MemberShip Demo

1. Create a new empty web project. Call it “MembershipDemo”.
   1. Note this creates a Web Application.
   2. Many things in this demo are actually easier if you create a Web Site instead.
   3. WebSites handle some of the things we do here automatically.
2. If you want to use an existing database follow the directions at <http://www.asp.net/web-forms/tutorials/security/membership/creating-the-membership-schema-in-sql-server-vb> or <http://www.4guysfromrolla.com/articles/040506-1.aspx>
   1. Go to directory: C:\Windows\Microsoft.NET\Framework\v2.0.50727
   2. Run aspnet\_regsql.exe
   3. Use (LocalDB)\V11.0 for the server and the full path for your database in it’s App\_Data folder.
   4. Data Source=(LocalDB)\v11.0;AttachDbFilename=C:\Users\rgarn\_000\Desktop\WebApplication1\WebApplication1\App\_Data\Database1.mdf;Integrated Security=True
3. Open the web.config file and add an authentication element:

<configuration>

<system.web>

<compilation debug="true" targetFramework="4.5" />

<httpRuntime targetFramework="4.5" />

<authentication mode="Forms" />

</system.web>

</configuration>

# Add a Forms Element

1. Add a forms element to authentication element:

<authentication mode="Forms" >

<forms name=".ASPXAUTH"

loginUrl="Login.aspx"

protection="All"

timeout="30"

path="/"

requireSSL="false"

slidingExpiration="true"

cookieless="UseDeviceProfile" />

</authentication>

1. Add an appSettings element to web.config:

<appSettings>

<add key="ValidationSettings:UnobtrusiveValidationMode" value="None"/>

</appSettings>

# Using the CreateUserWizard Server Control

1. Create a new webForm. Call it “NewAccount”
2. Add a CreateUserWizardControl:

<asp:CreateUserWizard ID="CreateUserWizard1" Runat="server"

BorderWidth="1px" BorderColor="#FFDEAD" BorderStyle="Solid" BackColor="#FFFBD6" Font-Names="Verdana">

<TitleTextStyle Font-Bold="True" BackColor="#990000" ForeColor="White">

</TitleTextStyle>

</asp:CreateUserWizard>

1. View page with Ctrl-F5.
   1. Create a user
2. From visual studio open your database.
   1. If you can’t see the database in the solution you may need to click on the “ShowAllFiles” icon at the top of the solution viewer.
   2. UPGRADING TO LOCALDB If SQL Server is not installed on your system, or the version does not match the version of the ﬁle, Server Explorer may refuse to open the aforementioned database ﬁle, stating that it isn't compatible with the current SQL Server instance.
   3. In that case, the Microsoft knowledge base article at <http://msdn.microsoft.com/en-us/library/hh873188(v=vs.110).aspx> provides instructions to upgrade the database ﬁle to either a more recent version of SQL Server Express or to the new LocalDB format.
   4. Once the database appears in the server explorer, view data of the user table.

# Working with the CreateUserWizard Control

1. Add a webForm to the page. Call it “Default.aspx”
2. On the new page add some text “Default Page” so the page has something to show.
3. Back on the new account page add the following event handler:

protected void CreateUserWizard1\_ContinueButtonClick(object sender, EventArgs e)

{

Response.Redirect("Default.aspx");

}

1. View Page with Ctrl-F5

# Incorporating Personalization Properties

1. Create a class that inherits from ProfileBase to represent our profile:

public class UserProfile : ProfileBase

{

public static UserProfile GetUserProfile(string username)

{

return Create(username) as UserProfile;

}

public static UserProfile GetUserProfile()

{

return Create(Membership.GetUser().UserName) as UserProfile;

}

[SettingsAllowAnonymous(false)]

public string FirstName

{

get { return base["FirstName"] as string; }

set { base["FirstName"] = value; }

}

[SettingsAllowAnonymous(false)]

public string LastName

{

get { return base["LastName"] as string; }

set { base["LastName"] = value; }

}

[SettingsAllowAnonymous(false)]

public string LastVisited

{

get { return base["LastVisited"] as string; }

set { base["LastVisited"] = value; }

}

[SettingsAllowAnonymous(false)]

public string Age

{

get { return base["Age"] as string; }

set { base["Age"] = value; }

}

[SettingsAllowAnonymous(false)]

public string Member

{

get { return base["Member"] as string; }

set { base["Member"] = value; }

}

}

1. Note that this class employs SettingsAllowAnonymous attributes (see <http://msdn.microsoft.com/en-us/library/aa287992(v=vs.71).aspx>)
2. Add the following to web.config under the <system.web> tag

<profile defaultProvider="AspNetSqlProfileProvider" inherits="MembershipDemo.UserProfile">

</profile>

1. Change the CreateUserControl as follows:

<asp:CreateUserWizard ID="CreateUserWizard1" Runat="server" BorderWidth="1px" BorderColor="#FFDFAD" BorderStyle="Solid"

BackColor="#FFFBD6" Font-Names="Verdana" LoginCreatedUser="true" OnCreatedUser="CreateUserWizard1\_CreatedUser" OnContinueButtonClick="CreateUserWizard1\_ContinueButtonClick" >

<WizardSteps>

<asp:WizardStep ID="WizardStep1" Runat="server" Title="Additional Information" StepType="Start">

<table width="100%">

<tr>

<td> Firstname: </td>

<td> <asp:TextBox ID="Firstname" Runat="server"></asp:TextBox> </td>

</tr>

<tr>

<td> Lastname: </td>

<td> <asp:TextBox ID="Lastname" Runat="server"></asp:TextBox> </td>

</tr>

<tr>

<td> Age: </td>

<td> <asp:TextBox ID="Age" Runat="server"></asp:TextBox> </td>

</tr>

</table>

</asp:WizardStep>

<asp:CreateUserWizardStep ID="CreateUserWizardStep1" Runat="server" Title="Sign Up for Your New Account">

</asp:CreateUserWizardStep>

<asp:CompleteWizardStep Runat="server" Title="Complete"> </asp:CompleteWizardStep>

</WizardSteps>

<StepStyle BorderColor="#FFDFAD" Font-Names="Verdana" BackColor="#FFFBD6" BorderStyle="Solid" BorderWidth="1px"></StepStyle>

<TitleTextStyle Font-Bold="True" BackColor="#990000" ForeColor="White"></TitleTextStyle>

</asp:CreateUserWizard>

1. This adds wizard steps to collect user profile information. It also adds an OnCreatedUser="CreateUserWizard1\_CreatedUser" attribute that calls an OnCreatedUser event:
2. Add the event handler in the code behind page:

protected void CreateUserWizard1\_CreatedUser(object sender, EventArgs e)

{

UserProfile pc = new UserProfile();

pc.Initialize(CreateUserWizard1.UserName.ToString(), true);

pc.FirstName = Firstname.Text;

pc.LastName = Lastname.Text;

pc.Age = Age.Text;

pc.Save();

}

1. View the page with Ctrl-F5.
   1. Create a new user.
   2. Fill in the profile data.
2. View aspnet\_Profile table in the database.
   1. Note how the profile data is stored.

# Incorporating Personalization Properties(WebSITE Approach)

1. This is provided for reference and in case your project is a WebSite instead of a a WebApplication:
2. If you are using a WebSite instead of a WebApplication you do not need to create a class to store profile data.
3. To create a WebSite you click on File|New|WebSite instead of File|New|Project when starting visual studio.
4. Instead of defining the default provider and inherited class in the web.config file, you define profile properties:

<profile>

<properties>

<add name="FirstName" />

<add name="LastName" />

<add name="LastVisited" />

<add name="Age" />

<add name="Member" />

</properties>

</profile>

1. ASP will then automatically create all the classes and settings needed.

# Customize Create User Step link

1. From the design view click on CreateUserWizardControl smart tag and “Customize User Steps”
2. Set step to “Sign up for your new accoung”
3. Delete the Email and Security Question Lines from the Control.
4. View Page with Ctrl-F5.
   1. Notice you get an error
5. View Properties. Set RequireEmail to False.
6. In web.config set requiresUniqueEmail attribute to false.
7. Set step to “Additional Information”
8. View Page with Ctrl-F5

# Adding Users Programmatically

1. Create a new webForm. Call it “NewAccountProgramatic”.
2. Add the following to the webForm:

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

<title>Creating a User</title>

</head>

<body>

<form id="form1" runat ="server">

<h1>Create User</h1>

<p>Username<br />

<asp:TextBox ID="TextBox1" Runat="server"></asp:TextBox>

</p>

<p>Password<br />

<asp:TextBox ID="TextBox2" Runat="server" TextMode="Password">

</asp:TextBox>

</p>

<p>

<asp:Button ID="Button1" Runat="server" Text="Create User" onClick="Button1\_Click" />

</p>

<p>

<asp:Label ID="Label1" Runat="server"></asp:Label>

</p>

</form>

</body>

</html>

1. Add the following button event handler:

protected void Button1\_Click(object sender, EventArgs e)

{

try

{

Membership.CreateUser(TextBox1.Text, TextBox2.Text);

Label1.Text = "Successfully created user " + TextBox1.Text;

}

catch (MembershipCreateUserException ex)

{

switch (ex.StatusCode)

{

case MembershipCreateStatus.DuplicateEmail:

Label1.Text = "You have supplied a duplicate email address.";

break;

case MembershipCreateStatus.DuplicateUserName:

Label1.Text = "You have supplied a duplicate username.";

break;

case MembershipCreateStatus.InvalidEmail:

Label1.Text = "You have not supplied a proper email address.";

break;

default:

Label1.Text = "ERROR: " + ex.Message.ToString();

break;

}

}

}

1. View page with Ctrl-F5.
   1. Create a new user.
   2. Create a new user but enter too short a password.
   3. Create a new user but use a previous userid.
   4. Note how error handler reports problem.

# Turning Off Access with the <authorization> Element

1. Add an authorization element to the web.config file:

<authorization> <deny users="?" /> </authorization>

1. Open default.aspx and view with Ctrl-F5.
   1. Notice you get an error since you do not have a login.aspx page yet.
2. Add a Login Page
3. Create a new webForm. Call it “Login.aspx”.
4. Add a login control to the form:

<asp:Login ID="Login1" Runat="Server">

</asp:Login>

1. View page with Ctrl-F5
2. View default.aspx with Ctrl-F5
   1. Notice that it brings you to the login page.

# Logging In Users Programmatically

1. Rename the existing login page to “LoginOld.aspx”
2. Add a new webForm. Call it “Login.aspx”
3. Add two text boxes, a button and a label.

<p>

UserID: <asp:TextBox ID="TextBoxUserID" runat="server"></asp:TextBox>

</p>

<p>

Password: <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>

</p>

<p>

<asp:Button ID="ButtonSubmit" runat="server" Text="Submit" />

</p>

<p>

<asp:Label ID="LabelResults" runat="server" Text=""></asp:Label>

</p>

* 1. Note the password text box uses the TextMode attribute “Password”

1. Add a button event handler:

protected void ButtonSubmit\_Click(object sender, EventArgs e)

{

if (Membership.ValidateUser(TextBoxUserID.Text, TextBoxPassword.Text))

{

FormsAuthentication.RedirectFromLoginPage(TextBoxUserID.Text, false);

}

else

{

LabelResults.Text = "You are not registered with the site.";

}

}

1. View page with Ctrl-F5.

# Locking Out Users Who Provide Bad Passwords

1. Add the following to web.config under meberships/providers/add:

maxInvalidPasswordAttempts ="3"

passwordAttemptWindow ="15"

1. Change the button event handler to:

if (CheckBoxUnlock.Checked == true)

{

MembershipUser user = Membership.GetUser(TextBoxUserID.Text);

user.UnlockUser();

}

if (Membership.ValidateUser(TextBoxUserID.Text, TextBoxPassword.Text))

{

LabelResults.Text = "You are logged on!";

}

else

{

MembershipUser user = Membership.GetUser(TextBoxUserID.Text);

LabelResults.Text = "Locked out value: " + user.IsLockedOut.ToString();

}

1. View the page with Ctrl-F5
   1. Try logging in multiple times incorrectly.
   2. Click the checkbox to unlock the account.

# The LoginStatus Server Control

1. Remove the <deny> element from the web.config ﬁle
2. Add a LoginStatus control to default.aspx:

<body>

<form id="form1" runat="server">

<div>

<asp:LoginStatus ID="LoginStatus1" Runat="server" />

</div>

</form>

</body>

1. View page with Ctrl-F5

# The LoginName Server Control

1. Make the old login page the current one
   1. Change the programmatic login page name to “LoginProg.aspx”
   2. Change the old login page name back to “Login.aspx”
2. Add a LoginName control to default.aspx:

<p>

<asp:LoginStatus ID="LoginStatus1" Runat="server" />

</p>

<p>

<asp:LoginName ID="LoginName1" Runat="server" Font-Bo1d="True" Font-Size="XX-Large" />

</p>

1. View page with Ctrl-F5

# Showing the Number of Users Online

1. Add some text and a label to the default.aspx page to show number of users online:

<p>

There are <asp:Label ID="LabelUsers" runat="server" Text="TBD"></asp:Label> users online!

</p>

1. Add the following code to the page-load event:

protected void Page\_Load(object sender, EventArgs e)

{

LabelUsers.Text = Membership.GetNumberOfUsersOnline().ToString();

}

1. View page with Ctrl-F5

# The ChangePassword Server Control

1. Create a new webForm. Call it “ChangePassword.aspx”
2. Add a login status and a ChangePassword control:

<p>

<asp:LoginStatus ID="LoginStatus1" Runat="server" />

</p>

<p>

<asp:ChangePassword ID="ChangePassword1" Runat="server" SuccessPageUrl="~/Default.aspx"> </asp:ChangePassword>

</p>

1. View page with Ctrl-F5

# The PasswordRecovery Server Control

1. Configure web.config for email. Add a system.net element :

<system.net>

<mailSettings>

<smtp from="demo@example.com">

<network host="localhost" port="25" defaultCredentials="true" />

</smtp>

</mailSettings>

</system.net>

* 1. Note: these settings will likely not work but will be used for demo purposes.

# LoginView Demo

1. Create a new WebForm. Call it “LoginViewDemo.aspx”.
2. Add a LoginStatus and LoginView control:

<asp:LoginStatus ID="LoginStatus1" Runat="server" />

<p>

<asp:LoginView ID="LoginView1" Runat="server">

<LoggedinTemplate>

Here is some REALLY important information that you should know

about all those people that are not authenticated!

</LoggedinTemplate>

<AnonymousTemplate>

Here is some basic information for you.

</AnonymousTemplate>

</asp:LoginView>

</p>

1. View page with Ctrl-F5.

# Add a Role Group

1. Add a role group:

<asp:LoginView ID="LoginView1" Runat="server">

<LoggedinTemplate>

Here is some REALLY important information that you should know

about all those people that are not authenticated!

</LoggedinTemplate>

<AnonymousTemplate>

Here is some basic information for you.

</AnonymousTemplate>

<RoleGroups>

<asp:RoleGroup Roles="Admins">

<ContentTemplate> You are an Admin! </ContentTemplate>

</asp:RoleGroup>

<asp:RoleGroup Roles="CoolPeop1e">

<ContentTemplate> You are cool! </ContentTemplate>

</asp:RoleGroup>

</RoleGroups>

</asp:LoginView>

1. If you View page with Ctrl-F5 it will work but you can’t see roles yet because you still need to define them for the site.

# Making Changes to the web.conﬁg File

1. Add the following location element to the web.config file under the configuration tag:

<location path="AdminPage.aspx">

<system.web>

<authorization>

<allow roles="AdminPageRights" />

<deny users="\*" />

</authorization>

</system.web>

</location>

1. Add the following element to web.config under root system.web :

<roleManager enabled="true"/>

1. Add a new webForm. Call it “RoleManagement.aspx”.

<h1>Role Manager</h1>

Add Role:<br />

<asp:TextBox ID="TextBox1" Runat="Server"></asp:TextBox>

<p>

<asp:Button ID="Button1" Runat="server" Text="Add Role to Application" onClick="Button1\_Click" />

</p>

Roles Defined:<br />

<asp:ListBox ID="ListBox1" Runat="Server"> </asp:ListBox>

1. Add the following methods to the code behind page:

protected void Page\_Load(object sender, EventArgs e)

{

if (!Page.IsPostBack)

{

ListBoxDataBind();

}

}

protected void Button1\_Click(object sender, EventArgs e)

{

Roles.CreateRole(TextBox1.Text);

ListBoxDataBind();

}

protected void ListBoxDataBind()

{

ListBox1.DataSource = Roles.GetAllRoles();

ListBox1.DataBind();

}

1. View page with Ctrl-F5
   1. Add a few roles including:
   2. Admins
   3. CoolPeople

# Delete Roles

1. Add button to form:

<p>

<asp:Button ID="ButtonDelete" runat="server" Text="Delete" onClick="ButtonDelete\_Click"/>

</p>

1. Add event handler:

protected void ButtonDelete\_Click(object sender, EventArgs e)

{

foreach (ListItem li in ListBox1.Items)

{

if (li.Selected == true)

{

Roles.DeleteRole(li.ToString());

}

}

ListBoxDataBind();

}

1. Add the following attribute to the listBox:

SelectionMode="Multiple"

1. View Page with Ctrl-F5
   1. Add and remove some roles.

# Add Users to Roles

1. Create a new webForm. Call it “UsersByRole.aspx”.
2. Add the following controls:

Roles:

<asp:DropDownList ID="DropDownListRoles" Runat="server" AutoPostBack="True" OnSelectedIndexChanged="DropDownListRoles\_SelectedIndexChanged"> </asp:DropDownList>

<br /><br />

<asp:ListBox ID="ListBoxUsersInRole" Runat="Server" SelectionMode="Multiple" > </asp:ListBox>

<p>

<asp:DropDownList ID="DropDownListUsers" runat="server"></asp:DropDownList>

</p>

<p>

<asp:Button ID="ButtonAddUser" runat="server" Text="Add User To Role" OnClick="ButtonAddUser\_Click" />

</p>

1. Add the following code behind:

protected void Page\_Load(object sender, EventArgs e)

{

if (!Page.IsPostBack)

{

DropDownListRoles.DataSource = Roles.GetAllRoles();

DropDownListRoles.DataBind();

DropDownListUsers.DataSource = Membership.GetAllUsers();

DropDownListUsers.DataBind();

ListBoxUsersInRole.DataSource = Roles.GetUsersInRole(DropDownListRoles.SelectedValue);

ListBoxUsersInRole.DataBind();

}

}

protected void ButtonAddUser\_Click(object sender, EventArgs e)

{

Roles.AddUserToRole(DropDownListUsers.SelectedValue, DropDownListRoles.SelectedValue);

ListBoxUsersInRole.DataSource = Roles.GetUsersInRole(DropDownListRoles.SelectedValue);

ListBoxUsersInRole.DataBind();

}

protected void DropDownListRoles\_SelectedIndexChanged(object sender, EventArgs e)

{

ListBoxUsersInRole.DataSource = Roles.GetUsersInRole(DropDownListRoles.SelectedValue);

ListBoxUsersInRole.DataBind();

}

1. View page with Ctrl-F5.
   1. Add some users to roles.

# Remove Users From Role

1. Add button:

<p>

<asp:Button ID="ButtonRemove" runat="server" Text="Remove User(s) From Role" OnClick="ButtonRemove\_Click" />

</p>

1. Add event handler:

protected void ButtonRemove\_Click(object sender, EventArgs e)

{

foreach (ListItem li in ListBoxUsersInRole.Items)

{

if (li.Selected == true)

{

Roles.RemoveUserFromRole(li.ToString(),DropDownListRoles.SelectedValue);

}

}

ListBoxUsersInRole.DataSource = Roles.GetUsersInRole(DropDownListRoles.SelectedValue);

ListBoxUsersInRole.DataBind();

}

1. View page with Ctrl-F5
   1. Add a user into the admin role and cool person role
2. View LoginViewDemo.aspx
   1. Login to users you assigned into roles

# USING THE WEB SITE ADMINISTRATION TOOL

1. Open the web site administration tool from Visual Studio: Project|Web Site Administration Tool
2. Use the tool to add some roles and users.
3. Use tools we created above to see changes.