Articles » Web Development » ASP.NET » General

ASP.NET Membership and Role Provider

By **S V Saichandra**, 27 Nov 2011

4.80 (12 votes)

Download demo - 2.12 MB

Introduction

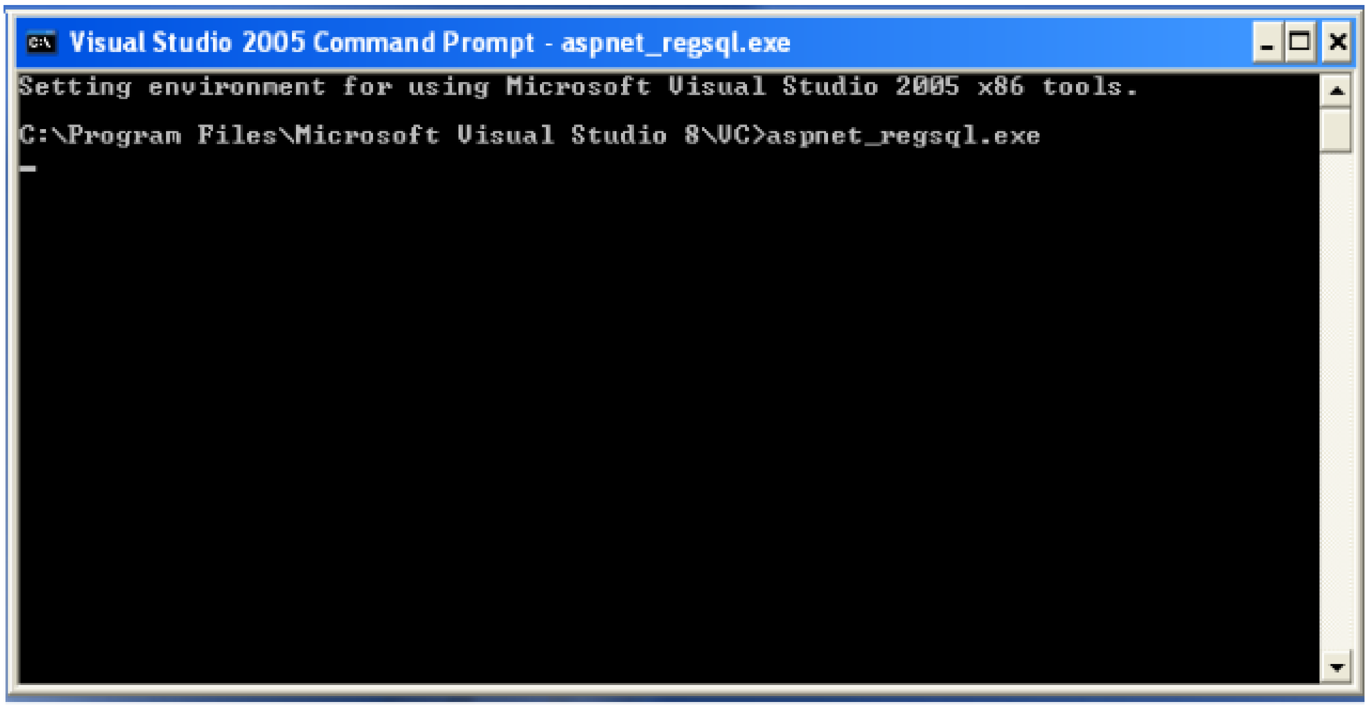
ASP.NET 2.0 provides built in controls to manage Membership in Web Applications. All these controls use ASP.NET providers that are loaded via *web.config* file. Membership provider and Role provider allow a complete system to maintain users information, authenticate and authorize the users. This article demonstrates how to use and configure the default Member ship and Role provider.

Implementing the Membership and Role Provider

Initially by using the Visual Studio 2005/2008/2010, create an ASP.NET website/web application. If you are using Visual Studio 2010, login and registration pages are available by default in the application. Create Registration page and then drag the Create User Wizard control from the Login controls section of the Toolbox. Now to store the user information, we need to create the database in the SQL Server. Follow the steps given below to use built in user store schema for maintaining the user information.

1. Go to Visual Studio, Visual Studio tools and then open the Visual Studio Command Prompt.
2. Use the *aspnet_regsql.exe* command to run the ASP.NET SQL Server Setup Wizard.
3. Check the option “Configure SQL Server for application services”.
4. Select the Server Instance and the database name for the application, if the database name is not provided, default `aspnetdb` database is created.
5. Click the confirm settings and finish button to create the database store.

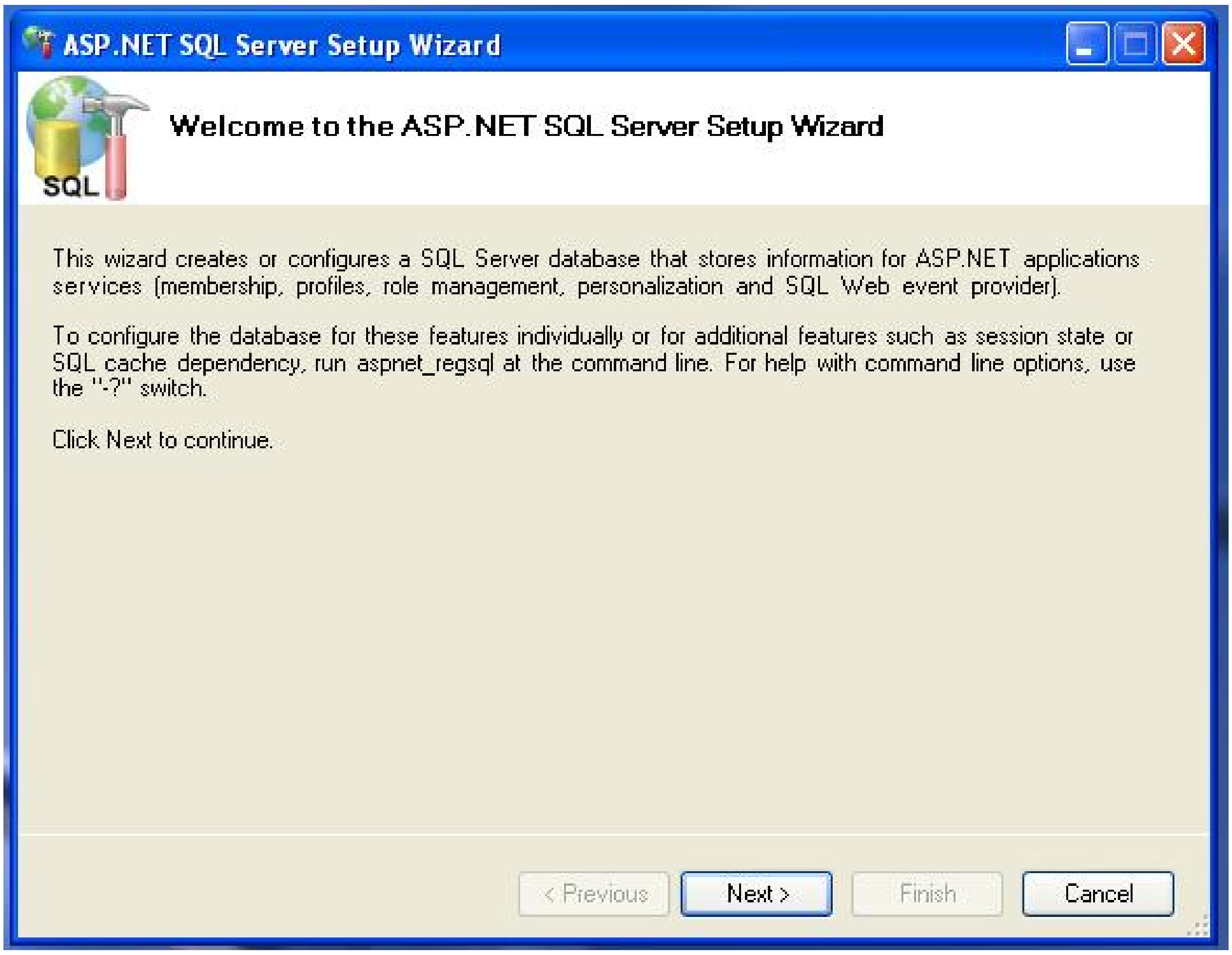
Step 1:



```
Setting environment for using Microsoft Visual Studio 2005 x86 tools.

C:\Program Files\Microsoft Visual Studio 8\VC>aspnet_regsql.exe
_
```

Step 2:



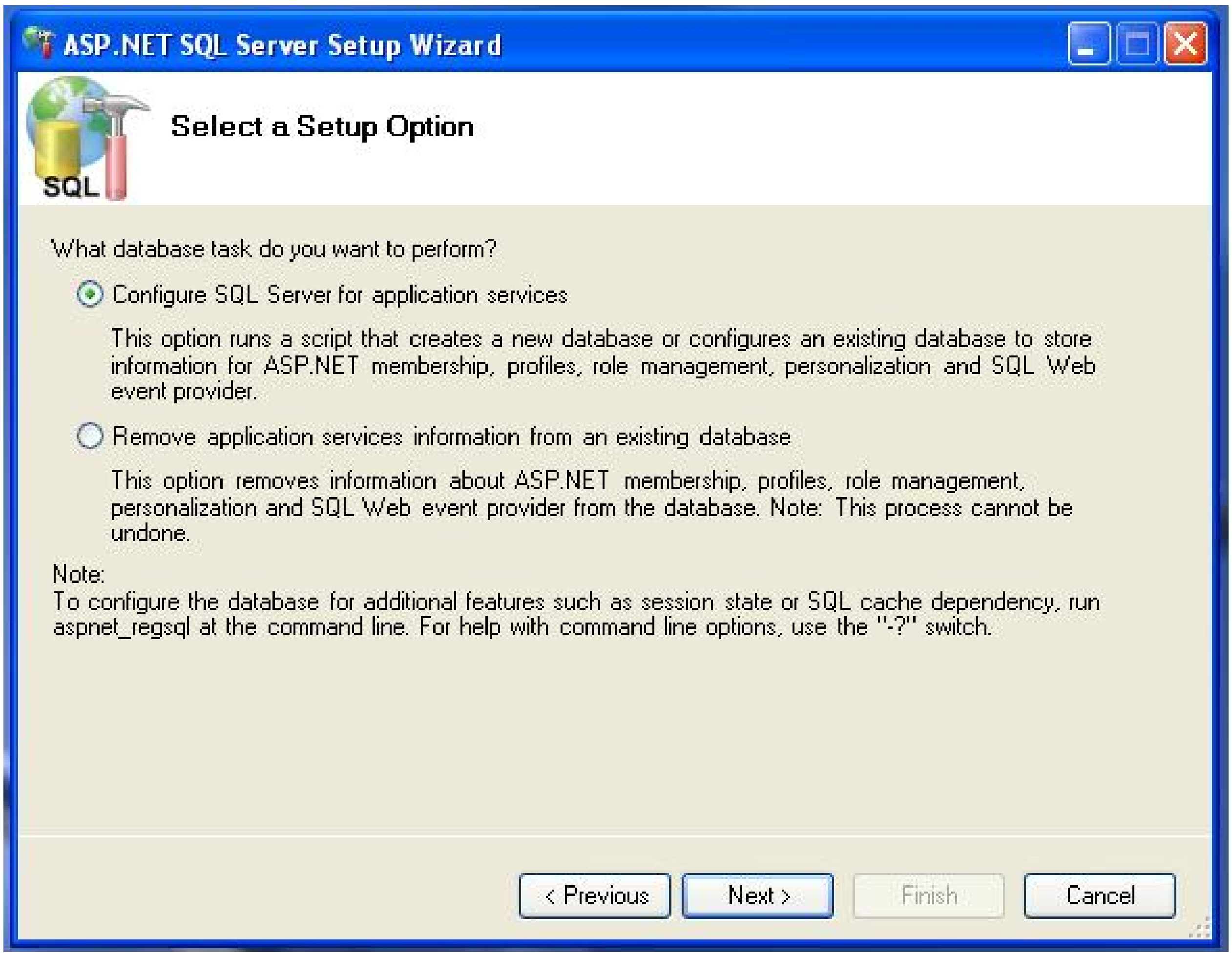
Welcome to the ASP.NET SQL Server Setup Wizard

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.

Step 3:



Select a Setup Option

What database task do you want to perform?

[✓ Configure SQL Server for application services]

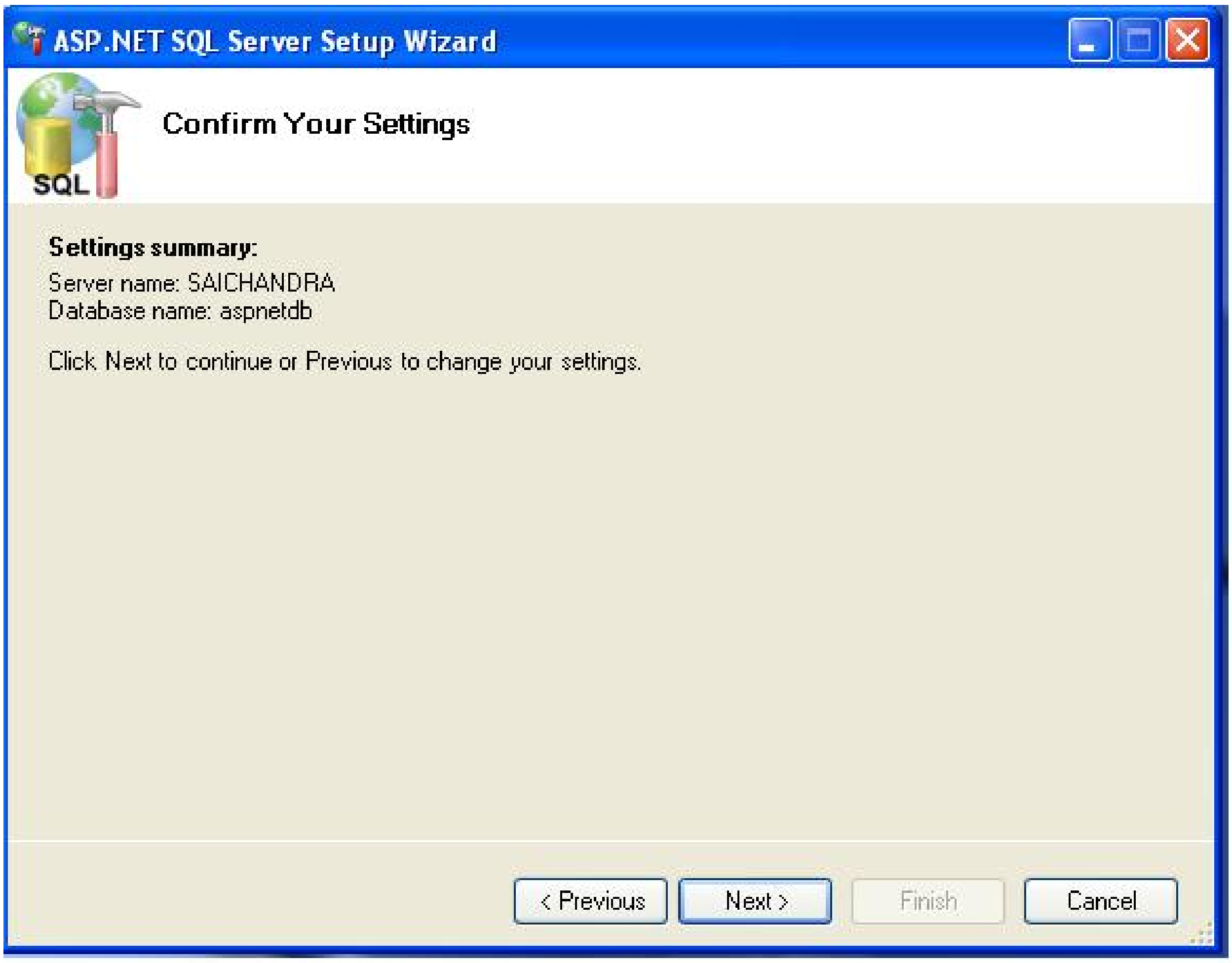
This option runs a script that creates a new database or configures an existing database to store information for ASP.NET membership, profiles, role management, personalization and SQL Web event provider.

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.

Step 4:

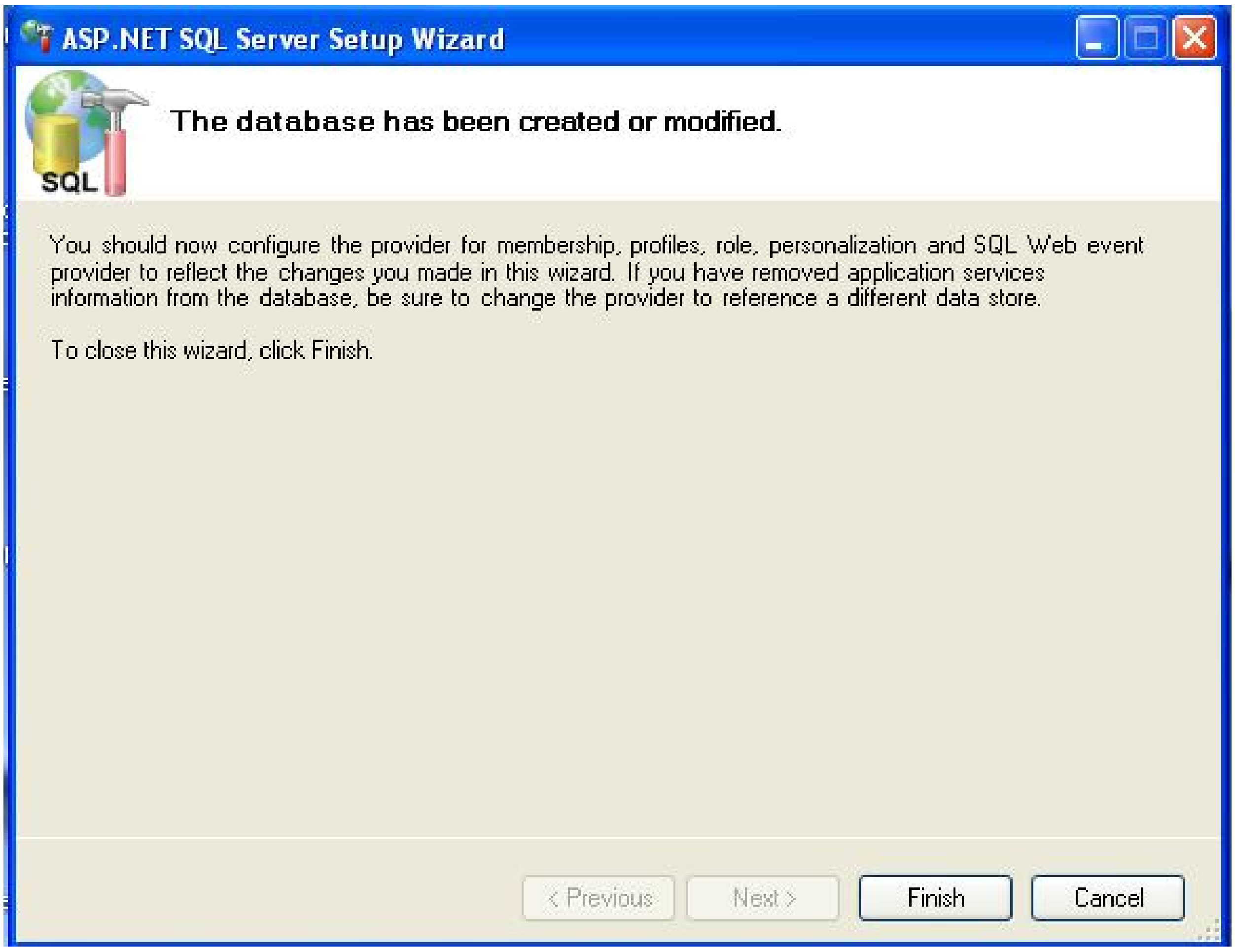


Confirm Your Settings

Settings summary:
Server name: SAICHANDRA
Database name: aspnetdb

Click Next to continue or Previous to change your settings.

Step 5:



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.

Preparing to build the security system for use in application, we need to configure the membership provider in *web.config* file. The following settings for Forms Authentication, Membership and Role provider are applied in the *web.config* file.

Forms Authentication Settings

The authentication mode under *system.web* tag is set to "`Forms`" and the elements included in are `loginUrl`, `defaultUrl`, `timeout`, `cookieless` and `protection` which specifies the login page URL, default page URL, cookie expiration time and protection level respectively. The settings in *web.config* file would look similar to the code shown below:

```
<authentication mode="Forms">
     <forms cookieless="UseCookies" defaultUrl="HomePage.aspx"
        loginUrl="UnAuthorized.aspx" protection="All" timeout="30">
          </forms>
</authentication>
```

Membership Provider Settings

Some of the important elements to be considered in the Membership provider are name – name of the provider, type – namespace of the provider, connectionStringName – name of the connectionstring and the most important password format. The password format is available in three formats, Hashed, Encrypted and Clear. Hashed format provides one way of storing password in encrypted format which cannot be brought back to original state, whereas Encrypted format provides both to encrypt and decrypt the password.

```
<membership defaultProvider="Demo_MemberShipProvider">
        <providers>
                <add name="Demo_MemberShipProvider"
                    type="System.Web.Security.SqlMembershipProvider"
                    connectionStringName="cnn"
                    enablePasswordRetrieval="false"
                    enablePasswordReset="true"
                    requiresQuestionAndAnswer="true"
                    applicationName="/"
                    requiresUniqueEmail="false"
                    passwordFormat="Hashed"
                    maxInvalidPasswordAttempts="5"
                    minRequiredPasswordLength="5"
                    minRequiredNonalphanumericCharacters="0"
                    passwordAttemptWindow="10" passwordStrengthRegularExpression="">
        </providers>
</membership>
```

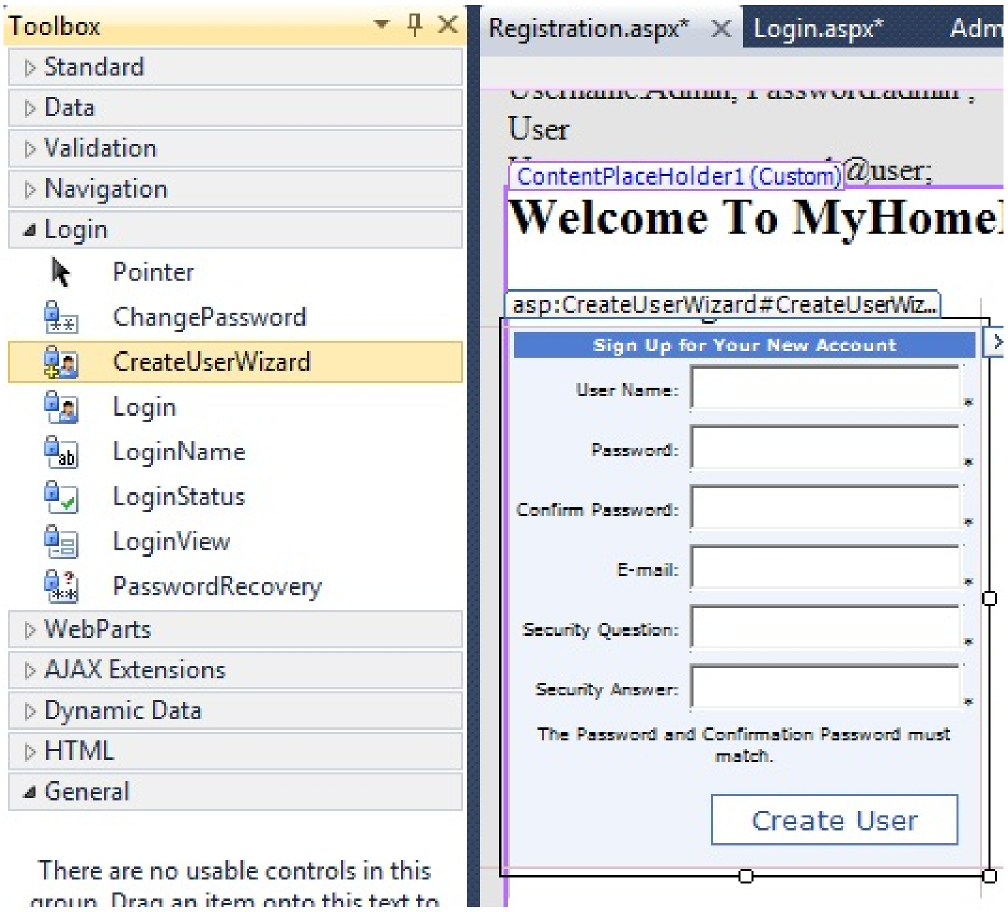
Role Provider Settings

The similar way is to specify the settings for default Provider under *system.web* tag of the *web.config* file as shown below. The settings are simple and self explanatory.

```
<roleManager enabled="true" cacheRolesInCookie="true"
        cookieName="TBHROLES" defaultProvider="Demo_RoleProvider">
              <providers>
                  <add connectionStringName="dld_connectionstring"
                  applicationName="/" name="Demo_RoleProvider"
                  type="System.Web.Security.SqlRoleProvider, System.Web,
                  Version=2.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"/>
             </providers>
</roleManager>
```

In the *login.aspx* and *Registration.aspx* pages, we need to use the providers to complete the membership system for the application.

Registering the Users

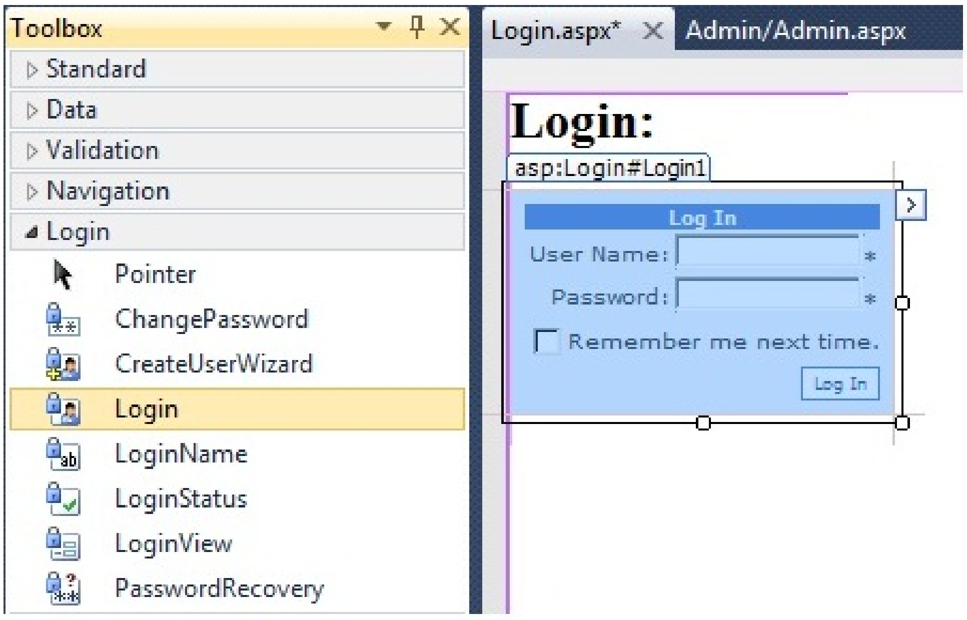


Registration page for the users can be easily created by using the available create user wizard and the following event handlers:

```
protected void CreateUserWizard1_CreatedUser(object sender, EventArgs e)
{
     MembershipCreateStatus p = MembershipCreateStatus.Success;
     Membership.CreateUser(CreateUserWizard1.UserName,
                CreateUserWizard1.Password, CreateUserWizard1.Email,
     CreateUserWizard1.Question, CreateUserWizard1.Answer, true, out p);
}

protected void CreateUserWizard1_ContinueButtonClick(object sender, EventArgs e)
{
  Response.Redirect("login.aspx");
}
```

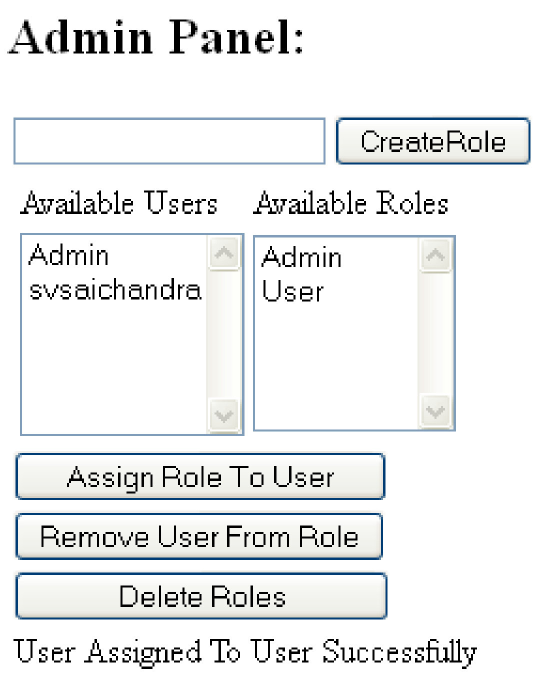
Authenticate the Users



The users can be authenticated by using the `login_Authenticate` event of the Login control. The code to authenticate users goes here:

```
protected void Login1_Authenticate(object sender,AuthenticateEventArgs e)
{
 if (Membership.ValidateUser(Login1.UserName, Login1.Password) == true)
    {
        Login1.Visible = true;
        Session["user"] = User.Identity.Name;
        FormsAuthentication.RedirectFromLoginPage(Login1.UserName, true);
    }
 else
    {
        Response.Write("Invalid Login");
    }
}
```

Creating the Admin Panel



In the Admin Panel, the features to Add, Edit, Delete and Assign Roles to users are provided to the administrator.

Creating the Roles

The following code snippet shows you how to create Roles:

```
Public void createRoles()
{
    try
    {
        if (!Roles.RoleExists(txtrolename.Text))
        {
            Roles.CreateRole(txtrolename.Text);
            BindUsers();
            BindRoles();
            Label1.Text = "Role(s) Created Successfully";
        }
        else
        {
            Label1.Text = "Role(s) Already Exists";
        }
    }
    catch (Exception ex)
    {
        Label1.Text = ex.Message;
    }
}
```

BindRoles

The `BindRoles` method is used to bind the available roles in the store to the user control.

```
public void BindRoles()
{
    SqlDataAdapter da = new SqlDataAdapter("select RoleName from aspnet_Roles", cnn);
    DataSet ds = new DataSet();
    da.Fill(ds, "Roles");
    lstRoles.DataSource = ds;
    lstRoles.DataTextField = "RoleName";
    lstRoles.DataValueField = "RoleName";
    lstRoles.DataBind();
}
```

BindUsers

The `BindUsers` method is used to bind the available users in the store to the user control.

```
public void BindUsers()
{
    SqlDataAdapter da = new SqlDataAdapter("select UserName from aspnet_users", cnn);
    DataSet ds = new DataSet();
    da.Fill(ds, "Roles");
    lstusers.DataSource = ds;
    lstusers.DataTextField = "UserName";
    lstRoles.DataValueField = "RoleName";
    lstusers.DataBind();
}
```

The following methods take `username` and `rolename` as parameters.

Assign Roles To User

The available roles can be assigned to the user in the following way:

```
private void AssignRoles()
    {
        try
        {
            if (!Roles.IsUserInRole(lstRoles.SelectedItem.Text))
            {
                Roles.AddUserToRole(lstusers.SelectedItem.Text,
                                lstRoles.SelectedItem.Text);
                BindUsers();
                BindRoles();
                Label1.Text = "User Assigned To User Successfully";
            }
            else
            {
                Label1.Text = "Role(s) Already Assigned To User";
            }
        }
        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
    }
```

Remove Roles from the User

You can remove the user from a role in the following manner:

```
private void RemoveuserFromRole()
{
    try
    {
        Roles.RemoveUserFromRole(lstusers.SelectedItem.Text, lstRoles.SelectedItem.Text);
        BindUsers();
        BindRoles();
        Label1.Text = "User Is Removed From The Role Successfully";
    }
    catch (Exception ex)
    {
        Label1.Text = ex.Message;
    }
}
```

Delete Roles

The code is used to delete the existing Roles, if they are not in use.

```
public void RemoveRole()
{
 try
        {
            Roles.DeleteRole(lstRoles.SelectedItem.Text);
            BindUsers();
            BindRoles();
            Label1.Text = "Role(s) Removed Successfully";
        }
        catch (Exception ex)
        {
            Label1.Text = ex.Message;
        }
}
```

Restrict the users depending on the roles by using *web.config* settings as follows:

```
<authorization
        <allow roles ="Admin"/>
        <deny users ="*"/>
</authorization>
```

In the above code, if you write `deny users =”*”` and then `allow roles =”Admin”`, there seems to be no difference, but the code wouldn’t work for you because writing the `deny user =”*”` at the beginning would even restrict the admin to access the folders.

Show/Hide The Menu Items to The Users Depending on Roles

```
if (Roles.IsUserInRole("Admin"))
{
    Menu1.Items[0].Text = "Admin";
}
else
{
    Menu1.Items[0].Text = "";
}
```

Conclusion

We have seen an overview of using the out of the box providers available to implement the Membership and Roles for the ASP.NET Application. For more details about Forms Authentication, Membership and Role provider, you can refer to the following links:

- MSDN Library
- MSDN Library

History

- 9th November, 2011: Initial version
- 27th November, 2011: Updated code and added images to the article

License

About the Author

S V Saichandra
Software Developer
India

S V Sai Chandra is a Software Engineer from Hyderabad Deccan. He started Embedded Programing in his college days and now he is a Web Developer by Profession. He Loves coding and his passion is always been towards Microsoft Technologies. Apart from coding his other hobbies include reading books, painting and hang out with friends is his most favorite past time hobby.
He blogs at
http://technowallet.blogspot.com
Technical Skills:
C#,Ado.Net,Asp.Net,Sql Server,JavaScript,XML,Web services.

Follow on Twitter

Comments and Discussions

18 messages have been posted for this article Visit **http://www.codeproject.com/Articles/281573/ASP-NET-Membership-and-Role-Provider** to post and view comments on this article, or click **here** to get a print view with messages.