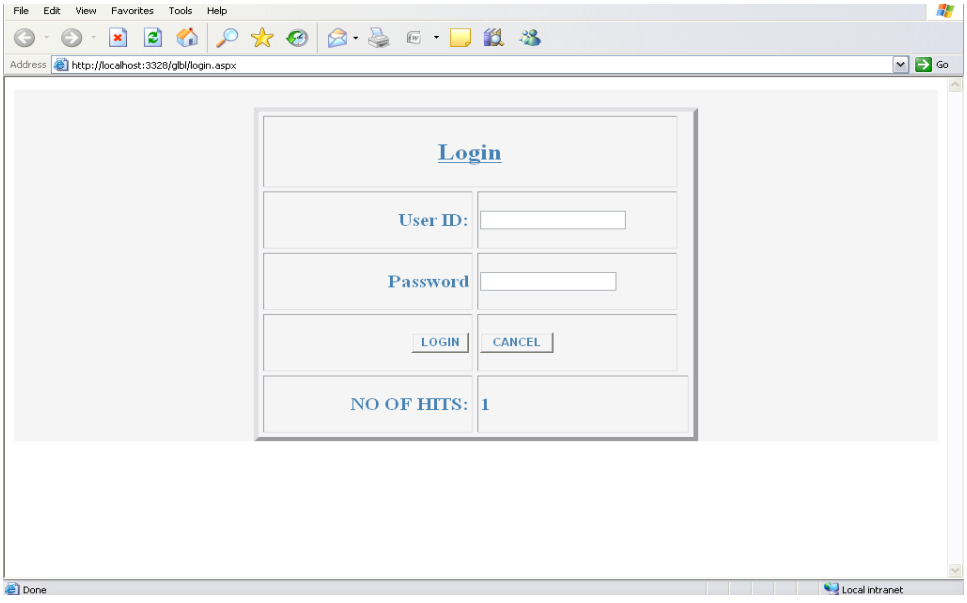


Figure 8.2-1: Login Page.

A generic error occurred which is handled using error page and Application_Error event .

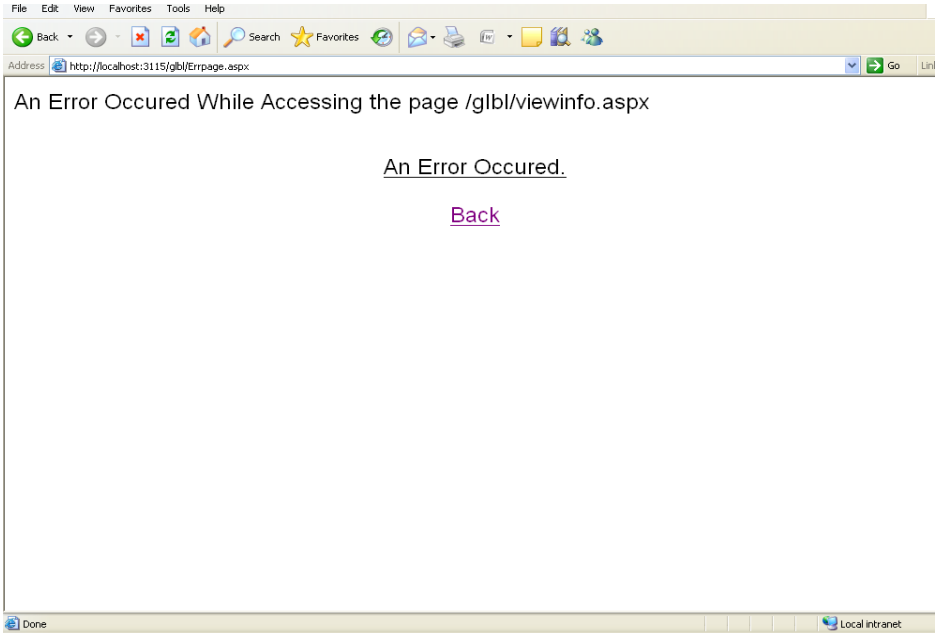


Figure 8.3-2: Error page.

Step 1: Open visual studio 2005 and create a website and selecting add new items add Global Application Class i.e. Global.asax file. In Global.asax file there are some predefined events defined as shown below

The **Application_Start** event is fired the first time when an application starts.

```
void Application_Start(object sender, EventArgs e)
{
        //  Code that runs on application startup
       int hitcount = 0;
       Application["hitcount"] = hitcount;
}
```

The **Application_End** event is last event of its kind that is fired when the application ends or times out. It typically contains application cleanup logic.

```
void Application_End(object sender, EventArgs e)
{
    // Code that runs on application shutdown
}
```

The **Application_Error** event is fired when an unhandled error occurs within the

application.

```
void Application_Error(object sender, EventArgs e)
{
   string err;
        // get the path of the error page
        err = Request.Path;
        Session["err"] = err;
        Response.Redirect("Errpage.aspx");
        //  Code that runs when an unhandled error occurs
 }
```

The **Session_Start** event is fired the first time when a user's session is started. This typically contains for session initialization logic code

.

```
void Session_Start(object sender, EventArgs e)
 {
        //  Code that runs when a new session is started
        long htcnttemp =
        (long.Parse(Application["hitcount"].ToString()));
        htcnttemp++;
        Application["hitcount"] = htcnttemp;
 }
```

The **Session_End** Event is fired whenever a single user Session ends or times out.

```
 void Session_End(object sender, EventArgs e)
{
    //  Code that runs when a session ends.
    //  Note: The Session_End event is raised only when the session
    state  mode
    //  is set to InProc in the Web.config file. If session mode is
    set to StateServer
     //  or SQLServer, the event is not raised.
}
```

Step 2: Now create a login page by taking 2 textboxes, 2 labels and 2 buttons as shown in screenshots. In page load event of login page to display number of hit count you have to write as

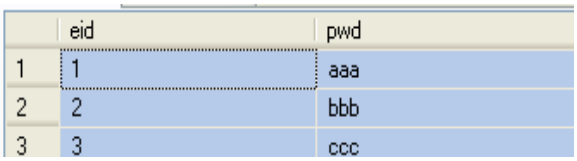
```
        lblHitcnt.Text = Application["hitcount"].ToString();
```

Step 3: To create an error page just add a new item web form and in its page load event write

```
        Response.Write(("An Error Occured While Accessing the page
        " + Session["err"]));
```

Control type	ID	Property values to be modified
asp:label	lblLogin	Text:Login
asp:label	lblUserid	Text:User Id
asp:label	lblPassword	Text:Password
asp:label	lblNoofhits	Text:No of Hits
asp:button	btnLogin	Text:Login
asp:button	btnCancel	Text:Cancel

Tables Used: emplogindetails table
Database: Northwind



	eid	pwd
1	1	aaa
2	2	bbb
3	3	ccc

Figure 8.3-3:Table content of emplogindetails table.

Procedures:

```
create proc loginprocess @eid int,@pwd varchar(30),@reply int output
as
begin
select @reply=count(*) from emplogindetails where eid=@eid and
pwd=@pwd
end
```

Context :

- Use Global.asax in a web application.
- Across platforms where there is support of http client (browser).
- Use of different events in global.asax.

Practice Session/s :

Create a login page which uses global.asax and implement no of users online . Create a table and insert some data. Table must have field such as status for online /offline, userid, password.

Check List :

- Importance of global application class.
- Importance of application and session events.

Common Error/s :

- Not properly defined session and application events.
- Trying to handle any UI related processing or process individual page level requests

Exception/s :

- Any illegal entry within Global.asax may not allow to build a website.

Lesson/s Learnt :

- ☑ Not properly defined session and application events.
- ☑ Trying to handle any UI related processing or process individual page level requests.

Best Practice/s :

- Use Global.asax to handle various application-wide tasks such as user authentication, application start up, and dealing with user sessions.

Changing global.asax: If a change is made to the Web application's global.asax file when the application is currently hosted, the application 'shuts down' dynamically. When a new page request comes in, the Web application will restart.

Optional file: global.asax is an optional file. But if it exists, it must be located in the root of the Web application. (i.e. c:\inetpub\wwwroot\<project_name>).

Crossword: Unit-8 **Estimated Time: 10 mins.**

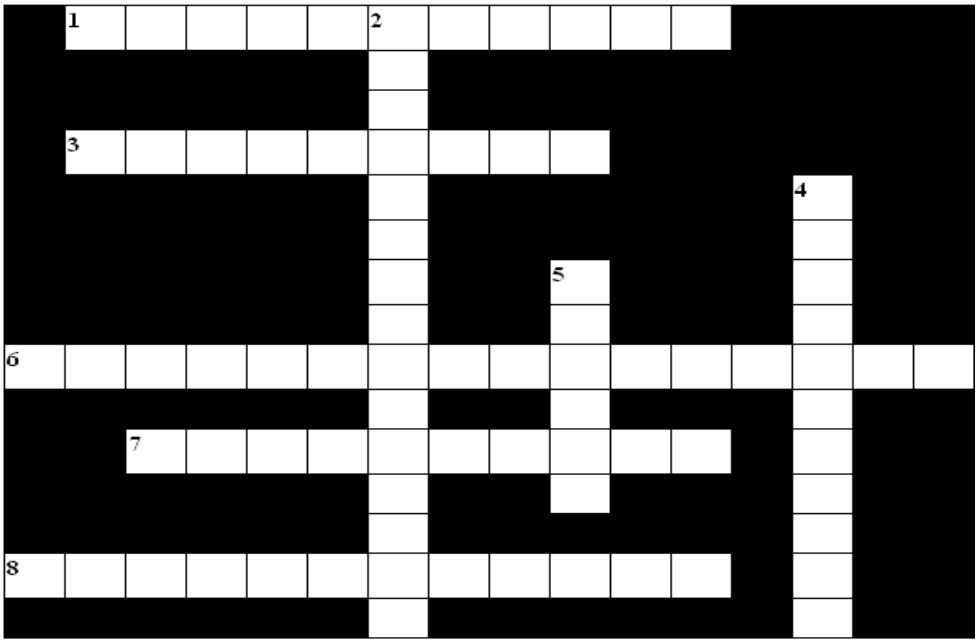


Fig. 8-1

Across:

1. A type of client side state management which provide a simple but limited way of maintaining some state information using which we can easily pass information from one page to another.(11)

3. A type of client side state management which enables you to retain page and control specific values between round trips and It is implemented with a hidden field called _VIEWSTATE.(9)

6. A type of server side state management which means storing global application- specific information, which is stored in a key value pair and is used to maintain data consistency between server round trips and between pages.(16)

7. Which type of state management is used for server.(10)

8. A type of server side state management, which stores information in server memory and the server stores a different copy of the variables for each browser session.(12)

Down:

2. Which is the process by which you maintain application and session related information when multiple users request for the same or different web pages of an ASP.net application.(15)

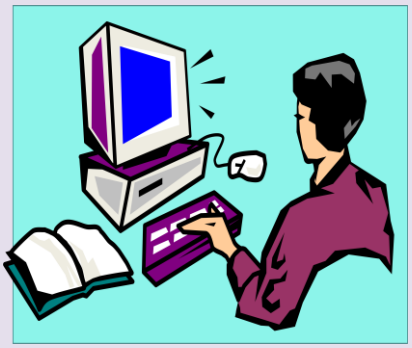
4. In which file, there are many events to work with such as Application_Start, Application_End, Session_Start, Session_End, and so on. Which resides in the root directory of the application.(11)

5. A type of client side state management, which contains page-specific information that a web server sends to a client along with the page output.(6)

9.0 Configuring, Optimizing ASP.Net Web Application

Topics

- 9.1 Caching
- 9.2 Web.Config and Machine.Config
- 9.3 Custom Error Handling
- 9.4 Crossword



Topic: Caching

Estimated Time: 60 mins.

Objectives : At the end of the activity, the participant should understand

- Basics of Caching
- Page Output Caching
- Page Fragment Caching
- Data Caching

Presentation :

- Cache is a thread-safe object and does not require to explicitly lock() and unlock() before access.
- Caching is a technique of persisting data in memory for immediate access to requesting program calls.
- Caching allows requests to be served from memory directly. It minimizes the usage of server resources to a great extent.
- It can cache [store in memory] the output generated by a page and will serve this cached content for future requests.
- Different types of caching are:-
  - Page output caching
  - Page fragment caching
  - Data caching
- Page's output can be cached using an **@OutputCache** directive at the top of the page

```
<%@ OutputCache Duration=5 VaryByParam="id"%>
```

Duration - The time in seconds, how long the output should be cached.
VaryByParam - This attribute is compulsory and specifies the querystring parameters to vary the cache

.

```
<%@ OutputCache Duration=5 VaryByParam="id" VaryByCustom="browser" %>
```

This will vary the cached output not only for the browser but also its major versions. i.e., IE5, IE 6, Netscape 4, Netscape 6 will all get different cached versions of the output.

Page output caching:

- Page output caching allows the output of the pages to be cached using an output cache engine.

Page fragment caching:

- Page fragment caching allows specific portions of the page to be cached rather than caching the whole page.
- Fragment caching has limitations as user controls need to be used for caching.

Data caching:

- Data caching allows caching of frequently accessed data in server side memory variables.
- Lifetime of cached object is same as that of application.

Page output caching:

Scenario :

Manager John of Tata Motors has a website with some good features. John's company is facing some network traffic problem due to increased number of users. In order to reduce network traffic John asks developers to cache entire page data. So for this developers uses Output Caching.

Demonstration/Code Snippet :

OutputCaching.aspx form displaying current date and time indicating the time when the Web Form is cached.

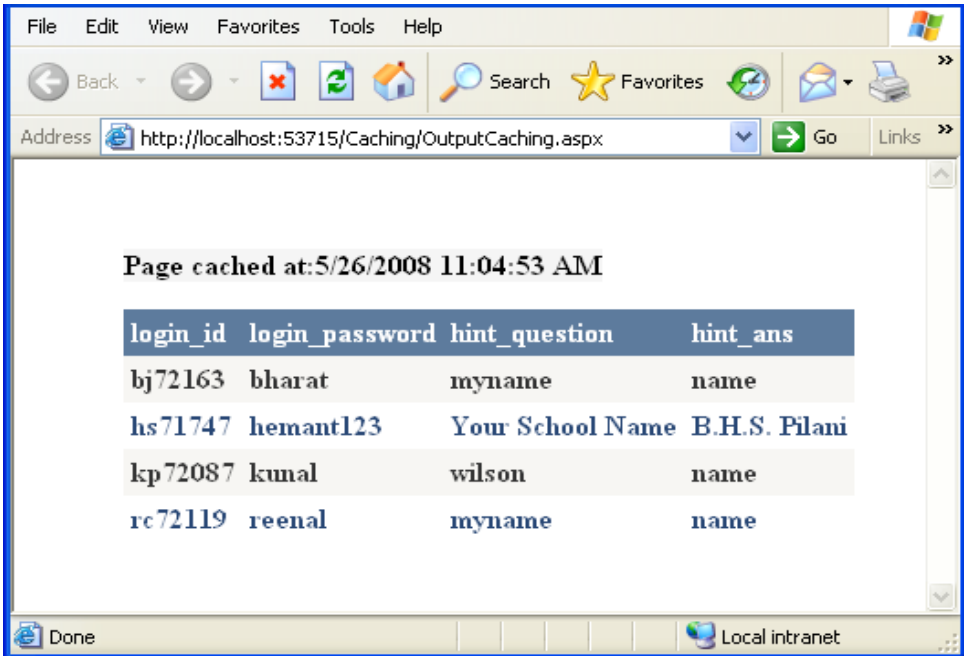


Figure 9.1-1: OutputCaching.aspx showing cached time and date.

OutputCaching.aspx Form showing data cached based on some critera (here by login_id)

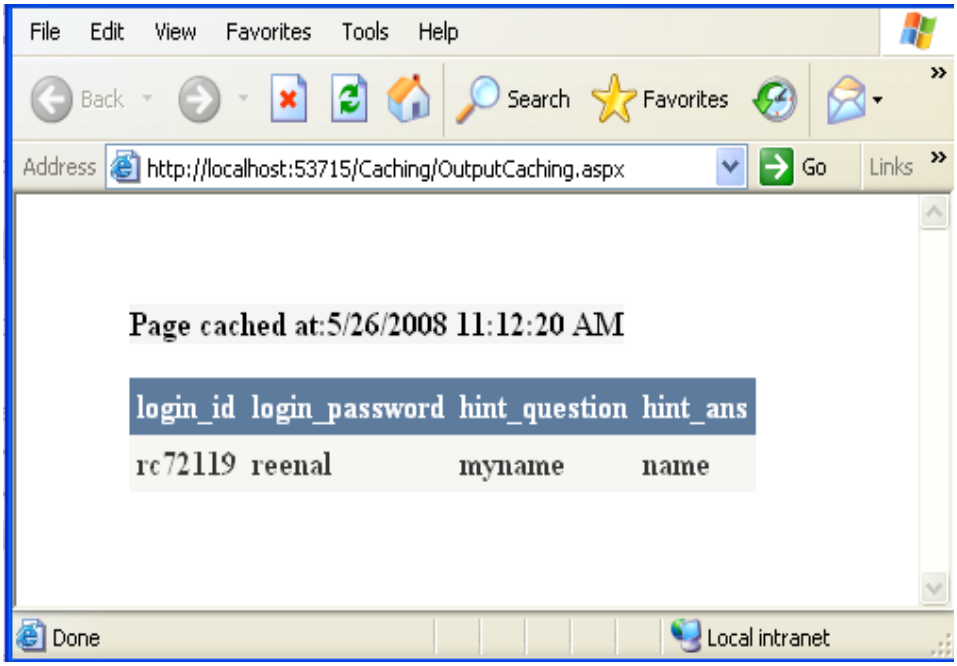


Figure 9.1-2: OutputCaching.aspx Caching done on login criteria.

Step 1: Create a new Web application project called Caching using visual studio 2005. Rename Default.aspx to OutputCaching.aspx

Step 2: Insert HTML Table from toolbox in OutputCaching.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblDate	-
asp:DataGrid	dtgDetails	-

Step 3: Add the following in the HTML view of OutputCaching.aspx Form:

```
<%@ OutputCache Duration="360" VaryByParam="none" %>
```

Step 4: Write in the Page Load event handler by double clicking on the OutputCaching.aspx Form.

```
using System.Data.SqlClient;
Using  System.Windows.Forms;//Add  system.windows.forms  reference  for
messagebox

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    try {
```

```
        string connstr, query;
        connstr =
         "server=hstslc011;uid=sa;password=satyam;database=northwind";
         query = "select * from login ";
         SqlConnection con = new SqlConnection(connstr);
         SqlDataAdapter adap = new SqlDataAdapter(querystring,con);
         DataSet ds = new DataSet();
         adap.Fill(ds);
         dtgDetails.DataSource = ds.Tables[0];
         dtgDetails.DataBind();
         lblDate.Text = "Page cached at:" + DateTime.Now.ToString();
          }
    catch (Exception ex)
    {
     MessageBox(ex);
    }
    }
    }
```

This code displays the data from **login** table and label displays the current date and time, which indicates the time when the Web Form is cached.

Since duration attribute of **OutputCache** element is set to 360, after refreshing OutputCaching.aspx Form within the next 6 minutes, the same time will be displayed. This indicates that OutputCaching.aspx Form is served from the cache and not from the server.

Step 5: To cache the Web Form based on criteria, set **VaryByParam** attribute to the required query string. For example to retrieve and display login table for a specific login_id, modify the @OutputCache as follows:

```
<%@ OutputCache Duration="360" VaryByParam="login_id" %>
```

The screenshot for the same has been shown in figure 9.1-2:
For this make changes in coding as follows:

```
Query= "select * from login where login_id='rc72119'";
```

Tables used: login

Database: Northwind

	login_id	login_password	hint_question	hint_ans
▶	bj72163	bharat	myname	name
	hs71747	hemant123	Your School Name	B.H.S. Pilani
	kp72087	kunal	wilson	name
	rc72119	reenal	myname	name

Figure 9.1-3: Login table.

Page fragment caching:

Scenario :

Manager John of Tata Motors has a website in which john wants to cache some part of design so that it can be used in subsequent pages. So for this John suggests his developers to use Fragment Caching to implement the same.

Demonstration/Code Snippet :

CachingFragment.aspx showing cached time and date.

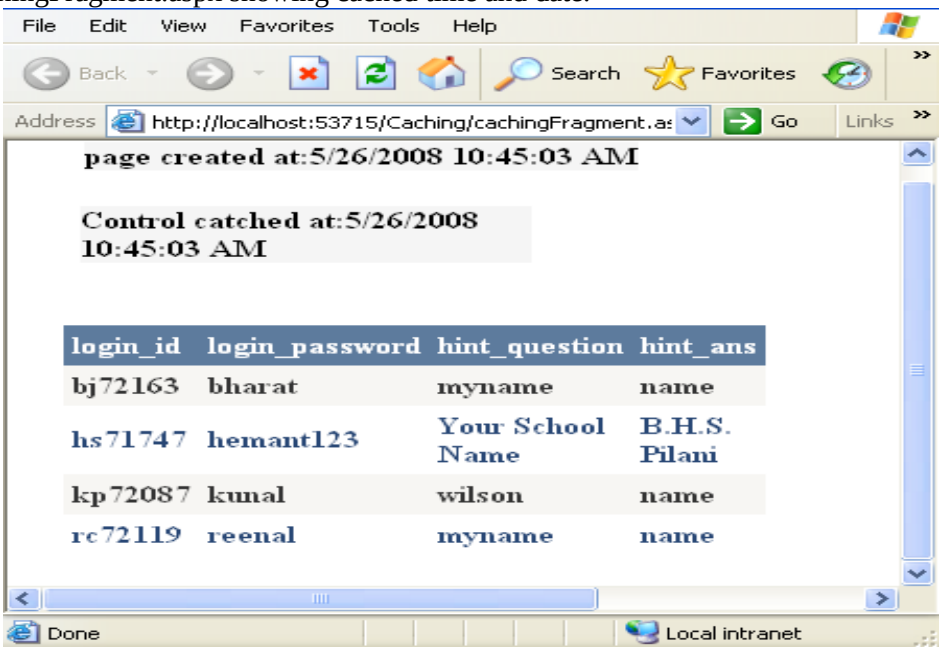


Figure 9.1-4: CachingFragment.aspx design.

Step 1: Create a new Web application project called Caching using visual studio 2005. Create a new Web User Control form in the same project called FragmentCaching.ascx. In the Solution Explorer, right click on the Web Application (Caching) and select Add New Item/Web User Control.

Step 2: Insert HTML Table from toolbox in FragmentCaching.ascx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblUDate	-
asp:DataGrid	dtgdetails	-

Step 3: Add the following in the HTML view of the Web Form:

```
<%@ OutputCache Duration="360" VaryByParam="none" %>
```

Step 4: Right-click the .ascx file, and then click **ViewCode** to display the code-behind page source. Add the following code to **Page_Load** event and build the page.

```
using System.Data.SqlClient;
using System.Windows.Forms;//Add system.windows.forms reference for
messagebox

public partial class FragmentCaching : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {
    try {
        string querystring,connstr,query;
         connstr =
        "server=hstslc011;uid=sa;password=satyam;database=northwind";
         query="Select * from login";
         SqlConnection con = new SqlConnection(connstr);
         querystring=Request .QueryString ["OrderID"];
         SqlDataAdapter adap = new SqlDataAdapter(query,con);
         DataSet ds = new DataSet();
         adap.Fill(ds);
         dtgDetails.DataSource = ds.Tables[0];
         dtgDetails.DataBind();
         lblUDate.Text = "Control catched at:" +
        DateTime.Now.ToString();
        }
   catch (Exception ex)
         {
          MessageBox(ex);
         }
    }
 }
```

This code displays the data from **login** table and label displays the date and time when the control is cached.

Step 5: Create a new Web form in the same project called CachingFragment.aspx. In the Solution Explorer, right click on the name of your Web Application (Caching) and select Add New Item/Web Form.

Step 6: Drag and drop the server controls on CachingFragment.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblDate	-

Step 7: Drag and drop WebUserControl we created onto CachingFragment.aspx.

Step 8: Write in the Page Load event handler of CachingFragment.aspx and build the page.

```
protected void Page_Load(object sender, EventArgs e)
  {
        lblDate.Text = "page created at:" + DateTime.Now.ToString();
  }
```

Step 9: Run the Application. After the page appears in the browser, refresh the page .Notice that the time on the Web Form has been updated, but the user controls still display the time when their associated cache entry was made.

Tables used: login
Database: Northwind

	login_id	login_password	hint_question	hint_ans
▶	bj72163	bharat	myname	name
	hs71747	hemant123	Your School Name	B.H.S. Pilani
	kp72087	kunal	wilson	name
	rc72119	reenal	myname	name

Figure 9.2-5: Login table.

Data caching:

Scenario:

Manager John of Tata Motors has a website. John wants to cache the frequently used data from one page. So for this John suggests developers to use features of Data Caching.

Demonstration/Code Snippet :

Data Caching form

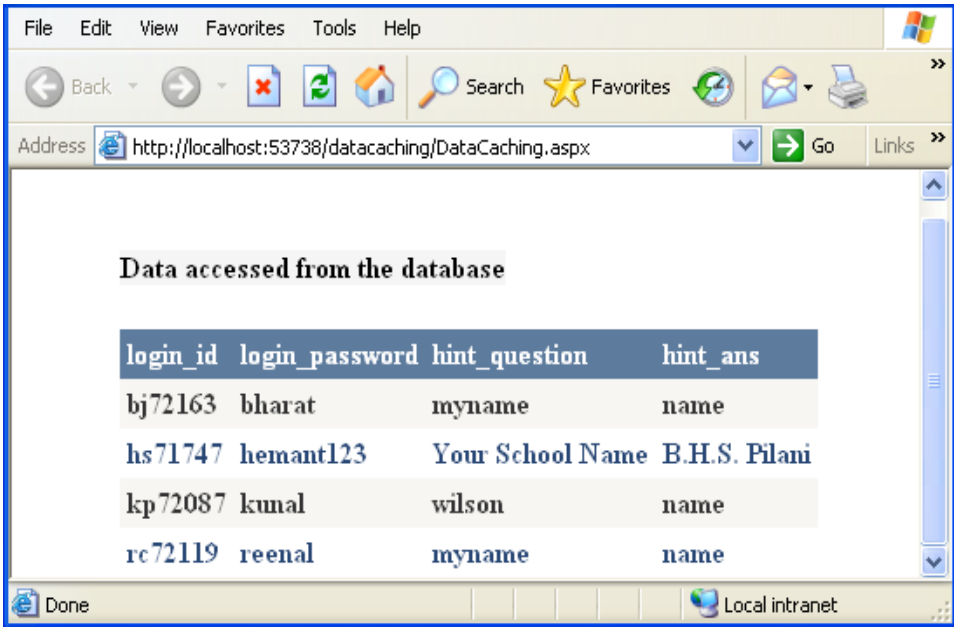


Figure 9.1-6: DataCaching.aspx when No data cached.

Even after changing the table contents, the contents of the DataGrid remains same.

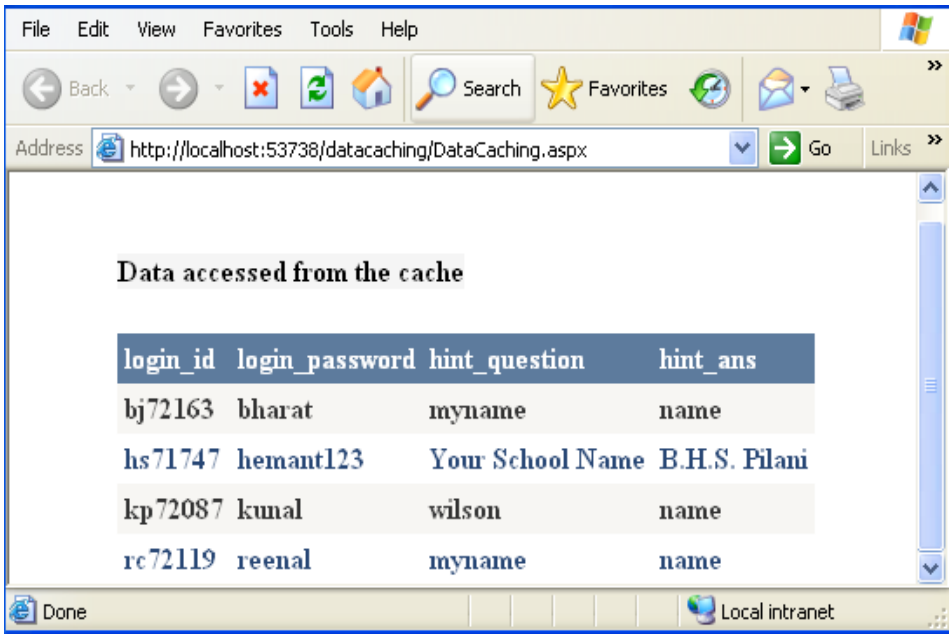


Figure 9.1-7: DataCaching.aspx when data cached.

Step 1: Create a new Web application project called DataCaching using visual studio 2005. Rename Default.aspx to DataCaching.aspx

Step 2: Insert HTML Table from toolbox in DataCaching.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblDate	-
asp:DataGrid	dtgDetails	-

Step 3: Add the following in the HTML view of the Web Form:

```
<%@ OutputCache Duration="360" VaryByParam="none" %>
```

Step 4: Add the following code to the Page Load event of the Web Form.

```
using System.Data.SqlClient;
using System.Data;
Using System.Windows.Forms;//Add system.windows.forms reference for
messagebox

public partial class DataCaching : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
    try {
         string connstr,query;
```

```
        connstr =
        "server=hstslc011;uid=sa;password=satyam;database=northwind
        ";
        query = "select * from login";
        DataSet ds;
        ds=(DataSet)Cache["What i need to store"];
        if (ds== null)
        {
            SqlConnection con = new SqlConnection(connstr);
            SqlDataAdapter adap = new SqlDataAdapter(query,con);
            ds = new DataSet();
            adap.Fill(ds);
            Cache.Insert("What i need to store",ds);
            lblDate.Text = "Data accessed from the database";
        }
        else
        {
            lblDate.Text = "Data accessed from the cache";
        }
        dtgDetails.DataSource = ds.Tables [0];
        dtgDetails.DataBind();
        }
    catch (Exception ex)
    {
        MessageBox(ex);
    }
   }
}
```

If(ds== null):- Check is done to see whether the cache object is null or not. If so, records are retrieved from login table and are stored in dataset and its cache is stored in the variable ds.

Tables Used: login
Database: Northwind

	login_id	login_password	hint_question	hint_ans
▶	bj72163	bharat	myname	name
	hs71747	hemant123	Your School Name	B.H.S. Pilani
	kp72087	kunal	wilson	name
	rc72119	reenal	myname	name

Figure 9.2-8: Login table.

Context :

- Caching is used for faster page rendering and it minimizes database hits and consumption of server resources.
- Page output caching reduces the network traffic by caching data for entire page.
- Fragment caching caches data for only specific section of page.
- Data caching caches frequently accessed data in server side memory variables .

Practice Session/s :

Logistics department wants to choose a vehicle for transportation , the supervisor may not know the list of vehicles currently available for transportation. In this case, we can keep a data table which keeps track of the available vehicles. Create a Website of logistics department and also keeps track of available vehicles. (Hint: Can use ComboBox/ListBox/DataGrid to bind the database to application).

Check List :

- Importance of caching.
- Using ‘cache’ directive and its attributes.
- Using different types of caching.

Common Error/s :

- Non inclusion of cache directive.
- Non inclusion of varybyparam attributes.

Exception/s :

- "Exception Details: System.Web.HttpException: Cache is not available "
  Try using "HttpContext.Current.Cache" instead of just "Cache" in your code. The compiler probably wasn't able to resolve the Cache type.
- The type or namespace name 'CacheDependency' could not be found (are you missing directive or an assembly reference?)

  Use namespace System.Web.Caching or refer to it as
  System.Web.Caching.CacheDependency

Lessons Learnt :

☑ Importance of storing data in the memory.
☑ Inclusion of directive and its attributes.

Best Practice/s :

- Caching is a technique that definitely improves the performance of web applications if one is careful to have a balance in terms of which data needs to be cached and parameter values for expiration policy.

VaryByParam attribute is mandatory. If you omit it, a run-time exception is always thrown. However, if you don't need to vary by parameters, set the attribute to **None.** The empty string is not an acceptable value for the **VaryByParam** attribute.

A cached ASP.NET page is served more quickly than a processed page, but not as quickly as a static HTML page.

The Cache class offers powerful features that allow customizing how items are cached and how long they are cached. For example, when system memory becomes scarce, the cache automatically removes seldom-used or low-priority items to free memory. This technique is referred to as scavenging, and is one of the ways that the cache ensures that out-of-date data does not consume valuable server resources.

Topic: Web.Config and Machine.Config

Estimated Time: 45 mins.

Objectives : At the end of the activity, the participant should understand the basics of

- Web.Config, Machine.Config and their use in an application.

Presentation :

Web.Config:

- Web.Config file is a configuration file for the Asp .net web application.
- Web.Config acts as a central location for storing the information that is to be accessed by web page.

- Web.Config can be used for any of the following configurations:-
  - Database connections
  - Session States
  - Error Handling
  - Security

Machine.Config:

- The Machine.Config file is at highest level of the configuration hierarchy.
- Machine.Config file is used to configure the application according to a particular machine.
- Usually, Machine.Config is not altered and only Web.Config is used while configuring applications.
- Web.Config inherits settings from Machine.Config

Scenario :

Mr. Shawn an administrator of Texas Company is developing a website. The administrator wants to get the Employee Details in the web pages. Administrator does not want to write connection string to database in every web page. So administrator decides to use web.config where connection string is specified once and if any changes in the connection string are required it can be done at the same place.

Demonstration/Code Snippet :

On click event, details are shown in grid view using the connection string stored in web.config.

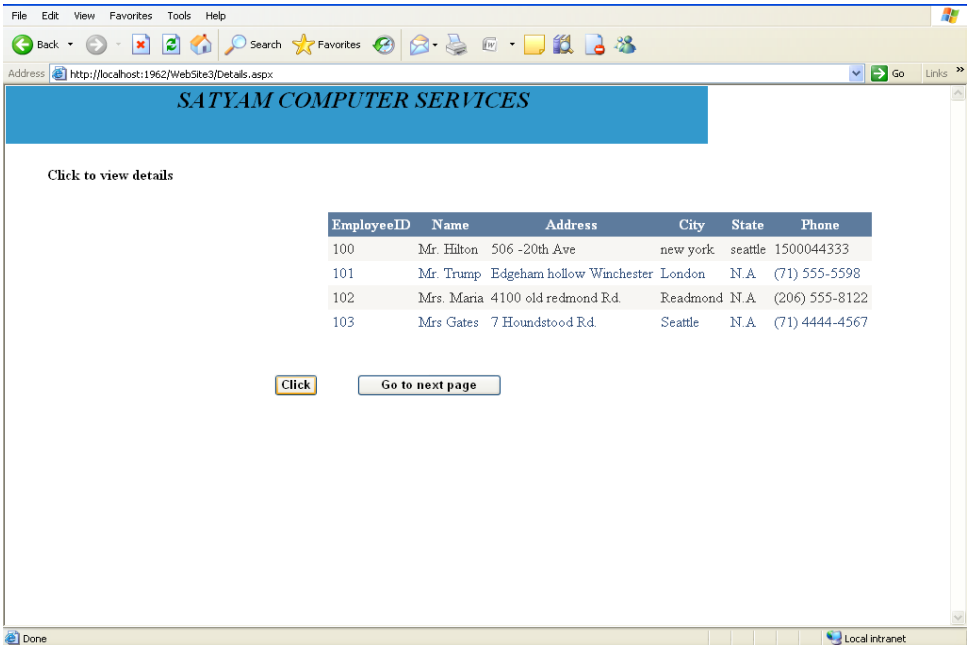


Figure 9.2-1: Employees Details viewed on Click event in Details.aspx.

After navigating to next page, click event shows the same details in grid view from the same connection string stored in the web.config file.

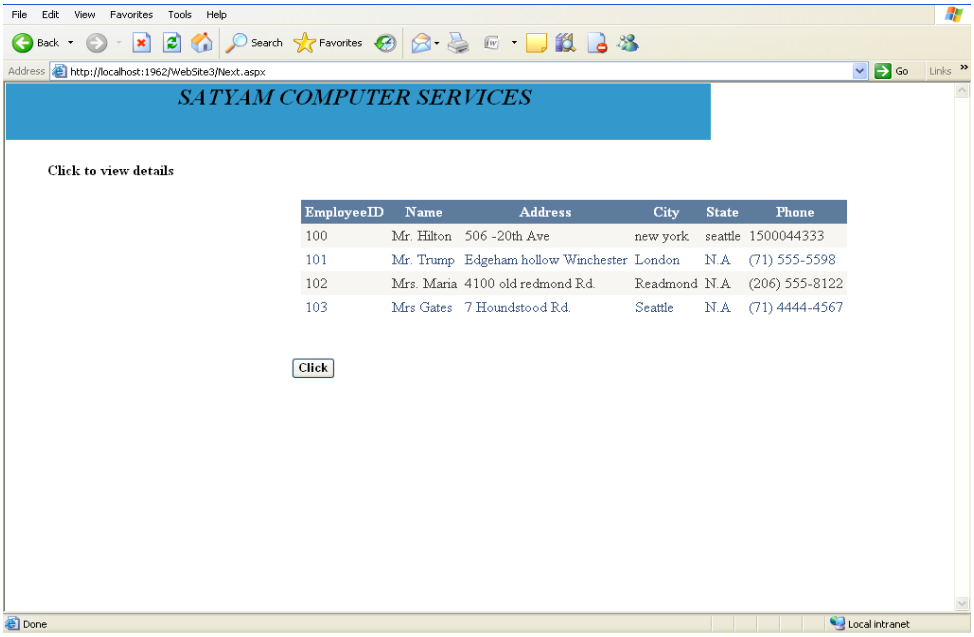


Figure 9.2-2: Employees Details viewed on Click event in Next.aspx.

Step 1: Create a new Web application project using visual studio 2005. Rename Default.aspx to Details.aspx.

Step 2: Drag and drop the server controls and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblView	Text:Click to view details
asp:button	btnClick	Text:Click
asp:button	btnNext	Text:Go to next page
asp:Grid View	gdvDetails	-

Step 3: Write the following code in page Load event handler, Click button event handler and Next button Event handler by double clicking on WebForm, click button and next button respectively of Details.aspx. On clicking next button Details.aspx page gets navigated to the Next.aspx page.

```
// Defining Namespaces
using System.Data.SqlClient;
using System.Windows.Forms;

public partial class _Default : System.Web.UI.Page
{  // Declaration
    SqlConnection mycon;
    SqlDataAdapter da = new SqlDataAdapter();
    SqlCommand cmd = new SqlCommand();

    protected void Page_Load(object sender, EventArgs e)
    {

        mycon = new SqlConnection();
        //Connection to the database Northwind
        mycon.ConnectionString =
       Convert.ToString(ConfigurationManager.ConnectionStrings["empl
       oyee"]);

    }
    protected void btnClick_Click(object sender, EventArgs e)
    {
        try
        {
            mycon.Open();
            cmd.Connection = mycon;
            cmd.CommandText = "select * from employee_details";
            cmd.CommandType = CommandType.Text;
            da.SelectCommand = cmd;
            DataSet ds = new DataSet();
            da.Fill(ds);
           gdvDetails.DataSource = ds.Tables[0];   // Binds the data
            to Gridview
            gdvDetails.DataBind();
        }
         catch(SqlException ex)
         {
            MessageBox.Show(ex.Message,"connection could not be
            opened");
         }
```

```
            finally
            {
            mycon.Close();// close connection to database
            mycon.Dispose();//Release all resources used by the object
            }
    }

     protected void  btnNext_Click(object sender, EventArgs e)
    {
         Response.Redirect("Next.aspx");// Navigate to next page
    }
```

Step 4: Create a new Web form in the same project called Next.aspx. In the Solution Explorer, right click on the name of your Web Application (Web.config) and select Add New Item/Web Form.

Step 5: Drag and drop the server controls and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblView	Text:Click to view details
asp:button	btnClick	Text:Click
asp:Grid View	gdvDetails	-

Step 6: Write in the Page load event handler and Click Button event handler by double clicking on Web form and Click button respectively of Next.aspx

```
// Defining Namespaces
using System .Data .SqlClient ;
using System .Windows .Forms ;

public partial class Next : System.Web.UI.Page
{
// Declaration
    SqlConnection mycon;
    SqlDataAdapter da = new SqlDataAdapter();
    SqlCommand cmd = new SqlCommand();

 protected void Page_Load(object sender, EventArgs e)
    {
        mycon = new SqlConnection();
        //Connection to the database Northwind
        mycon.ConnectionString =
       Convert.ToString(ConfigurationManager.ConnectionStrings["empl
       oyee"]);
    }
    protected void btnClick_Click(object sender, EventArgs e)
    {
        try
        {
            mycon.Open();
            cmd.Connection = mycon;
            cmd.CommandText = "select * from employee_details";
            cmd.CommandType = CommandType.Text;
```

```
            da.SelectCommand = cmd;
            DataSet ds = new DataSet();
            da.Fill(ds);
            // Binds the data to Gridview
            gdvDetails.DataSource = ds.Tables[0];
            gdvDetails.DataBind();
        }
        catch(SqlException ex)
        {
        MessageBox.Show(ex.Message,"connection could not beopened");
        }
        finally
        {
        mycon.Close();// close connection to database
        mycon.Dispose();//Release all resources used by the object
        }
    }
}
```

Step 7: In the Solution Explorer, right click on the name of the Web Application (Web.config) and select Add New Item/Web Configuration File. Write the following code in the <connectionStrings>….</connectionStrings> tag.

```
<connectionStrings>
<add name="employee" connectionString="Data Source=HSTSLC011;Initial
Catalog=Northwind;Persist Security Info=True;User
ID=sa;Password=satyam" providerName="System.Data.SqlClient"/>
</connectionStrings>
```

Tables used: login_col
Database: Northwind

EmployeeID	Name	Address	City	State	Phone
100	Mr. Hilton	506 -20th Ave	new york	seatte	1500044333
101	Mr. Trump	Edgeham hollow ...	London	N.A	(71) 555-5598
102	Mrs. Maria	4:00 old redmon...	Readmond	N.A	(206) 555-8122
103	Mrs Gates	7 Houndstood Rd.	Seattle	N.A	(71) 4444-4567

Figure9.2-3: login_col table.

Context :

- Use the same connection string in an entire application.
- To prevent the session data from being lost, even if the server crashes.
- For authorizing users and authenticating them.

Practice Session/s :

Create a Login form to validate users accessing the list of the products available for sale on the WebShoppe site. The login form should have Username and Password fields. The

connection string should be defined in the Web.Config file and the required table(s) can be used from the Northwind database.

Check List :

- Importance of different attributes in web.config.
- Importance of properties of different attributes.
- Use of Machine.Config

Common Errors :

- Non inclusion of closing tags.
- Connection String is not defined properly.

Exception/s :

- Any illegal entry within Web.config might not allow to build your web application.

Lessons Learnt :

- ☑ Understood the basics of Web.Config and Machine.Config
- ☑ Understood different attributes and its properties.

Best Practices :

- Always define the connection string in Web.Config file to make an application more efficient.

Asp.net runtime environment automatically restarts the application when Changes occurred in global.asax, machine.config or web.config in the application root.

Multiple Web.config files: You can have any number of Web.config files for each Web application. The rule is that each folder in your application can have only 1 such file.

Security alert: With the standard ASP.NET Machine.Config file, all configuration files are secured and cannot be downloaded by a client system. This allow for some protection of critical information such as user IDs and passwords for DSN sources, but keep in mind that any system can be hacked with enough time and effort. Always keep security in mind when planning the Web application.

Topic: Custom Error Handling

Estimated Time: 30 mins.

Objectives : At the end of the activity, the participant should understand

- Custom Error Handling

Presentation :

- To customize the default error page, one will have to change the default configuration settings of the application.
- There are three error modes in which an ASP.Net application can work:
- **Off Mode:** 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.
- **On Mode:** In case of "On" Mode, ASP.Net uses user-defined custom error page instead of its default error page for both local and remote users. If a custom error page is not specified, ASP.Net shows the error page describing how to enable remote viewing of errors.
- **RemoteOnly:** ASP.Net error page is shown only to local users. Remote requests will first check the configuration settings for the custom error page or finally show an IIS error.

Scenario:

In Koswik company manager Sang has good website with nice feature. But due to some errors which occurred at runtime, company is losing its clients. So in order to handle these errors, Sang now uses custom error handling. So Sang uses two methods Web.config and Application error in global.asax.

Demonstration/Code Snippet :

ErrDisplay.aspx page displays error message for error in code .

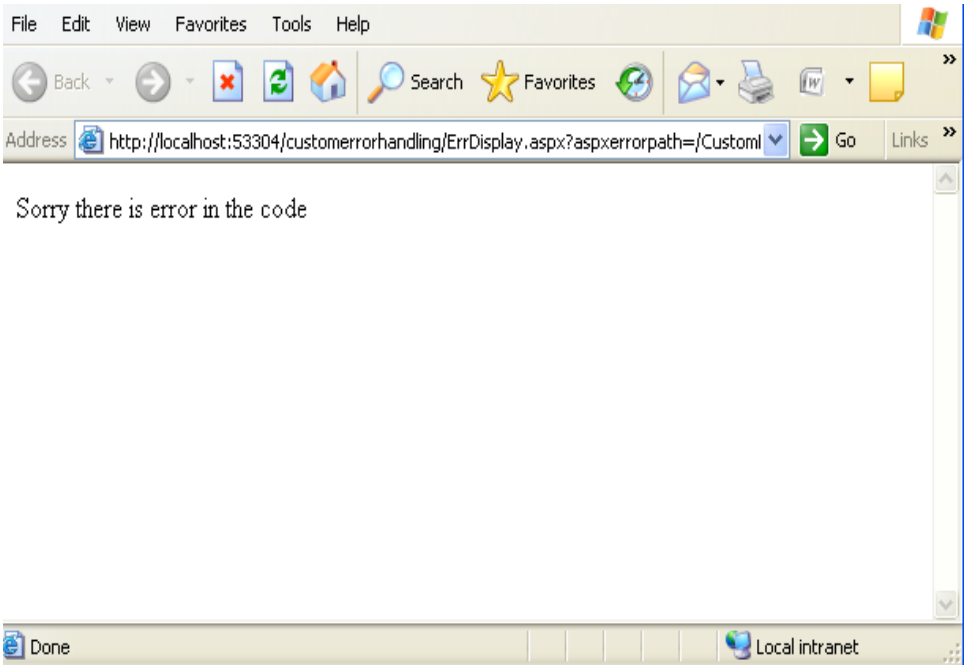


Figure 9.3-1: ErrDisplay.aspx displaying error for code.

Display.aspx displaying error for page not found.

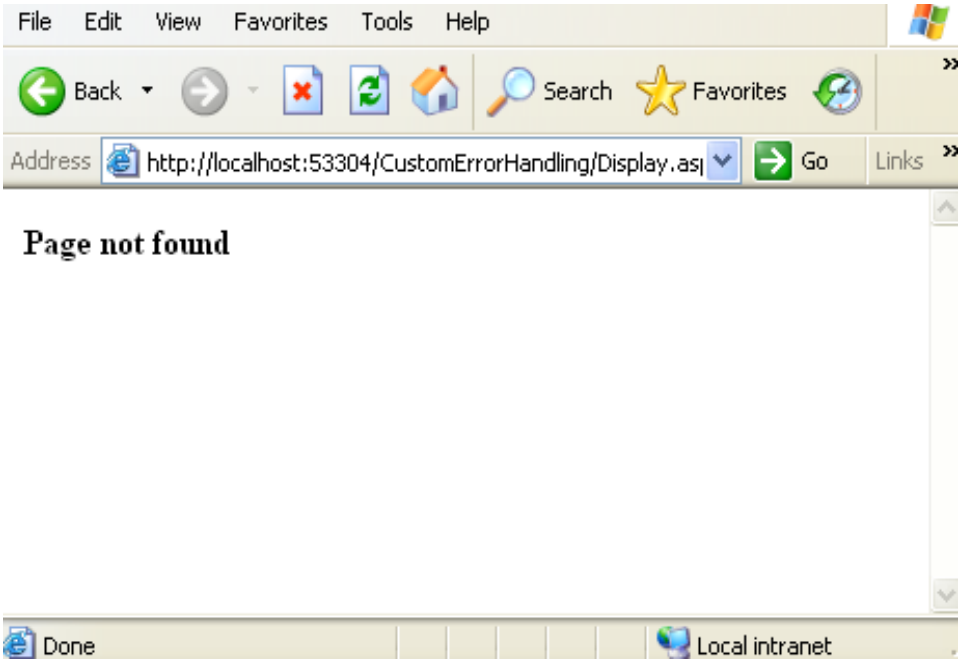


Figure 9.3-2: Display.aspx displaying error page when page is not found.

Using web.config:

Step 1: Create a new Web application project (CustomErrorHandling) using visual studio 2005. Rename Default.aspx to CustomError.aspx.

Step 2: Create a new Web form in the same project called ErrDisplay.aspx. In the Solution Explorer, right click on the name of Web Application (CustomErrorHandling) and select Add New Item/Web Form. In its design view write "Sorry there is error in the code" and then save it.

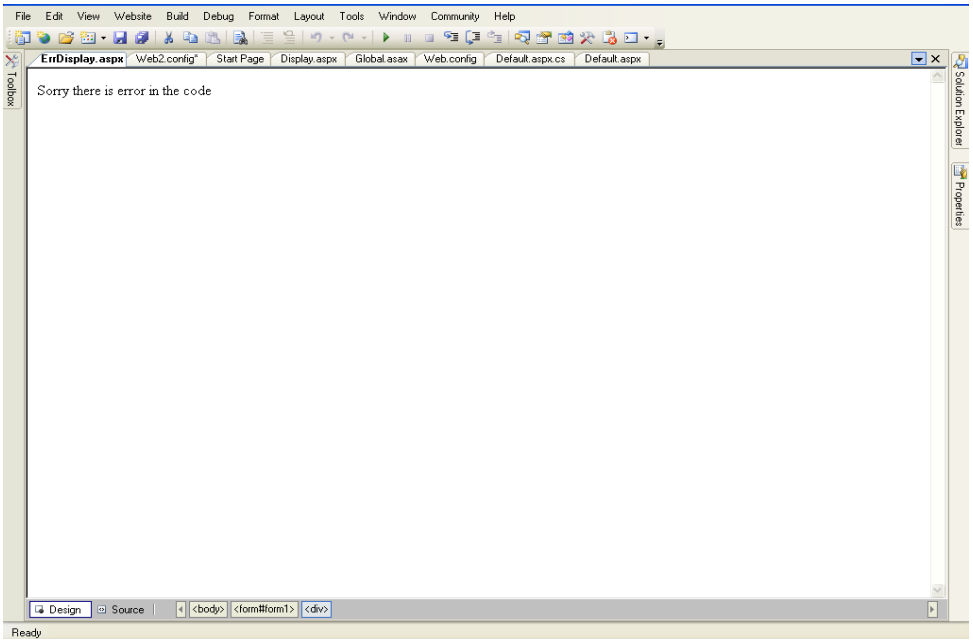


Figure 9.3-3: Design of ErrDisplay.aspx page.

Step 3: Write the event handler for Page load by double clicking on the Web Form.

```
protected void Page_Load(object sender, EventArgs e)
    {
        Dim //This is written so that error can occur
    }
```

Step 4: Add web configuration file. In the Solution Explorer, right click on the name of Web Application (CustomErrorHandling) and select Add New Item/Web Configuration File. Modify the same as follows:

```
<configuration>
    <appSettings/>
    <connectionStrings/>
    <system.web>
<!--The <customErrors> section enables configuration of what to do
if/when an unhandled error occursduring the execution of a request.
Specifically,it enables developers to configure html error pages
to be displayed in place of a error stack trace.-->

        <customErrors mode="On" defaultRedirect="ErrDisplay.aspx">
```

```
        <!--<customErrors mode="RemoteOnly"
        defaultRedirect="GenericErrorPage.htm">
        <error statusCode="403" redirect="NoAccess.htm" />
        <error statusCode="404" redirect="FileNotFound.htm" />-->
        </customErrors>
      </system.web>
      </configuration>
```

Step 5: Executing the application output "Sorry there is error in the code" is displayed.

Step 6: After modifying the Web.config file to the default Redirect attribute, the user is directed to the same custom error message irrespective of the type of the error. The type of the error is identified by the HTTP status code. Specific error messages can be specified, such as "Page not found" or "server crash" for specific status codes, as shown in the following code:

```
<configuration>
   <appSettings/>
   <connectionStrings/>
  <system.web>
    <!--    The <customErrors> section enables configuration
            of what to do if/when an unhandled error occurs
            during the execution of a request. Specifically,
            it enables developers to configure html error pages
            to be displayed in place of a error stack trace.
     <customErrors
      defaultRedirect="http://host1/MyError.aspx" mode="RemoteOnly">
      <!--defaultRedirect ="ErrDisplay.aspx">-->
      <error statusCode="500"
      redirect="Display.aspx"/>
      <error statusCode="403"
      redirect="accessdenied.aspx"/>
       <!--<error statusCode="403" redirect="NoAccess.htm" />
      <error statusCode="404" redirect="FileNotFound.htm" />-->
    </customErrors>
    <!--</customErrors>-->
  </system.web>
</configuration>
```

In this code, the error tag takes two attributes, status Code and redirect. The status Code attribute represents the value of the HTTP status code. The redirect attribute points to the error message file.

Using Global.asax:

Step 1: Add Global Application class in same project. In the Solution Explorer, right click on the name of Web Application (CustomErrorHandling) and select Add New Item/Global Application Class. Modify the same as follows:

```
void Application_Error(object sender, EventArgs e)
  {  // Code that runs when an unhandled error occurs
     Response. Redirect ("ErrDisplay.aspx");
  }
```

The Output for this is also same as for web.config.i.e. Sorry there is error in the code.

Context :

- Use Custom error handling to handle specific errors.

Practice Session/s :

- Identify the way to handle Specific error.

Check List :

- Importance custom error handling using web.config. and global.asax.

Common Errors :

- Not specifying mode in web.config.

Exception/s :

- No Specific exception.

Lessons Learnt :

☑ Understand the basics of custom error handling.

Best Practices :

It is advisable to use Custom Error.

Crossword: Unit-9 **Estimated Time: 10 mins.**

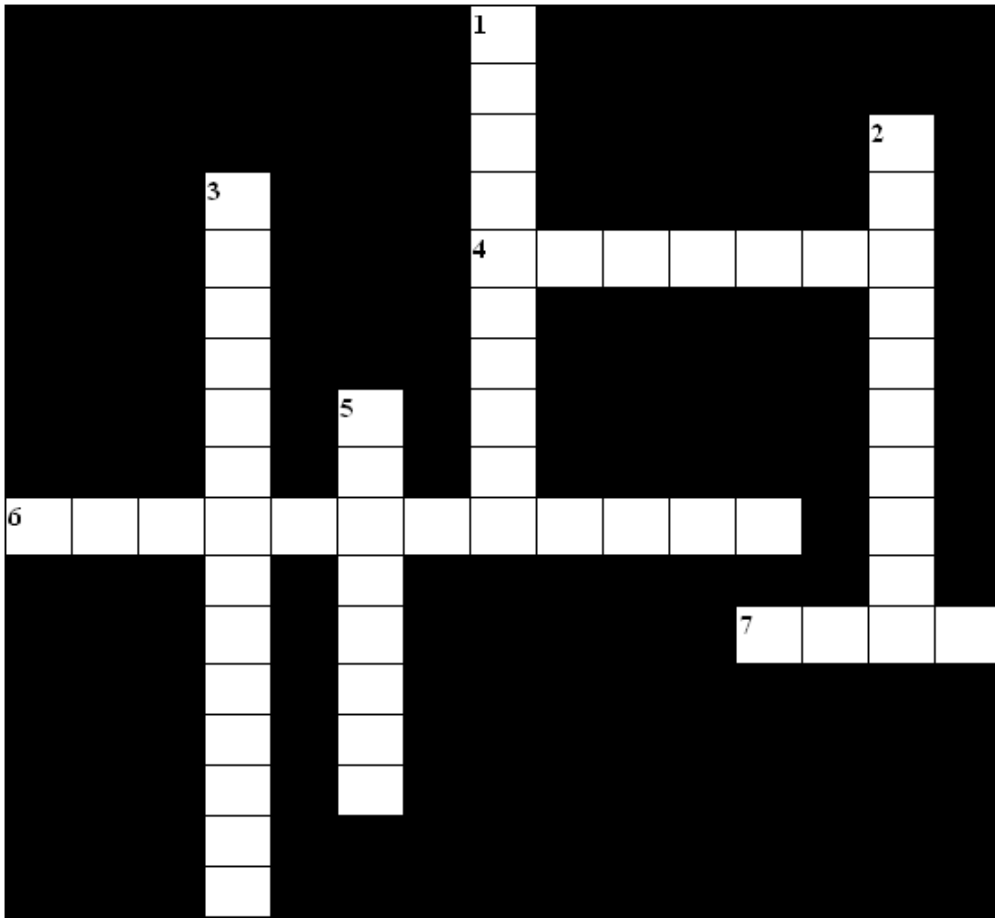


Fig. 9-1

Across:

4. Which is a technique of persisting data in memory for immediate access to requesting program calls, which is used to improve the performance of the website.(7)

6. Which type of caching allows specific portions(fragment) of the page to be cached rather than caching the whole page.(12)

7. Which type of caching allows, caching of frequently accessed data in server side memory variables.(4)

Down:

1. Which file is used as a configuration file for the Asp .net web application, which acts as a central location for storing the information which is to be accessed by web page.(10)

2. Which type of caching allows the output of the pages to be cached using an output cache engine.(10)

3. This file is at highest level of the configuration hierarchy used to configure the application according to a particular machine.(14)

5. Which parameter is used to store the time (in seconds), of how long the output should be cached.(8)

10.0 Securing ASP.Net Web Application

Topics

- 10.1 Web Application Security
- 10.2 MMC Snap-In
- 10.3 Website Administration Tool
- 10.4 Web Parts
- 10.5 Crossword

Topic: Web Application Security

Estimated Time: 60 mins.

Objectives : At the end of the activity, the participant should understand

- Authorization
- Authentication including windows, form and passport based.

Presentation :

- The following illustration shows the relationship among the security systems in ASP.NET.

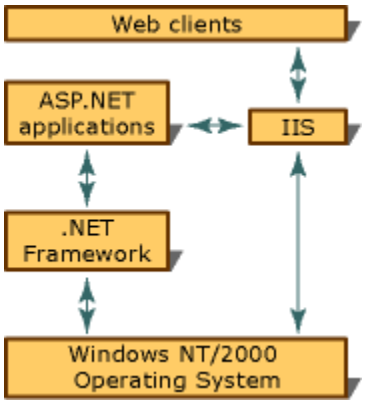


Figure 10.1-1: Relationship among security system.

IIS

- IIS deciphers and optionally authenticates the request.
- IIS always assumes that a set of credentials map to a Windows NT account and uses them to authenticate a user.
- There are three different kinds of authentication available in IIS 5.0
  - Basic Authentication.
  - Digest Authentication.
  - Integrated Windows Authentication.

Web Clients

- A program capable of communicating with Web servers, requesting and receiving information from them, and processing it for display or other uses.

Types of Security:

- **Authorization:** Determines whether an identity should be granted the requested type of access to a given resource.

- **Authentication:** It is the process of obtaining identification credentials from a user (such as name and password), and validating those credentials against some authority. Types:
  - **Windows Based:** use any of the authentication methods that might have already been performed by IIS before passing the request to the ASP.NET application.
  - **Form Based:** used for personalization, where content is customized for a known user.
  - **Passport Based:** is a centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites.

Authorization:

Scenario :

Mr. Johnson the owner of WallMart, wants to create an application that accepts Username and Password from a user before the user is able to login to the WallMart website. It allows only specified users to have access to all the pages of WallMart website. So Johnson asks Lui, the developer to add this functionality. So Lui uses the concept of Authorization.

Demonstration/Code Snippet :

Login page is shown as:

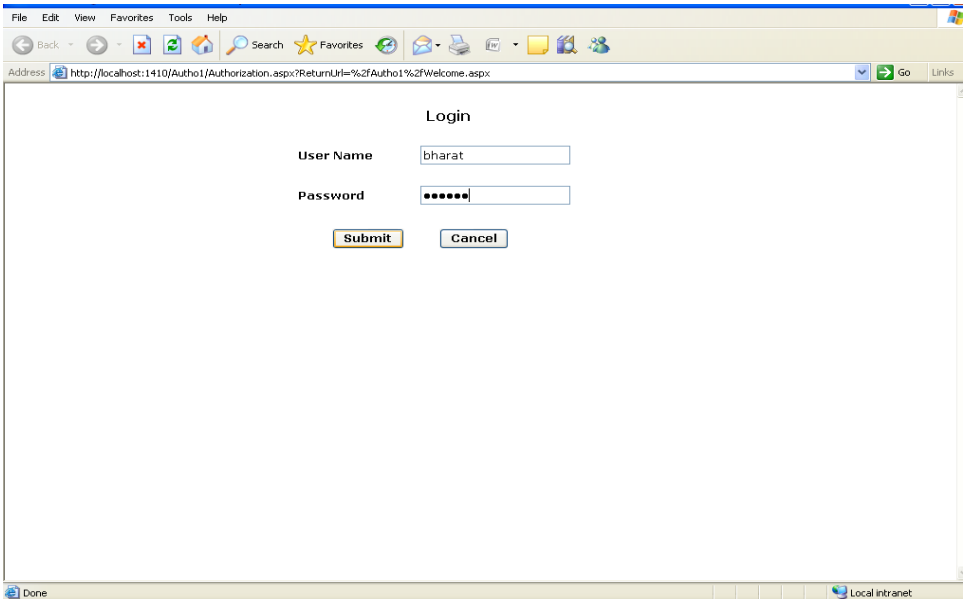


Figure 10.1-2 : Welcome.aspx page.

Step 1: Create a new Web application project called Authorization using visual studio 2005. Rename Default.aspx to Welcome.aspx.

Step 2: Insert HTML Table from toolbox in Welcome.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Login
asp:label	lblUsername	Text:User Name
asp:label	lblPwd	Text:Password
asp:textbox	txtUsername	-
asp:textbox	txtPwd	-
asp:button	btnSubmit	Text:Submit
asp:button	btnCancel	Text:Cancel

Authorization enables to restrict the access of an authenticated user to parts of the application or Web site for which a user is already authenticated.

Step 3: In the Solution Explorer, right click on the name of the Web Application (Authorization) and select Add New Item/Web Configuration File. Write the following code in Web.Config file:

```
<!-- Athentication is used to check whether intended user is getting logged in or not-->

 <authentication mode ="Forms">
   <forms loginUrl ="Authorization.aspx">
```

```
    <credentials passwordFormat ="Clear">
      <user name ="bharat" password ="bharat"/>
      <user name ="kunal" password ="kunal"/>
    </credentials>
  </forms>
</authentication>

<!-- Authorization is used to check whether the Authenticated user have access to all part
of the website or not-->
<authorization>
  <!-- Here bharat user have rights to access the whole website-->
  <allow users ="bharat"/>
  <!-- Rest of the users are deny-->
  <deny users ="*"/>
</authorization>
```

The **<*authorization*>** element is specified in the Web.config file. The **<*authorization*>** element includes two child elements, **<*allow*>** and **<*deny*>.** Specifying the list of users allowed to access the Web site in the <***allow***> element and the list of users not allowed to access the Web site in the <***deny***> element. Code the **<*authorization*>** element immediately after the <***authentication***> element.

Step 4: Write the event handler for Submit button event handler by double clicking on Submit button of Welcome.aspx:

```
protected void btn_Sumbit_Click(object sender, EventArgs e)
{
 if (FormsAuthentication.Authenticate(txtUserName.Text, txtPwd.Text)
  == true)
 {
 FormsAuthentication.RedirectFromLoginPage(txt_UserName.Text,
  true);
 }
}
```

Authenticate: The **Authenticate** method is used for checking the credentials supplied by a user against a specified data source.

RedirectFromLoginPage : The **RedirectFromLoginPag*e*** method is used to redirect a user to the resource that the user has initially requested.

Step 5: Run the application. When the Submit button is clicked Username and Password will be checked in the web.config file and the user is authenticated. And then authenticated user will be allowed to veiw the website.

The form that must be viewed to user after user successully logs in, that form must be set as Start Page.

Authentication:

a) Window Based :

Scenario:

Mr.Wilson of Fortune enterprises ltd. wants to allow the specific users having domain login to access the company Website. If a user has logged onto a local computer as a domain user, user won't need to be authenticated again when accessing a network computer in that domain. So Wilson suggests developer about the idea and implements it using windows authentication.

Demonstration/Code Snippet :

If the user has entered into some other domain user account other than the **bubbles** and tries to access the website sees a large **"Access is denied"** statement in browser window.

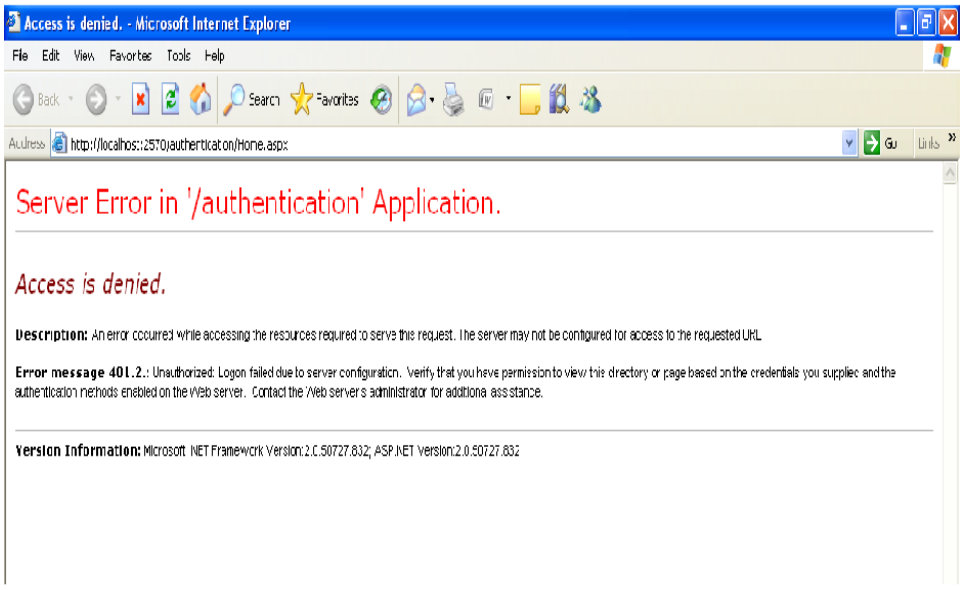


Figure 10.1-3: Page showing Access is denied.

Step 1: Create Users from Control panel, open administrative tools/Computer management utility. Further expand system tools node / local users / groups node and select the users folder.

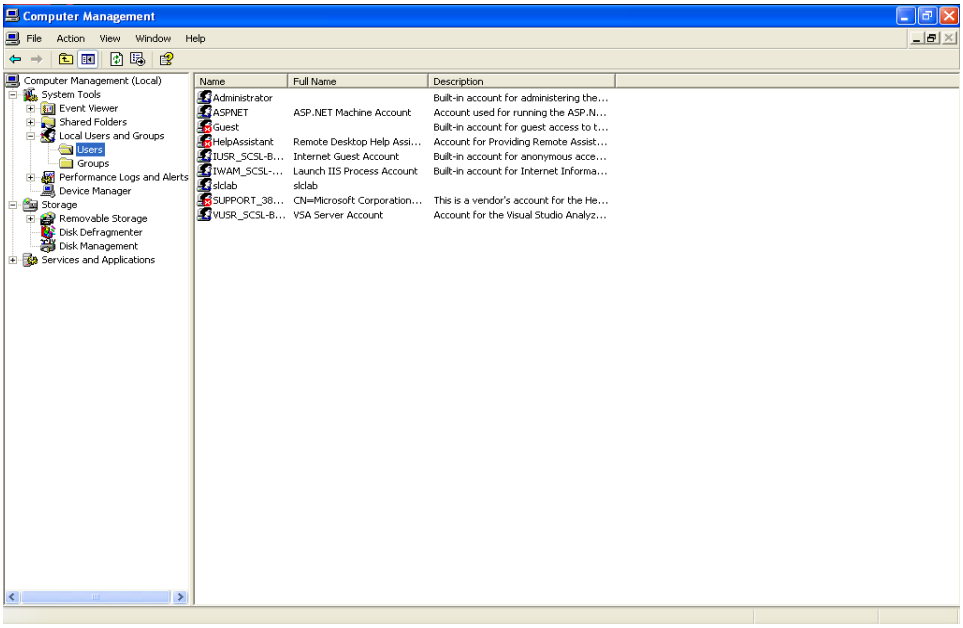


Figure 10.1-4: Creating a new user.

Step 2: Right-click user folder and select new user.

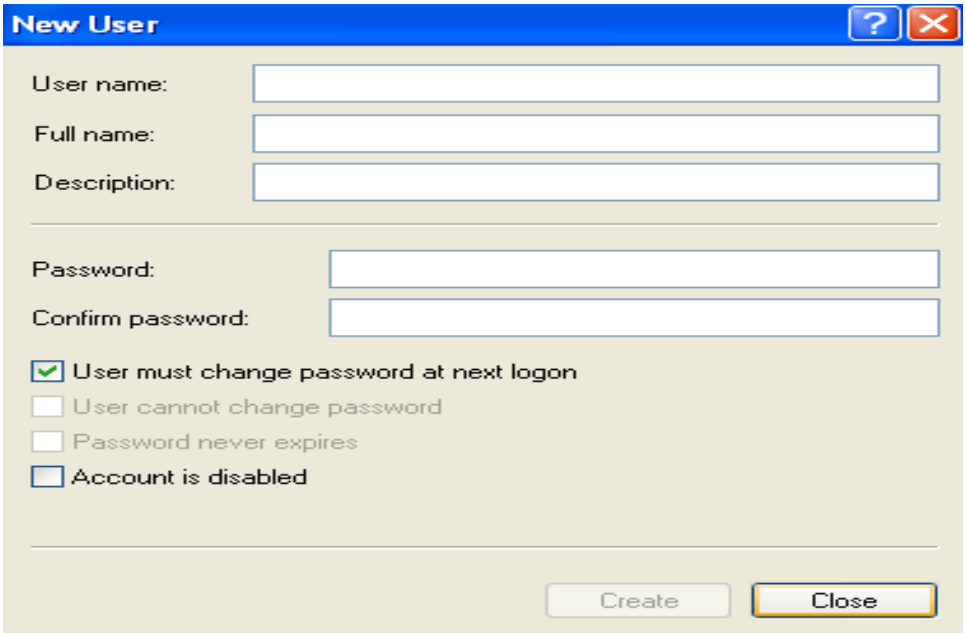


Figure 10.1-5: Fill the details.

Step 3: Fill the details and uncheck the check box to change the password and click the Create button. User **bubbles** is created.

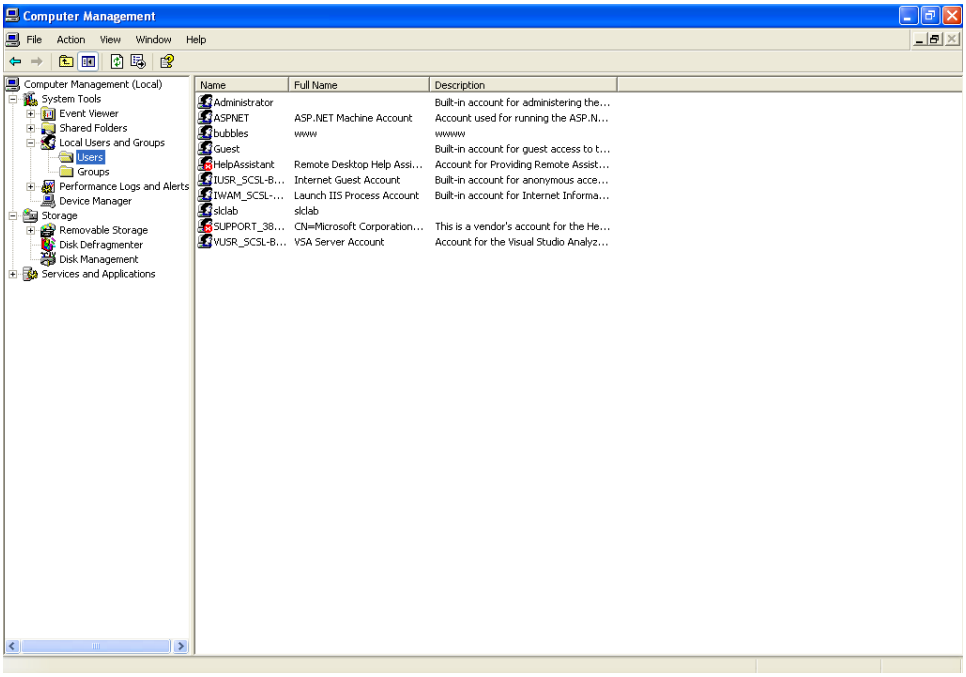


Figure 10.2-6: New User is created.

Windows authentication allows bubbles user to be authenticated based on the Windows account on the Windows domain. While Windows authentication might be convenient for intranet Web applications where all users are expected to have a Windows user account in the domain, it is not applicable for general Web users from the Internet who may or may not even be accessing your Web application from a Windows machine.

Step 4: Create a new Web application project called WindowAuthentication using visual studio 2005. Rename Default.aspx to Welcome.aspx.

Step 5: In the Solution Explorer, right click on the name of the Web Application (Window Authentication) and select Add New Item/Web Configuration File. Write the following code in Web.Config file

```
<!--The <authentication> section enables configuration
   of the security authentication mode attribute set to
windows is used by ASP.NET to identify an incoming user.-->
<authentication mode="Windows" />

<!--The <authorization> element is used to define
   specifics about the users or groups who are
   permitted access to the application.-->
     <authorization >

<!--The <allow> element is used to allow a specific user.-->
```

```
        <allow users ="bubbles"/>

    <!--The <deny> element specifies that all users are denied access to the application.-->
        <deny users ="*"/>
     </authorization>
```

Authentication mode as ***window*** specifies that users will be authenticated using any form of IIS authentication.

b) Form Based:

Scenario:

Mr. Brown of Kansas Company wants to restrict the users from accessing the intermediate pages of the Web site. It is required that all the users access Login page before navigating to intermediate pages. Brown would like to provide access only to the customers of the company under a given domain of **bubbles**.

Demonstration/Code Snippet :

Login page comes on executing the application.

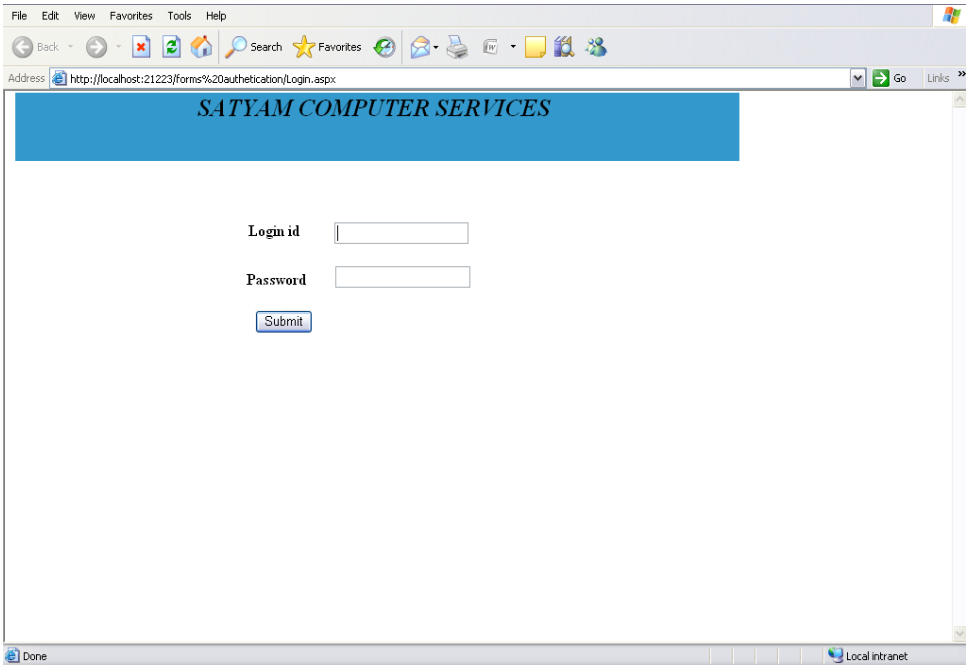


Figure 10.1-7: Login.aspx page.

On successful login user views home page.

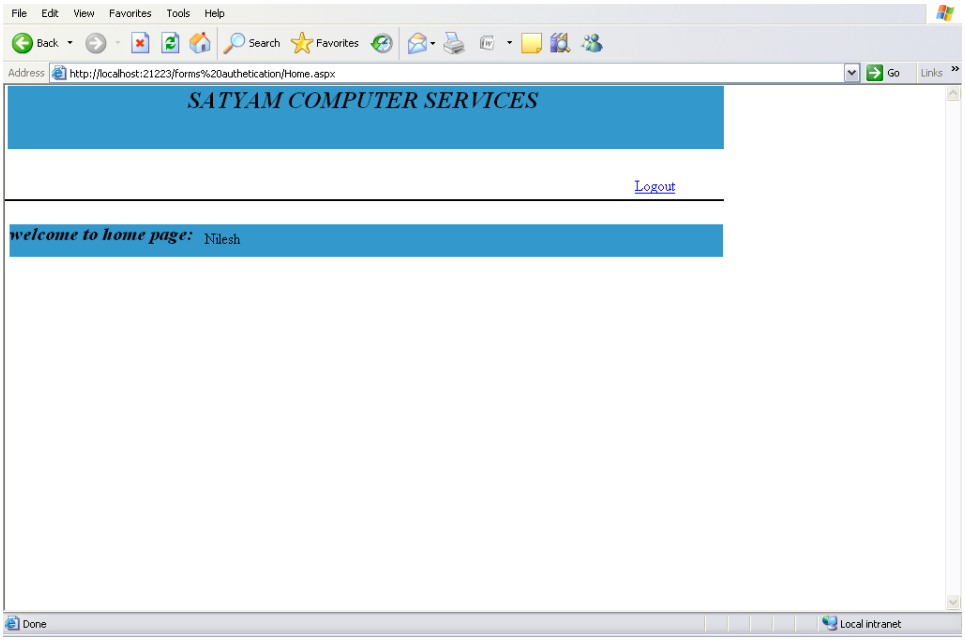


Figure 10.1-8: Home.aspx page.

If any user tries to open the application directly through the home page url, it will automatically redirect to the login page. So, the user will not be able to access the application without logging in. Here, it is shown by copying the URL of home page in browser.

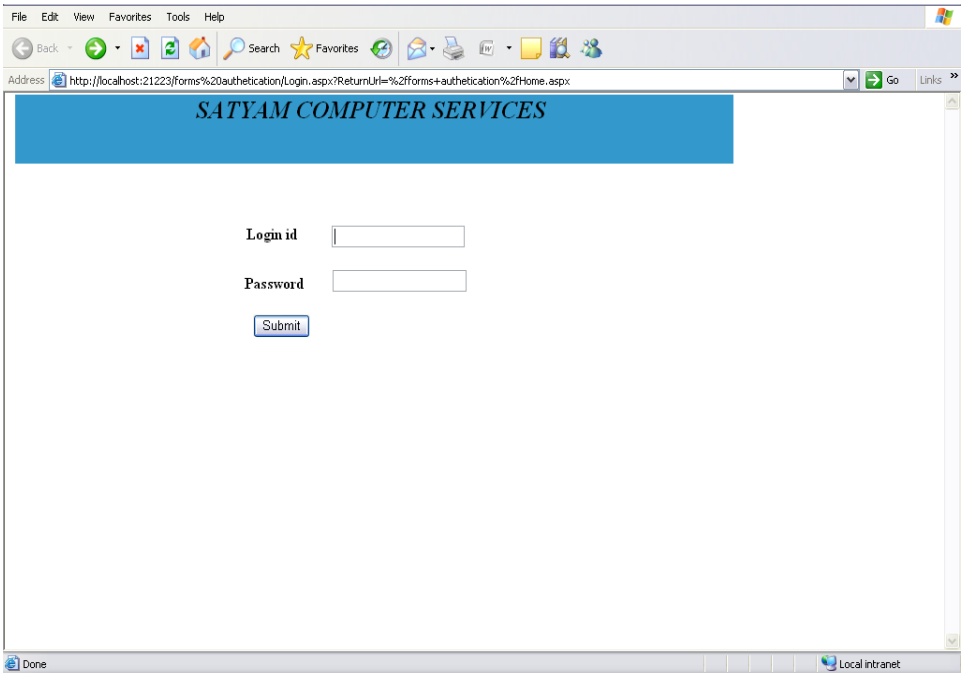


Figure 10.1-9: Showing Security.

The **Forms Authentication** class is a .NET framework library class in the

System.Web.Security namespace. The Authenticate method is a static (or shared in VB.NET terminology) method of this class which takes in the user name and password as strings, and returns a Boolean value depending on whether the credentials are valid based on the credential store used.

Step 1: Create a new Web application project called FormAuthentication using visual studio 2005. Rename Default.aspx to Login.aspx.

Step 2: In the Solution Explorer, right click on the name of the Web Application (Form Authentication) and select Add New Item/Web Configuration File. Write the following code in Web.Config file:

```
<!--  The <authentication> section enables configuration
          of the security authentication mode used by
          ASP.NET to identify an incoming user.-->
         <authentication mode="Forms">
 <!--    The attributes like name gives the form name,
          loginUrl shows that if any user tries to open any
          unauthenticated page it will redirect to the login page,
          timeout abandon the session by itself.-->
         <forms name="auth" loginUrl="Login.aspx"
        timeout="20"></forms>

          </authentication>
   <!--The <authorization> element is used to define
          specifics about the users or groups who are
          permitted access to the application.-->

      <authorization>
     <!--Deny user does not allow other users to access
        the application except the authenticated users.-->
          <deny users="?"/>
    </authorization>
```

Use the <***forms***> child element of the <***authentication***> element to specify the name of the login form in its ***loginUrl*** attribute. In addition, specify the default extension of the cookie that is issued to an authenticated user by using the ***name*** attribute of the <***forms***> element.

Step 3: Insert HTML Table from toolbox in Login.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Satyam Computer Services
asp:label	lblLoginId	Text:Login Id

asp:label	lblPwd	Text:Password
asp:textbox	txtLoginId	-
asp:textbox	txtPwd	-
asp:button	btnSubmit	Text:Submit

Step 4: Write the event handler for Page Load and Submit button by double clicking on WebForm, submit button respectively of Login.aspx.

```
// Defining Namespaces
using System.Data.SqlClient;
using System.Windows.Forms;

public partial class _Default : System.Web.UI.Page
{

SqlConnection con;//connection to database Northwind
string
connectionstr="server=hstslc011;database=northwind;uid=sa;pwd=satyam
";


    protected void Page_Load(object sender, EventArgs e)
    {
        txtlogin.Focus();
    }

  protected void btnsubmit_Click(object sender, EventArgs e)
    {
       con = new SqlConnection(connectionstr);
       DataSet ds;
       ds = new DataSet();
       SqlDataAdapter adap;
       string sql;
       sql = string.Format("select * from login_col");
       adap = new SqlDataAdapter(sql, con);
       adap.Fill(ds);// fill the dataset
        if(IsPostBack)
        {
           for (int i = 0; i < ds.Tables[0].Rows.Count - 1; i++)
           {
           if (txtlogin.Text ==
      ds.Tables[0].Rows[i].ItemArray[0].ToString() && txtpwd.Text ==
      ds.Tables[0].Rows[i].ItemArray[1].ToString())
              {
      FormsAuthentication.RedirectFromLoginPage(txtlogin.Text,
      false);//redirect the login page to home page
      Session["name"]  =  txtlogin.Text;//  Store  the  username  in
      session variable.
               }
              Else
             {
              MessageBox.Show("Invalid user");
             }
```

```
            }
        }
    }
}
```

The **RedirectFromLoginPage** static/shared method of FormsAuthentication redirects an authenticated user back to the originally-requested URL.This method looks at the URL's ReturnUrl parameter to determine where is the original URL to redirect to. In the event that this parameter is missing, this method will redirect the user to Login.aspx.

Step 4: Create a new Web form in the same project called Home.aspx. In the Solution Explorer, right click on the name of your Web Application (Form Authentication) and select Add New Item/Web Form.

Step 5: Drag and drop the server controls in Home.aspx and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text:Satyam Computer Services
asp:label	lblName	-
asp:label	lblwelcome	Text:Welcome to the Home Page:
asp:linkbutton	lnbtnlogout	Text:Logout

Step 6: Write the event handler for Page Load and logout linkbutton by double clicking on WebForm, logout linkbutton respectively of Home.aspx.

```
public partial class Home : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        lblName.Text = Session["name"].ToString();
        Session.Abandon();
    }
    protected void lnbtnlogout_Click(object sender, EventArgs e)
    {
        FormsAuthentication.SignOut();//Removes the authentication
       ticket
        FormsAuthentication.RedirectToLoginPage();
    }
}
```

SignOut method is used to log a user off from the Web application.

Tables Used: login_col

Database: northwind

	loginid	password
▶	Nilesh	nilesh
	Wilson	wilson

Figure 10.1-9: Login_col table.

Context :

- Use when security to an application is needed.
- Using different types of Authentication.

Practice Session/s :

Create a login form to validate users accessing the discount list of the products available for sale on the Web Shoppe site. The login form should have username and password field. A Welcome message should be displayed on correct login entry.

Check List :

- Importance of IIS.
- Importance of Authorization and Authentication.
- Importance of Windows, Form and Passport based authentication.

Common Error/s :

- Non installation of SDK.
- Non inclusion of 'mode' attributes.
- In case of Windows authentication impersonates is not set to **true.**

Exception/s :

- If we provide invalid login credentials, we will get Server Error '/authentication' Application and Access Denied.

Lesson/s Learnt :

☑ Understanding the architecture of security systems.
☑ Role of Authentication and Authorization.
☑ Role of Windows, Form and Passport based authentication.

Best Practice/s :

- Always use authorization and authentication where login is required.

- Always hash password values for credentials stored in the web.config file. Hashing is nothing more than applying one-way encryption to the password.

Tricky Authorization: The ***<authorization>*** element is processed from top to bottom. What this means is that you can place the <deny> element before the <allow> element, and the users specified in the <deny> element will 'override' those specified in the <allow> element since it appears after <deny>.

The <authentication> element can be used only in the web.config that is in the root directory of an application. Attempting to use it in a subdirectory will cause an error. This means that only one authentication type can be defined for each application.

Unlike the <authentication> element, the <authorization> element is not limited to the web.config file in the root of the web application. Instead, it can be use in any subdirectory, thereby allowing to set different authorization settings for different groups of pages.

Topic: MMC Snap-In

Estimated Time: 20 mins.

Objectives :

At the end of the activity, the participant should understand the use of

- Basics of MMC Snap-In.

Presentation :

- The Microsoft Management Console (MMC) snap-in for ASP.NET provides a graphical user interface (GUI) for manipulating ASP.NET configuration settings at the global, Web site, and application levels on the local computer.
- The tool prevents from making invalid settings, allows to control whether settings can be inherited by Web applications, and helps to manage the dependencies between levels of the configuration hierarchy.
- The ASP.NET MMC snap-in is integrated with the Internet Information Services (IIS) Manager snap-in, making it easy to work with IIS settings and ASP.NET configuration settings that apply to a Web site or application.
- The ASP.NET MMC Snap-In includes the following tabs:
  - **General**—enables to configure connection strings and application settings.
  - **Custom Error**—enables to configure custom error pages.
  - **Authorization**—enables to configure authorization rules.
  - **Authentication**—enables to configure Forms, Windows, or Passport authentication.
  - **Application**—enables to configure application settings such as application-wide Master Pages and Themes.
  - **State Management**—enables to configure Session state.
  - **Location**—enables to apply configuration settings to a particular folder or page.

Scenario :

Administrator of Johnson Company is developing the website for his company. Administrator wants to configure Website or application.

Demonstration/Code Snippet :

Go to control panel/ administration tool/ Internet Information Services (IIS) Manager and right click the virtual directory of your application and select properties.

Default Website property window is displayed.

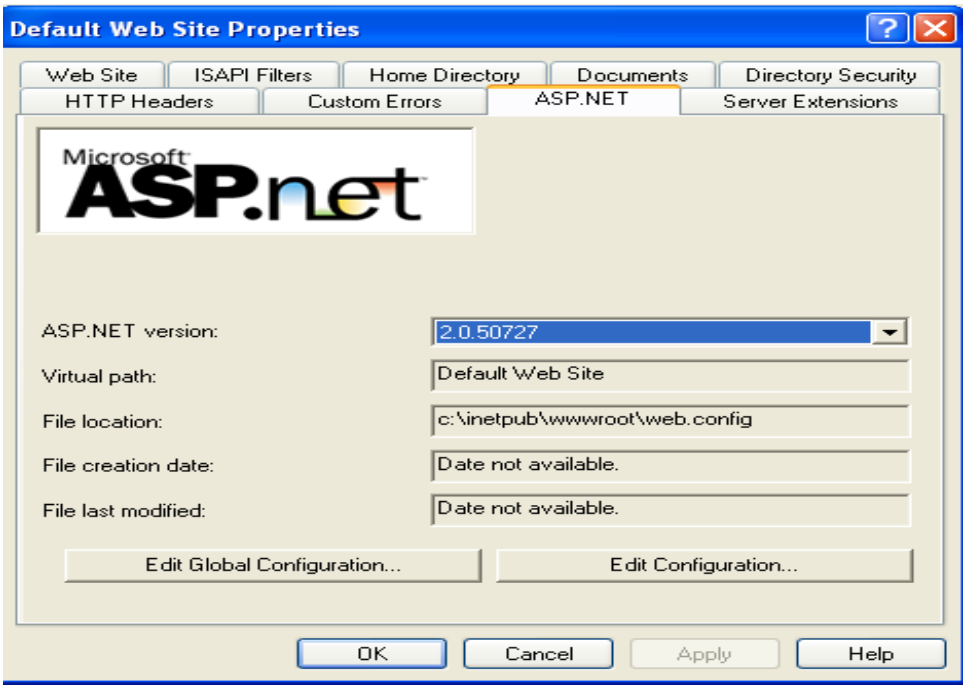


Figure 10.2-1: Default Website Property.

On ASP.NET Tab, Click Edit Configuration button, ASP.NET Configuration Settings dialog button appears. To Add custom application specific settings, Use General Tab to add key/value pairs. Remaining Tabs are self-describing. These tabs are used to configure settings.

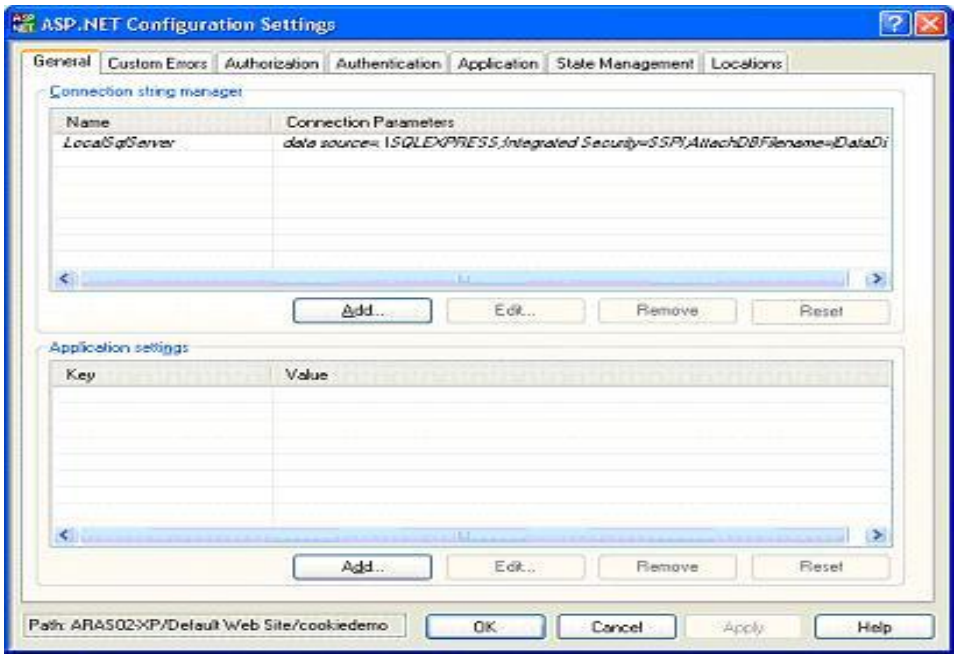


Figure 10.2-2: ASP.NET Configuration Settings.

Context :

- Use MMC Snap-In for ASP.Net configuration settings.
- Use MMC Snap-In for configuring IIS settings.

Practice Session/s :

- Identify MMC Snap-in use in Configuring Website.

Check List :

- Importance of MMC Snap-In.
- Inheritance of settings by Web Applications.

Common Error/s :

- Ensure that you had selected latest version of ASP.Net in Default website properties window.

Exception/s :

- If you don't select the latest version of ASP.Net, it might not alow to deploy your website.

Lesson/s Learnt :

- ☑ Basics of MMC Snap-In.
- ☑ Using GUI for manipulating configuration settings.

Best Practice/s :

- It is recommended to configure website with latest version of ASP.Net.

Topic: Website Administration Tool

Estimated Time: 40 mins.

Objectives : At the end of the activity, the participant should understand the use of

- Administration tool
- Role based security

Presentation :

- The Web Site Administration Tool allows one with administrative privileges for a Web site, to use a Web browser to manage the ASP.NET application locally or remotely.
- The Web Site Administration Tool includes a tabbed interface with tabs for **Security**, **Application**, and **Provider**.
  - Security, for managing user accounts and roles.
  - Application, for managing general application settings.
  - Provider, for testing and assigning providers for membership and role management
- The Web Site Administration Tool is useful for two main reasons.
  - First, it abstracts the XML configuration into an easy-to-use interface.
  - Second, it provides administration features via a browser.

Scenario:

In K-Lite Company owned by Wilson have important informations contained in company's website, accessed by administrator and other users. Wilson wants that some important informations should only be accessed by administrator and not by other users. So for this Wilson asks website developer to use Administration tools to provide role based security.

Demonstration/Code Snippet :

Login.aspx page for Admin and User

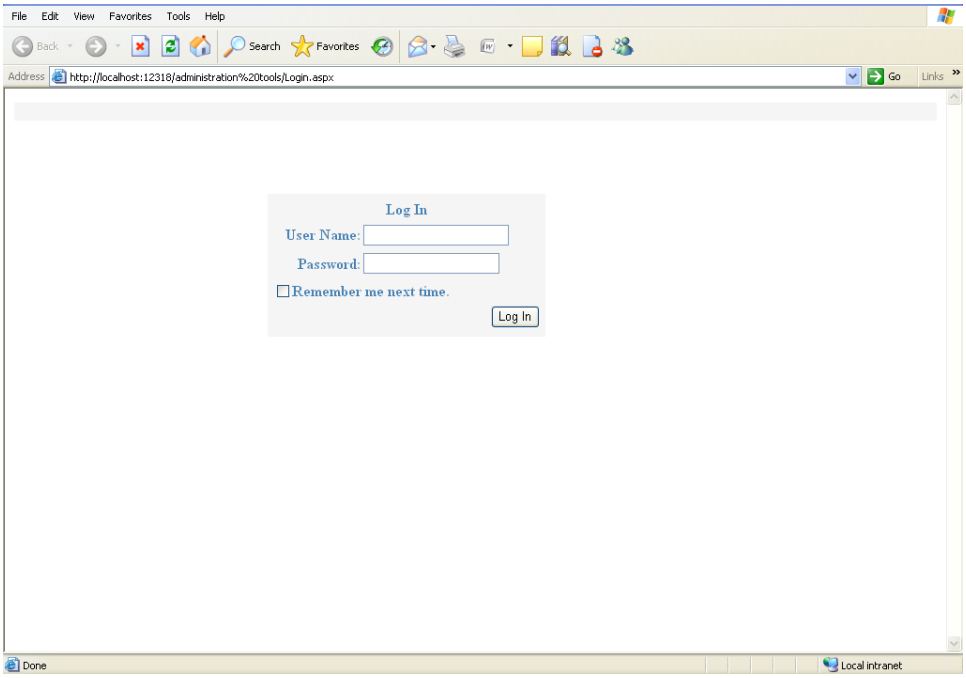


Figure 10.3-1: Login.aspx for user and Admins.

After Login admin view Home page as:

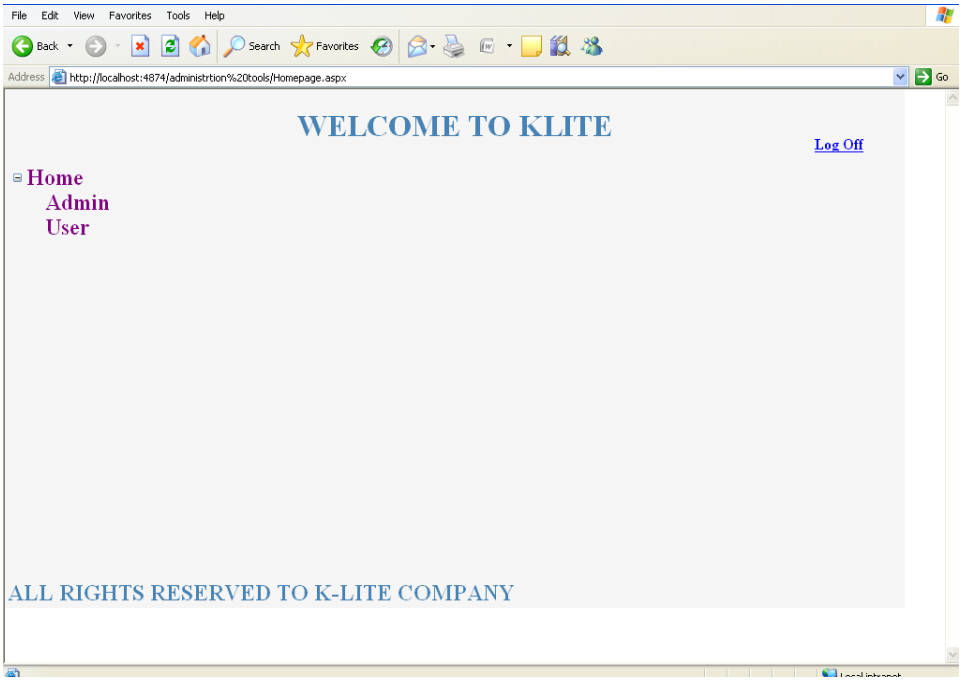


Figure 10.3-2: Homepage.aspx for Admins.

After login User will see home page as:

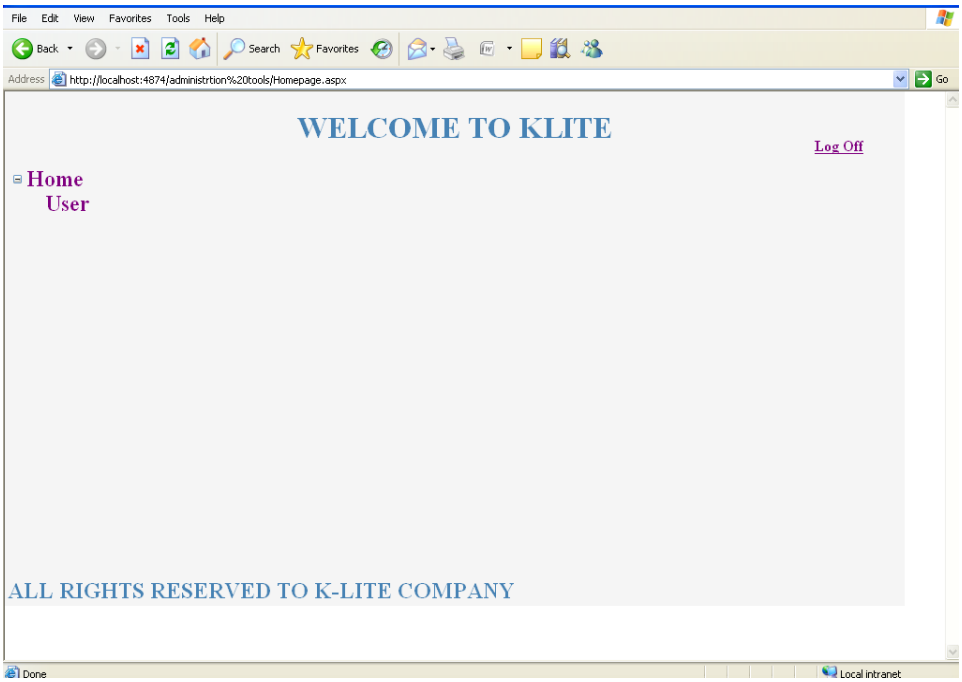


Figure 10.3-3:Homepage.aspx for users.

Step 1: Create a new Web application project called Administration Tool using visual studio 2005. Rename Default.aspx to Login.aspx

Step 2: Insert HTML Table from toolbox in Login.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:login	loginKlite	DestinationUrl: Homepage.aspx

Step3: Create a new Web form in the same project called Homepage.aspx. In the Solution Explorer, right click on the name of Web Application (Administration Tool) and select Add New Item/Web Form.

Step 4: Insert HTML Table from toolbox in Homepage.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:treeview	trvNavigation	-

Step 5: Create two folders Admin and Users by right clicking on the name of Web Application (Administration Tool) in solution explorer and add new folders(Admin and Users). Selecting Admin folder create webform(Admins.aspx) by right clicking on the Admin folder and select Add New Item/Web Form. Similarly create webform (User.aspx)

Step 6: Create a site map in the same project. In the Solution Explorer, right click on the name of Web Application (Administration Tool) and select Add New Item/site map and write the following.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-
1.0" >
<siteMapNode url="Homepage.aspx" title="Home"description="Homepage">
<siteMapNode url="~/Admin/Admins.aspx" title="Admin"
description="Adminpage" />
<siteMapNode url="~/Users/User.aspx" title="User"
description="Userpage" />
</siteMapNode>
</siteMap>
```

Step 7: In tree view smart tag select new data source and select site map provider. It will automatically show the menu.

Step 8: Select from menu/ website/ ASP.NET configuration.

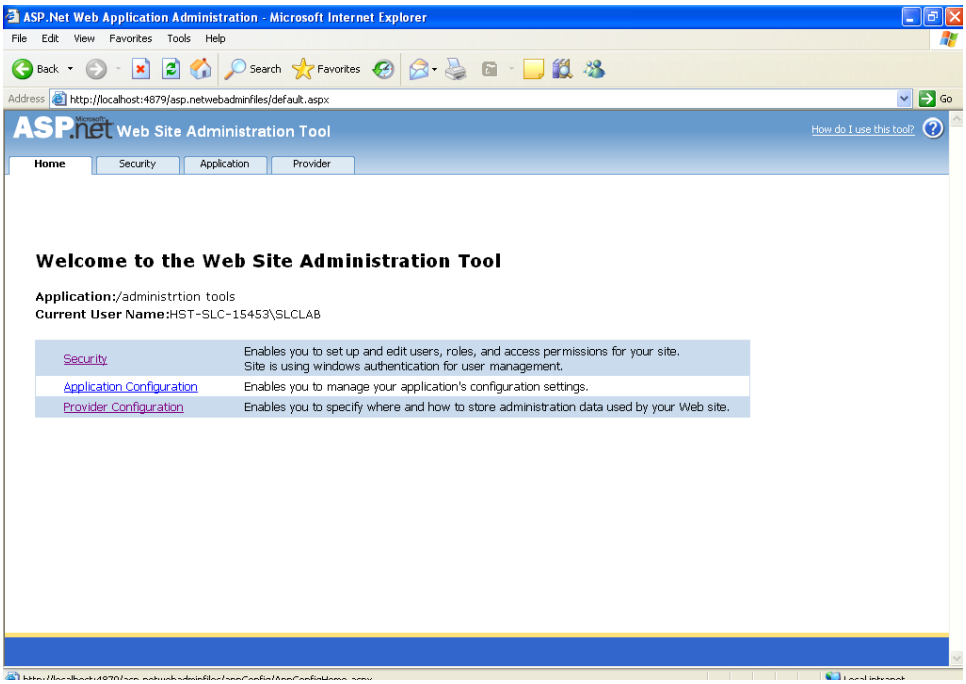


Figure 10.3-4: Website Administration Tool.

Step 9: Select provider configuration and select single provider for all site management data.

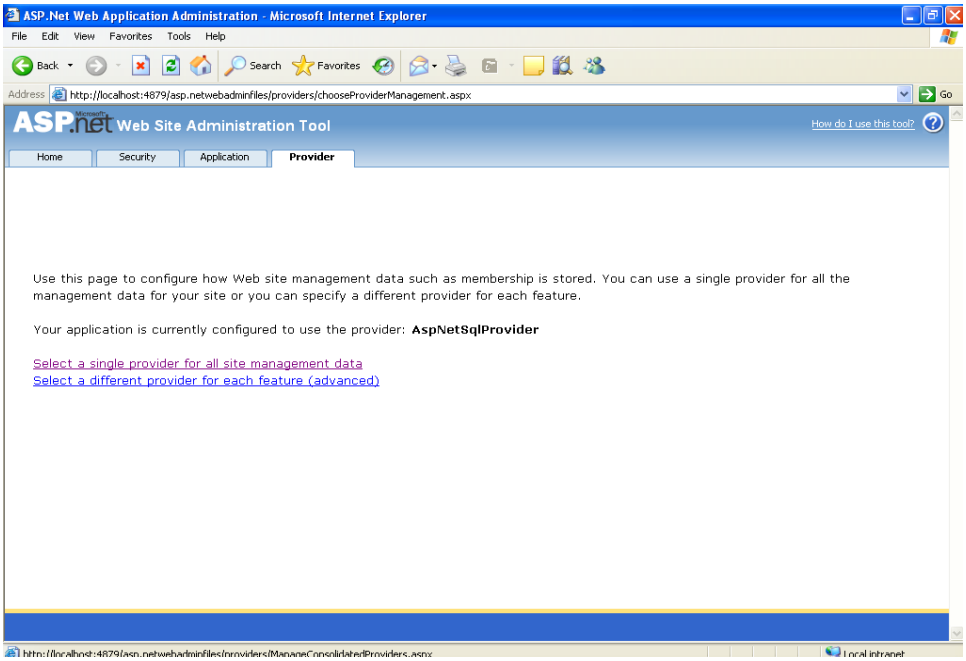


Figure 10.3-5: Selecting Provider.

Step 10: Test the connection. If successful connection established then click on ok and if connection fails then open visual studio command prompt window and write **aspnet_regsql.exe** and configure your database. Why you need to do so as default provider is SQL in asp.net and it automatically create a database named **ASPNETDB** in SQL and table where all information is stored in respective tables.

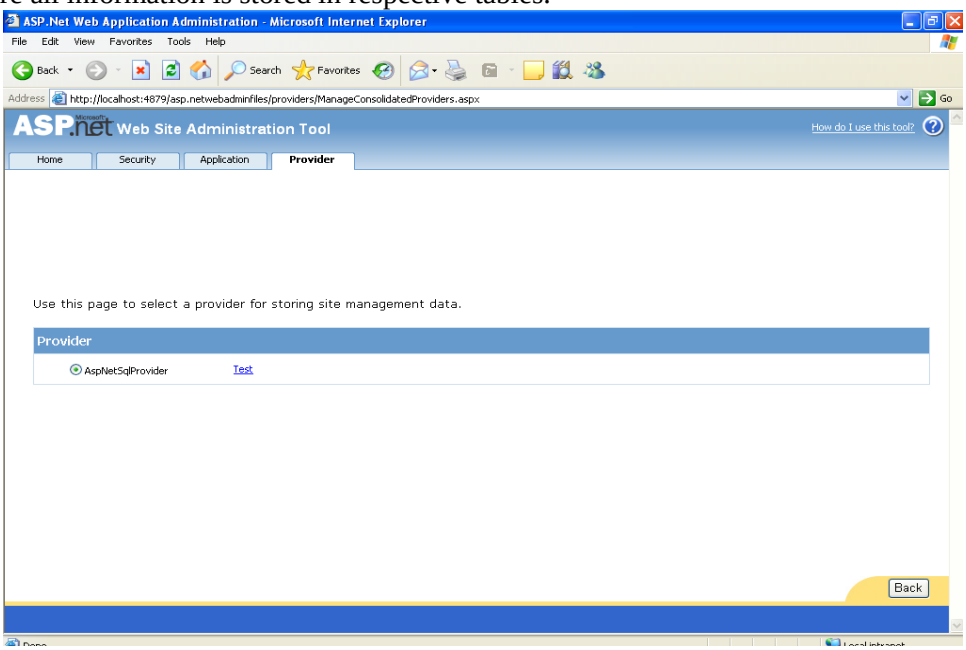


Figure 10.3-6: Testing connection to database.

Step 11: In security select authentication type and select from internet and click on done.

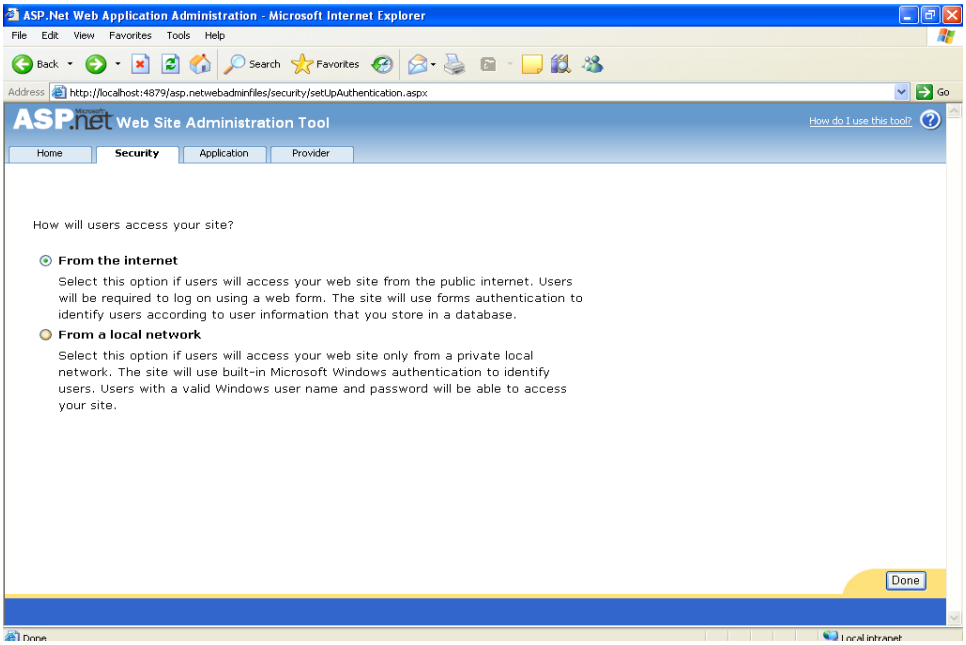


Figure 10.3-7: In Security tab selecting From the internet and clicking done.

Step 12: Enable the role and click on create or manage roles.

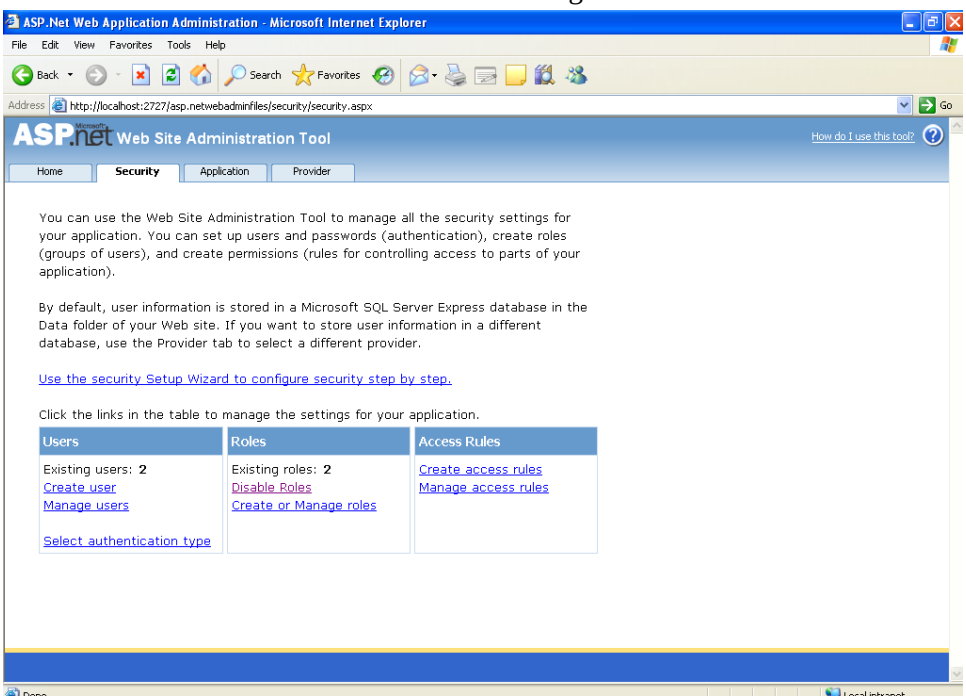


Figure 10.3-8: After enabling role, create or manage roles.

Step 13: Create two roles, **admin** and **user** and click on back.

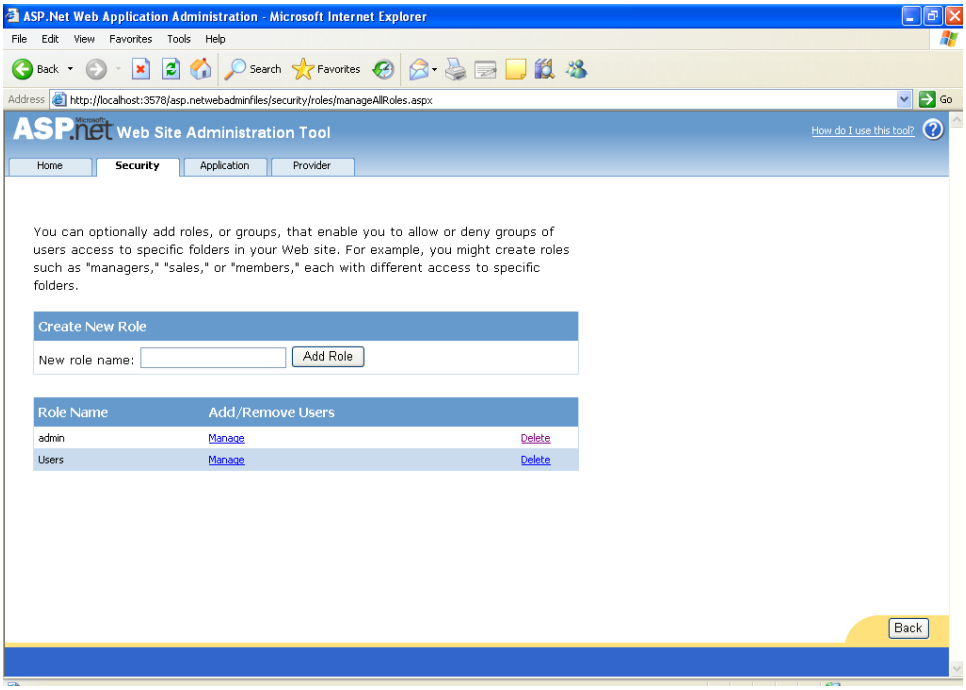


Figure 10.3-9: Admin and Users role created.

Step 14: In security tab click on create users. Here two users have been created, one is assign admin role and another is assign user role.

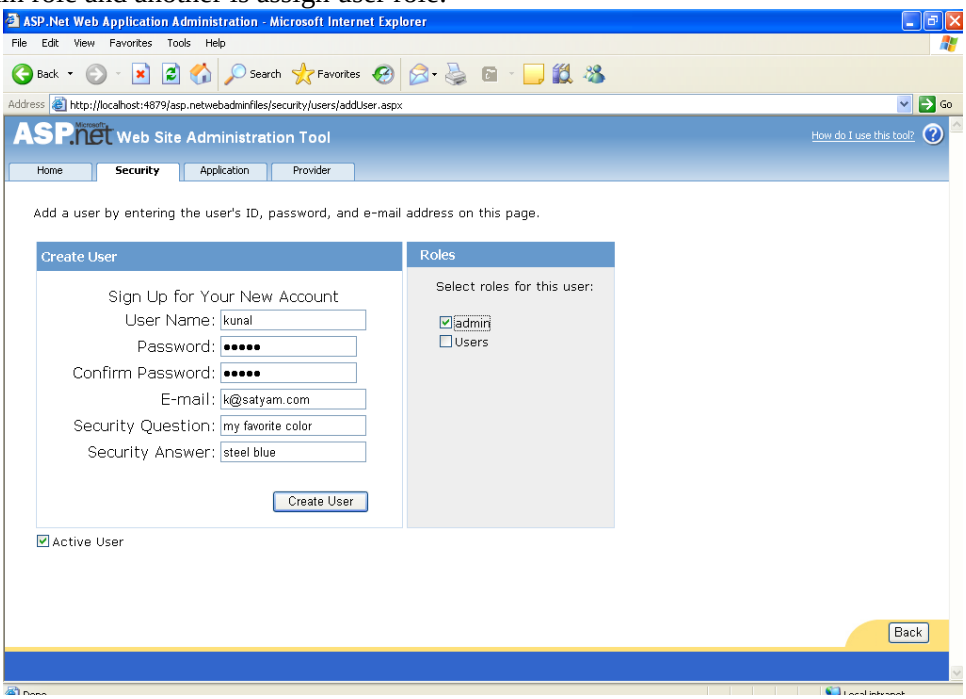


Figure 10.3-10: Creating user.

By creating a user through the Web Admin you created a SQL Server Express database

contained within your application.

Step 15: Create access rule on two folders, admin and users, created in the application. Selecting admin folder select role **admin** and give permission as allow and click ok. Again create access rule on Admin folder selecting all users and give permission as deny and click ok. Similary create access rule for user folder selecting **all user** and give permission as allow and click ok.

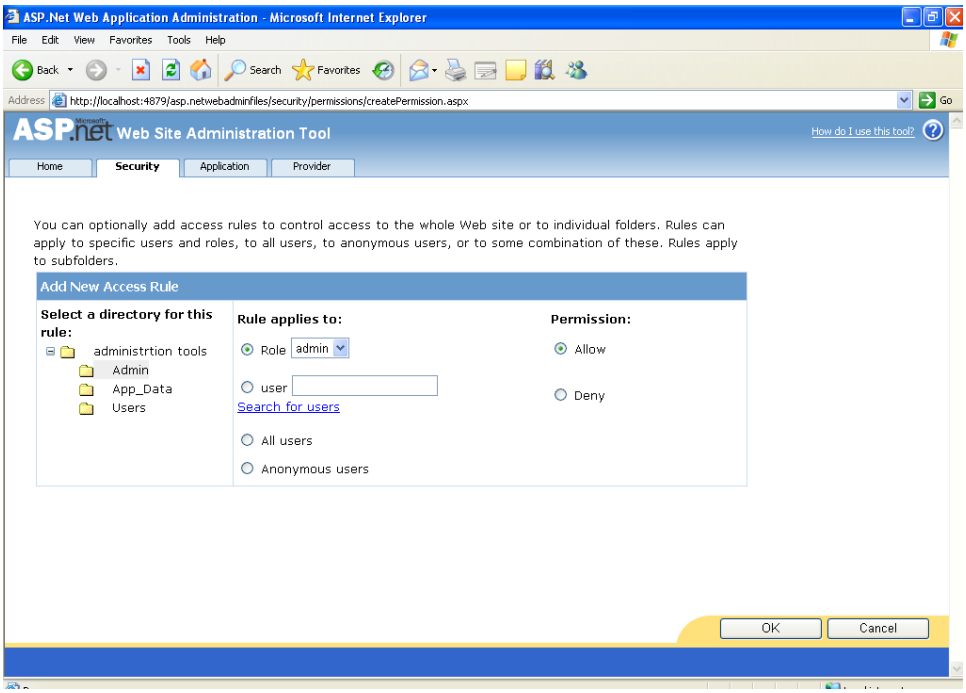


Figure 10.3-11: Creating Access rule on Admin and Users folder.

Step 16: Close the **Asp.net** configuration tool. In order to have role based security write in web.config file.

```
<siteMap defaultProvider="default">
<providers>
<clear />
<add name="default" type="System.Web.XmlSiteMapProvider"
siteMapFile="web.sitemap" securityTrimmingEnabled="true" />
</providers>
</siteMap>
```

On making **securityTrimmingEnabled=true** role based security is done . If its not made true then user can see on link for admin page as well as for user page. By enabling true only user page link is seen as shown earlier in screenshots.

Context :

- Use administration tool to manage accounts and general application settings.
- Allows configuring every possible web.config setting.
- configure authorization rules by using the Web Site Administration

Practice Session/s :

Create a website having employee leave transfer page where employee can generate leave by entering start date, end date and reason for leave. Create a Admin grant/Reject leave page where administrator can grant or reject the leave. Create database wherever necessary. Admin should be able to see both the links while employee is able to see only employee link.

Check List :

- Importance of Website Administration Tool for role based security.

Common Error/s :

- Creation of improper access rules
- Giving improper navigation URL in sitemap
- Not Configuring database properly

Exception/s :

- If you specify any other role than that specified you will get Server Error.

Lesson/s Learnt :

- ☑ Use of Web site Administration Tool.
- ☑ Implementing role based security.
- ☑ Using site map for navigation.

Best Practice/s :

- Use Administrative tools to have role based security.
- Most changes to configuration settings that you make in the Web Site Administration Tool take effect immediately. This requires the Web site to which the change applies to be restarted. Because this will cause currently active sessions in the Web site to be lost, you should make configuration changes to a staged or development version of the Web site before publishing these changes to the production server.

Choose **From the Internet** option for authentication type (if forms-based authentication is used). Choose **From a local network** option as authentication type (Integrated Windows authentication is used).

Topic: Web Parts

Estimated Time: 45 mins.

Objectives : At the end of the activity, the participant should understand

- Customizing a page using Web Parts
- Use of custom user controls in Web Parts

Presentation :

- Web Parts is an integrated set of controls for creating Web sites that enable end users to modify the content, appearance, and behavior of Web pages directly from a browser.
- Users can add new Web Parts controls to a page, remove them, hide them, or minimize them like ordinary windows.
- Users can import or export Web Parts control settings for use in other pages or sites, retaining the properties, appearance, and even the data in the controls. This reduces data entry and configuration demands on end users.
- Custom user control used in a web part has to be derived from the Web Part Class. When using the Web Part class import the **System.Web.UI.WebControls** Web Part namespace.
- There are two basic things in web parts:
  - **Web part manager** important of all the Web Part controls is responsible for managing and coordinating all controls inside WebPartZones.
  - **Web part zones**. There are four kinds of zones in web parts:
    - ✓ **Web Part Zone:** is used to define zones, which serve as containers for Web Parts.
    - ✓ **Editor Zone:** is used to allow end users to customize Web Parts pages by editing the properties of the page's Web Parts.
    - ✓ **Catalog Zone:** is used to allow end users to customize Web Parts pages by adding Web Parts to them. A CatalogZone control becomes visible only when a user switches a Web page to catalog display mode (CatalogDisplayMode).
    - ✓ **Connection Zone:** provides a UI for making connections.

Scenario:

Manager Wilson of Kmart Company have features like showing timezone, next holiday and lot more in company website to its users. Wilson wants to make website more interactive to the users and allows users to see what they want to see at runtime. So Wilson asks his developer to use Web Parts.

Demonstration/Code Snippet :

HomePage.aspx shows webpart implementation and its design.

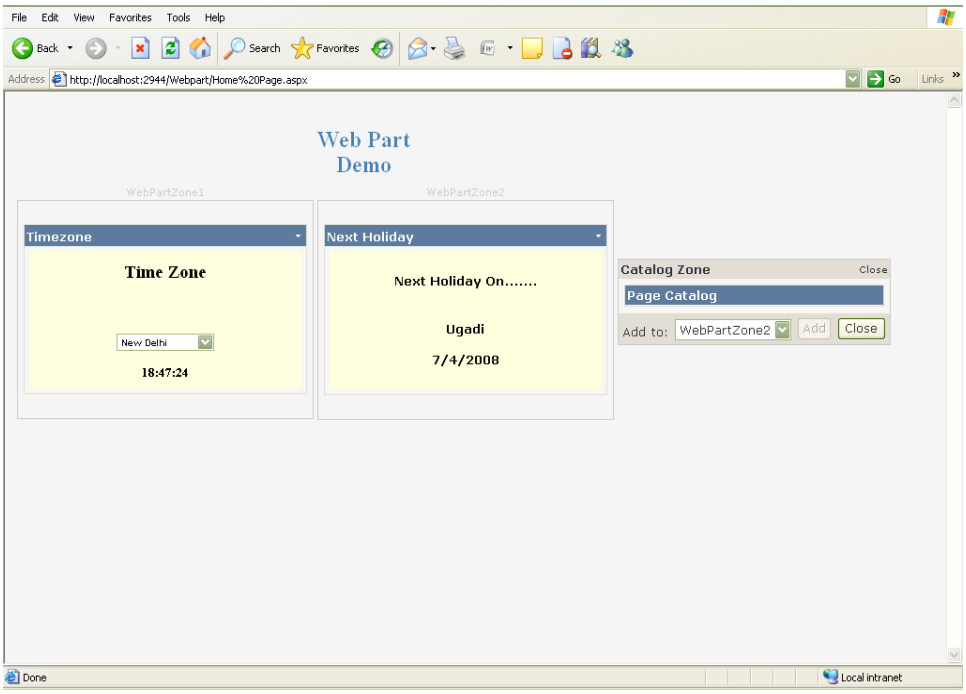


Figure 10.4-1: Homepage.aspx showing webpart.

Closing a WebPart.

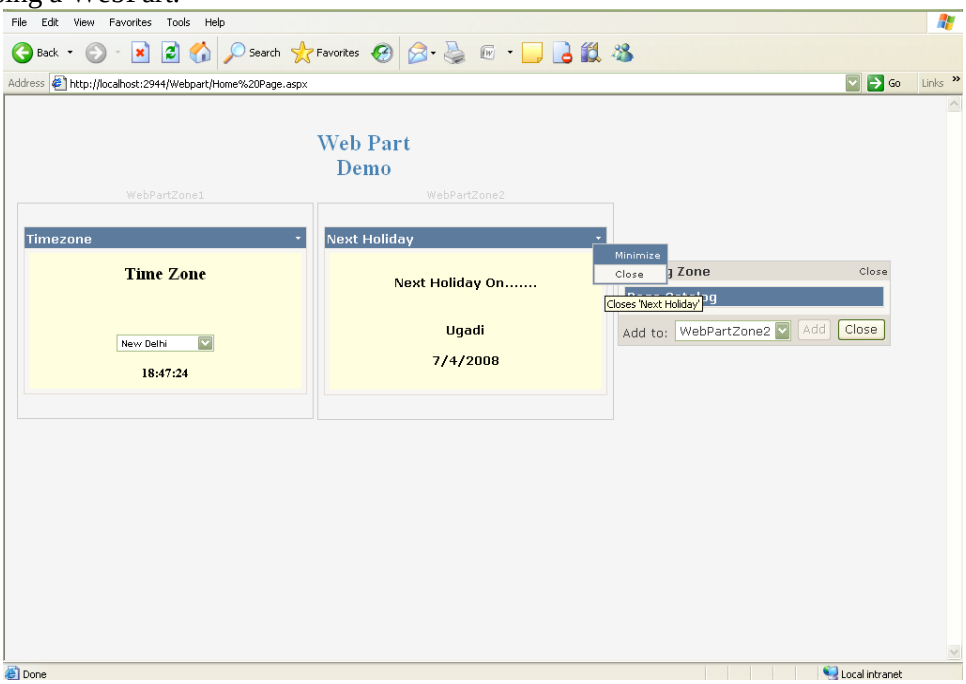


Figure 10.4-2: How to close webpart.

On closing webpart it will be in page catalog.

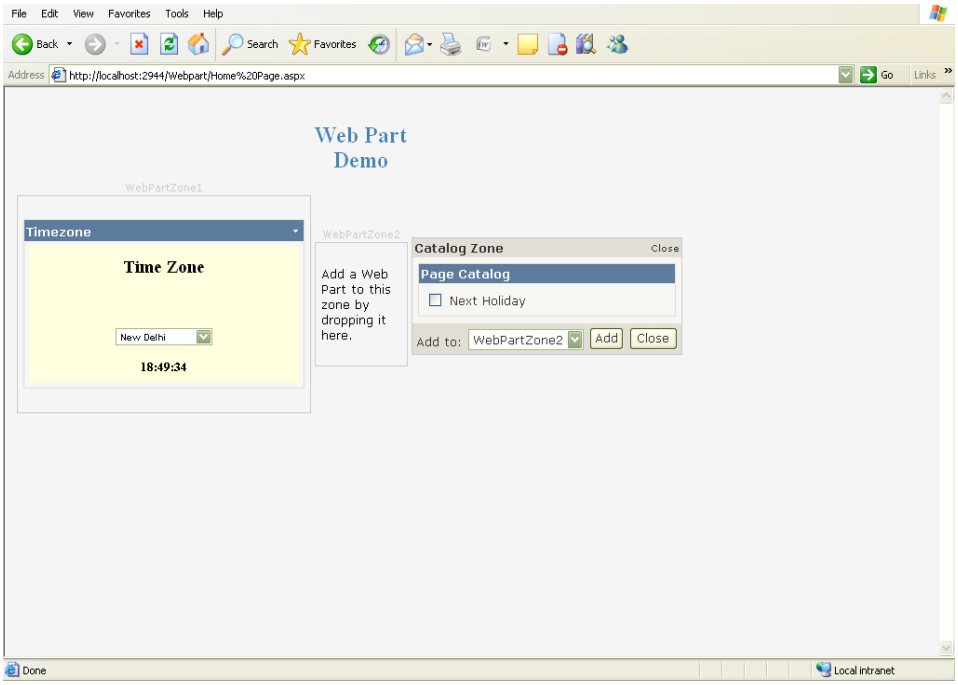


Figure 10.4-3: HomePage.aspx shows that closed web part is in page catalog.

Check on the webpart and then user can add to any of the webpartzone selected from dropdown list and then click on add.

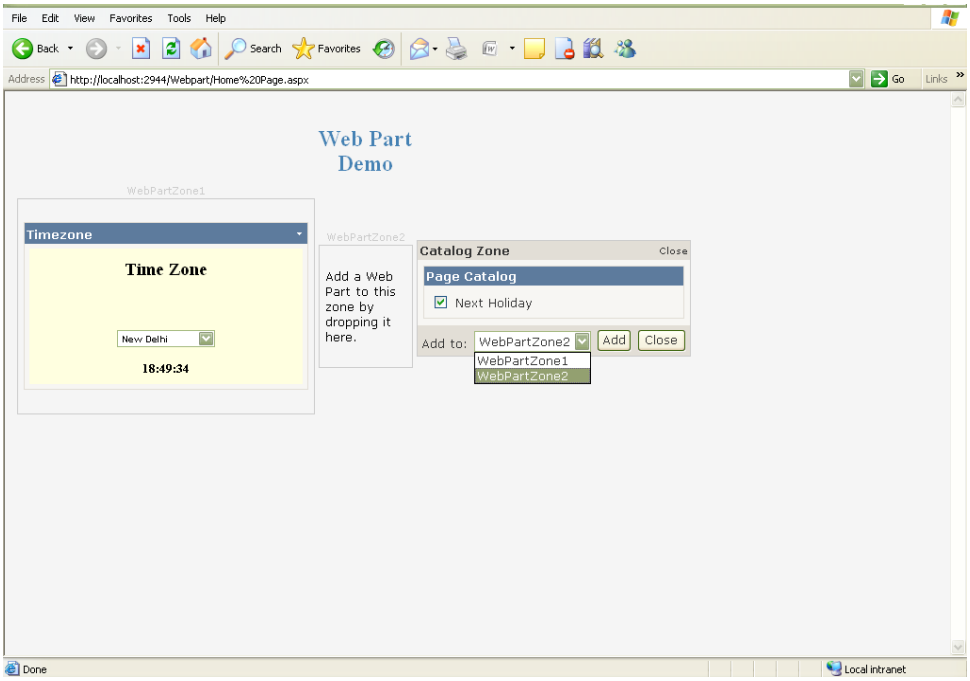


Figure 10.4-4: Adding Web Part in webpart zone.

On adding Webpart to webpartzone 1 then it is viewed as shown below.

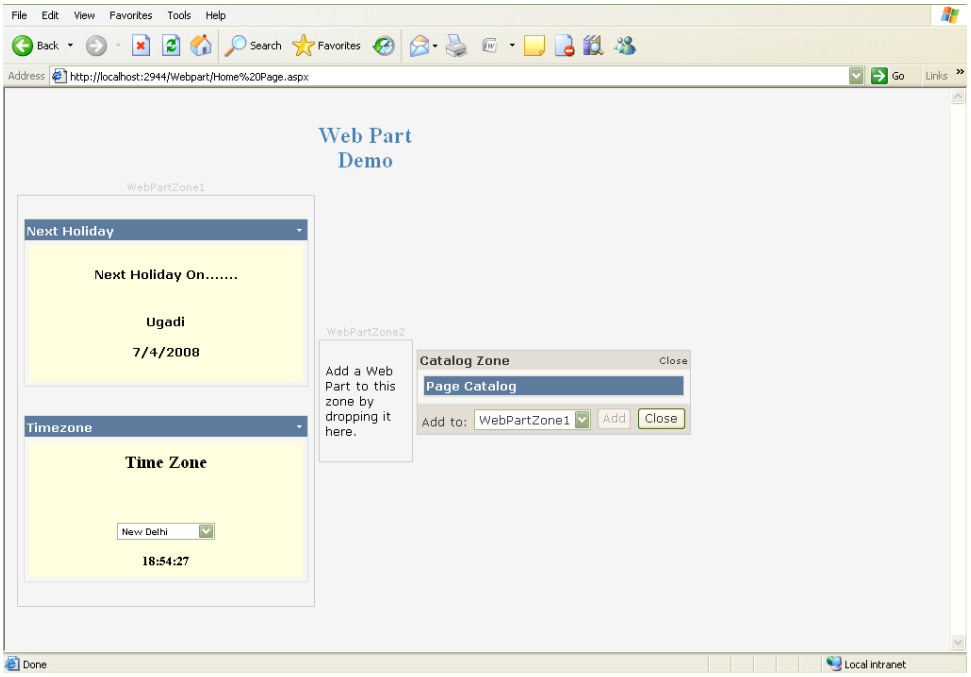


Figure 10.4-5: HomePage.aspx showing NextHoliday webpart added in webpartzone 1.

Step 1: Create a new Web application project called WebPart using visual studio 2005. Rename Default.aspx to HomePage.aspx.

Step 2: In the Solution Explorer, right click on the name of the Web Application (WebPart) and select Add New Item/WebUserControl File. Rename it to TimeZone.ascx.

Step 3: Insert HTML Table from toolbox in TimeZone.ascx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text: Time Zone
asp:dropdownlist	ddlLocation	AutoPostBack: True
		Items/Add/Text: New Delhi Add/Text: Sydney Add/Text: Singapore Add/Text: London
asp:label	lblTime	-

Step 4: Write the event handler for Page Load of TimeZone.ascx.

```
protected void Page_Load(object sender, EventArgs e)
{
TimeSpan ts;
switch (ddl_location.SelectedValue)
{
case "New Delhi":
lblTime.Text= System.DateTime.Now.TimeOfDay.ToString().Substring(0,
8);
break;

case "Sydney":
ts = new TimeSpan(4, 30, 0);
lblTime.Text =
System.DateTime.Now.TimeOfDay.Add(ts).ToString().Substring(0, 8);
break;

case "Singapore":
ts = new TimeSpan(2, 30, 0);
lblTime.Text =
System.DateTime.Now.TimeOfDay.Add(ts).ToString().Substring(0, 8);
break;

case "London":
ts = new TimeSpan(4, 30, 0);
lblTime.Text =
System.DateTime.Now.TimeOfDay.Add(ts).ToString().Substring(0, 8);
break;
}
}
```

Step 5: In the Solution Explorer, right click on the name of the Web Application (WebPart) and select Add New Item/WebUserControl File. Rename it to Holiday.ascx.

Step 6: Insert HTML Table from toolbox in TimeZone.ascx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblHeading	Text: Next Holiday on…
asp:label	lblDay	-
asp:label	lblDate	-

Step 7: Write the event handler for Page Load of Holiday.ascx.

```
protected void Page_Load(object sender, EventArgs e)
    {
        lblDay.Text = "Ugadi";
        lblDate.Text = "7/4/2008";
    }
```

Step 8: Open HomePage.aspx. Insert HTML Table from toolbox in HomePage.ascx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:webpartmanager	WebPartManager1	-
asp:label	lblheading	Text: Web Part Demo
asp:webpartzone	webpartzone1	-
asp:webpartzone	webpartzone2	-
asp:catalogzone	catalogzone1	-

Step 9: In Catalog Zone Add PageCatalogPart. In order to run in the Catalog display mode write in page load event of HomePage.aspx.

```
protected void Page_Load(object sender, EventArgs e)
{
WebPartManager1.DisplayMode
WebPartManager.CatalogDisplayMode;
}
```

CatalogZone is used for catalog display mode. Catalog Zone is used for managing WebPartZone content and Page CatalogPart contains the Webparts which has been closed by user and which can be added later on to any webpartzone using catalogzone.

Step 10: In order to run webparts make personalization enabled=true in source code of HomePage.aspx (design).

```
<asp:WebPartManager ID="WebPartManager1" runat="server">
        <Personalization Enabled="true" />
</asp:WebPartManager>
```

Step 11: Drag and drop WebUserControl TimeZone.ascx and Holiday.ascx in webpartzone1 and webpartzone2 respectively.

Step 12: Following code will be therebelow @page directive in HomePage.aspx source code as soon as webusercontrols are added

```
<%@ Register Src="Holiday.ascx" TagName="Holiday" TagPrefix="uc2" %>

<%@ Register Src="TimeZone.ascx" TagName="TimeZone" TagPrefix="uc1"
%>
```

Step 13: Title can be added as shown below:

```
<ZoneTemplate>
<uc1:Timezone ID="Timezone1" runat="server" title="Timezone"/>
</ZoneTemplate>
```

```
<ZoneTemplate>
<uc2:Holiday ID="Holiday1" runat="server" title="Next Holiday"/>
</ZoneTemplate>
```

Step 14: Run the Application setting Homepage.aspx as startup page.

Context :

- Using different controls to customize a page.
- Establishing connections between controls.

Practice Session/s :

- Create two different Web Sites having different Web Parts and import one of them to the other one.
- Create a Web Site with a Web Part and establish connection between different controls of that Web Part to reflect their dependencies.

Check List :

- Importance of connections between controls in a Web part.
- Using System.Web.UI.WebControls. WebParts namespace.
- Creating complex Web Parts.
- Importance of web part class.

Common Error/s :

- Non inclusion of System.Web.UI.WebControls.WebParts.
- Non inclusion of Webparts Class.

Exception/s :

- **(Provider: SQL Network Interfaces, error: 26 - Error Locating Server/Instance Specified)**
  Basically the WebPartManager is trying to contact to the profile store to save your personalization settings, which happens to be SQL Server by default. Now your app is clean and you have not specified anywhere that you wanted to have SQL as your profile storer, perhaps you don't have SQL even installed. So to get around this just switch off the webpartmanager personalization by setting its personalization attribute to false :

```
<asp:WebPartManager ID="WebPartManager1" runat="server">
    <Personalization Enabled="False" />
</asp:WebPartManager>
```

Lesson/s Learnt :

- ☑ Importance of Web parts.
- ☑ Inclusion of System.Web.UI.WebControls.WebParts and WebParts Class.
- ☑ Customizing Web pages using Web Parts.

Best Practice/s :

- Use WebParts for better user interface.

Crossword: Unit-10 **Estimated Time: 10 mins.**

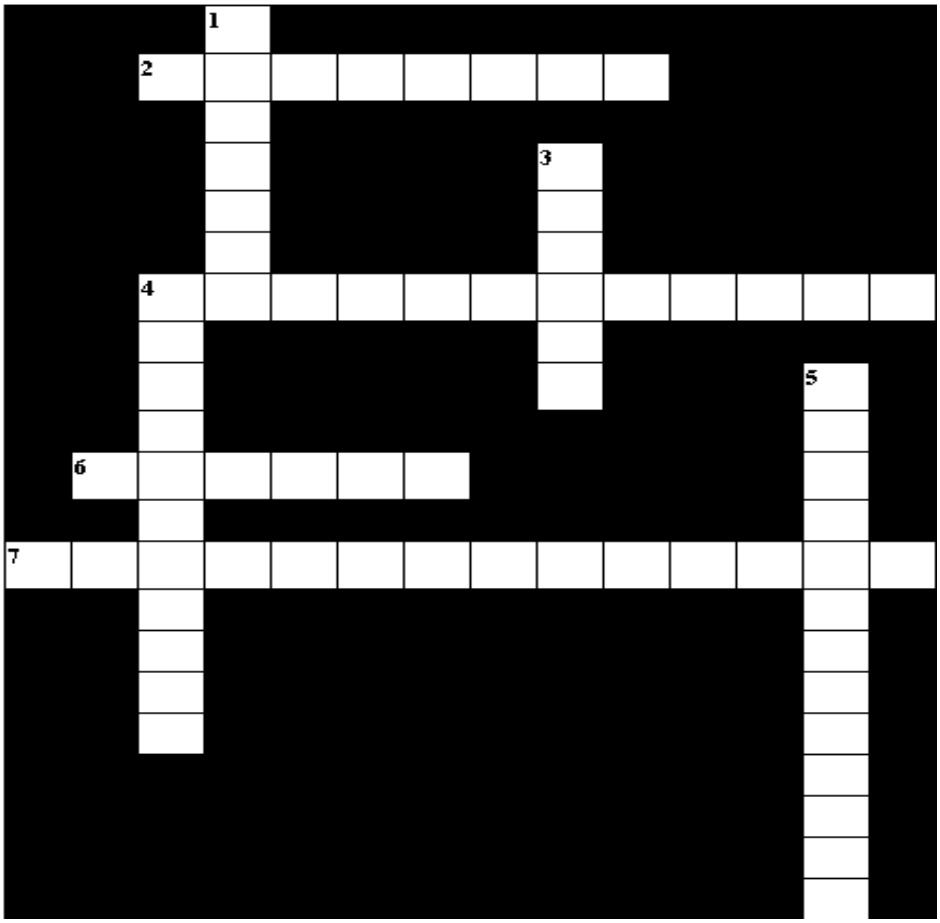


Fig. 10-1

Across:

2. A page used in the Web Site Administration Tool.(8)

4. Attribute that allows a web part to display property in web browser in edit mode.(12)

6. Which mode gives us option to add/remove web parts on runtime.(6)

7. The process of obtaining identification credentials from a user (such as name and password), and validating those credentials against some authority.(14)

Down:

1. One of the pages used in the Web Site Administration Tool (7).

3. Which is the default mode used for web parts.(6)

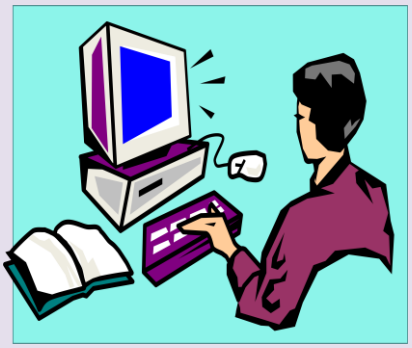
4. This is a basic things in web parts(11):

5. The process of controlling access to resources based on the authenticated identification credentials (such as role).(13)

11.0 Deploying an ASP.Net Web Application

Topics

- 11.1 Creating and Deploying Web Setup Project
- 11.2 Web Services
- 11.3 Crossword



Topic: Web setup Project and Deployment

Estimated Time: 45 mins.

Objectives : At the end of the activity, the participant should understand

- Create web setup project
- Deployment

Presentation :

- There are six types of setup and deployment projects in VS.NET, but only three of them are of significant importance, which are "Setup Project", "Web Setup Project" and "Merge Module Project".
- **Setup Project:** Generic type of project that could be used for all type of applications including web based application.
- **Web Setup Project:** This project type helps in creating virtual directories for web based applications during installation.
- **Merge Module Project:** When you want to install some additional third party software like MSDE along with your application then you can use this type of project. Use a *.msm extension based merge module for installation of MSDE along with your own and setup is created.

Scenario :

Mr. Steve, the owner of the Inversion Technologies wants to build and deploy the website of his organization on the web so that his employees can view the website anywhere within the organizational Domain.

Demonstration/Code Snippet :

Step 1: Create Website to deploy. After application is created open Web.config and make debug=”false”.

Step 2: Check whether, if it is in release mode, if not then change from debug mode to release mode which is on the top, as shown below.

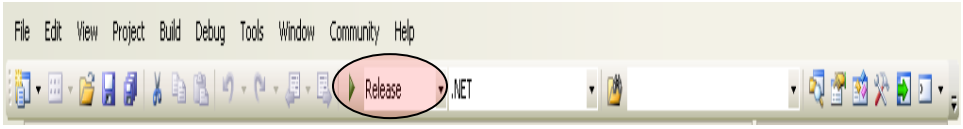


Figure 11.1-1: Menu bar Showing Release.

Step 3: If it is not giving release option then don't worry. Right click on the project in solution explorer/add new project and select setup and deployment and select web setup Project for deploying website.

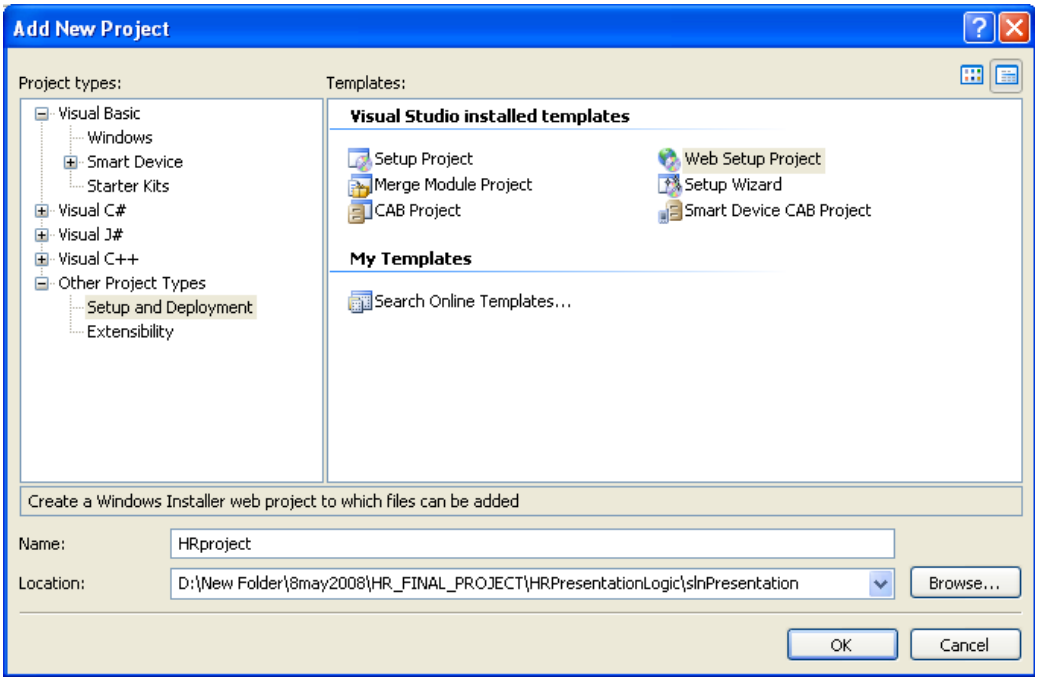


Figure 11.1-2: Selecting Web setup Project.

Step 4: Click on web application folder and add project output.

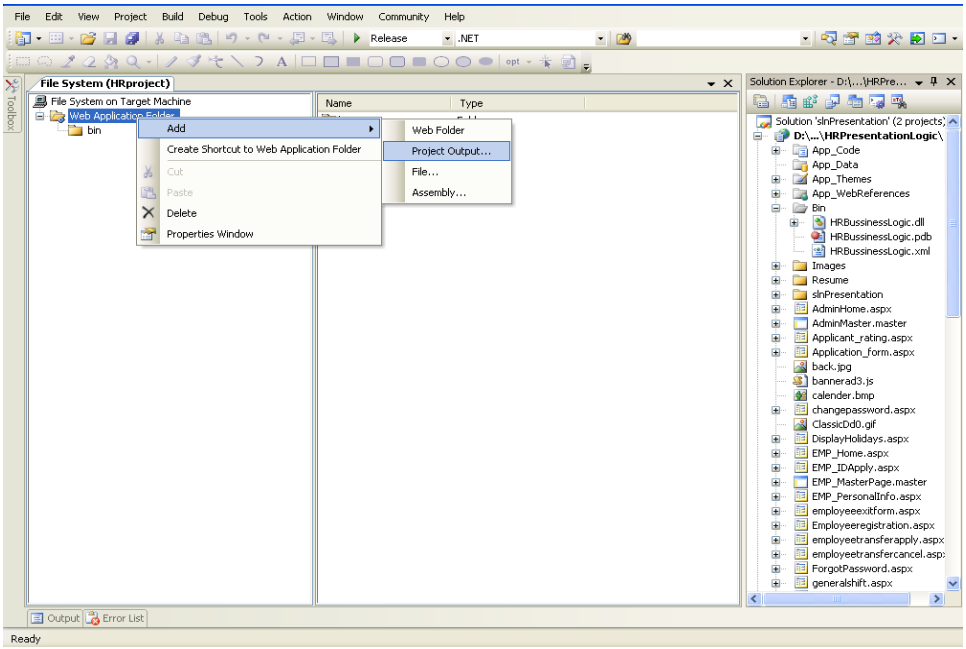


Figure 11.1-3: Adding Project output.

Step 5: Make configuration active.

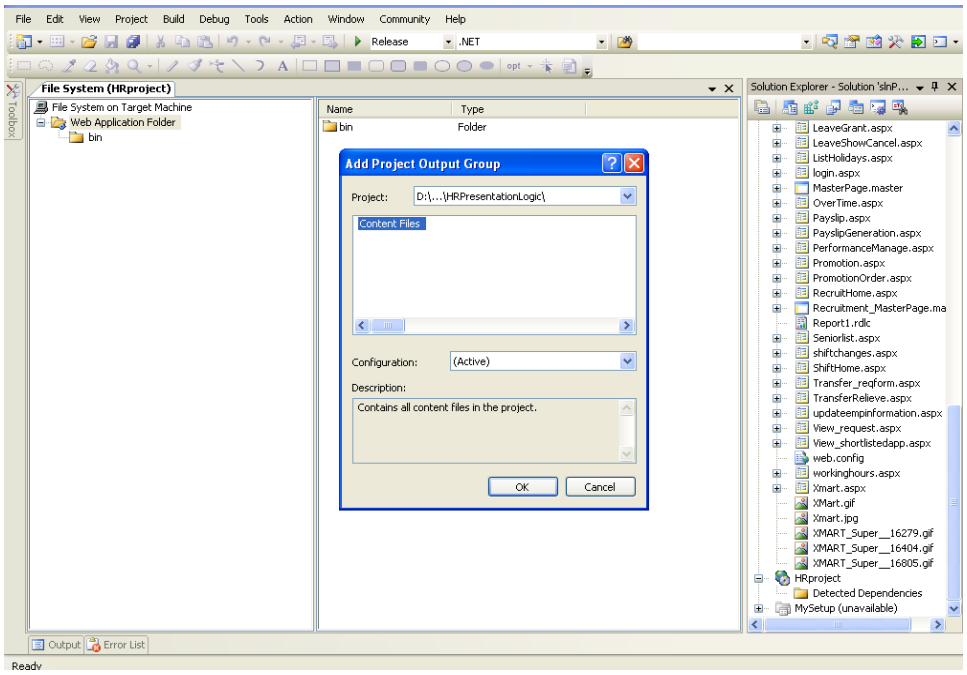


Figure 11.1-4: Making configuration active.

Step 6: Create a shortcut to bin and rename it.

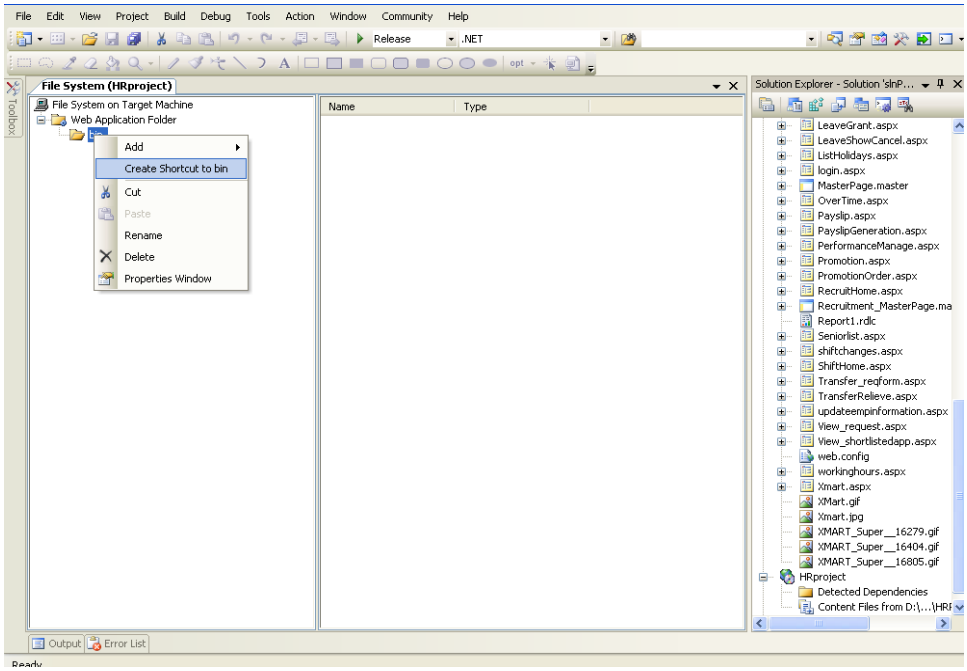


Figure 11.1-5: Creating shortcut.

Step 7: It looks like this. Check virtual directory by right click on web application folder and selecting properties and also check default document it should not be empty.

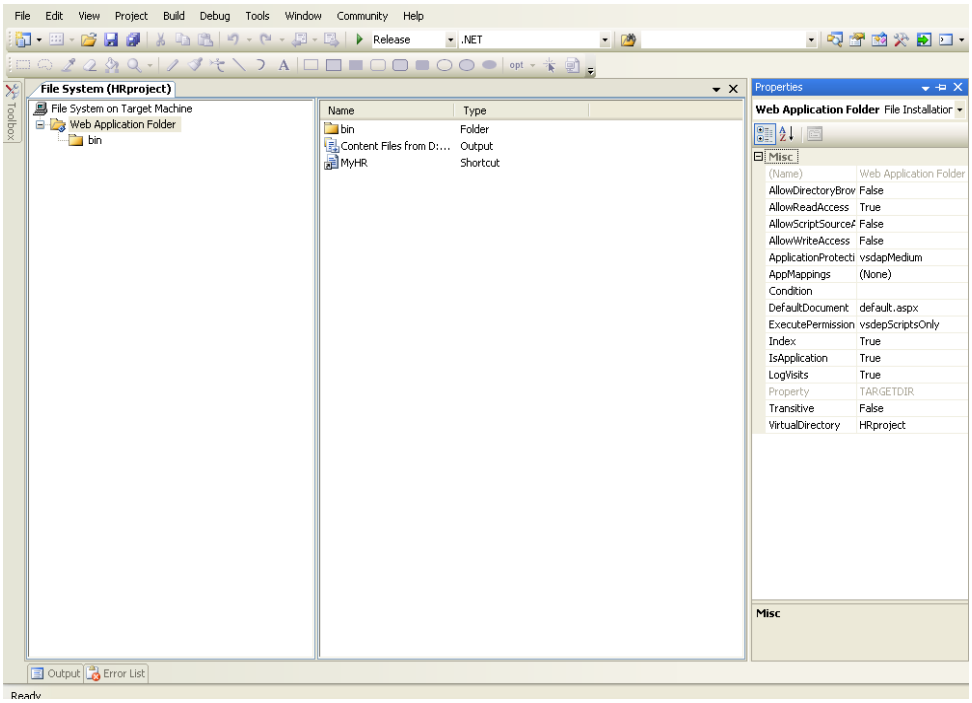


Figure 11.1-6: Checking virtual directory created.

Step 8: Right click on setup in solution explore and check its property.

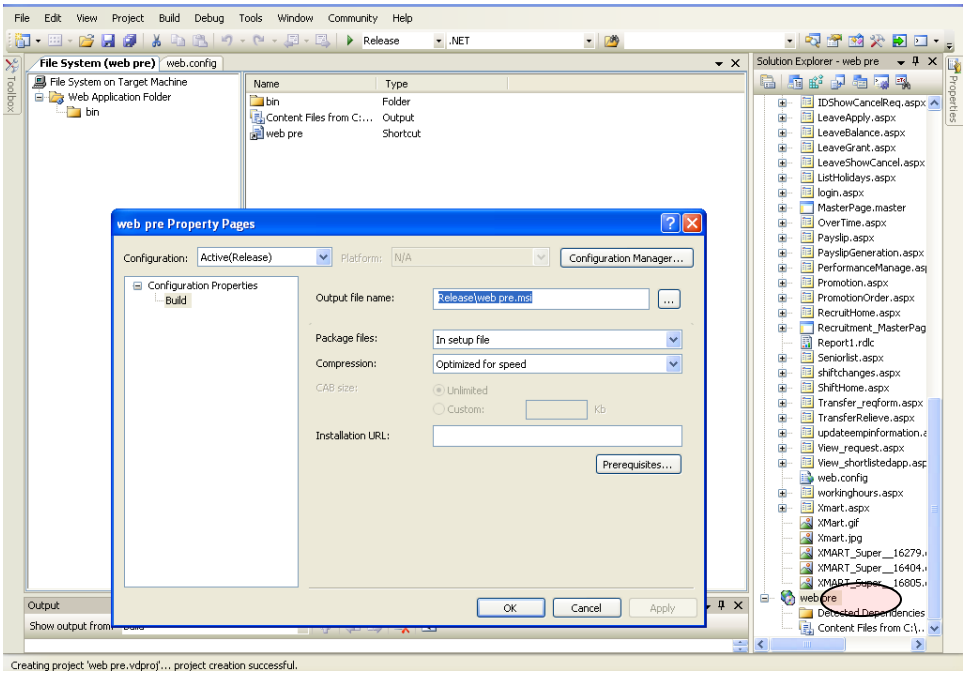


Figure 11.1-7: It shows web pre property page.

Step 9: If output filename path is not "Release\" then select configuration manager and do as shown below:

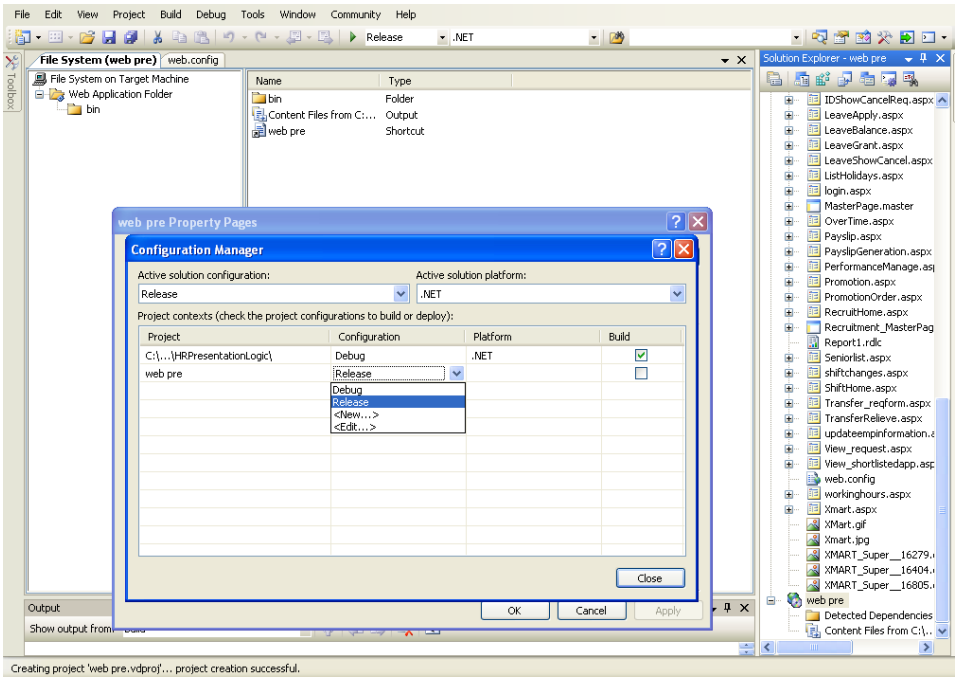


Figure 11.1-8: Selecting 'Release' from configuration manager.

Step 10: Right click on set up in solution explorer and install it. Now open the IIS from control panel/ Administrative tools/ IIS and check. Project is available and can run by clicking on project and selecting any form with extension .aspx.

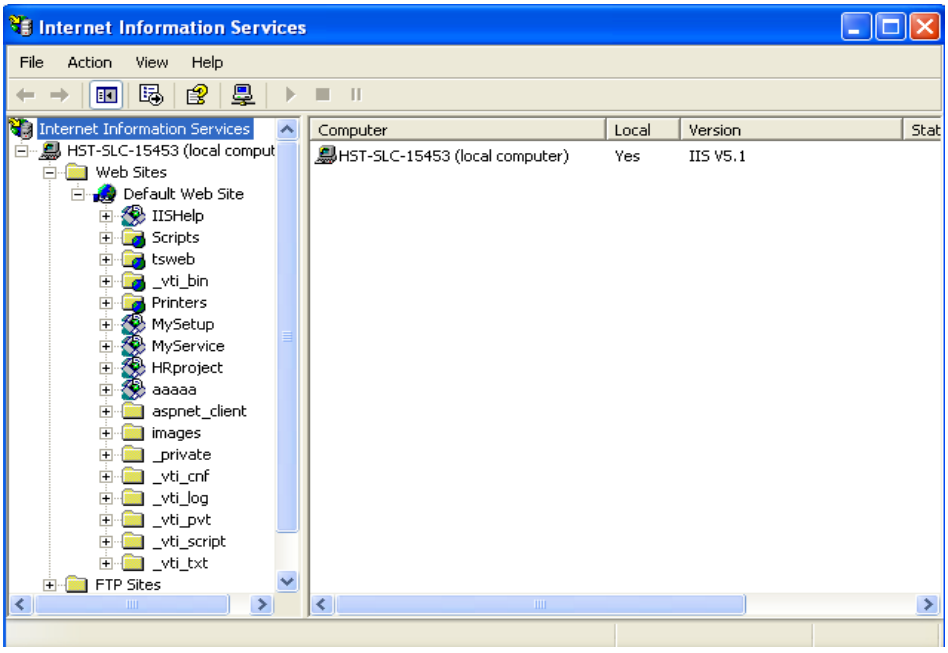


Figure 11.1-9: Checking installed project in system.

Step 11: If you are using the web service you need to deploy the web service first in the similar way and then run it from the IIS and write in the browser instead of local host your computer IP to host the service and add its reference to your website and the build it in release mode and the deploy it.

Context :

- Deploying of WebSite is used so make use of application globally.

Practice Session/s :

Create a demo site for viewing employee information like date of birth, name, department etc. and try to deploy it and view in browser from another system.

Check List :

- Importance of website deployment.

Common Error/s :

- Deploying in debug mode and then trying to view in browser from another pc in network.

Exception/s :

- Deploying in debug mode and then trying to view in browser from another pc in network throws a run time exception.

Lesson/s Learnt :

- ☑ Creating Web Setup Project.
- ☑ Deployment of website.

Best Practice/s :

- Deployment of site in release mode.

Once created, the type of a project cannot be changed between Web and standard. If you have created a standard deployment project and later decide to deploy it to a Web, you will need to create a new project.

Request a local page using the loopback address. The loopback address is 127.0.0.1, and the alias is localhost. The loopback address and alias always point to the current computer and are extremely useful while testing. Try to type this and run login page as shown below from local machine.
http://127.0.0.1/login.aspx

Topic: Web Service

Estimated Time: 40 mins.

Objectives : At the end of the activity, the participant should understand

- Basics of Web services

Presentation :

- Web services are programmable business logic components that provide access to functionality through the Internet. It exposes a number of methods to provide functions.
- Web services provide interoperability, dynamic integration, enable applications to communicate using accepted industry standards and in a secure environment.
- Service provider, Service Client and Service broker play different roles in a web service.

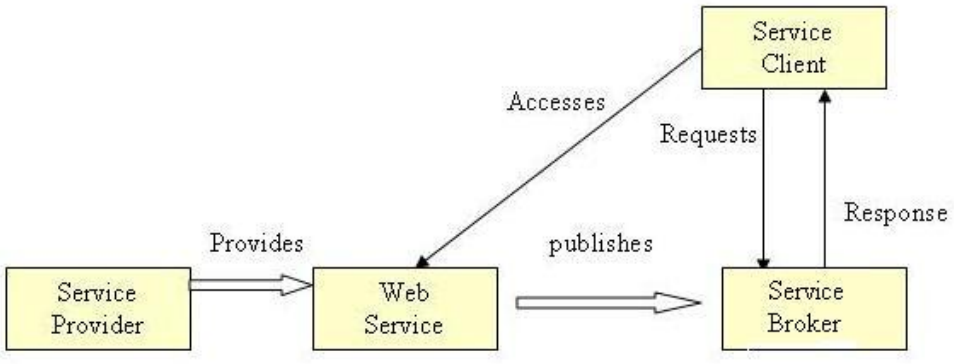


Figure 11.2-1: Functioning of Web Service.

- XML, SOAP, WSDL and UDDI are the different elements of web service.
  - XML is a markup language designed to carry data.
  - SOAP is a communication protocol used for sending messages.
  - WSDL is a descriptive language which defines and locates a web service.
  - UDDI is a directory for storing information about web services.

Scenario:

In a K Solution company manager Ryan wants to make company's site to be more secure. Ryan doesn't want any user to know what is happening with the database. So Ryan suggests developers to make application more secured by using web services.

Demonstration/Code Snippet :

Web service page looks like this:

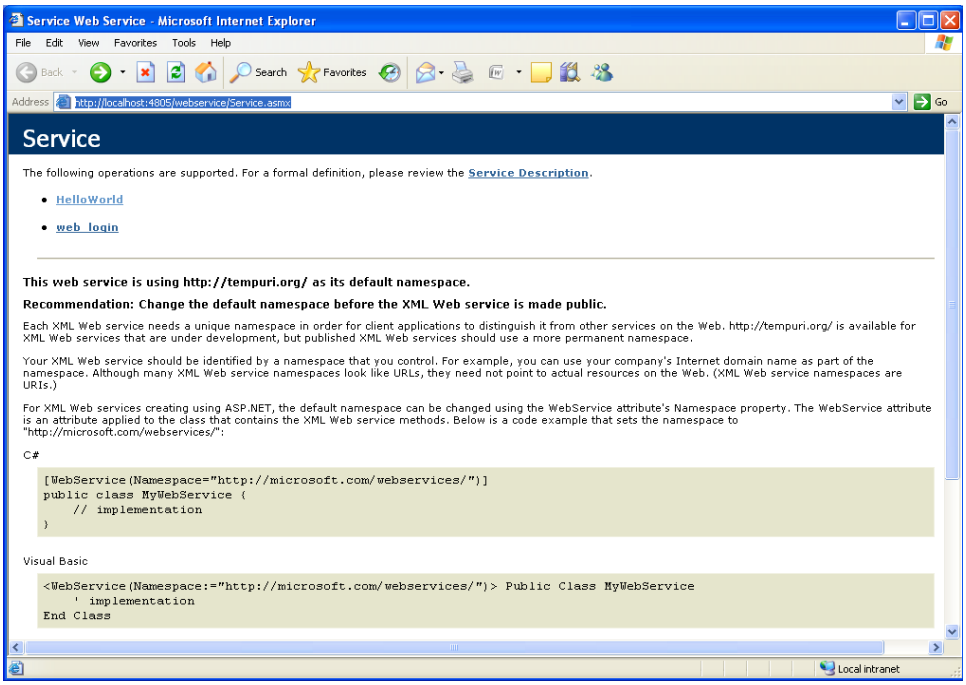


Figure 11.2-2: Web service page.

Click on web_login method it shows as shown below:

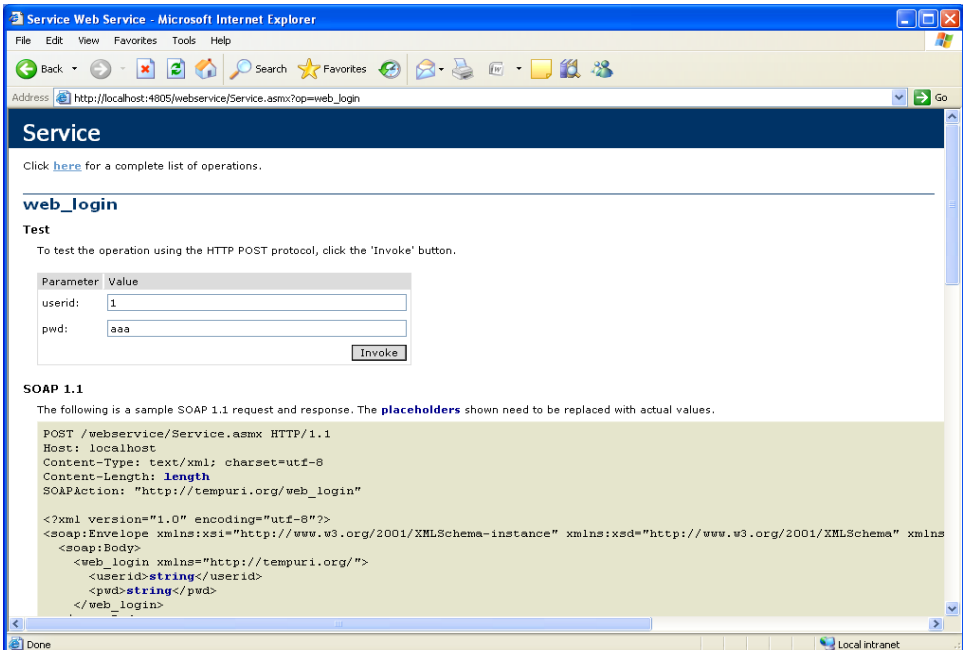


Figure 11.1-3: Web_login page.

On Clicking invoke it shows the result in xml format as shown below:

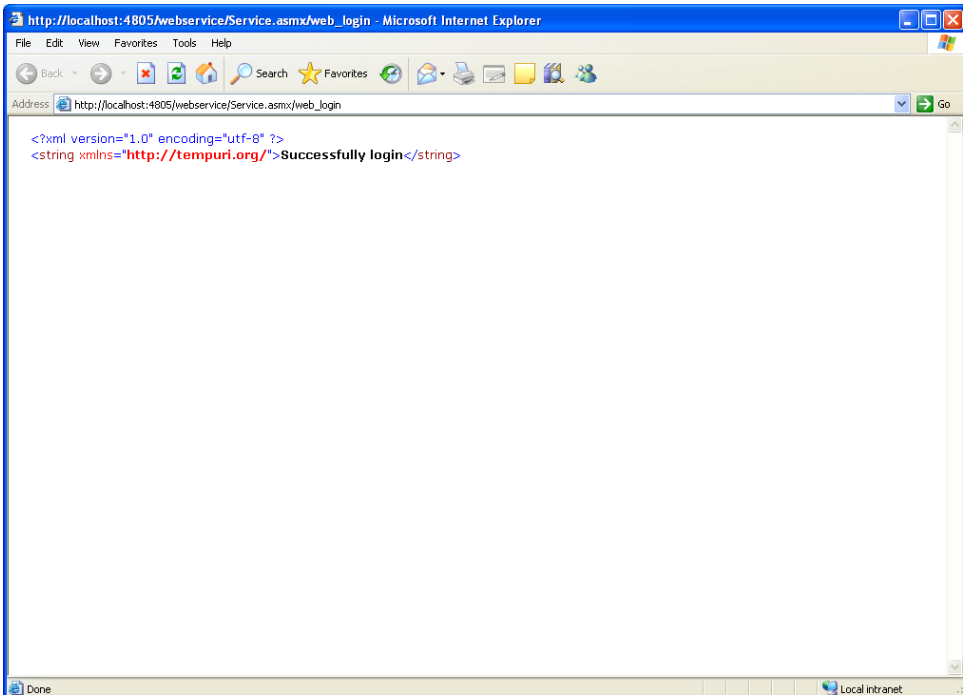


Figure 11.2-4: Result in XML format.

For Unsuccessful Login it shows unsuccessful login as shown below.

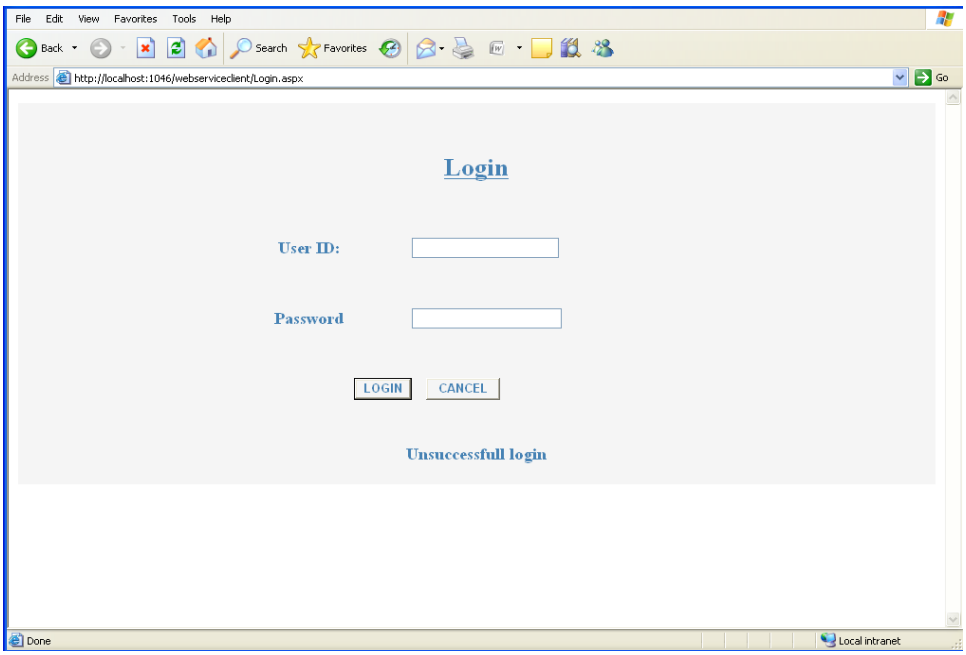


Figure 11.2-5: Unsuccessful login.

For successful login it redirect to homepage as shown below.

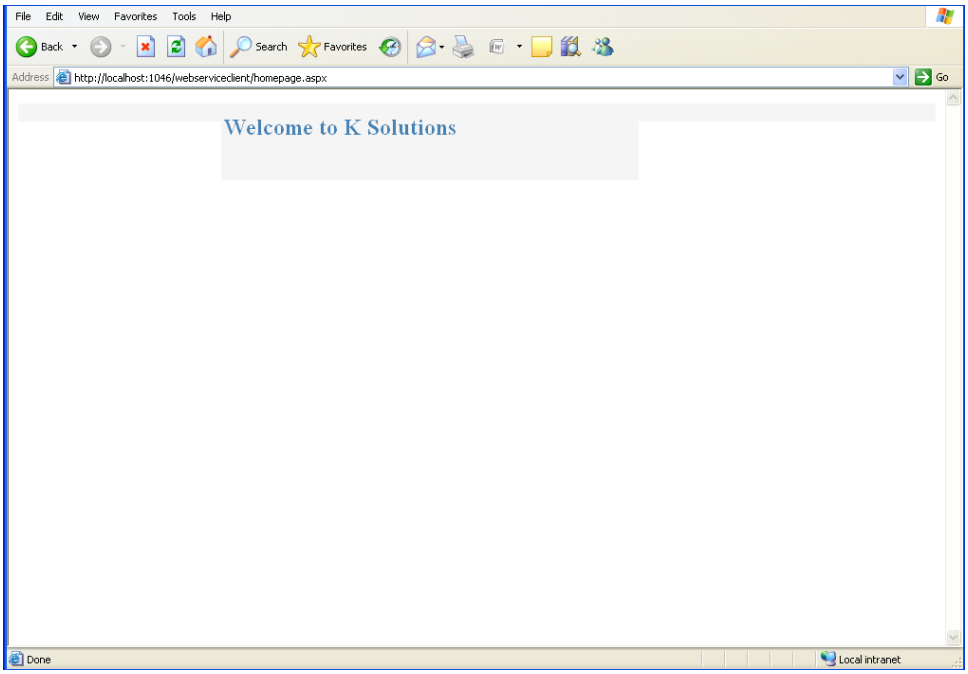


Figure 11.2-6: Home page.

Step 1: Create WebService by creating new WebSite /selecting asp.net web services using visual studio 2005.

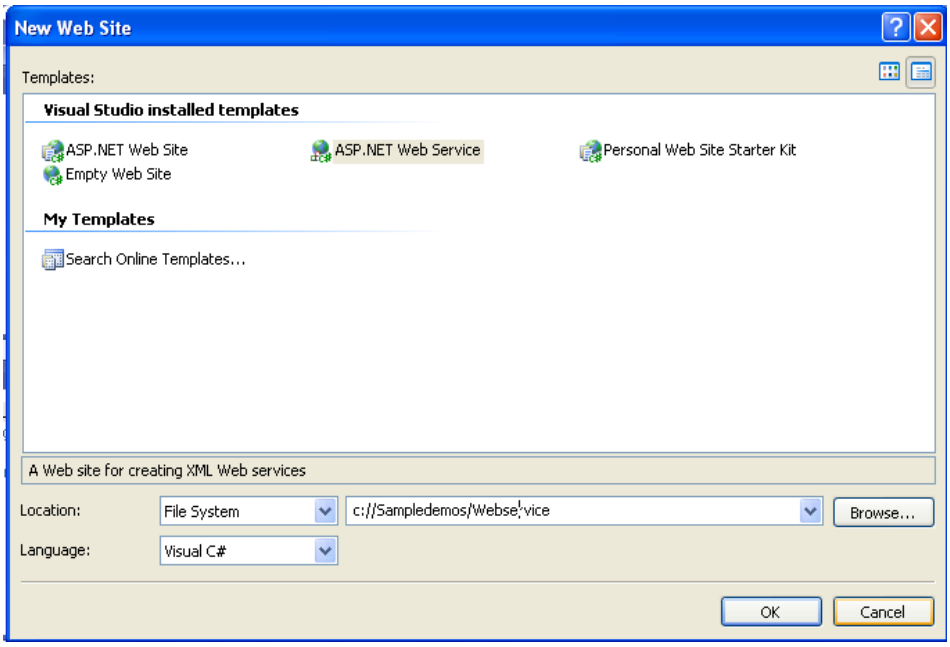


Figure 11.2-7: Creating Webservice.

Step 2: Create Webmethod named web_Login

```
string reply;
SqlCommand cmd;
public static string connectionstr =
"server=hstslc011;uid=sa;pwd=satyam;database=northwind";
 SqlConnection con= new SqlConnection(connectionstr);
```

```
[WebMethod]
    public string web_login(string userid,string pwd)
    { // return type is string and 2 parameters pass to web method
userid and pwd
     try
     {
     cmd = new SqlCommand("loginprocess",con );
     cmd.CommandType = CommandType.StoredProcedure;
     cmd.Parameters.Add("@eid", SqlDbType.Int).Value =
    int.Parse(userid.ToString());
     cmd.Parameters.Add("@pwd", SqlDbType.VarChar, 30).Value =
    pwd.ToString();
     cmd.Parameters.Add("@reply", SqlDbType.Int).Direction =
    ParameterDirection.Output;
     con.Open();
     cmd.ExecuteNonQuery();
     }
     catch (Exception ex)
     {
        reply = ex.Message;
     }
     finally
     {
        con.Close();
     }
     if ((int.Parse(cmd.Parameters["@reply"].Value.ToString()) > 0))
     {
        reply = "Successfully login";
     }
     else
     {
        reply = "Unsuccessfull login";
     }
     return reply;
    }
```

Step 3: Check whether webmethod created is running or not. If it is running then copy the address

Example:
http://localhost:4805/webservice/Service.asmx

Step 4: Create a new Web application project called webserviceclient using visual studio 2005. Rename Default.aspx to Login.aspx

Step 5: Insert HTML Table from toolbox in Login.aspx. Drag and drop the server controls in HTML table and change the (ID) property and other control property values as indicated in the following table:

Control type	ID	Property values to be modified
asp:label	lblLogin	Text:Login
asp:label	lblUserid	Text:User ID
asp:label	lblPwd	Text:Password
asp:label	lblReply	-
asp:textbox	txtUserid	-
asp:textbox	txtPwd	-
asp:button	btnLogin	Text:LOGIN
asp:button	btnCancel	Text: CANCEL

Step 6: Add web refrence to webserviceclient project. In the Solution Explorer, right click on the name of Web Application (webserviceclient) and select Add Web Refrence.

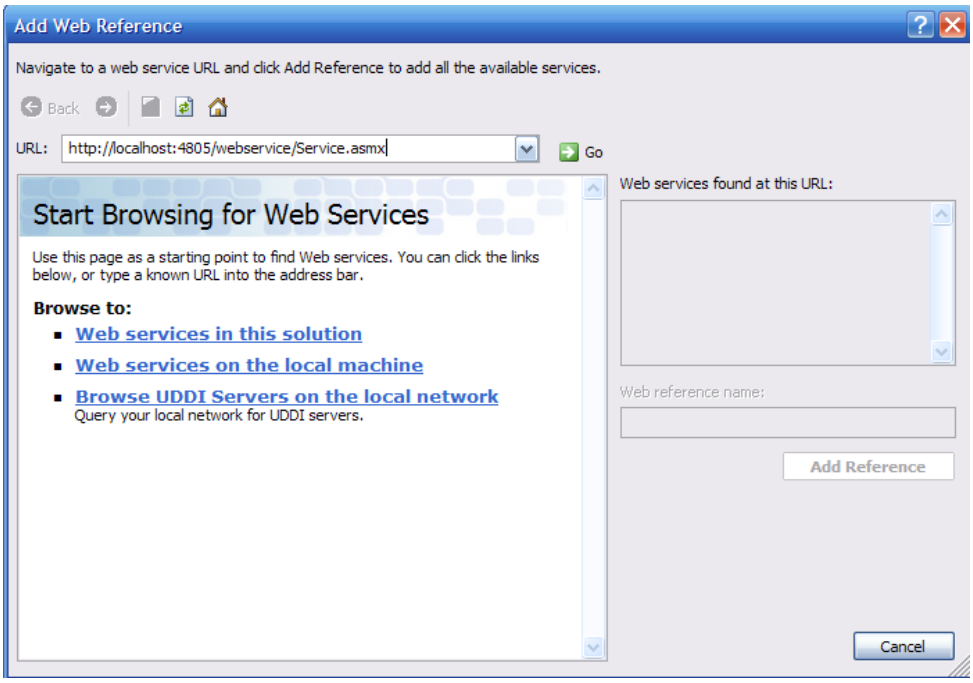


Figure 11.2-8: Adding web refrence in URL.

Step 7: Click on go web service is found and in web refrence name give name localhost and click on Add Reference.

Step 8: In Login.aspx page in order to use websevice localhost add namespace.

```
using localhost;
```

Step 9: Create the object for the localhost in class as shown below:

```
Service objbusiness = new Service();
```

Step 10: Call the webmethod in login submit button as shown below:

```
protected void btnLogin_Click(object sender, EventArgs e)
{
try
{
//webmethod called here
reply = (objbusiness.web_login(txtUserid.Text,
txtPwd.Text).ToString());
}
catch (Exception ex)
{
lblreply.Text = ex.Message;
}
if (reply == "Successfully login")
Response.Redirect("homepage.aspx");
else
{
     txtUserid.Text = "";
     txtUserid.Focus();
     lblreply.Text = reply;
}
}
```

Tables used: emplogindetails
Database: Northwind

	eid	pwd
1	1	aaa
2	2	bbb
3	3	ccc

Figure 11.2-8: Emplogin details table.

Procedures:

```
create proc loginprocess @eid int,@pwd varchar(30),@reply int output
as
begin
select @reply=count(*) from emplogindetails where eid=@eid and
pwd=@pwd
end
```

Context :

- Use the web method in different web application.
- Across platforms where there is support of http client(browser).
- Especially for n (>2) -tier application integration.

Practice Session/s :

- Create a web service that accepts Fahrenheit and convert into Celsius scale.
- Create a web service to compute compound interest.
- Create a web service that converts centimeters into feet. Assume that 1 foot is equal to 30cms.
- Create an ASP.NET page that accepts a value in centimeters from the user and uses the web service created in Q1 to return the equivalent value in feet.
- Create a web service that returns the phone number of an author, given the author's lastname. Use the authors' table in the pubs database. Create a client ASP.NET page for the web service.

Check List :

- Importance of associating [WebMethod] tag to each of the web method.
- Able to invoke a web service.
- Importance of WSDL document.
- Inherits WebServices class.

Common Error/s :

- Non inclusion of [WebMethod].
- Running the application without running web service atleast once.

Exception/s :

- Web service cannot be accessed because IIS is not installed.

Lesson/s Learnt :

- ☑ Understand the architecture of web service.
- ☑ The roles of XML, UDDI (Broker), SOAP, WSDL, Service Provider, Service Consumer .
- ☑ Including the namespace of System.Web.Services.WebServices.
- ☑ Inheriting WebServices class.
- ☑ Importance of [WebMethod] attribute.

Best Practice/s :

- Use webservice in application from security point of view.

- .aspx pages have references to **System. Drawing** since their purpose is to generate a user interface while .asmx pages have references to **System.Web.Services** since their purpose is to generate an interface for external programs.
- .aspx pages usually begin with an @Page directive to designate while .asmx pages usually begin with an @WebService directive.
- Using the wrong @ directive with the wrong type of file extension will generate an error.
- Before running the application check whether web service is running or not.

Crossword: Unit-11 **Estimated Time: 10 mins.**

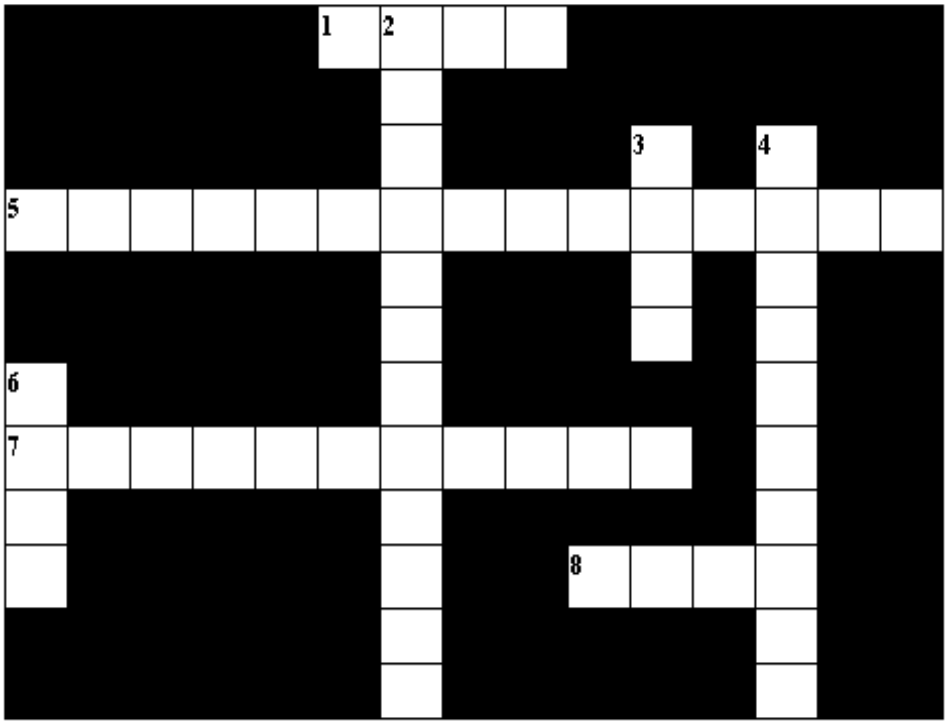


Fig. 11-1

Across:

1. One of the element of web service, which is a descriptive language, which defines and locates a web service.(4)

Down:

2. Which is the generic type of project that could be used for all type of applications including web based application.(12)

5. Which project type helps in creating virtual directories for web based applications during installation.(15)

7. Which type of project is used, when we want to install some additional third party software like MSDE along with the application.(11)

8. One of the element of web service, which is a directory for storing information about web services.(4)

3. One of the element provided by web service, which is a communication protocol used for sending messages.(4)

4. This is a service, which is programmable business logic components that provide access to functionality through the Internet.(10)

6. To create merge module installation of MSDE, which extension is used.(4)

Answers For Crosswords

Unit-1

Across	**2) Garbage collection 5) Class Library 6)Application Code 7)Bin**
Down	**1) Code Access Security 3) Platform 4)CLR**

Unit-2

Across	**5)Skinfile 6)Masterpage**
Down	**1)Validation 2)Master 3)Cascading 4)Webserver**

Unit-3

Across	**1).Aspx 3).Cs 6)Inlinecoding**
Down	**2)Serverside 4)Codebehind 5)Clientside**

Unit-4

Across	**3)Pagelevel 5)Tracecontext 6)Ten 7)Runtime**
Down	**1)Application 2)Tracing 4)Compiletime**

Unit-5

Across	**3)Range 4)Custom 5)Requiredfield 6)Compare**
Down	**1)Validation 2)Regularexpression**

Unit-6

Across	**2).Ascx 4)Src 5)Login 7)Register**
Down	**1)Usercontrol 3)Tagprefix 6)Form**

Unit-7

Across	**2)Dataadapter 4)Dataset 6)Datareader 7)Xml**
Down	**1)Steredprocedure 2)Disconnected 3)Ado.net 5)System.xml**

Unit-8

Across	**1)Querystring 3)View state 6)Applicationstate 7)Serverside 8)Sessionstate**
Down	**2)Statemanagement 4)Global.asax 5)Cookie**

Unit-9

Across	**4)Caching 6)Pagefragment 7)Data**
Down	**1)Web.config 2)Pageoutput 3)Machine.config 5)Duration**

Unit-10

Across	**2)Provider 4)Webbrowsable 6)Catalog 7)Authentication**
Down	**1)Profile 3)Browse 4)Webpartzone 5)Authorization**

Unit-11

Across	**1)Wsdl 5)Websetupproject 7)Mergemodule 8)Uddi**
Down	**2)Setupproject 3)Soap 4)webservice 6).Msm**

Team Members

Srikanth Nivarthi
47275

Sreenivas Ram
66223

Seshu Babu Barma
56150

Veerendra Kumar Ankem
77964

Teena Arora
74572

Contributors

Amit Puranik
66861

Pravesh Agrawal
66829

Kunal Parikh
72087

Bharat Joshi
72163

Nilesh Jain
72143

Bhavik Gosrani
72128

Reenal Chaudhari
72119

Sumit Phadnis
72144