Unit-IV

ADVANCED FEATURES OF ASP.NET

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic, C#, JScript .NET, and J#. These languages enable you to develop ASP.NET applications that benefit from the common language runtime, type safety, inheritance, and so on.

ASP.NET includes:

- A page and controls framework
- The ASP.NET compiler
- Security infrastructure
- State-management facilities
- Application configuration
- Health monitoring and performance features
- Debugging support
- An XML Web services framework
- Extensible hosting environment and application life cycle management
- An extensible designer environment

Page and Controls Framework

The ASP.NET page and controls framework is a programming framework that runs on a Web server to dynamically produce and render ASP.NET Web pages. ASP.NET Web pages can be requested from any browser or client device, and ASP.NET renders markup (such as HTML) to the requesting browser. As a rule, you can use the same page for multiple browsers, because ASP.NET renders the appropriate markup for the browser making the request. However, you can design your ASP.NET Web page to target a specific browser, such as Microsoft Internet Explorer 6, and take advantage of the features of that browser. ASP.NET supports mobile controls for Web-enabled devices such as cellular phones, handheld computers, and personal digital assistants (PDAs).

ASP.NET Web pages are completely object-oriented. Within ASP.NET Web pages you can work with HTML elements using properties, methods, and events. The ASP.NET page framework removes the implementation details of the separation of client and server inherent in Web-based applications by presenting a unified model for responding to client events in code that runs at the server. The framework also automatically maintains the state of a page and the controls on that page during the page processing life cycle.

ASP.NET Compiler

All ASP.NET code is compiled, which enables strong typing, performance optimizations, and early binding, among other benefits. Once the code has been compiled, the common language runtime further compiles ASP.NET code to native code, providing improved performance.

ASP.NET includes a compiler that will compile all your application components including pages and controls into an assembly that the ASP.NET hosting environment can then use to service user requests.

Security Infrastructure

In addition to the security features of .NET, ASP.NET provides an advanced security infrastructure for authenticating and authorizing user access as well as performing other security-related tasks. You can authenticate users using Windows authentication supplied by IIS, or you can manage authentication using your own user database using ASP.NET forms authentication and ASP.NET membership. Additionally, you can manage the authorization to the capabilities and information of your Web application using Windows groups or your own custom role database using ASP.NET roles. You can easily remove, add to, or replace these schemes depending upon the needs of your application.

ASP.NET always runs with a particular Windows identity so you can secure your application using Windows capabilities such as NTFS Access Control Lists (ACLs), database permissions, and so on.

State-Management Facilities

ASP.NET provides intrinsic state management functionality that enables you to store information between page requests, such as customer information or the contents of a shopping cart. You can save and manage application-specific, session-specific, page-specific, user-specific, and developer-defined information. This information can be independent of any controls on the page.

ASP.NET offers distributed state facilities, which enable you to manage state information across multiple instances of the same application on one computer or on several computers.

ASP.NET Configuration

ASP.NET applications use a configuration system that enables you to define configuration settings for your Web server, for a Web site, or for individual applications. You can make configuration settings at the time your ASP.NET applications are deployed and can add or revise configuration settings at any time with minimal impact on operational Web applications and servers. ASP.NET configuration settings are stored in XML-based files. Because these XML files are ASCII text files, it is simple to make configuration changes to your Web applications. You can extend the configuration scheme to suit your requirements.

Health Monitoring and Performance Features

ASP.NET includes features that enable you to monitor health and performance of your ASP.NET application. ASP.NET health monitoring enables reporting of key events that provide information about the health of an application and about error conditions. These events show a combination of diagnostics and monitoring characteristics and offer a high degree of flexibility in terms of what is logged and how it is logged.

ASP.NET supports two groups of performance counters accessible to your applications:

- The ASP.NET system performance counter group
- The ASP.NET application performance counter group

Debugging Support

ASP.NET takes advantage of the run-time debugging infrastructure to provide cross-language and cross-computer debugging support. You can debug both managed and unmanaged objects, as well as all languages supported by the common language runtime and script languages.

In addition, the ASP.NET page framework provides a trace mode that enables you to insert instrumentation messages into your ASP.NET Web pages.

XML Web Services Framework

ASP.NET supports XML Web services. An XML Web service is a component containing business functionality that enables applications to exchange information across firewalls using standards like HTTP and XML messaging. XML Web services are not tied to a particular component technology or object-calling convention. As a result, programs written in any language, using any component model, and running on any operating system can access XML Web services.

Extensible Hosting Environment and Application Life-Cycle Management

ASP.NET includes an extensible hosting environment that controls the life cycle of an application from when a user first accesses a resource (such as a page) in the application to the point at which the application is shut down. While ASP.NET relies on a Web server (IIS) as an application host, ASP.NET provides much of the hosting functionality itself. The architecture of ASP.NET enables you to respond to application events and create custom HTTP handlers and HTTP modules.

Extensible Designer Environment

ASP.NET includes enhanced support for creating designers for Web server controls for use with a visual design tool such as Visual Studio. Designers enable you to build a design-time user interface for a control, so that developers can configure your control's properties and content in the visual design tool.

USING THE NEW SECURITY CONTROLS IN ASP.NET 2.0

Implementing security in a site has the following aspects:

- **Authentication** â€“ it is the process of ensuring the user’s identity and authenticity. ASP.Net allows four types of authentication system:
    - Windows Authentication

  - Forms Authentication
  - Passport Authentication
  - Custom Authentication

- **Authorization** â€“ it is the process of defining and allotting specific roles to specific users.
- **Confidentiality** â€“ it involves encrypting the channel between the client's browser and the web server.
- **Integrity** â€“ it involves maintaining the integrity of data. For example, implementing digital signature.

ASP.NET comes with several new security controls (located under the Login tab in the Toolbox; see Figure 1) that greatly simplify the life of a Web developer. Using the new security controls, we can now perform tasks such as user logins, registration, password changes, and more, with no more effort than dragging and dropping controls onto Web form.

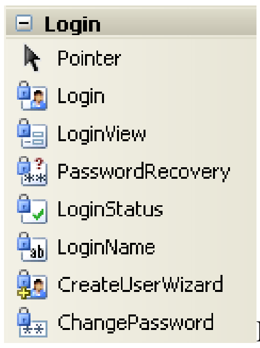


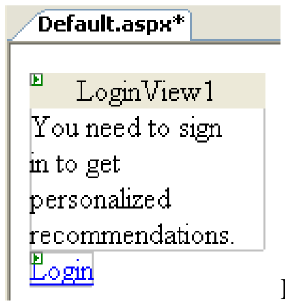
Figure 1: The new security controls in ASP.NET 2.0.

To begin, lets explore using the **LoginView**, **LoginStatus** and **LoginName** controls. First, let's build a Web project using Visual Studio 2005 Beta 2, so go ahead and launch the Visual Studio IDE. From the File menu, click New Web Site to create a new Web project. Name the project C:\SecurityControls.

You need to set the ContinueDestinationPageURL property of the CreateUserWizard control so that when the Continue button is clicked the user can be redirected to another page, such as a welcome page.

In the Default.aspx Web form, drag and drop the **LoginView** control. The **LoginView** control is a container control that displays different information depending on whether the user is logged in or not.

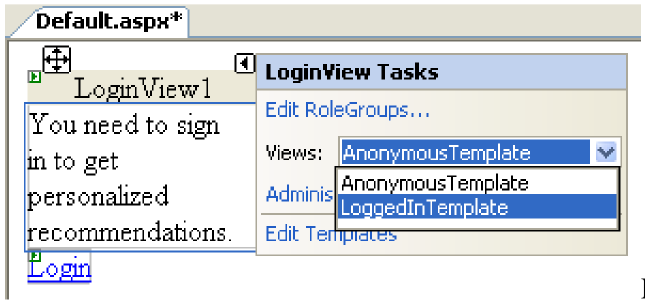
Populate the **LoginView** control with the text shown in **Figure 2**. Also, drag and drop the Login control onto the **LoginView** control. The text that you have just typed will be displayed when the user is not yet authenticated (anonymous). The Login control displays a link to allow the user to be redirected to another page to log into the application.



You need to sign in to get personalized recommendations.

Figure 2: Populating the LoginView control.

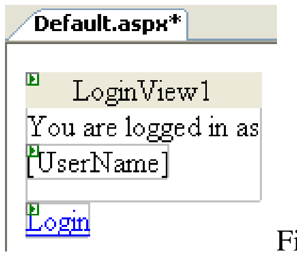
In the Smart Tasks menu of the **LoginView** control, change the Views to "LoggedInTemplate" (**Figure 3**).



You need to sign in to get personalized recommendations.

Figure 3: Changing the view of the LoginView control.

With the view changed, enter the text shown in **Figure 4** into the LoginView control. This text will be displayed once the user has been authenticated. Drag and drop the **LoginName** control onto the **LoginView** control. The **LoginName** control will display the name of the user that is used to log into the application.



You are logged in as [UserName]

Figure 4: This text will display when the user is authenticated.

Using the Login Control

Let's now add a new Web form to the project (right-click on project name in Solution Explorer and select Add New Item...) and name it Login.aspx. Your application will use this form to let users log into the application.

Drag and drop the Login control onto Login.aspx. You can apply formatting to the Login control to make it look more professional. Click on the Smart Tag of the Login control and select the Auto Format...link (**Figure 5**).

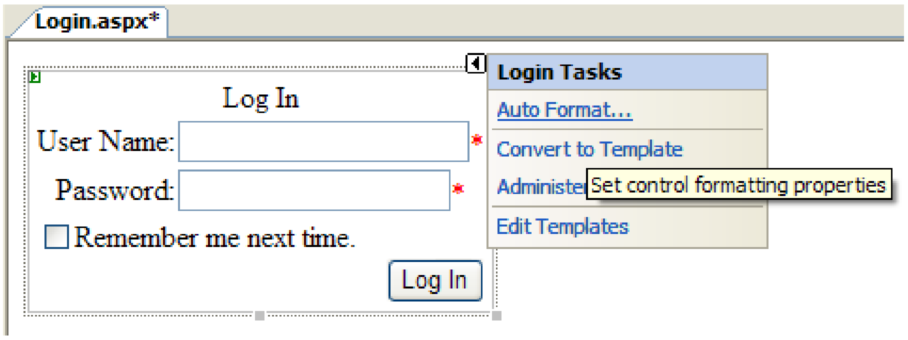


Figure 5: Applying auto format to the Login control.

Select the Colorful scheme and the Login control should now look like **Figure 6**.

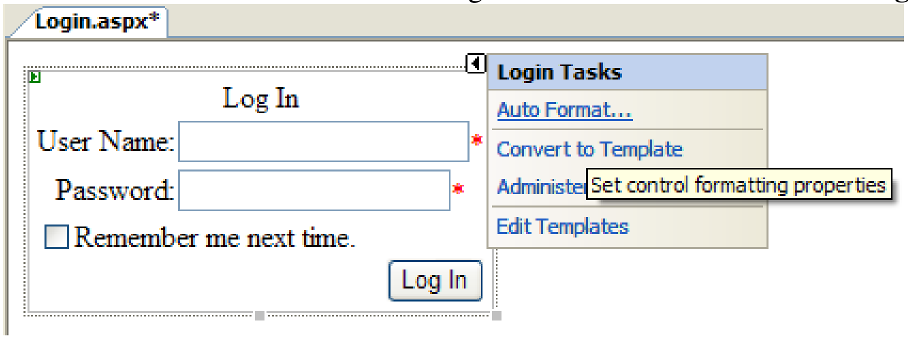


Figure 6: The new look of the Login control after applying the Colorful scheme.

By default, ASP.NET 2.0 uses Windows authentication, which is not very flexible if you are targeting Internet users. And so you will change the default authentication mode from Windows to Forms.

Add a Web.config file to your project (right-click on project name in Solution Explorer and select Add New Item.... From the list of available choices select Web Configuration File).

In Web.config, change the authentication mode from Windows to Forms by adding the following line of code. You use forms authentication so that you can add users to your Web site without needing to create the user accounts in Windows.

```
<system.web>

  <authentication mode="Forms"/>

...
```

Adding a New User to Your Application

Before you proceed to test the application, you need to create a new user for the application. You can use the ASP.NET Web Site Administration Tool (WAT) to add a new user to your application. To invoke the WAT, select Website and then choose ASP.NET Configuration (**Figure 7**).

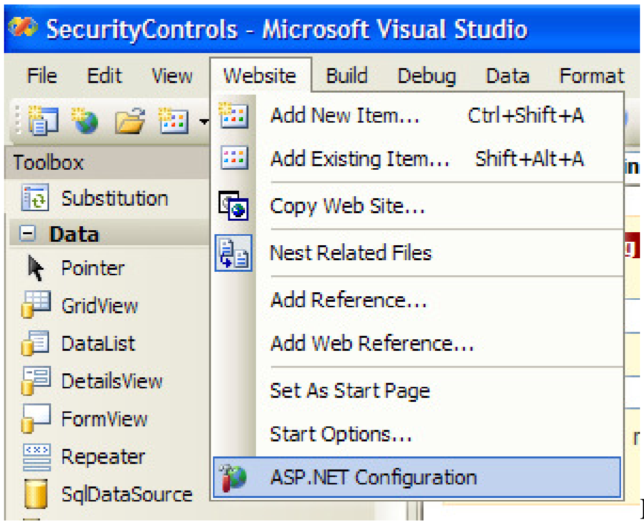
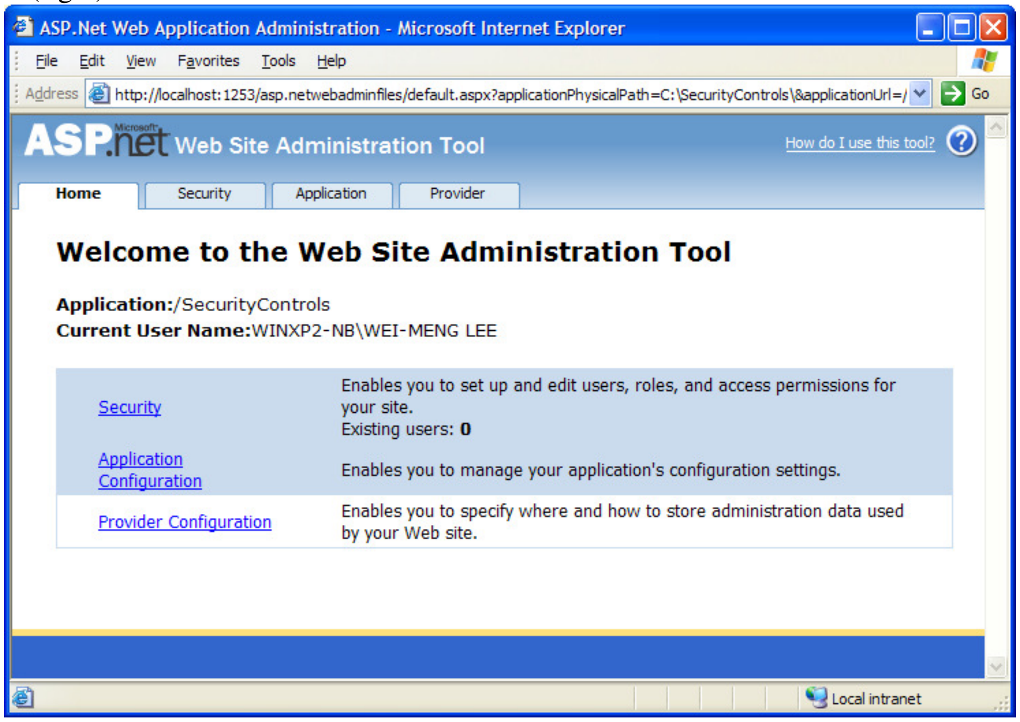


Figure 7: Invoking the WAT.

The WAT will be displayed in a new Web page. Click the Security link to go to the Security tab(fig.8)


ASP.Net Web Application Administration - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Address
http://localhost:1253/asp.netwebadminfiles/default.aspx?applicationPhysicalPath=C:\SecurityControls\&applicationUrl=/
Go
ASP.net Web Site Administration Tool
How do I use this tool?
Home
Security
Application
Provider
Local intranet

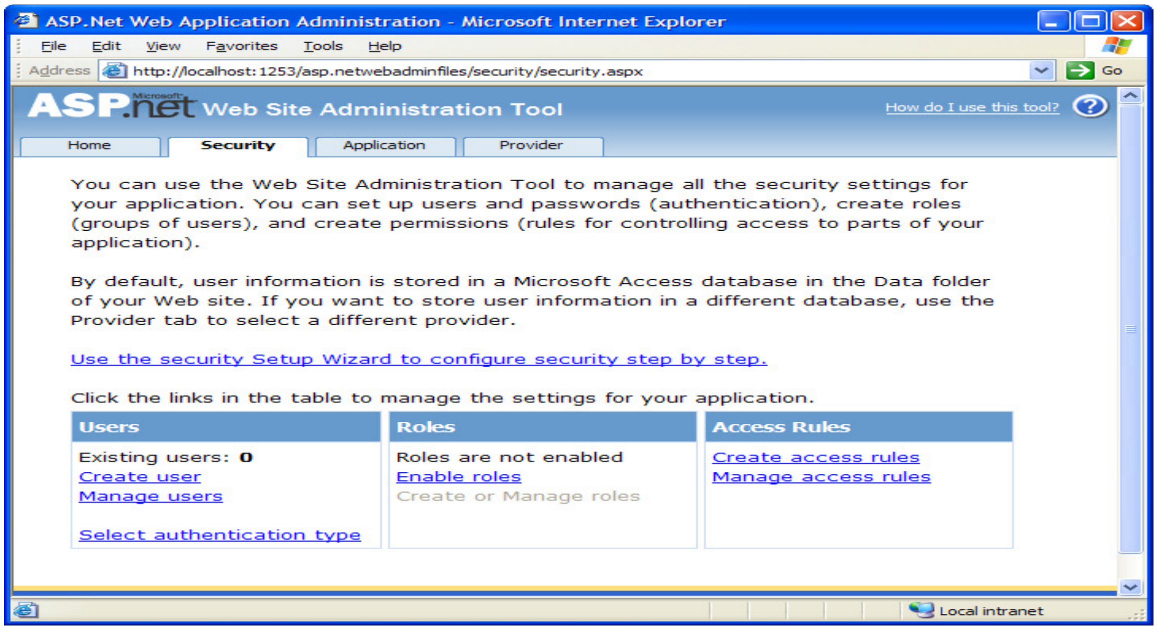
Welcome to the Web Site Administration Tool

Application:/SecurityControls
Current User Name:WINXP2-NB\WEI-MENG LEE

Security	Enables you to set up and edit users, roles, and access permissions for your site. Existing users: **0**
Application Configuration	Enables you to manage your application's configuration settings.
Provider Configuration	Enables you to specify where and how to store administration data used by your Web site.

Figure8:The WAT.

The Security tab allows you to perform tasks such as creating and deleting users as well as creating roles and access rules for your application. Click on the Create user link to add a new user to your application (**Figure 9**).


ASP.Net Web Application Administration - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:1253/asp.netwebadminfiles/security/security.aspx
Go
ASP.net Web Site Administration Tool
How do I use this tool?
Home
Security
Application
Provider
Local intranet

You can use the Web Site Administration Tool to manage all the security settings for your application. You can set up users and passwords (authentication), create roles (groups of users), and create permissions (rules for controlling access to parts of your application).

By default, user information is stored in a Microsoft Access database in the Data folder of your Web site. If you want to store user information in a different database, use the Provider tab to select a different provider.

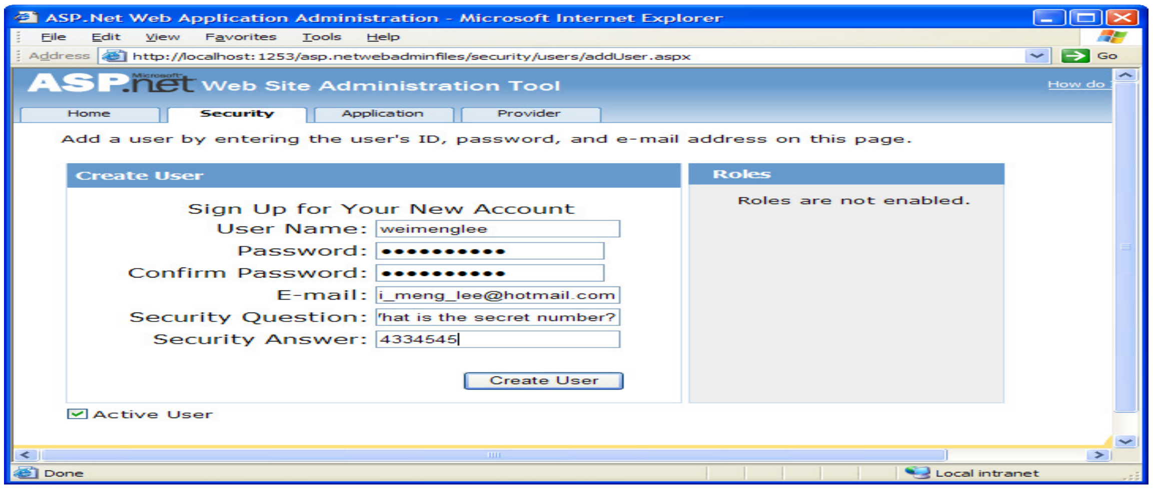
Use the security Setup Wizard to configure security step by step.

Click the links in the table to manage the settings for your application.

Users	Roles	Access Rules
Existing users: **0**	Roles are not enabled	Create access rules
Create user	Enable roles	Manage access rules
Manage users	Create or Manage roles	
Select authentication type		

Figure 9: The Security tab in the WAT.

Supply the required information for the new user account (**Figure 10**). Note that the password must have a combination of numeric, alphabetical, and special characters. Be sure to supply at least seven characters for the password. Click Create User to add the new user.


ASP.Net Web Application Administration - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:1253/asp.netwebadminfiles/security/users/addUser.aspx
Go
ASP.net Web Site Administration Tool
How do
Home
Security
Application
Provider
Create User
Roles
Done
Local intranet

Add a user by entering the user's ID, password, and e-mail address on this page.

Sign Up for Your New Account

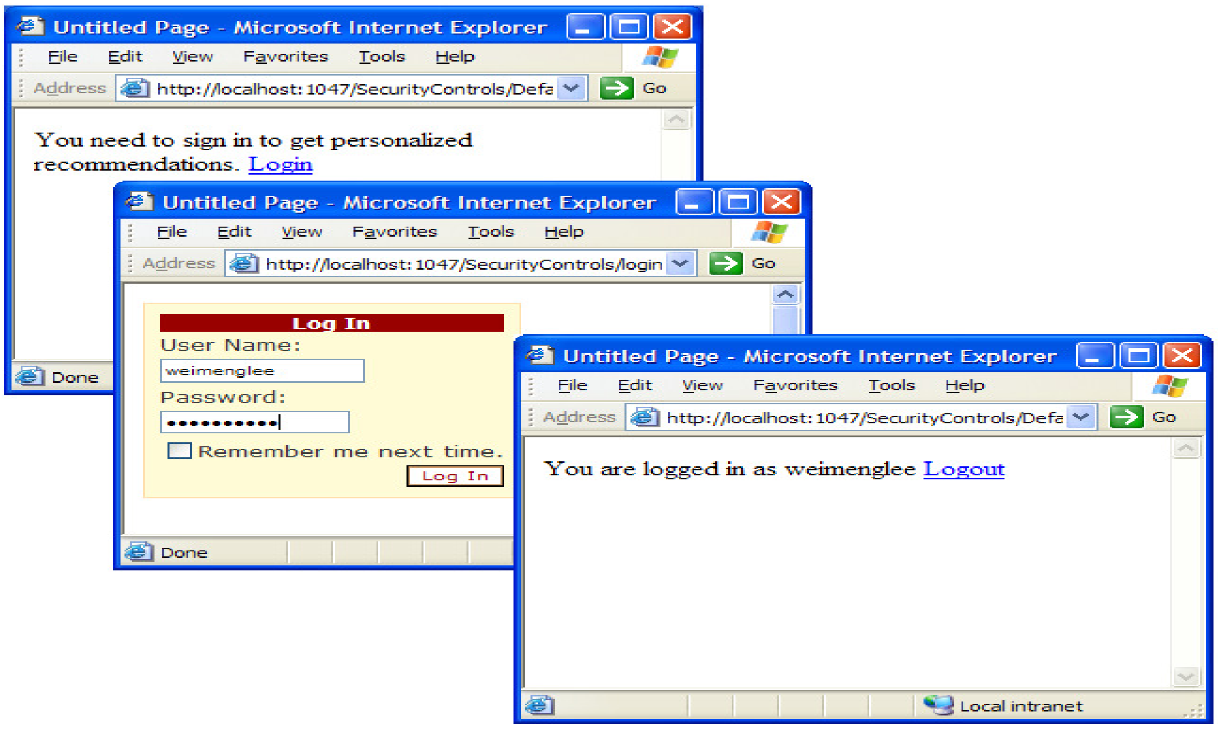
User Name: weimenglee
Password: ●●●●●●●●●●●
Confirm Password: ●●●●●●●●●●●
E-mail: i_meng_lee@hotmail.com
Security Question: 'hat is the secret number?
Security Answer: 4334545

Roles are not enabled.

☑ Active User

Figure 10: Adding a new user account to your application.

You are now ready to test the application. Select Default.aspx in Solution Explorer and press F5. Click the Login link to log into the application and then enter the account information. When you have successfully logged into the application, the Login link changes to Logout. **Figure 11** shows the sequence of events.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:1047/SecurityControls/Defa
Go
Done
Local intranet

You need to sign in to get personalized recommendations. Login

Address http://localhost:1047/SecurityControls/login

Log In
User Name: weimenglee
Password: ••••••••••
Remember me next time.
Log In

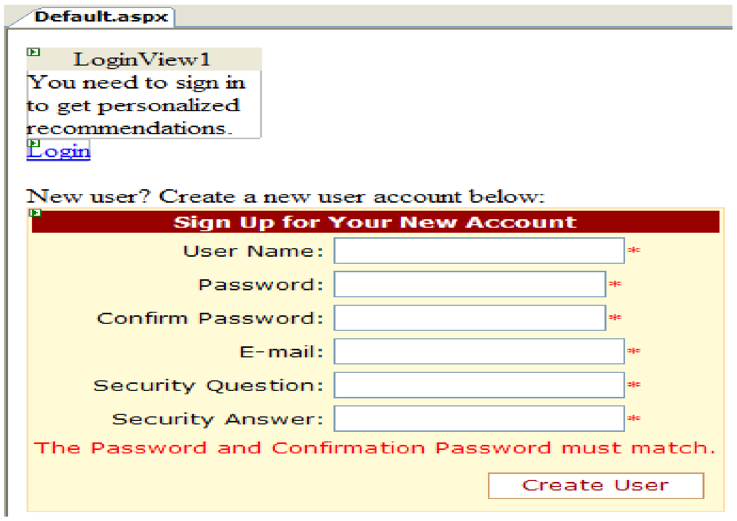
You are logged in as weimenglee Logout

Figure 11: Logging in to the application.

Creating New Users

Besides creating user accounts for users, you can also allow users to create new accounts themselves. This is useful in scenarios where you allow users to create free accounts in order to access your application, such as in a discussion forum.

To allow users to create new accounts, use the **CreateUserWizard** control. Drag and drop the **CreateUserWizard** control onto Default.aspx and apply the **Colorful** scheme. The control should now look like **Figure 12**.



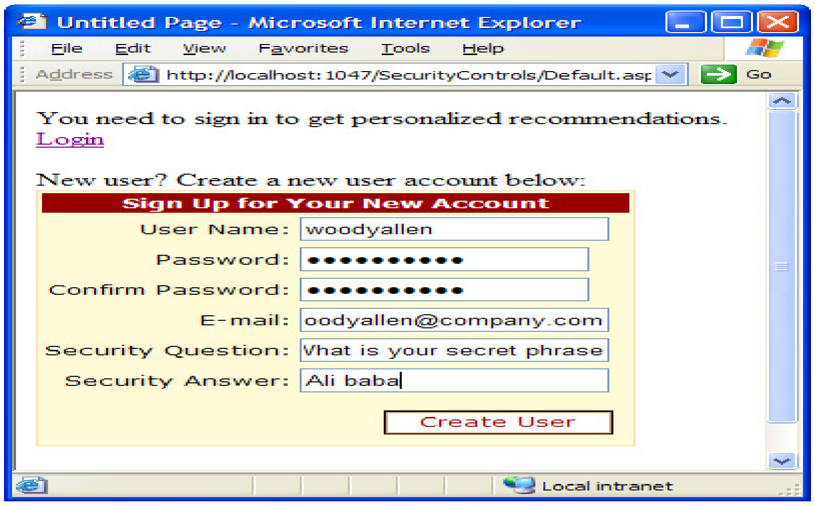
LoginView1
You need to sign in to get personalized recommendations.
Login

New user? Create a new user account below:

Sign Up for Your New Account
User Name: *
Password: *
Confirm Password: *
E-mail: *
Security Question: *
Security Answer: *
The Password and Confirmation Password must match.
Create User

Figure 12: Creating new user accounts using the CreateUserWizard control.

To test the application, press F5. You can now create a new user account yourself (**Figure 13**). Supply the needed information and click Create user


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:1047/SecurityControls/Default.asp
Go
Create User
Local intranet

You need to sign in to get personalized recommendations.
Login

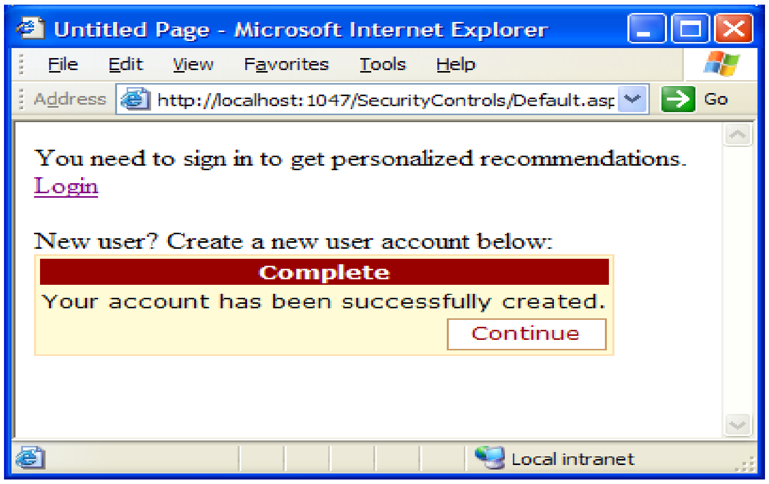
New user? Create a new user account below:

Sign Up for Your New Account

User Name:	woodyallen
Password:	●●●●●●●●●●●
Confirm Password:	●●●●●●●●●●●
E-mail:	oodyallen@company.com
Security Question:	Vhat is your secret phrase
Security Answer:	Ali baba

Figure 13: Creating a new user account.

When the user is created successfully, you will see the screen as shown in **Figure 14**.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:1047/SecurityControls/Default.asp
Go
Continue
Local intranet

You need to sign in to get personalized recommendations.
Login

New user? Create a new user account below:

Complete

Your account has been successfully created.

Figure 14: A new account is successfully created.

Where Is the User's Information Stored?

So far you have seen how to create users using the WAT as well as using the CreateUserWizard control. You're probably wondering where this information is stored. If you now examine the Solution Explorer and refresh the App_Data folder (right-click on it and select Refresh Folder), you will see an item named ASPNETDB.MDF (**Figure 15**).

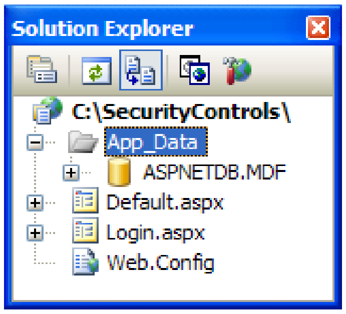


Figure 15: The ASPNETDB.MDF database file.

The ASPNETDB.MDF is a SQL Server 2005 Express database that ASP.NET 2.0 uses by default to store application-related data such as user accounts, profiles, etc. To examine the database, double-click it and you'll see its content displayed in the Database Explorer (**Figure 16**). Specifically, the aspnet_Membership and aspnet_Users tables will store the user accounts information that you have just created in the previous sections. To view the content of the tables, right-click on the table name and select Show Table Data.

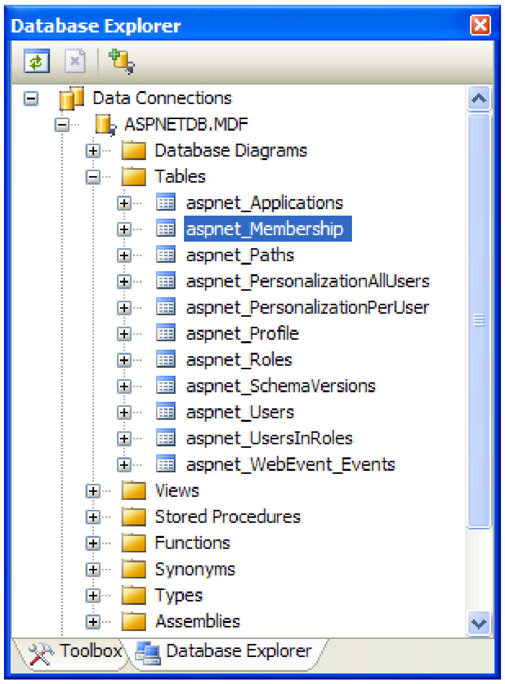


Figure 16: Examining the ASPNETDB.MDF database.

One really nice feature of ASP.NET 2.0 is that there is no need to create custom databases to store your users' information. And you don't even need to worry about hashing the users' password to store them securely. ASP.NET 2.0 does this automatically for you.

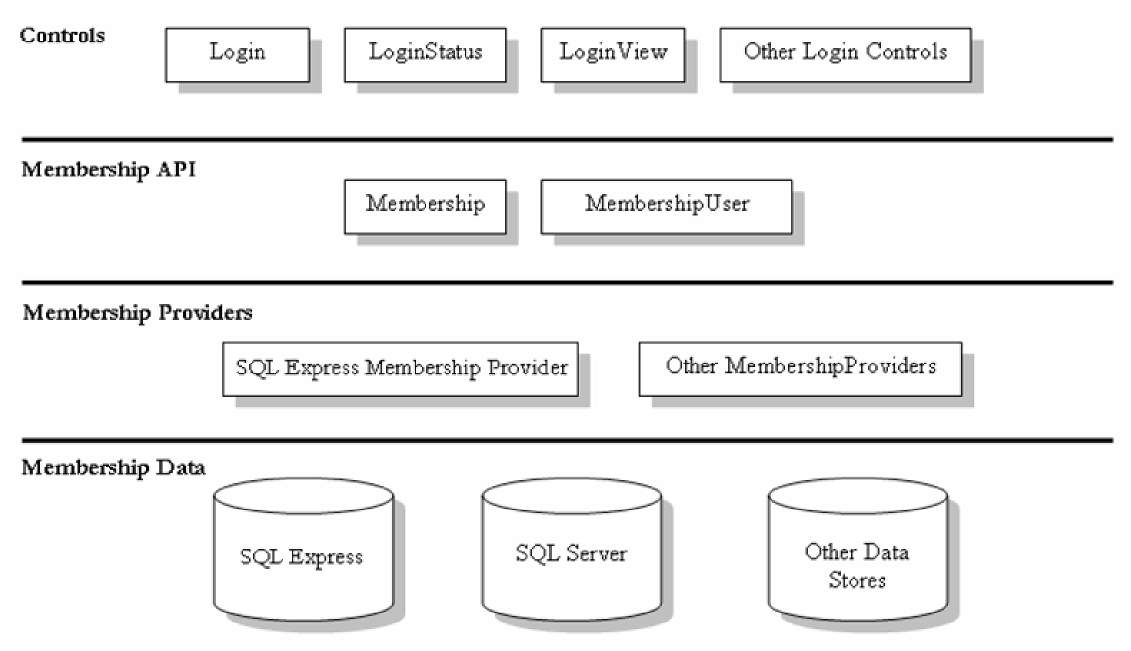


Figure 17: The Membership Provider model.

Recovering Lost Passwords

Recovering/resetting lost passwords is a common task that you need to perform as an administrator. The **PasswordRecovery** control allows users to perform this mundane task themselves by automatically retrieving the password and then sending it to the user via e-mail.

Password recovery makes sense only if you store the password as plain text and not its hashed value. However, by default, the settings in the machine.config file specify that all passwords be hashed before they are stored in the member database. Machine.config also disables password retrieval by default.

To store the user's password in plain text, add the following entry in Web.config.

```
...
<system.web>
  <membership
    defaultProvider="SqlProvider"
    userIsOnlineTimeWindow="15">
    <providers>
      <clear />
        <add name="SqlProvider  type="System.Web.Security.SqlMembershipProvider"
connectionStringName="LocalSqlServer"
```

```
applicationName="SecurityControls"

enablePasswordRetrieval="true"

enablePasswordReset="true"

requiresQuestionAndAnswer="true"

requiresUniqueEmail="true"

passwordFormat="Clear" />     </providers>   </membership>

...
```

Specifically, you are clearing all the Membership Providers and then adding a new **SqlMembershipProvider**. Note that you need to set the **enablePasswordRetrieval** (to true) and **passwordFormat** (to Clear) attributes in order to allow passwords to be retrieved.

If you set the **passwordFormat** as Hashed, then you must set **enablePasswordReset** to false.

Now drag and drop the **PasswordRecovery** control onto Default.aspx and then apply the Colorful scheme. The **PasswordRecovery** control now looks like **Figure 18**.

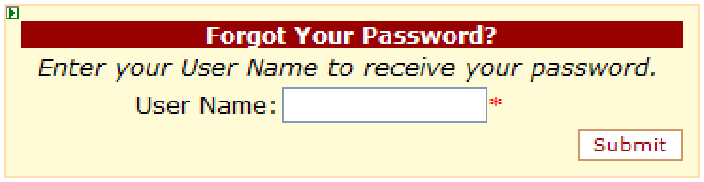
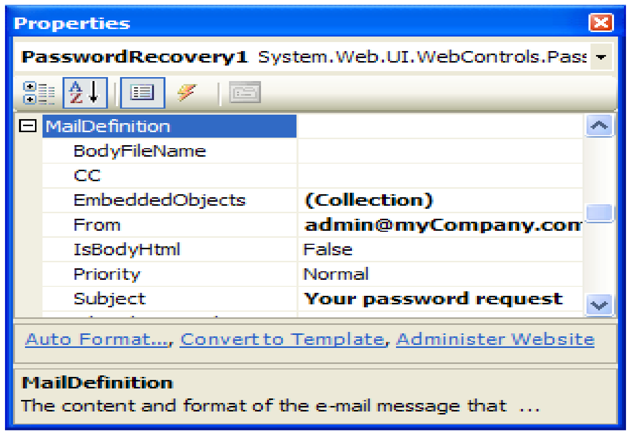


Figure 18: The PasswordRecovery control.

In the Properties window of the **PasswordRecovery** control, set the From and Subject fields under the **MailDefinition** property as shown in **Figure 19**.


Properties
PasswordRecovery1 System.Web.UI.WebControls.Pass
Auto Format..., Convert to Template, Administer Website

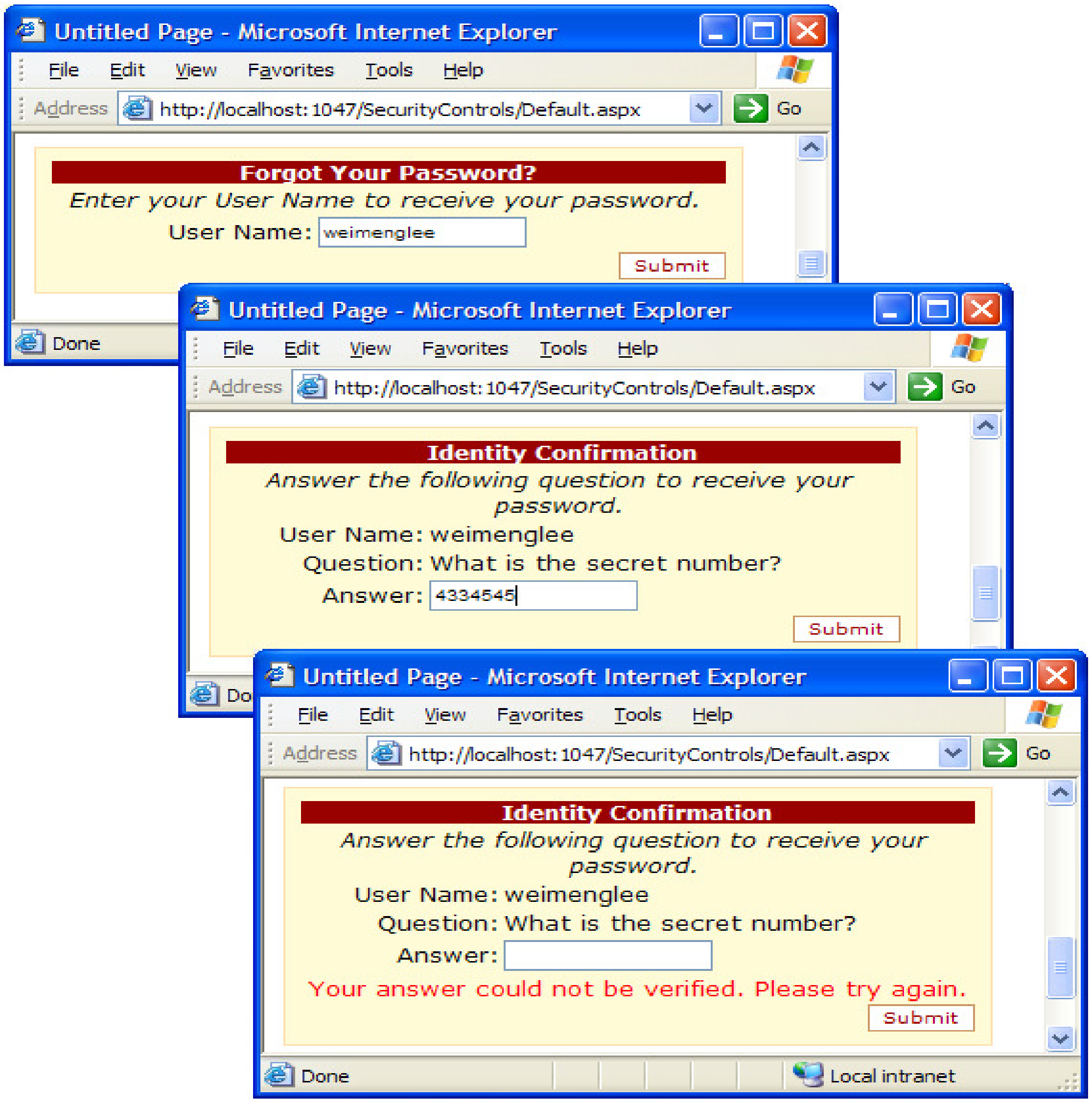
MailDefinition	
BodyFileName	
CC	
EmbeddedObjects	(Collection)
From	admin@myCompany.com
IsBodyHtml	False
Priority	Normal
Subject	Your password request

MailDefinition
The content and format of the e-mail message that ...

Figure 19 : Configuring the PasswordRecovery control.

You also need to have SMTP service configured on your machine for the **PasswordRecovery** control to send an e-mail. To configure SMTP service on your machine, start WAT, choose Application, then choose Configure SMTP e-mail settings.

To test the application, press F5. You will be prompted for your user name and then your security question. If the answer to the security question is correct, the password will be e-mailed to you; otherwise you will get an error message on the page like that shown in **Figure 20**.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost:1047/SecurityControls/Default.aspx
Go
Done
Local intranet

Forgot Your Password?
Enter your User Name to receive your password.
User Name: weimenglee
Submit

Identity Confirmation
Answer the following question to receive your password.
User Name: weimenglee
Question: What is the secret number?
Answer: 4334545
Submit

Identity Confirmation
Answer the following question to receive your password.
User Name: weimenglee
Question: What is the secret number?
Answer:
Your answer could not be verified. Please try again.
Submit

Figure 20 : Recovering lost password.

For security reasons, it is not a good idea to send a user's password through e-mail. Hence, you really need to consider using this option very carefully.

Changing Passwords

Besides recovering lost passwords, you also need to allow users to change their passwords. In ASP.NET 2.0, you can do so using the **ChangePassword** control.

Since a user can only change their password after they have logged in, you will now create a new folder in your application that is accessible to only authenticated users.

You can add a new folder to your application by right-clicking on the project name in Solution Explorer, choose Add Folder, and then choose Regular Folder. Name the folder "Members." Now add a new Web form to this new folder (right-click on Members and then select Add New Item...). Name the new Web form ChangePassword.aspx (**Figure 21**).

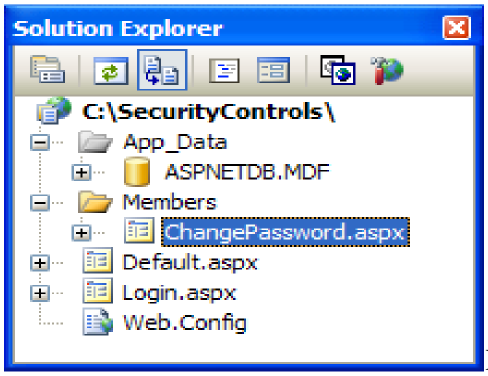


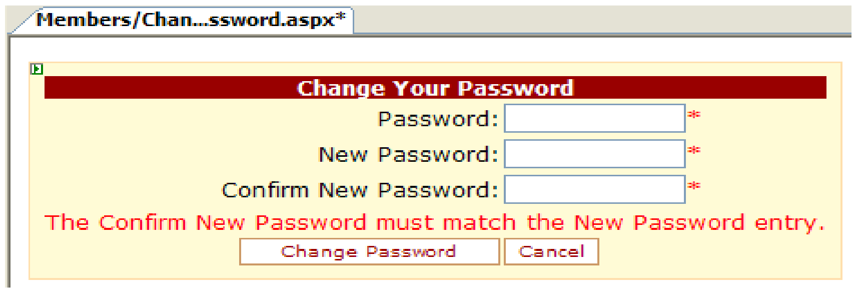
Figure 21 : Adding a new folder to the project.

To restrict accesses to the Members folder, add the following <location> element to Web.config.

```
...
</system.web>
  <location path="Members">
    <system.web>
      <authorization>
        <deny users="?" />
      </authorization>
    </system.web>
  </location>
</configuration>
```

Essentially, pages within the Members folder are only accessible to authorized users (all anonymous users (?) will be denied access).

Drag and drop the **ChangePassword** control onto ChangePassword.aspx and apply the **Colorful** scheme (**Figure 22**).


Members/Chan...ssword.aspx*

Change Your Password

Password: *

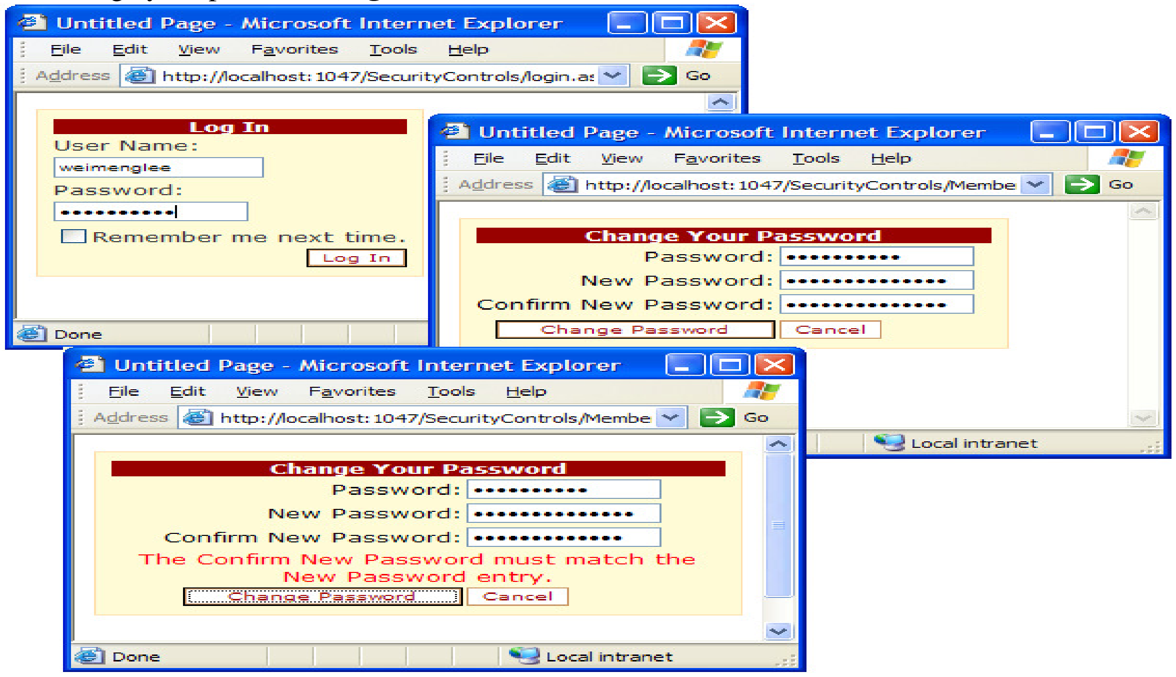
New Password: *

Confirm New Password: *

The Confirm New Password must match the New Password entry.

Figure 22: The ChangePassword control.

To test the application, in Solution Explorer select the ChangePassword.aspx file in the Members folder and press F5. You will first be redirected to the login.aspx page (for authentication) and once authenticated the ChangePassword.aspx page will be loaded. You can now change your password (**Figure 23**).


Untitled Page - Microsoft Internet Explorer

Address http://localhost:1047/SecurityControls/login.as

Log In

User Name: weimenglee

Password: ●●●●●●●●●●●

Remember me next time.

Untitled Page - Microsoft Internet Explorer

Address http://localhost:1047/SecurityControls/Membe

Change Your Password

Password: ●●●●●●●●●●

New Password: ●●●●●●●●●●●●●●

Confirm New Password: ●●●●●●●●●●●●●●

Untitled Page - Microsoft Internet Explorer

Address http://localhost:1047/SecurityControls/Membe

Change Your Password

Password: ●●●●●●●●●●

New Password: ●●●●●●●●●●●●●●

Confirm New Password: ●●●●●●●●●●●●●

The Confirm New Password must match the New Password entry.

Figure 23: Changing passwords using the ChangePassword control.

STATE MANAGEMENT

HTTP is a stateless protocol. Once the server serves any request from the user, it cleans up all the resources used to serve that request. These resources include the objects created during that request, the memory allocated during that request, etc. For a guy coming from a background of Windows application development, this could come as a big surprise because there is no way he could rely on objects and member variables alone to keep track of the current state of the application.

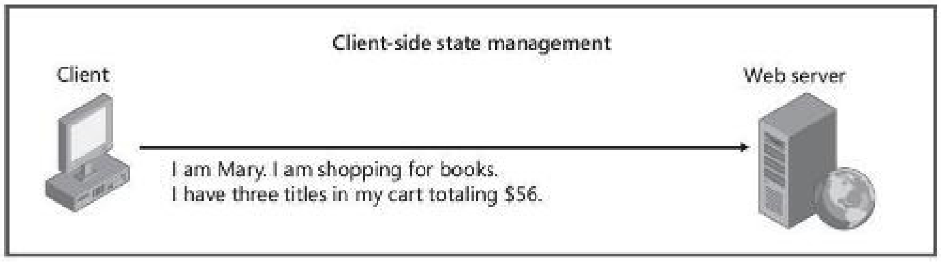
If we have to track the users' information between page visits and even on multiple visits of the same page, then we need to use the State management techniques provided by ASP.NET. State management is the process by which ASP.NET let the developers maintain state and page information over multiple request for the same or different pages.

Types of State Management

There are mainly two types of state management that ASP.NET provides:

1. Client side state management
2. Server side state management

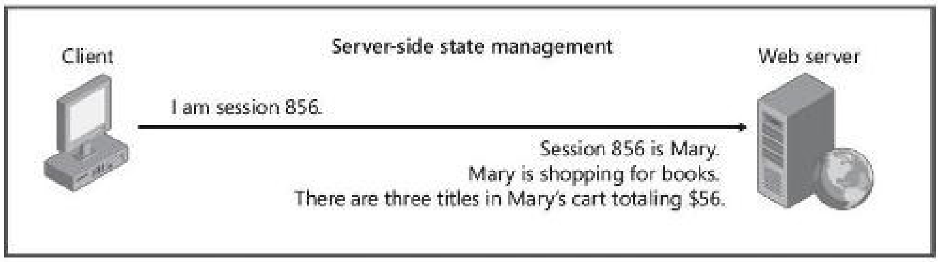
When we use client side state management, the state related information will be stored on client side. This information will travel back and forth with every request and response. This can be visualized as:


Client-side state management
Client
Web server
I am Mary. I am shopping for books.
I have three titles in my cart totaling $56.

Note: Image taken from Microsoft press' Book.

The major benefit of having this kind of state management is that we relieve the server from the burden of keeping the state related information, it saves a lot of server memory. The downside of client side state management is that it takes more bandwidth as considerable amount of data is traveling back and forth. But there is one more problem which is bigger than the bandwidth usage problem. The client side state management makes the information travel back and forth and hence this information can be intercepted by anyone in between. So there is no way we can store the sensitive information like passwords, creditcard number and payable amount on client side, we need server side state management for such things.

Server side state management, in contrast to client side, keeps all the information in user memory. The downside of this is more memory usage on server and the benefit is that users' confidential and sensitive information is secure.


Server-side state management
Client
Web server
I am session 856.
Session 856 is Mary.
Mary is shopping for books.
There are three titles in Mary's cart totaling $56.

Note: Image taken from Microsoft press' Book.

We cannot say that we will use any one type of state management in our application. We will have to find a mix of client side and server side state management depending on the type and size of information. Now let us look at what are the different ways we can manage state on client side and server side.

Client side state management techniques

- View State
- Control State
- Hidden fields
- Cookies
- Query Strings

Server side state management techniques

- Application State
- Session State

View State

ASP.NET uses this mechanism to track the values of the controls on the web page between page request for same page. We can also add custom values to view state. ASP.NET framework takes care of storing the information of controls in view state and retrieving it back from viewstate before rendering on postback.

If we need to use viewstate to store our information, we just need to remember that the viewstate is a dictionary object. We can have our data stored as key value pair in viewstate (see code below). The controls information is also being hashed into this dictionary during request and populated back during response.

Since this information is stored in the web page itself, ASP.NET encrypts the information. We can tweak the encryption related parameters from *web.config*.

```
<Configuration>
     <system.web>
      <pages viewStateEncryptionMode="Always"/>
    </system.web>
</configuration>
```

or page declarative:

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

Let us now look at a small implementation for viewstate. We have a simple web page with a textbox and a button. The idea is that we will write something in the text box and see how ASP.NET stores this information in view state. We will store our own information in the view state too. When we run the page and write my name in the textbox and press the button, a postback occurs but my name still remains in the textbox. Viewstate made that possible so after postback, the page looks like:

When we look at the source, the view state looks like:

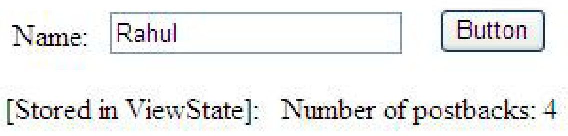
```
<input type="hidden" name="__VIEWSTATE"
id="__VIEWSTATE"
value="/wEPDwUKMTkwNjc4NTIwMWRkfIZa4Yq8wUbdaypyAjKouH5Vn1Y=" />
```

Now let us try to add our own information in the viewstate. Let's keep track of users' postback on this page. Whenever user will hit a button, we will add 1 to the stored pot back value. The way to do would be:

```
protected void Page_Load(object sender, EventArgs e)
{
   if(IsPostBack == true)
   {
      if (ViewState["number"] != null) //Lets retrieve, increase and store again
      {
         ViewState["number"] = Convert.ToInt32(ViewState["number"]) + 1;
      }
      else //First postback, lets store the info
      {
         ViewState["number"] = 1;
      }

      Label1.Text = ViewState["number"].ToString();
   }
}
```

When we run the page and hit the button to do a postback, the web will show us the postbacks being done so far which is being stored in viewstate:



View State is enabled by default, but we can disable it by setting the EnableViewState property for each web control to false. This reduces the server processing time and decreases page size.

Control State

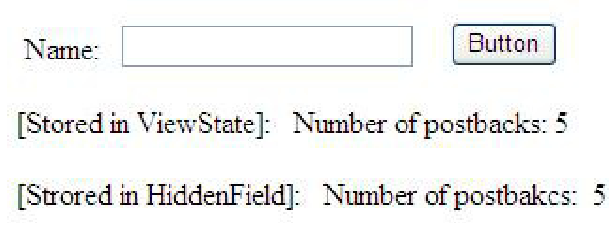
We now know what a viewstate is and we also know that we can disable viewstate for controls on the page. But imagine if we are developing a custom control and we internally are using viewstate to store some information but the user of the control can disable the viewstate for our control. To avoid this problem, we can have viewstate like behavior which cannot be disabled by control users and it is called ControlState. Control states lies inside custom controls and work the same as viewstate works.

To use control state in a custom control, we have to override the OnInit method and call the RegisterRequiresControlState method during initialization. Then we have to override the SaveControlState and LoadControlState methods.

Hidden Fields

Hidden field are the controls provided by the ASP.NET and they let use store some information in them. The only constraint on hidden filed is that it will keep the information when HTTP post is being done, i.e., button clicks. It will not work with HTTP get. Let us do the same exercise of keeping track of postbacks using HiddenFields now.
(**Note**: ViewState also uses hidden field underneath.)

```
//Store in Hidden Field ---------------------------------------------------------
int newVal = Convert.ToInt32(HiddenField1.Value) + 1; //Hidden field default value was 0
HiddenField1.Value = newVal.ToString();
Label2.Text = HiddenField1.Value;
```



[Stored in ViewState]: Number of postbacks: 5

[Strored in HiddenField]: Number of postbakcs: 5

When we run the page and hit the button to do a postback, the web will show us the postbacks being done so far which is being stored in Hiddenfields (See code for details).

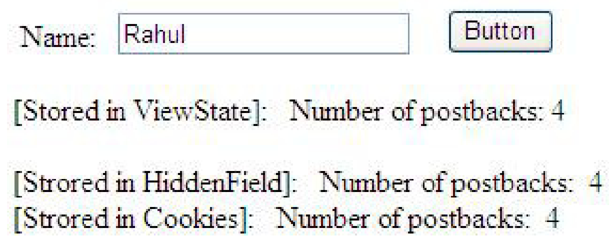
Cookies

There are scenarios when we need to store the data between page requests. So far, the techniques we have discussed store the data for the single page requests. Now we look at the techniques that store information between page requests.

Cookies are small pieces of information that can be stored in a text file on users' computer. The information can be accessed by the server and can be utilized to store information that is required between page visits and between multiple visits on the same page by the user. Let us do the same exercise of keeping track of postback by using cookies.

```
int postbacks = 0;
if (Request.Cookies["number"] != null) //Lets retrieve, increase and store again
{
    postbacks = Convert.ToInt32(Request.Cookies["number"].Value) + 1;
}
else //First postback, lets store the info
{
    postbacks = 1;
}
Response.Cookies["number"].Value = postbacks.ToString();

Label3.Text = Response.Cookies["number"].Value;
```



[Stored in ViewState]: Number of postbacks: 4

[Strored in HiddenField]: Number of postbacks: 4
[Strored in Cookies]: Number of postbacks: 4

We cannot keep track of postbacks using cookies as cookies will stay on user machine, so essentially we are looking at the number of times user POSTED back on their page so far since the beginning.

When we run the page and hit the button to do a postback, the web will show us the postbacks being done so far which is being stored in Cookies (see code for details). The cookies can have various parameters like how long they are valid and when should they expire. These parameters can be manipulated as:

```
Response.Cookies["number"].Expires = DateTime.Now.AddDays(1);
```

This cookie will expire after 1 day of its creation.

Query Strings

Query strings are commonly used to store variables that identify specific pages, such as search terms or page numbers. A query string is information that is appended to the end of a page URL. They can be used to store/pass information from one page to another to even the same page. Let us work on storing the postback information in querystrings now:

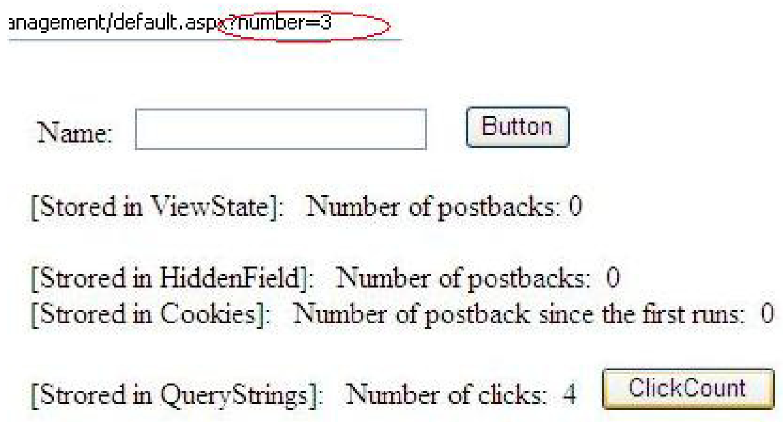
```
//GetDataItem from querystring
if (Request.QueryString["number"] != null) //Lets retrieve, increase and store again
{
    Label4.Text = Request.QueryString["number"];
}

//set in query string
int postbacks = 0;
```

```
if (Request.QueryString["number"] != null) //Lets retrieve, increase and store again
{
   postbacks = Convert.ToInt32(Request.QueryString["number"]) + 1;
}
else //First postback, lets store the info
{
   postbacks = 1;
}

Response.Redirect("default.aspx?number=" + postbacks);
```

One thing to notice here is that we can no way store the postback information in the query string we are dealing with same page. The reason is that the query string creates a new URL each time and it will be a fresh request each time we use query strings. SO we are now essentially tracking number of click here. The idea behind query string is to pass small information to OTHER pages that can be used to populate information on that page.


anagement/default.aspx?number=3

Name: [ ] Button

[Stored in ViewState]: Number of postbacks: 0

[Strored in HiddenField]: Number of postbacks: 0
[Strored in Cookies]: Number of postback since the first runs: 0

[Strored in QueryStrings]: Number of clicks: 4 ClickCount

NOTE: The use of cookies and querystring here are just for the purpose of demonstration. In real scenarios, they should never be used to store information required for same page. The Querystrings should be used to store the information between multiple page visits. Cookies should be used to store information between multiple visits to our website from the same computer.

Application State

ASP.NET allows us to save values using application state. A global storage mechanism that is accessible from all pages in the Web application. Application state is stored in the Application key/value dictionary. This information will also be available to all the users of the website. In case we need user specific information, then we better use sessionstate.

ASP.NET provides three events that enable you to initialize Application variables (free resources when the application shuts down) and respond to Application errors:

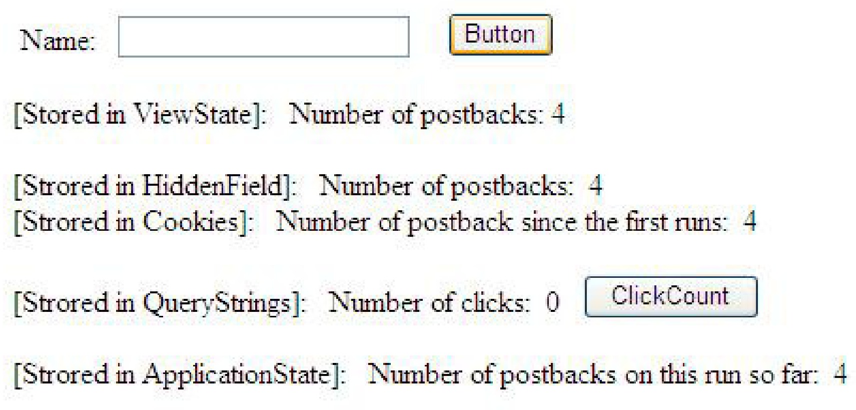
- Application_Start: Raised when the application starts. This is the perfect place to initialize Application variables.
- Application_End: Raised when an application shuts down. Use this to free application resources and perform logging.
- Application_Error: Raised when an unhandled error occurs. Use this to perform error logging.

Let us now store the information of postbacks in application state:

```
//global.asax
void Application_Start(object sender, EventArgs e)
{
   Application["number"] = 0;
}

//In web pages
Application.Lock();
Application["number"] = Convert.ToInt32(Application["number"]) + 1;
Application.UnLock();

Label5.Text = Application["number"].ToString();
```


Name:
Button

[Stored in ViewState]: Number of postbacks: 4

[Strored in HiddenField]: Number of postbacks: 4
[Strored in Cookies]: Number of postback since the first runs: 4

[Strored in QueryStrings]: Number of clicks: 0 ClickCount

[Strored in ApplicationState]: Number of postbacks on this run so far: 4

When we run the page and hit the button to do a postback, the web will show us the postbacks being done so far which is being stored in ApplicationState. We can use this object to keep track of clicks by all users on the entire website (see code for details).

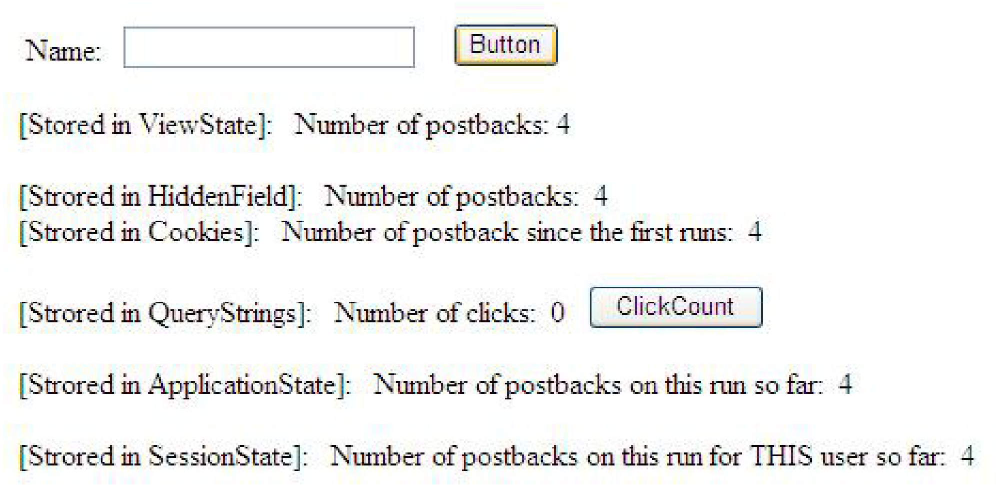
Session State

Like Application state, this information is also in a global storage that is accessible from all pages in the Web application. Session state is stored in the Sessionkey/value dictionary. This information will be available to the current user only, i.e., current session only.

```
//global.asax
void Session_Start(object sender, EventArgs e)
{
   // Code that runs when a new session is started
   Session["number"] = 0;
}

// Web forms
Session["number"] = Convert.ToInt32(Session["number"]) + 1;

Label6.Text = Session["number"].ToString();
```



```
[Stored in ViewState]:   Number of postbacks: 4

[Strored in HiddenField]:   Number of postbacks:  4
[Strored in Cookies]:   Number of postback since the first runs:  4

[Strored in QueryStrings]:   Number of clicks:  0

[Strored in ApplicationState]:   Number of postbacks on this run so far:  4

[Strored in SessionState]:   Number of postbacks on this run for THIS user so far:  4
```

When we run the page and hit the button to do a postback, the web will show us the postbacks being done so far which is being stored in SessionState. We can use this object to keep track of clicks by the current user, i.e., who owns the session for the entire website (see code for details).

Advantages of Client Side State Management

- Better scalability
- Support for multiple browser

Advantages of Server Side State Management

- Better security
- Reduced bandwidth

MOBILE APPLICATION DEVELOPMENT IN ASP.NET

Introduction

Mobile application development in ASP.NET is similar to traditional ASP.NET web application development. And it is very easy for ASP.NET developer to develop mobile application. All mobile web pages are inherit from **MobilePage** class which exists in **System.Web.UI.MobileControls** namespace.ASP.NET exposes a **System.Web.Mobile** namespace is for specifically to Web development.

Background

In this demonstration, you will create a mobile web page that dedicated to mobile device. The page will show a loan repament calculator and after passing valid parameter it will show repament amount of a pricipal amount with terms and rate.

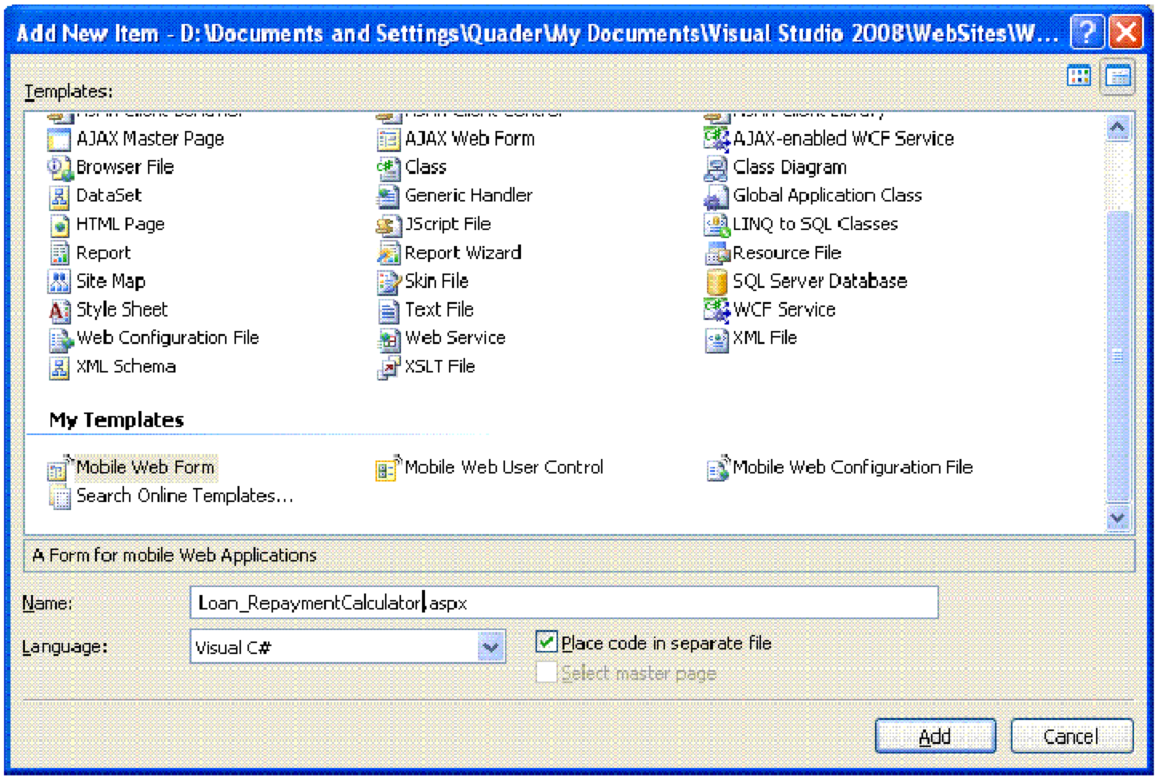
Creating Web Application in ASP.NET

1. Click to open Microsoft Visual Studio 2008
2. On the **File menu** , choose New, and then choose Web Site.
   The New Web Site dialog box appears.
3. Under Visual Studio installed templates, select ASP.NET Web Site.
4. Click **Browse** .
   The Choose Location dialog box appears.
5. Location **File System** and **LRC**
6. Language **Visual C#**
7. Click **OK** button

A Default.aspx is added in your solution and it is traditional ASP.NET page which is inherited from System.Web.UI.Page. But you need to create page which inherit from **MobilePage** class in **System.Web.UI.MobileControls** namespace. In this demonstration, you will use controls from the **System.Web.Mobile** namespace that are specifically designed for devices that cannot display as much information as a desktop browser.

Creating Mobile Web Page in Application

1. Right-click the Default.aspx page in Solution Explorer and choose **Delete**.
2. Click **OK** in the dialog box.
3. Right-click the application in Solution Explorer and choose **Add New Item**
4. Choose **Mobile Web Form** under **Visual Studio installed templates**.


Add New Item - D:\Documents and Settings\Quader\My Documents\Visual Studio 2008\WebSites\W...

Templates:

AJAX Master Page	AJAX Web Form	AJAX-enabled WCF Service
Browser File	Class	Class Diagram
DataSet	Generic Handler	Global Application Class
HTML Page	JScript File	LINQ to SQL Classes
Report	Report Wizard	Resource File
Site Map	Skin File	SQL Server Database
Style Sheet	Text File	WCF Service
Web Configuration File	Web Service	XML File
XML Schema	XSLT File	

My Templates

Mobile Web Form	Mobile Web User Control	Mobile Web Configuration File
Search Online Templates...		

A Form for mobile Web Applications

Name: Loan_RepaymentCalculator.aspx

Language: Visual C#

[✓] Place code in separate file

[ ] Select master page

Figure 1

Dowanload(Download MobileWebFromTemplate.zip - 16.12 KB) mobile page template if you do not have mobile form template in Add New Item box and place tempalate according to instruction provided in readme file. After extrating MobileWebFromTemplate.rar file you will get two folder a 'Web Application' and another is 'Web Site'. Place RAR files in Web Application folder to '[My Documents]\Visual Studio 2008\Templates\ItemTemplates\Visual C#' and RAR files in Web Site folder to '[My Documents]\Visual Studio 2008\Templates\ItemTemplates\Visual Web Developer'. Now you will get Mobile Web Form template.

5. Name **Loan_RepaymentCalculator.aspx**
6. Choose Language **Visual C#**
7. Check Place code in separate file.
8. Click **Add** in the dialog box

Right click on **Loan_RepaymentCalculator.aspx** choose View Code define namespace for **Loan_RepaymentCalculator** class.

```
namespace STL.Web.Mobile.UI
{
   public partial class Loan_RepaymentCalculator :
System.Web.UI.MobileControls.MobilePage
   {
    }
}
```

Set Inherits attribute value STL.Web.Mobile.UI.Loan_RepaymentCalculator in page directive of **Loan_RepaymentCalculator's** source file.

Collapse | Copy Code

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Loan_RepaymentCalculator.aspx.cs"
   Inherits="STL.Web.Mobile.UI.Loan_RepaymentCalculator" %>
```

Design Mobile Web Page

In solution explorer double click on **Loan_RepaymentCalculator.aspx** to view source code and you will find mobile form **form1** rename it as **frmInput**.From the **Mobile Web Forms** folder of the Toolbox, drag controls onto frmInput and set their properties as defined in the following.

1. **Label** control
   a. ID = "lblHeading"
   b. Runat = "Server"
   c. EnableViewState = "False"
   d. Wrapping = "Wrap"
   e. StyleReference="StyleHeader"

2. **Label** control
   a. ID = "lblPrincipal"
   b. Runat = "Server"
   c. EnableViewState = "False"
   d. Text = "1. Amount"
   e. StyleReference="StyleLabel"

3. **TextBox** control
   a. ID = "PrincipalAmount"
   b. Runat = "Server"
   c. Numeric = "True"
   d. MaxLength = "12"
   e. Size = "10"
   f. Title = "Principal Amount"
   g. StyleReference="StyleTextBox"

4. **RequiredFieldValidator** control for validating principal amount that expect input from user.
   a. ID = "rfvPrincipal"
   b. Runat = "Server"
   c. ControlToValidate ="PrincipalAmount"

d. ErrorMessage = "Amount Empty!"
e. StyleReference="StyleValidation"

5. RegularExpressionValidator control for validating principal amount that expect only numeric(fractional) value from user .
a. ID = "revPrincipal"
b. Runat = "Server"
c. ControlToValidate = "PrincipalAmount"
d. ErrorMessage = "Invalid Amount!"
e. ValidationExpression = "^([0-9]+)?\.?\d{1,2}"
f. StyleReference="StyleValidation"

6. Label control
a. ID = "lblTerm"
b. Runat = "Server"
c. EnableViewState = "False"
d. Text = "2. Term(Year)"
e. StyleReference="StyleLabel"

7. TextBox control
a. ID = "Term"
b. Runat = "Server"
c. Numeric = "True"
d. MaxLength = "6"
e. Size = "10"
f. Title = "Term"
g. StyleReference="StyleTextBox"

8.RequiredFieldValidator control for validating term that expect input from user.
a. ID = "rfvTerm"
b. Runat = "Server"
c. ControlToValidate ="Term"
d. ErrorMessage ="Term Empty!"
e. StyleReference="StyleValidation"

9. RegularExpressionValidator control for validating term that expect only numeric(not fractional) value from user .
a. ID = "revTerm"
b. Runat = "Server"
c. ControlToValidate = "Term"
d. ErrorMessage = "Invalid Amount!"
e. ValidationExpression = "^[1-9]([0-9]+)?"
f. StyleReference="StyleValidation"

10. Label control
a. ID = "lblRate"
b. Runat = "Server"
c. EnableViewState = "False"
d. Text = "3. Rate(%)"
e. StyleReference="StyleLabel"

11. TextBox control
a. ID = "Rate"
b. Runat = "Server"
c. Numeric = "True"
d. MaxLength = "5"
e. Size = "10"
f. Title = "Rate"
g. StyleReference="StyleTextBox"

12. RequiredFieldValidator control for validating rate that expect input from user.
a. ID = "rfvRate"
b. Runat = "Server"
c. ControlToValidate ="Rate"
d. ErrorMessage ="Rate Empty!"
e. StyleReference="StyleValidation"

13. RangeValidatorcontrol for validating rate that expect only numeric value between 1 to 100 from user.
a. ID = "rvRate"
b. Runat = "Server"
c. Type="Double"
d. ControlToValidate = "Rate"
e. ErrorMessage = "Invalid Rate!"
f. MinimumValue="0"
g. MaximumValue="100"
h. StyleReference="StyleValidation"

14. Command control
a. ID = "cmdRepayment"
b. Runat = "Server"
e. Text = "Repayment"
f. OnClick="cmdRepayment_Click"

The **Command** control provides a way to invoke ASP.NET event handlers from UI elements, thus posting user input from UI elements back to the server. The command is for calculate repayment. Event OnClick of cmdRepayment is bind with cmdPayment_Click event procedure, it will disscus later in this demonestration.

The Form mobile control enables you to break up complex pages into a collection of forms on a mobile Web page. With this ability, you can minimize the effort required to port Web-based applications to mobile devices.

ASP.NET mobile web page can contain more than one form control and mobile application displays only one form at a time. And a form control cannot be a inner element of another form control.

Add second form control into **Loan_RepaymentCalculator.aspx** page after **frmInput** from the **Mobile Web Forms** folder of the Toolbox, and define form control ID is **frmResult**

Now from the **Mobile Web Forms** folder of the Toolbox, drag controls onto **frmResult** and set their properties as defined in the following.

1. Label control
a. ID = "lblHeadingResult"
b. Runat = "Server"
c. EnableViewState = "False"
d. Wrapping = "Wrap"
e. StyleReference="StyleHeader"

2. TextView control to display result
a. ID = "tvLoanDetails"
b. Runat = "Server"
c. EnableViewState = "False"
d. StyleReference="StyleLabelResult"

3. Command control it is a navigation button to go previous form control
a. ID = "cmdBack"
b. Runat = "Server"
e. Text = "Back"
f. OnClick="cmdBack_Click"

Event OnClick of the cmdBack command button bind with cmdBack_Click event procedure will discuss later in this demonstration.

Add last form control into **Loan_RepaymentCalculator.aspx** page after **frmResult** from the **Mobile Web Forms** folder of the Toolbox, and define form control ID is **frmError**. If runtime error occurs application will show this error form.

Now from the Mobile Web Forms folder of the Toolbox, drag controls onto **frmError** and set their properties as defined in the following.

1. Label control
a. ID = "lblHeadingError"
b. Runat = "Server"
c. EnableViewState = "False"
d. Wrapping = "Wrap"
e. StyleReference="StyleHeader"

2. TextView control to display error
a. ID = "tvError"
b. Runat = "Server"
c. EnableViewState = "False"
d. Text ="Sorry For Inconvenience!"
e. StyleReference="StyleError"

3. Command control it is a navigation button to go previous form control
a. ID = "cmdHome"

b. Runat = "Server"
e. Text = "Home"
f. OnClick="cmdBack_Click"

Event OnClick of the cmdBack command button bind with cmdBack_Click event procedure will disscuss later in this demonstration.

StyleSheet

StyleSheet can be internal or external in mobile ASP.NET application. External stylesheet is for entire application while internal stylesheet only for page specific. The stylesheet control is need to implement style in application. Stylesheet control can contain any number of style elements, or elements that inherits from the style element. Each style element must have a unique **name** property. You can use the Name property to refer to each **Style** element in the **StyleSheet** control from other controls on the same **MobilePage** object.

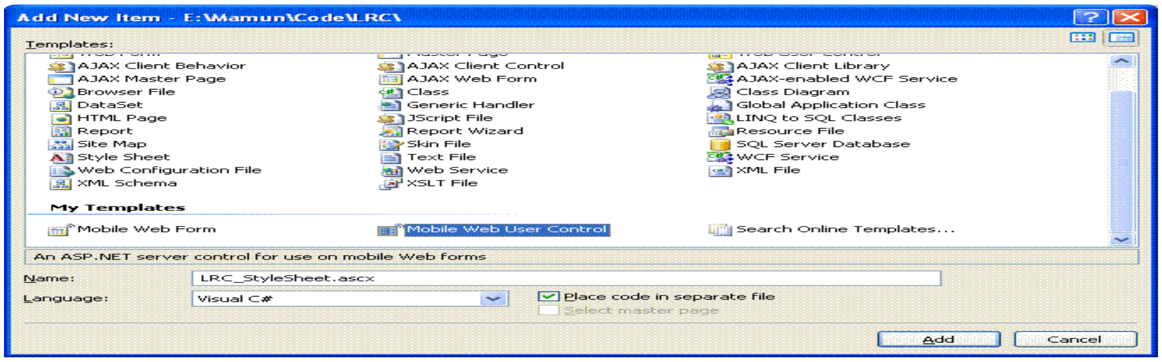
To create the external style sheet, you create a user control, in an .ascx file, and place a single style-sheet control with a set of styles in it. Then, to refer to this file, you place a style-sheet control on the page and set its ReferencePath property to the relative URL of the user control.

Now add a StyleSheet folder in LRC Application. To do this follow the below steps:
1. Right Click on LRC application
2. Choose New Folder
3. Rename it **StyleSheet**

Add Mobile Web User Control in StyleSheet folder. Follow the below steps:

1. Right Click on StyleSheet folder in LRC application
2. Choose **Add New Item**

Add New Item dialogbox appear as below,



An ASP.NET server control for use on mobile Web forms
Name: LRC_StyleSheet.ascx
Language: Visual C#
☑ Place code in separate file
☐ Select master page

Figure 2

3. Name **LRC_StyleSheet.ascx**
4. Labguage **Visual C#**
5. Click **Add** in the dialog box.

set **STL.Web.Mobile.UI** namespace for **LRC_StyleSheet** class in LRC_StyleSheet.ascx.cs file and Set
Inherit="STL.Web.Mobile.UI.LRC_StyleSheet" in control directive of **LRC_StyleSheet's** source file.

To define style sheet you need to add a StyleSheet control on the page from Toolbox under Mobile Web Forms Folder. And define styles as bellow:

```
  <mobile:StyleSheet ID="StyleSheet1"  runat="server">
      <mobile:Style Name="StyleForm" Font-Size="Small">
      </mobile:Style>
      <mobile:Style Name="StyleHeader" ForeColor="#999966" Font-Size="Small" Font-
Bold="True">
      </mobile:Style>
      <mobile:Style Name="StyleLabel" ForeColor="#cc3399" Font-Size="Small" Font-
Bold="False">
      </mobile:Style>
      <mobile:Style Name="StyleTextBox" ForeColor="#cc3399" Font-Size="Small" Font-
Bold="False">
      </mobile:Style>
      <mobile:Style Name="StyleValidation" ForeColor="Red" Font-Size="Small" Font-
Bold="False">
      </mobile:Style>
      <mobile:Style Name="StyleLabelResult" ForeColor="#cc0066" Font-Size="Small" Font-
Bold="False">
      </mobile:Style>
      <mobile:Style Name="StyleError" ForeColor="Red" Font-Size="Small">
      </mobile:Style>
  </mobile:StyleSheet>
```

To add style reference from this external StyleSheet **into Loan_RepaymentCalculator.aspx**, Just go to the source of this page and add a StyleSheet control from the Toolbox under Mobile Web Froms add set
ReferencePath="~/StyleSheet/LRC_StyleSheet.ascx"

```
   <mobile:StyleSheet ID="StyleSheet1"  runat="server"
ReferencePath="~/StyleSheet/LRC_StyleSheet.ascx">
   </mobile:StyleSheet>
```

Now you can a add StyleReference in elements of a mobile web page.

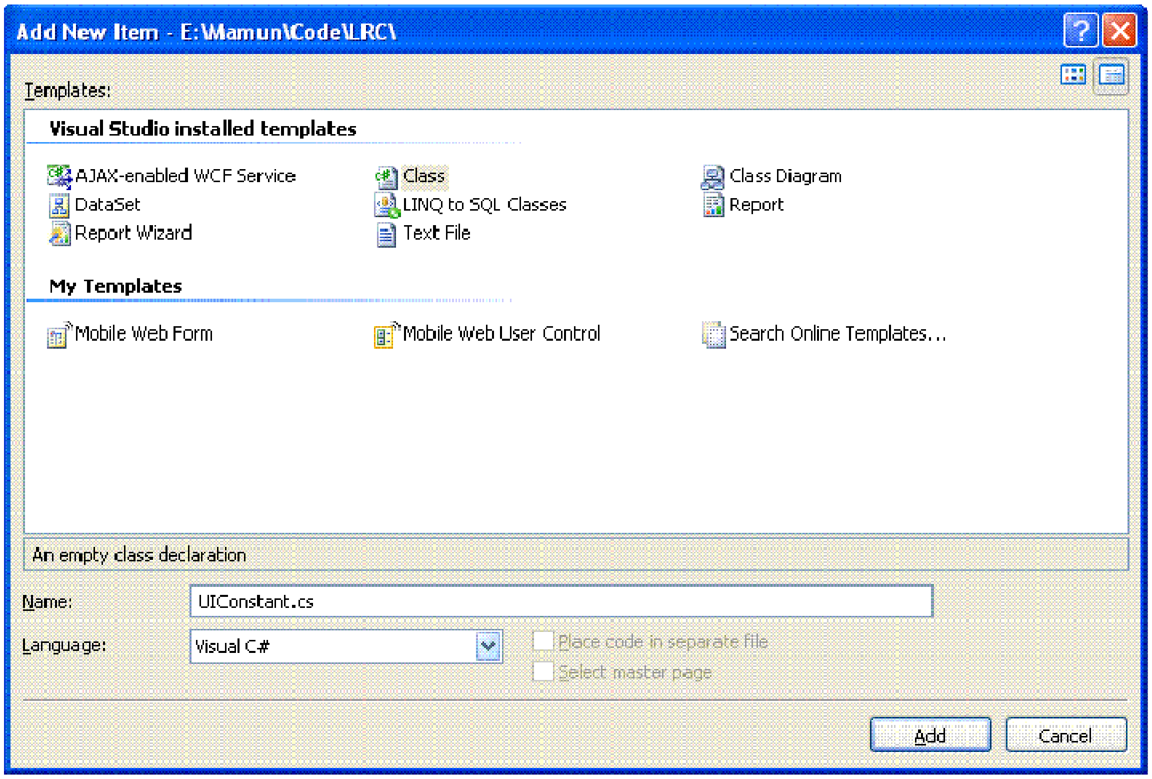
```
   <mobile:Label ID="lblHeading"  runat="server" EnableViewState="False"
StyleReference="StyleHeader" Wrapping="Wrap">
   </mobile:Label>
```

Class

Add a class under **STL.Web.Mobile.UI** namespace in LRC Application for UI constants

Steps:

1. Right Click on the App_Code folder
2. Choose **Add New Item**
3. Choose **Class**
4. Name **UIConstant.cs**
5. Click **Add** in the dialog box.



An empty class declaration

Name: UIConstant.cs

Language: Visual C#

Figure 3

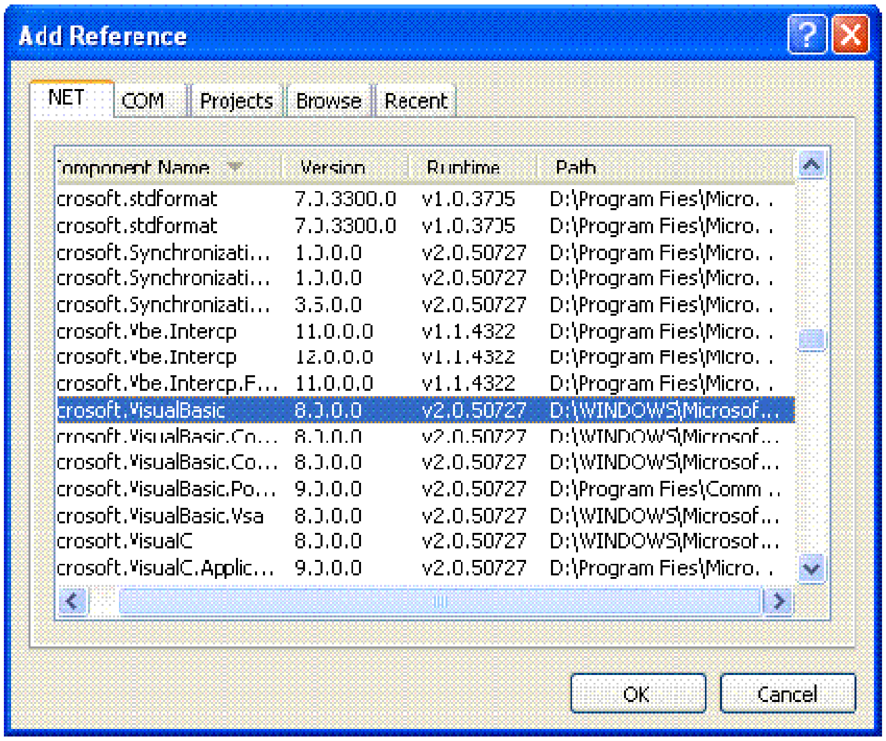
Add constants in UIConstant.cs files

```
namespace STL.Web.Mobile.UI
{
   public class UIConstant
   {
      private UIConstant()
      {
      }
      public const String TITLE_BAR="Loan Payment Calculator";
      public const String PAGE_TITLE = "Loan Payment Calculator";
   }}
```

Events

Add **Microsoft.VisualBasic.dll** reference in application to calculate monthly payment using Financial.Pmt method.

Steps:
1. Right Click on the LRC Application
2. Choose **Add Reference**
3. Choose **Microsoft.VisualBasic**
4. Click **Add** in the dialog box.



omponent Name	Version	Runtime	Path
crosoft.stdformat	7.0.3300.0	v1.0.3705	D:\Program Fies\Micro. .
crosoft.stdformat	7.0.3300.0	v1.0.3705	D:\Program Fies\Micro. .
crosoft.Synchronizati...	1.0.0.0	v2.0.50727	D:\Program Fies\Micro. .
crosoft.Synchronizati...	1.0.0.0	v2.0.50727	D:\Program Fies\Micro. .
crosoft.Synchronizati...	3.5.0.0	v2.0.50727	D:\Program Fies\Micro. .
crosoft.Vbe.Intercp	11.0.0.0	v1.1.4322	D:\Program Fies\Micro. .
crosoft.Vbe.Intercp	12.0.0.0	v1.1.4322	D:\Program Fies\Micro. .
crosoft.Vbe.Intercp.F...	11.0.0.0	v1.1.4322	D:\Program Fies\Micro. .
crosoft.VisualBasic	8.0.0.0	v2.0.50727	D:\WINDOWS\Microsof...
crosoft.VisualBasic.Co...	8.0.0.0	v2.0.50727	D:\WINDOWS\Microsof...
crosoft.VisualBasic.Co...	8.0.0.0	v2.0.50727	D:\WINDOWS\Microsof...
crosoft.VisualBasic.Po...	9.0.0.0	v2.0.50727	D:\Program Fies\Comm ..
crosoft.VisualBasic.Vsa	8.0.0.0	v2.0.50727	D:\WINDOWS\Microsof...
crosoft.VisualC	8.0.0.0	v2.0.50727	D:\WINDOWS\Microsof...
crosoft.VisualC.Applic...	9.0.0.0	v2.0.50727	D:\Program Fies\Micro. .

Figure 4

Using **Microsoft.VisualBasic** namespace in **Loan_RepaymentCalculator.aspx.cs** file

```
using Microsoft.VisualBasic;
```

OnClick event of cmdRepayment command in **frmInput** form is as bellow

```
        protected void cmdRepayment_Click(object sender, EventArgs e)
    {
      if (!Page.IsValid) return;
      try
      {
        Double dblPrincipal = double.Parse(this.PrincipalAmount.Text);
        Double dblApr = double.Parse(this.Rate.Text);
```

```
            Double dblMonthlyInterest = (Double)(dblApr / (12 * 100));
            Int64 intTermInMonths = Int64.Parse(this.Term.Text) * 12;
            Double dblMonthlyPayment;
                      //Calculate monthly payment
            dblMonthlyPayment = Microsoft.VisualBasic.Financial.Pmt(dblMonthlyInterest,
intTermInMonths, -dblPrincipal,                    0,
Microsoft.VisualBasic.DueDate.BegOfPeriod);
            this.ActiveForm = this.frmResult;
            StringBuilder sbDetailsSpec = new StringBuilder("");
            sbDetailsSpec.Append(String.Format("{0} @ {1}% for {2} years
 Payment: ", dblPrincipal.ToString                             ("C0"), dblApr.ToString(),
this.Term.Text));
            sbDetailsSpec.Append("" + dblMonthlyPayment.ToString("C") + "");
            this.tvLoanDetails.Text = sbDetailsSpec.ToString();
         }
         catch
         {
                      //If runtime error occurs then go to error form.
            this.ActiveForm = frmError;
         }
            }
```

OnClick event of cmdBack command in **frmInput** form is as bellow

```
            protected void cmdBack_Click(object sender, EventArgs e)
      {
               //To back to input form
         this.ActiveForm = this.frmInput;
      }
```

Initialize user method to initialize elements in the mobile web page

```
            private void Initialize()
      {
         this.frmInput.Title = UIConstant.TITLE_BAR;
         this.frmResult.Title = UIConstant.TITLE_BAR;
         this.frmError.Title = UIConstant.TITLE_BAR;

         this.lblHeading.Text = UIConstant.PAGE_TITLE;
         this.lblHeadingResult.Text = UIConstant.PAGE_TITLE;
         this.lblHeadingError.Text = UIConstant.PAGE_TITLE;
      }
```

Load event of the page

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

Application Level Errors

To handle application level error you need to add a error page. To add page

Steps:
1. Right click on the LRC application
2. Choose **Add New Item**
3. Name **ErrorPage.aspx**
4. Click **Add** in the dialog

Set **Inherits="STL.Web.Mobile.UI.ErrorPage"** in page directive of ErrorPage.aspx. And define STL.Web.Mobile.UI namespace for
ErrorPage

```
namespace STL.Web.Mobile.UI
{
   public partial class ErrorPage : System.Web.UI.MobileControls.MobilePage
   {
   }
}
```

Add a **StyleSheet** control in the page and set
ReferencePath="~/StyleSheet/LRC_StyleSheet.ascx"

Add a **form** control in the ErrorPage and set ID="frmError"

Add control in **frmError** that is defined as following:

1. Label control
- a. ID = "lblHeadingError"
- b. Runat = "Server"
- c. EnableViewState = "False"
- d. Wrapping = "Wrap"
- e. StyleReference="StyleHeader"

2. TextView control to display error
- a. ID = "tvError"
- b. Runat = "Server"
- c. EnableViewState = "False"
- d. Text ="Sorry For Inconvenience!"
- e. StyleReference="StyleError"

3. Command control it is a navigation button to go previous form control
a. ID = "cmdHome"
b. Runat = "Server"
e. Text = "Home"
f. OnClick="cmdBack_Click"

Code of the ErrorPage file is as follows:

```
public partial class ErrorPage : System.Web.UI.MobileControls.MobilePage
   {
       #region Event

       protected void Page_Load(object sender, EventArgs e)
       {
           Intitalize();
       }

       protected void cmdHome_Click(object sender, EventArgs e)
       {
                           //To redirect to Loan_RepaymentCalculator page
           Response.Redirect("~/Loan_RepaymentCalculator.aspx");
       }

       #endregion Event

       #region Method

       private void Intitalize()
       {
           this.frmError.Title = UIConstant.TITLE_BAR;
           this.lblHeadingError.Text = UIConstant.PAGE_TITLE;
       }

       #endregion Method
   }
```

Web.config

You nedd to change configuration in Web.config file to redirect to Error Page when application level error is occured.
set **mode="on"** and **defaultRedirect="~/ErrorPage.aspx"** in customErros element under System.Web element.

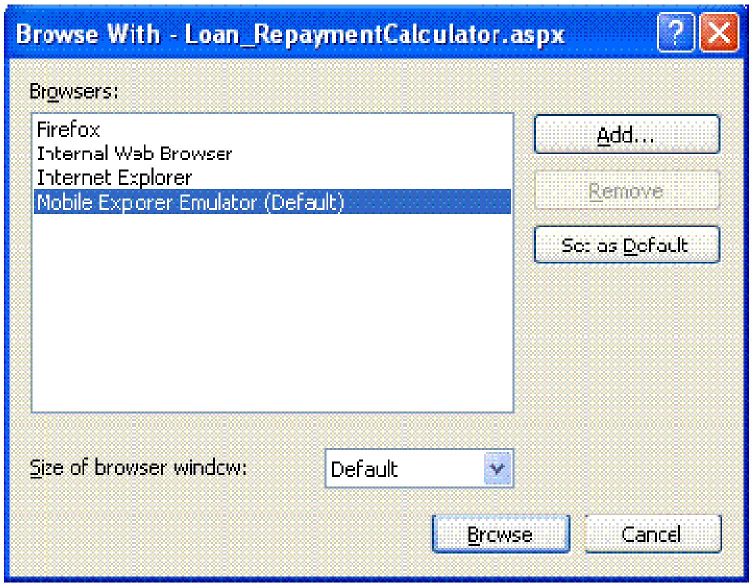
Test Application

To test the application you can use **Microsoft Mobile Explorer 3.0** . If not avilable Microsoft Mobile Explorer 3.0 you can use your desktop browser or free download it from net. Install **Microsoft Mobile Explorer 3.0** in your system.

To browse with **Microsoft Mobile Explorer 3.0** you need to do as follows:

1. Right click on **Loan_RepaymentCalculator.aspx** file
2. Choose **Browse With**

   (If Microsoft Mobile Explorer 3.0 is not avilable in Browsers list of Browse With dialog, you need add it)
3. Click **Add**
4. Browse your location where you installed **Microsoft Mobile Explorer 3.0 (mmeemu.exe)**
5. Select **Microsoft Mobile Explorer**
6. Click **Set as Default** in the dialog box.

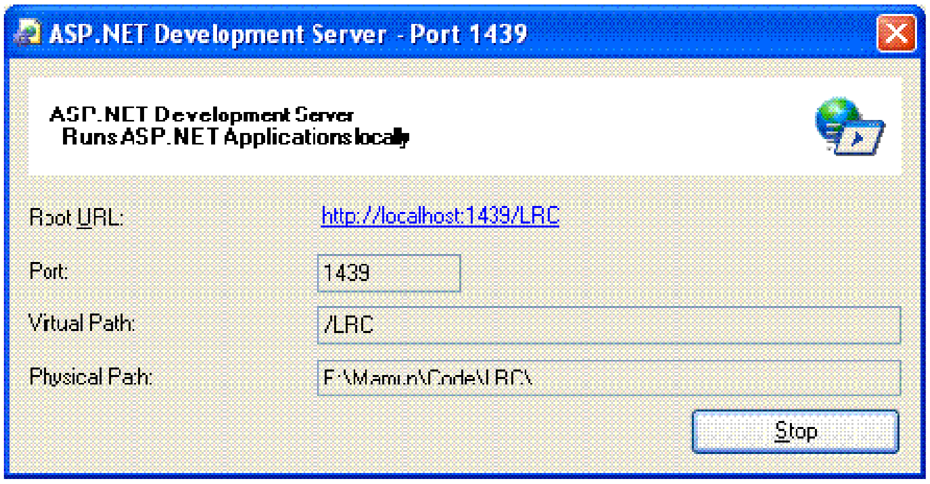


Firefox
Internal Web Browser
Internet Explorer
Mobile Exporer Emulator (Default)

Size of browser window: Default

Figure 5

Press F5 to run the application. Microsoft Mobile Explorer Emulator will appear. Click ASP.NET Development Server icon in the system tray to get application URL name and its port. It may be different in your system.


ASP.NET Development Server - Port 1439

ASP.NET Development Server
Runs ASP.NET Applications locally

Root URL: http://localhost:1439/LRC

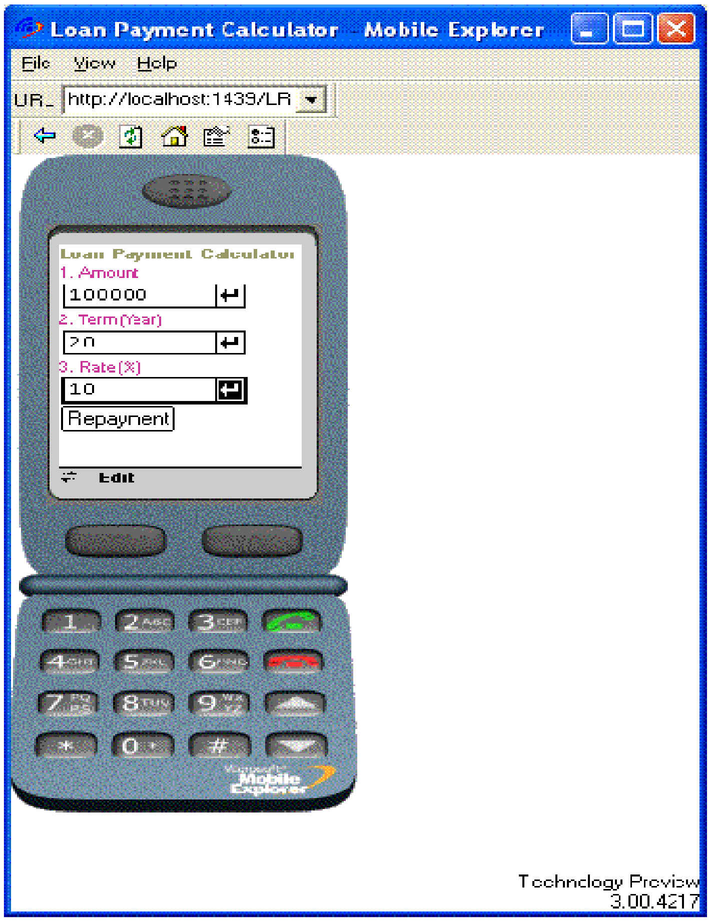
Port: 1439

Virtual Path: /LRC

Physical Path: F:\Mamun\Code\LRC\

Figure 6

In the Microsoft Mobile Explorer Emulator type URL as
http://localhost:1439/LRC/Loan_RepaymentCalculator.aspx

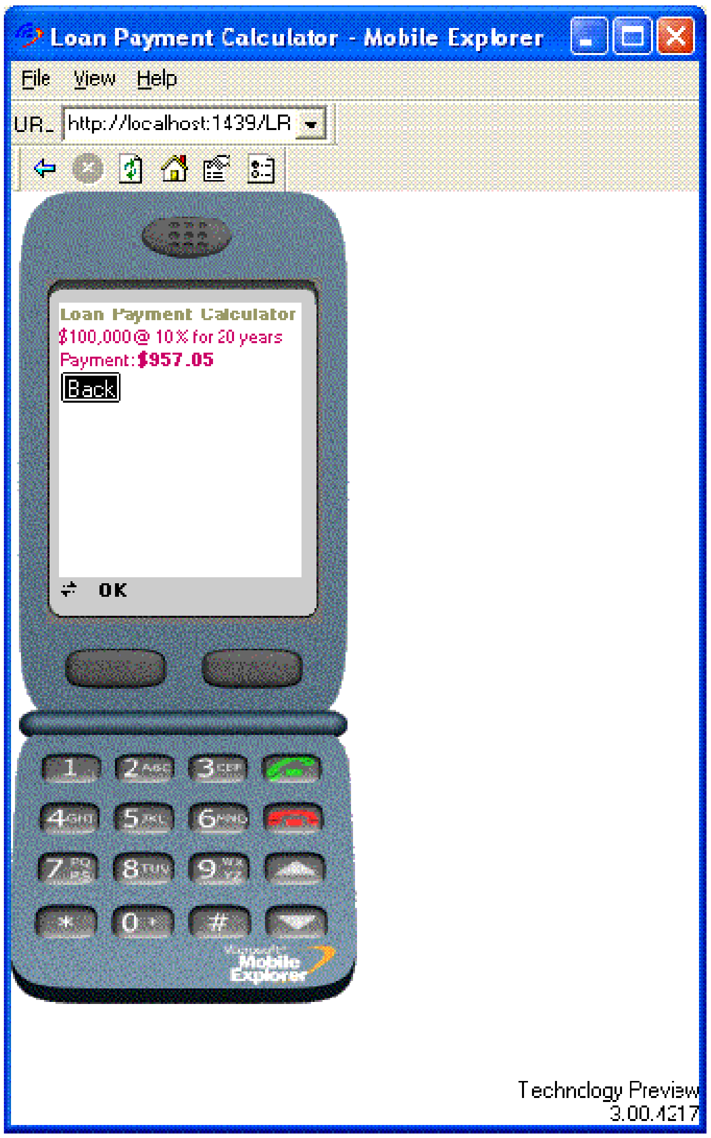

Loan Payment Calculator - Mobile Explorer

URL http://localhost:1439/LR

Loan Payment Calculator
1. Amount
100000
2. Term(Year)
20
3. Rate(%)
10

Figure 7

Enter **Amount, Term & Rate**. Click on Repayment button in the screen. You will get result like bellow,


Loan Payment Calculator - Mobile Explorer
File View Help
URL http://localhost:1439/LR
Back
OK
Technology Preview
3.00.1217

Loan Payment Calculator
$100,000@ 10% for 20 years
Payment:$957.05

Figure 8

Tools

For testing application in Mobile Emulator
http://devhood.com/tools/tool_details.aspx?tool_id=52