ASP.NET Membership Step by Step

By Md. Mahedee Hasan, Software Architect, Leadsoft Bangladesh Limited, Microsoft MVP, Visual Studio and Development Technologies.

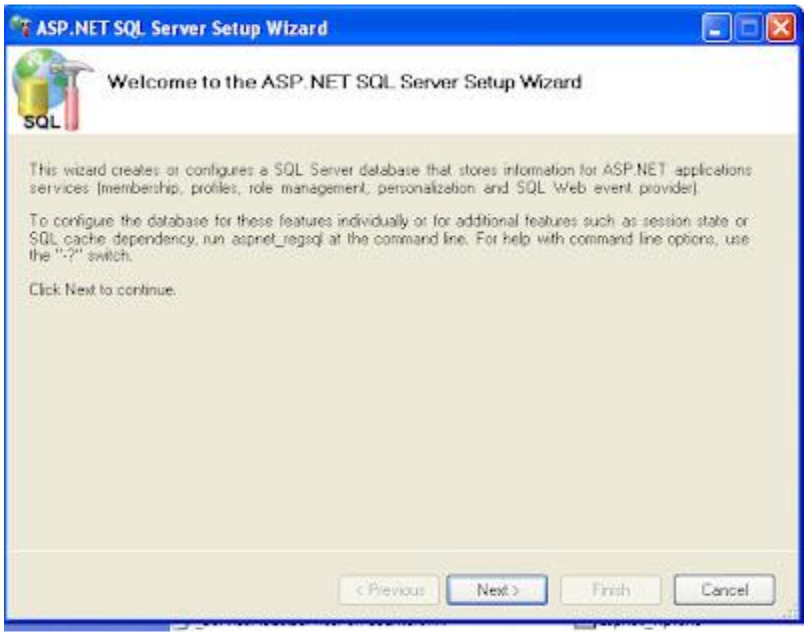
ASP.NET provides build in control to validate and store user credential. So, membership helps to manage user authentication and authorization in website. The membership provider is the glue between Login controls and membership database.

Step 1: Implementing the Membership and Role Provider
Suppose we are going to create a website by Visual Studio 2010. If you create a website by visual studio 2010, you will get login and Registration pages by default. But we want to create it manually so that we can understand its functionality.

So, what should we do? Don't worry. Let's create an empty website. To create an empty website, go to **Add → New Website → ASP.NET Empty Website**.
In the empty website, we will get only an web.config file.

To store user information, we need to create required tables, sp and other information into the database in the SQL Server. To create membership tables and sps follow the following steps.

a. Go to directory C:\WINDOWS\Microsoft.NET\Framework\v4.0.30319
b. Click aspnet_regsql.exe. You will see the following screen.

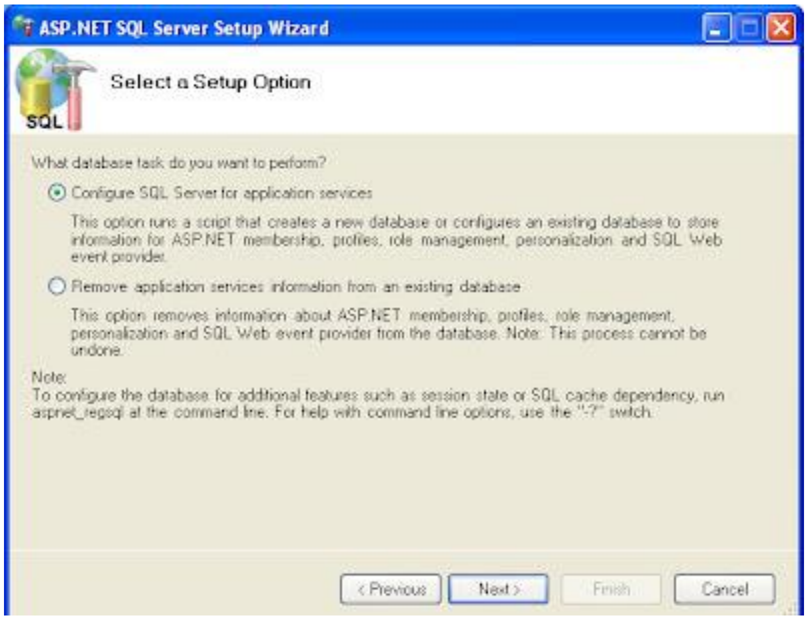

ASP.NET SQL Server Setup Wizard
Welcome to the ASP.NET SQL Server Setup Wizard
SQL
< Previous
Next >
Finish
Cancel

This wizard creates or configures a SQL Server database that stores information for ASP.NET applications services (membership, profiles, role management, personalization and SQL Web event provider).

To configure the database for these features individually or for additional features such as session state or SQL cache dependency, run aspnet_regsql at the command line. For help with command line options, use the "-?" switch.

Click Next to continue.

Click next to continue



Select a Setup Option

What database task do you want to perform?

(●) Configure SQL Server for application services

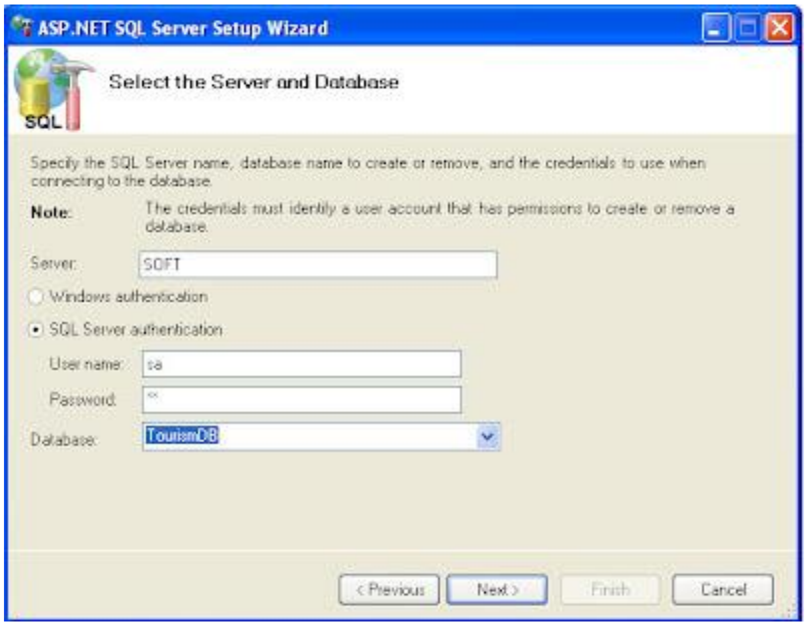
This option runs a script that creates a new database or configures an existing database to store information for ASP.NET membership, profiles, role management, personalization and SQL Web event provider.

( ) Remove application services information from an existing database

This option removes information about ASP.NET membership, profiles, role management, personalization and SQL Web event provider from the database. Note: This process cannot be undone.

Note:
To configure the database for additional features such as session state or SQL cache dependency, run aspnet_regsql at the command line. For help with command line options, use the "-?" switch.

Select Configure SQL Server for application services and Press Next



Select the Server and Database

Specify the SQL Server name, database name to create or remove, and the credentials to use when connecting to the database.

Note: The credentials must identify a user account that has permissions to create or remove a database.

Server: SOFT

( ) Windows authentication

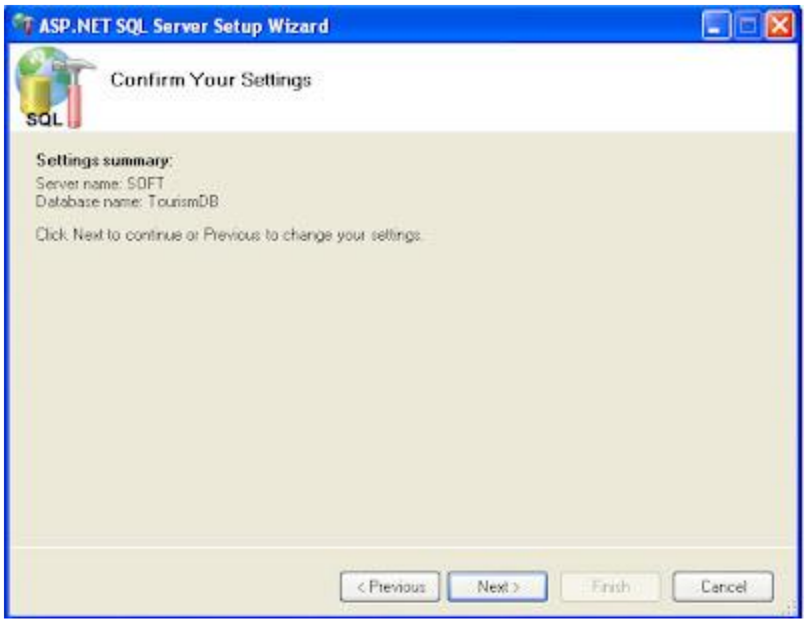
(●) SQL Server authentication

User name: sa

Password: **

Database: TourismDB

Type your server name. Here my server name is SOFT. Select SQL Server Authentication, type user name and password of server. Then select Database Name. Click Next to continue. You will see the following screen.

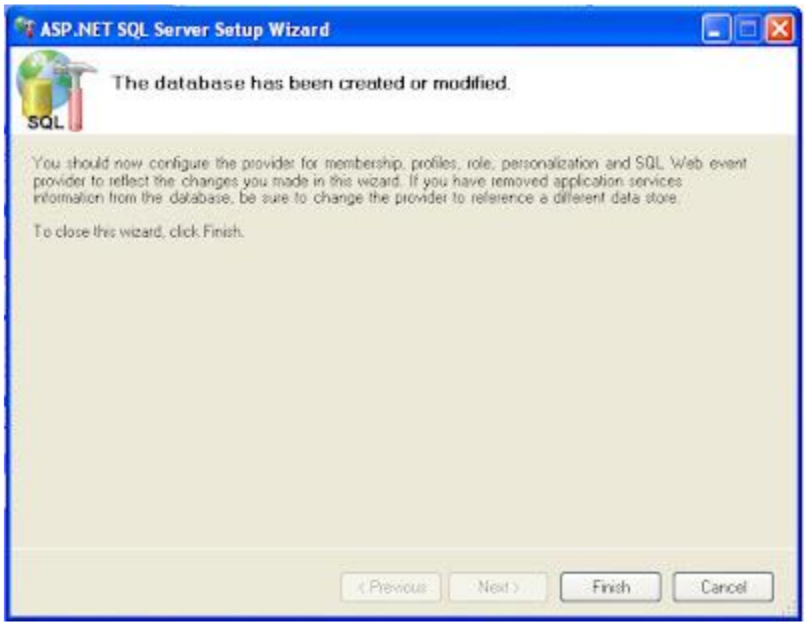


Confirm Your Settings

Settings summary:
Server name: SOFT
Database name: TourismDB

Click Next to continue or Previous to change your settings.

Click Next to continue.



The database has been created or modified.

You should now configure the provider for membership, profiles, role, personalization and SQL Web event provider to reflect the changes you made in this wizard. If you have removed application services information from the database, be sure to change the provider to reference a different data store.

To close this wizard, click Finish.

Click Finish. Now all tables and SPs are imported to desired database.

Step 2: Configuring Connection String: To retrieve membership information, create a connection string. This connection string can be used for retrieving other information from database for the website.

```
 <connectionStrings>
   <add name="ApplicationServices" connectionString="Data Source=SOFT;Initial Catalog=TourismDB; User ID=sa;Password=sa" providerName="System.Data.SqlClient"/>
 connectionStrings>
```

Step 3: Configuring Authentication: Authentication is the process by which system can identify who are you? ASP.NET framework supports 3 types of Authentication.

- Windows Authentication
- .NET Passport Authentication
- Form Authentication.

We will configure form authentication here. When form authentication is enabled, users are typically identified by the cookies. There are several options for configuring form authentication. One of the options is cookiless which enables you to use form authentication even when the browser does not support cookies.

In this case, Membership handles all details' of retrieving and storing user and role information. Add the following portion in web.config file to enable form authentication.

```
<system.web>
  <authentication mode="Forms">
    <forms loginUrl="~/Account/Login.aspx" timeout="2880" cookieless="AutoDetect">
    forms>
  authentication>
 system.web>
```

Step 4: Membership Provider Settings:
To enable membership, add the following code snippet to web.config file. There are many elements in membership provider. We will use only the important elements here. Basic elements are – name, type(name space of the provider), connectionStringName (name of connection string which is used to connect with membership database). You can use PasswordFormat either of the three – Hash, encrypted and clear. Default value of PasswordFormat attribute is Hashed. You can use maxInvalidPasswordAttempts – number of maximum attempts to enter invalid password, minRequiredPasswordLength – minimum password length.

```
<system.web>

  <membership>
    <providers>
      <clear/>

        <add name="AspNetSqlMembershipProvider" type="System.Web.Security.SqlMembershipProvider"
connectionStringName="ApplicationServices" enablePasswordRetrieval="false" enablePasswordReset="true"
requiresQuestionAndAnswer="false" requiresUniqueEmail="false" maxInvalidPasswordAttempts="5"
minRequiredPasswordLength="6" minRequiredNonalphanumericCharacters="0" passwordAttemptWindow="10"
applicationName="/"/>
    providers>
  membership>
system.web>
```

Step 5: Add a Login Form: Add a login form name Login.aspx in the website. Now add a Login control in Login page. You can change the style of the login control according to your requirements. Elegant format of login control is used here. For the first time, there is no user available. So we must create some users. We used here `CreateUserText="Register" CreateUserUrl="~/Register.aspx".` This two properties are used here so that we can create a new user by going to Register.aspx page.

```
<asp:Login ID="loginUser" runat="server" CreateUserText="Register"
            CreateUserUrl="~/Register.aspx"
            onauthenticate="loginUser_Authenticate" BackColor="#F7F7DE"
            BorderColor="#CCCC99" BorderStyle="Solid" BorderWidth="1px"
            Font-Names="Verdana" Font-Size="10pt" Height="135px" Width="286px">
            <TitleTextStyle BackColor="#6B696B" Font-Bold="True" ForeColor="#FFFFFF" />
        asp:Login>
```

Add the following code snippet for handling corresponding **onauthenticate** event

```
protected void loginUser_Authenticate(object sender, AuthenticateEventArgs e)
    {
        bool isLogin = Membership.ValidateUser(loginUser.UserName,
loginUser.Password);
        if (isLogin)
        {
            loginUser.Visible = true;
            Session["user"] = User.Identity.Name;
            FormsAuthentication.RedirectFromLoginPage(loginUser.UserName,
true);
            Response.Redirect("Default.aspx");
        }
    }
```

After log in successfully. Page will be redirected to Default.aspx page. So we need to add a Default.aspx page. We will add it later.

Step 6: Add Register Form: In step 5 I have added property `CreateUserUrl="~/Register.aspx" .`But this page is not exists. So create a page Register.aspx. In this form we will use a CreateUserWizard which enable to display a user registration form. Like login control you can also change style of CreateUserWizard control. I used here simple format.

```
<asp:CreateUserWizard ID="RegisterUser" runat="server"
            OnCreatedUser="RegisterUser_CreatedUser" BackColor="#E3EAEB"
            BorderColor="#E6E2D8" BorderStyle="Solid" BorderWidth="1px"
            Font-Names="Verdana" Font-Size="0.8em">


            <ContinueButtonStyle BackColor="White" BorderColor="#C5BBAF"
                BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
                ForeColor="#1C5E55" />
            <CreateUserButtonStyle BackColor="White" BorderColor="#C5BBAF"
                BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
                ForeColor="#1C5E55" />
            <TitleTextStyle BackColor="#1C5E55" Font-Bold="True" ForeColor="White" />
            <WizardSteps>
                <asp:CreateUserWizardStep ID="CreateUserWizardStep1" runat="server">
                asp:CreateUserWizardStep>
                <asp:CompleteWizardStep ID="CompleteWizardStep1" runat="server">
                asp:CompleteWizardStep>
            WizardSteps>
            <HeaderStyle BackColor="#666666" BorderColor="#E6E2D8" BorderStyle="Solid"
                BorderWidth="2px" Font-Bold="True" Font-Size="0.9em" ForeColor="White"
                HorizontalAlign="Center" />
            <NavigationButtonStyle BackColor="White" BorderColor="#C5BBAF"
                BorderStyle="Solid" BorderWidth="1px" Font-Names="Verdana"
                ForeColor="#1C5E55" />
            <SideBarButtonStyle ForeColor="White" />
            <SideBarStyle BackColor="#1C5E55" Font-Size="0.9em" VerticalAlign="Top" />
            <StepStyle BorderWidth="0px" />

        asp:CreateUserWizard>
```

Add the following code snippet for handling corresponding `OnCreatedUser` event.

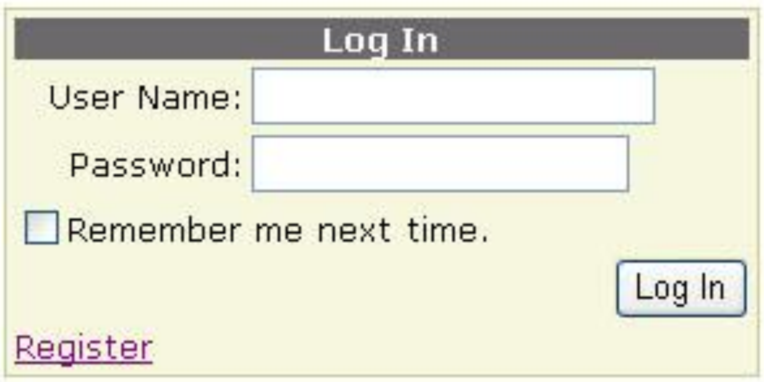
```
protected void RegisterUser_CreatedUser(object sender, EventArgs e)
    {
        FormsAuthentication.SetAuthCookie(RegisterUser.UserName, false /*
createPersistentCookie */);

        string continueUrl = RegisterUser.ContinueDestinationPageUrl;
        if (String.IsNullOrEmpty(continueUrl))
        {
            continueUrl = "Login.aspx";
        }
        Response.Redirect(continueUrl);
    }
```

Step 7: Add location tag for Register page: Add the following code snippet in web.config file. This means register page is allowed to all. Because, for the first time you have to create a user without login to the application.

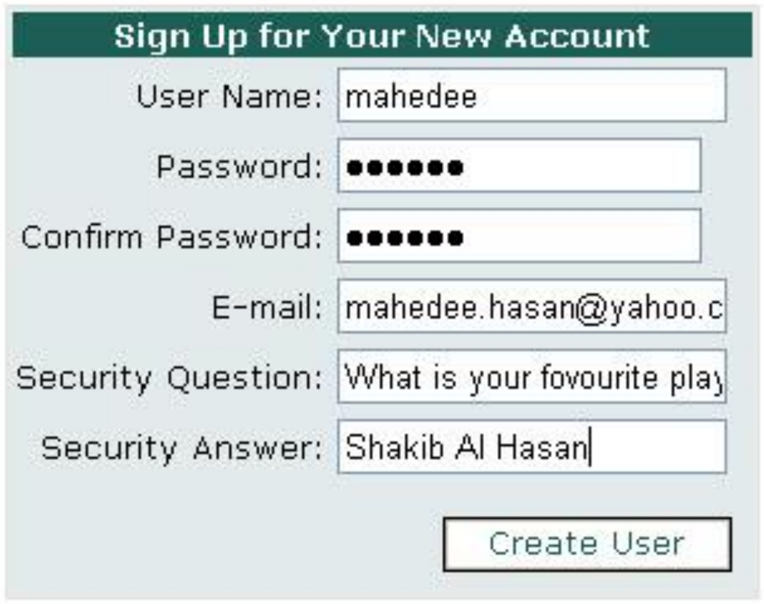
```
<location path="Register.aspx">
  <system.web>
    <authorization>
      <allow users="*"/>
    authorization>
  system.web>
location>
```

Step 8: Add Default Page: Add Default.aspx page. You can add here some message here because page will be redirected to the default page after successfully logged in.

Now run the application. You will see the following login screen.



But you don't have any user to login. So, click Register in login page and you will be redirected to Register page.



User Name: mahedee
Password: ●●●●●●
Confirm Password: ●●●●●●
E-mail: mahedee.hasan@yahoo.c
Security Question: What is your fovourite play
Security Answer: Shakib Al Hasan

Now create a user and try to login by that user from login page. Yes! you have logged in and redirected to Default.aspx.

Step 9: Add Login Status: We always see a message **"Welcome Mahedee!"** after login and **Login** before login to the site. We see this login status in all pages. If user want to login from any pages, he just need to click in **login** link and go to login page. Here he can login and enter the site as authentic user. Here we have created a master page and login status control to do exactly the same thing. Here I used LoginStatus and LoginView Control. It should be mentioned here what are the functions of LoginStatus and LoginView Control? LoginStatus control enables you to display a login or logout link depending on user authentication. It actually displays whether user is currently logged in or not. LoginView enables you to display different content depending on user Authentication status or role.

Add the following code snippet in MasterPage.master.

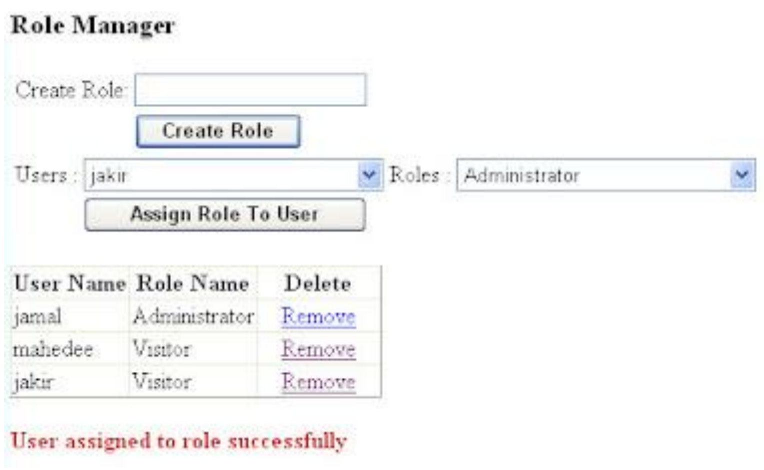
```
    <asp:LoginView ID="HeadLoginView" runat="server" EnableViewState="false">
        <AnonymousTemplate>
            [ <a href="Login.aspx" id="HeadLoginStatus" runat="server">Log Ina> ]
        AnonymousTemplate>
        <LoggedInTemplate>
            Welcome <span class="bold">
                <asp:LoginName ID="HeadLoginName" runat="server" />
            span>! [
             <asp:LoginStatus ID="HeadLoginStatus" runat="server" LogoutAction="Redirect"
LogoutText="Log Out"
                LogoutPageUrl="~/" />
            ]
        LoggedInTemplate>
    asp:LoginView>
```

Now if you add MasterPage.master to any of your pages. You will get login status after login to the application. Here I used MasterPage in Default.aspx.

Step 10: Add Role Manager: To add role manager add the following code snippet to web.config file.

```
<roleManager enabled="true">
   <providers>
     <clear/>
            <add         name="AspNetSqlRoleProvider"         type="System.Web.Security.SqlRoleProvider"
connectionStringName="ApplicationServices" applicationName="/"/>
       <add name="AspNetWindowsTokenRoleProvider" type="System.Web.Security.WindowsTokenRoleProvider"
applicationName="/"/>
   providers>
  roleManager>
```

Step 11: Create Role Manager Page: Create a page like below for managing roles.



User Name	Role Name	Delete
jamal	Administrator	Remove
mahedee	Visitor	Remove
jakir	Visitor	Remove

User assigned to role successfully

Step 12: Add Role Provider in web.config: Add the following code snippet for web.config for enable Role Provider.

```
<roleManager enabled="true">
     <providers>
       <clear/>
                    <add             name="AspNetSqlRoleProvider"             type="System.Web.Security.SqlRoleProvider"
connectionStringName="ApplicationServices" applicationName="/"/>
       <add name="AspNetWindowsTokenRoleProvider" type="System.Web.Security.WindowsTokenRoleProvider"
applicationName="/"/>
     providers>
   roleManager>
```

Step 13: Manage Role: In step 11 we saw Role manager page. Now we will create the corresponding code for managing (Create, delete roles & assign users to role) roles. Here I also add a grid to display user role mapping.

a. Create Roles: Add the following code for corresponding event to create role. I have added the code for corresponding Create button event.

?

```
protected void btnCreateRole_Click(object sender, EventArgs e)
    {
        try
        {
            if (!Roles.RoleExists(this.txtRole.Text))
            {
                Roles.CreateRole(txtRole.Text);
                BindRoles();
                BindUsers();
                this.lblMsg.Text = "Role created successfully";
            }
            else
            {
                lblMsg.Text = "Role already exists";
            }
        }
        catch (Exception ex)
        {
            lblMsg.Text = ex.Message;
        }
    }
```

b.Bind Roles: I have bind a dropdown list for Roles.

?

```
private void BindRoles()
{
    SqlConnection con = new
SqlConnection(Convert.ToString(ConfigurationManager.ConnectionStrings["Appli
cationServices"]));

    con.Open();
    SqlDataAdapter da = new SqlDataAdapter("select UserId, UserName from
aspnet_users", con);
    DataSet ds = new DataSet();
    da.Fill(ds);
    this.ddlUsers.DataSource = ds;
    ddlUsers.DataTextField = "UserName";
    ddlUsers.DataValueField = "UserName";
    ddlUsers.DataBind();

    con.Close();
}
```

c.Bind Users: A dropdown list is bound for roles in this occasion also.

?

```
private void BindUsers()
{
    SqlConnection con = new
SqlConnection(Convert.ToString(ConfigurationManager.ConnectionStrings["Appli
cationServices"]));

    con.Open();
    SqlDataAdapter da = new SqlDataAdapter("SELECT RoleId, RoleName FROM
aspnet_Roles", con);
    DataSet ds = new DataSet();
    da.Fill(ds);
    this.ddlRoles.DataSource = ds;
    ddlRoles.DataTextField = "RoleName";
    ddlRoles.DataValueField = "RoleName";
    ddlRoles.DataBind();

    con.Close();
}
```

d.Assign Roles to User: Use the following code snippet to assign roles to users.

?

```
protected void btnAssign_Click(object sender, EventArgs e)
    {
        try
        {
            if (!Roles.IsUserInRole(this.ddlRoles.SelectedValue))
            {
                Roles.AddUserToRole(this.ddlUsers.SelectedValue,this.ddlRole
s.SelectedValue);
                BindUsers();
                BindRoles();
                BindUserRole();
                this.lblMsg.Text = "User assigned to role successfully";
            }
            else
            {
                this.lblMsg.Text = "Role already exits for the user.";
            }
        }
        catch (Exception ex)
        {
            this.lblMsg.Text = ex.Message;
        }
    }
```

e.Remove role from the user: I have added a gridview in rolemanger page. Here is the code snippet of delete button event handler. Roles.RemoveUserFromRole(string username, string rolename) Method is responsible to remove role form the user.

?

```
protected void btnDelete_Click(object sender, EventArgs e)
    {
        LinkButton btnDelete = sender as LinkButton;

        //Identify the clicked row
        int rowIndex = Convert.ToInt32(btnDelete.Attributes["RowIndex"]);

        GridViewRow gvRow = this.gvRoleToUsers.Rows[rowIndex];

        try
        {
            Roles.RemoveUserFromRole(gvRow.Cells[0].Text,
gvRow.Cells[1].Text); //username, role name
            BindUsers();
            BindRoles();
            BindUserRole();
            this.lblMsg.Text = "User is removed from the role";
        }
        catch (Exception ex)
        {
            this.lblMsg.Text = ex.Message;
        }


    }
```

f.**Delete roles:** The following code snippet is responsible to delete a roles.

?

```
protected void btnDeleteRole_Click(object sender, EventArgs e)
{
    try
    {
        Roles.DeleteRole(this.ddlRoles.SelectedValue); //Responsible to
delete role
```

```
        this.BindRoles();
        this.BindUserRole();
        this.lblMsg.Text = "Role is deleted!";
    }
    catch (Exception exp)
    {
        this.lblMsg.Text = exp.Message;
    }
}
```

Step 14: Show and Hide Menu depending on Roles: We have already created roles and assigned roles to users. Now if we want to show and Hide the menus depending on roles, we have to follow the following steps.

a. Add sitemap: Add web.sitemap in the project. We have added this site map with a treeview. Have a look in the web.sitemap content.

```
xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="ASP.NET" title=""  description="" roles="Administrator">
    <siteMapNode url="Defult.aspx" title="Home"  description=""/>
    <siteMapNode url="Register.aspx" title="Register"  description="" />
    <siteMapNode url="RoleManager.aspx" title="Role Manager" description=""
roles="Administrator"/>
    <siteMapNode url="ManageUsers.aspx" title="User Manager" description=""
roles="Administrator,IT officer"/>
  siteMapNode>
siteMap>
```

You may understand above code snippet that only Administrator can view RoleManager.aspx page and Administrator and IT Officer can view ManageUsers.aspx page. There is no restriction in other pages.

b. Add Location element in web.config: To hide and show menu as described above add the following code snippet in web.config file.

```
  <location path="RoleManager.aspx">
    <system.web>
      <authorization>
        <allow roles="Administrator"/>
        <deny users="*"/>
      authorization>
    system.web>
  location>

  <location path="ManageUsers.aspx">
    <system.web>
      <authorization>
        <allow roles="Administrator,IT officer"/>
        <deny users="*"/>
      authorization>
    system.web>
  location>
```