Membership and Role Providers in ASP.NET

Membership and Role Providers

- Membership and role providers exist to provide authentication and authorization services to our applications.
- The provider model in ASP.NET 2.0 provides extensibility points for developers to plug their own implementation of a feature into the runtime. Both the membership and role features in ASP.NET 2.0 follow the provider pattern by specifying an interface, or contract.

Membership and Role Providers

```
<membership>
  <providers>
    <add
      name="AspNetSqlMembershipProvider"
      type="System.Web.Security.SqlMembershipProvider, ..."
      connectionStringName="LocalSqlServer"
      enablePasswordRetrieval="false"
      enablePasswordReset="true"
      requiresQuestionAndAnswer="true"
      applicationName="/"
      requiresUniqueEmail="false"
      passwordFormat="Hashed"
      maxInvalidPasswordAttempts="5"
      minRequiredPasswordLength="7"
      minRequiredNonalphanumericCharacters="1"
      passwordAttemptWindow="10"
      passwordStrengthRegularExpression=""
    />
  </providers>
</membership>
```

Membership and Role Providers

By default, the machine.config file configures membership and roles to work with a SQL Server Express database file in the App_Data directory.

```
<add name="LocalSqlServer" connectionString="data source=.\SQLEXPRESS;Integrated
Security=SSPI;AttachDBFilename=|DataDirectory|aspnetdb.mdf;User Instance=true"
providerName="System.Data.SqlClient" />
```

Membership and Role Providers

You can always override the default setting and point all providers using LocalSqlServer to a remote database, or a non-Express database on the local machine.

1. Use the ASP.NET Sql Server Registration Tool (aspnet_regsql.exe) to create a new “aspnetdb” database.
2. **modify the web.config file for your application to redefine the LocalSqlServer connection string to point to the new database.**

```
<connectionStrings>
  <remove name="LocalSqlServer"/>
  <add name="LocalSqlServer"
       connectionString="server=.;database=aspnetdb;integrated security=sspi;"/>
</connectionStrings>
```

Using the Membership Provider

```
string username = "SwedishChef";
string password = "bj#kbj1k";
string email = @"swede@mailinator.com";
string question = "The greatest band ever?";
string answer = "ABBA";
bool isApproved = true;
MembershipCreateStatus status;

Membership.CreateUser(
    username, password, email,
    question, answer, isApproved,
    out status);

if(status == MembershipCreateStatus.Success)
{
  // party!
}
```

Using the Role Provider

```
If(Roles.IsUserInRole("Admin") == true)
 {
      // perform an admin action
 }
else
 {
      // give user an error message

}
```

ASP.NET Data Binding

ASP.NET Data Binding

ASP.NET Data Binding

- Simplified data binding
- Data source controls
- Data controls
  - GridView and DetailsView controls
  - Editing with GridView and DetailsView
- Caching
  - Cache configuration

Simplified Data Binding

- Data binding expressions are now simpler

```
<!-- ASP.NET 1.x data binding expression -->
<%# DataBinder.Eval (Container.DataItem, "Price") %>

<!-- Equivalent ASP.NET 2.0 data binding expression -->
<%# Eval ("Price") %>
```

DataSource Controls

- Declarative (no-code) data binding

Name	*Description*
SqlDataSource	Connects data-binding controls to SQL databases
AccessDataSource	Connects data-binding controls to Access databases
XmlDataSource	Connects data-binding controls to XML data
ObjectDataSource	Connects data-binding controls to data components
SiteMapDataSource	Connects site navigation controls to site map data

SqlDataSource

- Declarative data binding to SQL databases
  - Any database served by a managed provider
- Two-way data binding
  - SelectCommand defines query semantics
  - InsertCommand, UpdateCommand, and DeleteCommand define update semantics
- Optional caching of query results
- Parameterized operation

Using SqlDataSource

```
<asp:SqlDataSource ID="Titles" RunAt="server"
  ConnectionString="server=localhost;database=pubs;integrated security=true"
  SelectCommand="select title_id, title, price from titles" />
<asp:DataGrid DataSourceID="Titles" RunAt="server" />
```

Key SqlDataSource Properties

Name	*Description*
ConnectionString	Connection string used to connect to data source
SelectCommand	Command used to perform queries
InsertCommand	Command used to perform inserts
UpdateCommand	Command used to perform updates
DeleteCommand	Command used to perform deletes
DataSourceMode	Specifies whether DataSet or DataReader is used (default = DataSet)
ProviderName	Specifies provider (default = SQL Server .NET provider)

Parameterized Commands

- XxxParameters properties permit database commands to be parameterized
  - Example: Get value for WHERE clause in SelectCommand from query string parameter or item selected in drop-down list
  - Example: Get value for WHERE clause in DeleteCommand from GridView
- XxxParameter types specify source of parameter values

XxxParameter Types

Name	*Description*
Parameter	Binds a replaceable parameter to a data field
ControlParameter	Binds a replaceable parameter to a control property
CookieParameter	Binds a replaceable parameter to a cookie value
FormParameter	Binds a replaceable parameter to a form field
QueryStringParameter	Binds a replaceable parameter to a query string parameter
SessionParameter	Binds a replaceable parameter to a session variable

SqlDataSource Example 1

```
<%@ Page Language="C#" %>
<html>
 <head runat="server">
 <title>GridView Bound to SqlDataSource</title>
 </head> <body> <form id="form1" runat="server">
 <asp:GridView ID="GridView1" DataSourceID="SqlDataSource1"
runat="server" /> <asp:SqlDataSource ID="SqlDataSource1"
runat="server" SelectCommand="SELECT [au_id], [au_lname], [au_fname],
[phone], [address], [city], [state], [zip], [contract] FROM [authors]"
ConnectionString="<%$ ConnectionStrings:Pubs %>" /> </form> </body>
</html>
```

SqlDataSource Example 2

```
<%@ Page Language="C#" %>
<html>
  <head id="Head1" runat="server">
    <title>Updating Data Using GridView</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <asp:GridView ID="GridView1" AllowSorting="true" AllowPaging="true" Runat="server"
        DataSourceID="SqlDataSource1" AutoGenerateEditButton="true" DataKeyNames="au_id"
        AutoGenerateColumns="False">
        <Columns>
          <asp:BoundField ReadOnly="true" HeaderText="ID" DataField="au_id" SortExpression="au_id" />
          <asp:BoundField HeaderText="Last Name" DataField="au_lname" SortExpression="au_lname" />
          <asp:BoundField HeaderText="First Name" DataField="au_fname" SortExpression="au_fname" />
          <asp:BoundField HeaderText="Phone" DataField="phone" SortExpression="phone" />
          <asp:BoundField HeaderText="Address" DataField="address" SortExpression="address" />
          <asp:BoundField HeaderText="City" DataField="city" SortExpression="city" />
          <asp:BoundField HeaderText="State" DataField="state" SortExpression="state" />
          <asp:BoundField HeaderText="Zip Code" DataField="zip" SortExpression="zip" />
          <asp:CheckBoxField HeaderText="Contract" SortExpression="contract" DataField="contract" />
        </Columns>
      </asp:GridView>
      <asp:SqlDataSource ID="SqlDataSource1" Runat="server" SelectCommand="SELECT [au_id], [au_lname],
[au_fname], [phone], [address], [city], [state], [zip], [contract] FROM [authors]"
        UpdateCommand="UPDATE [authors] SET [au_lname] = @au_lname, [au_fname] = @au_fname, [phone] = @phone,
[address] = @address, [city] = @city, [state] = @state, [zip] = @zip, [contract] = @contract WHERE [au_id] =
@au_id"
        ConnectionString="<%$ ConnectionStrings:Pubs %>" />
    </form>
  </body>
</html>
```

ObjectDataSource

- Instead of a ConnectionString property, ObjectDataSource exposes a **TypeName** property that specifies an object type (class name) to instantiate for performing data operations. Similar to the command properties of SqlDataSource, the ObjectDataSource control supports properties such as **SelectMethod**, **UpdateMethod**, **InsertMethod**, and **DeleteMethod** for specifying methods of the associated type to call to perform these data operations.
- Declarative binding to data components
  - Leverage middle-tier data access components
  - Keep data access code separate from UI layer
- Two-way data binding
  - SelectMethod, InsertMethod, UpdateMethod, and DeleteMethod
- Optional caching of query results
- Parameterized operation

Key OjbectDataSource Properties

Name	*Description*
TypeName	Type name of data component
SelectMethod	Method called on data component to perform queries
InsertMethod	Method called on data component to perform inserts
UpdateMethod	Method called on data component to perform updates
DeleteMethod	Method called on data component to perform deletes
EnableCaching	Specifies whether caching is enabled (default = false)

ObjectDataSource Example

```
<%@ Page Language="C#" %>
 <html>
<body>
 <form id="form1" runat="server">
 <asp:DropDownList ID="DropDownList1" Runat="server" DataSourceID="ObjectDataSource2" AutoPostBack="True" />
 <asp:ObjectDataSource ID="ObjectDataSource2" Runat="server" TypeName="AuthorsComponent"
SelectMethod="GetStates"/> <br /> <br />
 <asp:GridView ID="GridView1" Runat="server" DataSourceID="ObjectDataSource1" AutoGenerateColumns="False"
AllowPaging="True" AllowSorting="True">
 <Columns>
<asp:CommandField ShowEditButton="True" />
<asp:BoundField HeaderText="ID" DataField="ID" SortExpression="ID" />
<asp:BoundField HeaderText="Name" DataField="Name" SortExpression="Name" />
<asp:BoundField HeaderText="LastName" DataField="LastName" SortExpression="LastName" /> <asp:BoundField
HeaderText="State" DataField="State" SortExpression="State" />
</Columns>
 </asp:GridView>
 <asp:ObjectDataSource ID="ObjectDataSource1" Runat="server" TypeName="AuthorsComponent"
SelectMethod="GetAuthorsByState" UpdateMethod="UpdateAuthor" DataObjectTypeName="Author"
SortParameterName="sortExpression">
 <SelectParameters>
<asp:ControlParameter Name="state" PropertyName="SelectedValue“
ControlID="DropDownList1"></asp:ControlParameter>
 </SelectParameters>
 </asp:ObjectDataSource>
 </form>
 </body>
 </html>
```

The GridView Control

- Enhanced DataGrid control
  - Renders sets of records as HTML tables
- Built-in sorting, paging, selecting, updating, and deleting support
- Supports rich assortment of field types, including ImageFields and CheckBoxFields
  - Declared in <Columns> element
- Highly customizable UI

GridView Field Types

Name	*Description*
BoundField	Renders columns of text from fields in data source
ButtonField	Renders columns of buttons (push button, image, or link)
CheckBoxField	Renders Booleans as check boxes
CommandField	Renders controls for selecting and editing GridView data
HyperLinkField	Renders columns of hyperlinks
ImageField	Renders columns of images
TemplateField	Renders columns using HTML templates

The DetailsView Control

- Renders individual records
  - Pair with GridView for master-detail views
  - Or use without GridView to display individual records
- Built-in paging, inserting, updating, deleting
- Uses same field types as GridView
  - Declared in <Fields> element
- Highly customizable UI

Web Parts

Web Parts

- Framework for building portal-style apps
  - Patterned after SharePoint Portal Server
  - System.Web.UI.WebControls.WebParts
- Rich UIs with minimal code
  - Edit page layout using drag-and-drop
  - Edit appearance and behavior and more
- Seamless personalization
- Intercommunication ("connections")

The WebPartManager Control

- Orchestrates operation of Web Parts
  - Maintains list of Web Parts and zones
  - Manages page state (e.g., display mode) and fires events when page state changes
  - Facilitates communication between Web Parts
  - Manages personalization and much more
- One instance per page; no UI

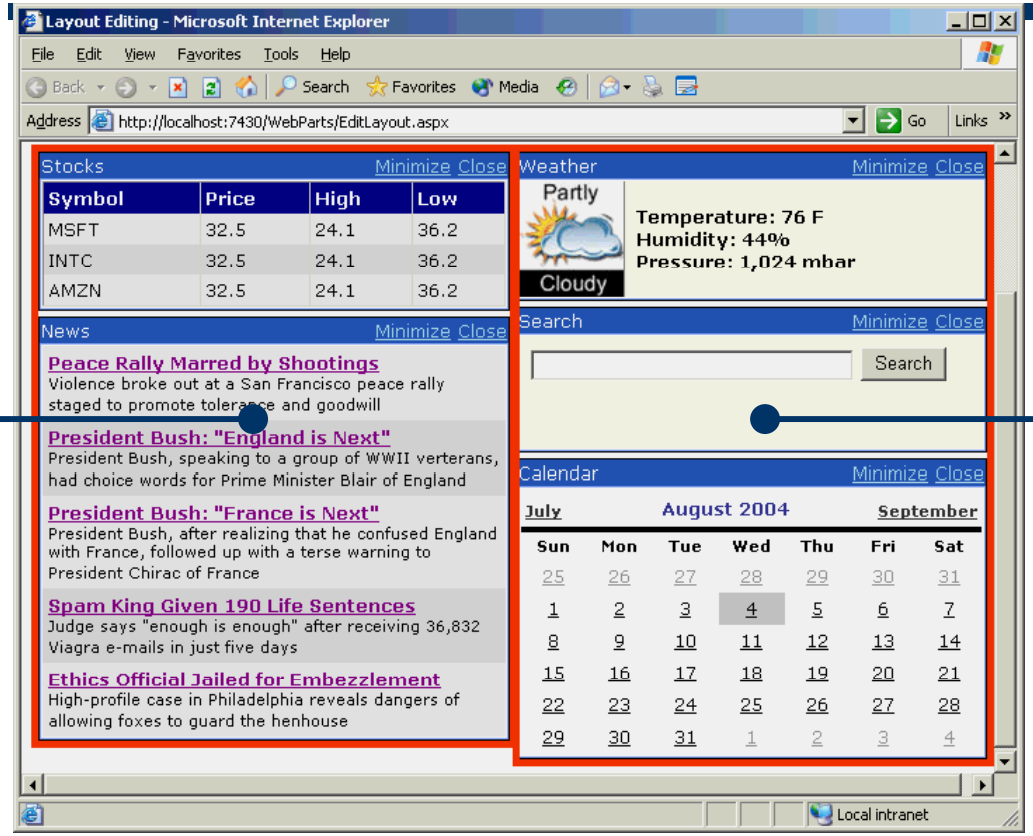
```
<asp:WebPartManager ID="WebPartManager1" RunAt="server" />
```

The WebPartZone Control

- Defines zones on a Web Parts page
- Defines default layout and appearance of Web Parts within each zone

```
<asp:WebPartZone ID="WeatherZone"
  DragHighlightColor="244,198,96" RunAt="server">
  <PartTitleStyle BackColor="#2254B1" ForeColor="White" />
  <PartStyle BorderColor="#81AAF2" BorderStyle="Solid" BorderWidth="1px" />
  <ZoneTemplate>
    <!-- Web Parts declared here -->
  </ZoneTemplate>
</asp:WebPartZone>
```

Web Part Zones



Zone 1

Zone 2

Web Parts

- Controls defined in a WebPartZone
  - Web controls, user controls, custom controls
- Controls that don't implement IWebPart are internally wrapped in GenericWebParts
  - Adds properties: Title, Description, etc.

```
<ZoneTemplate>
  <asp:Calendar Title="Calendar" ID="Calendar1" RunAt="server" />
  <user:Weather Title="Weather" ID="Weather1" RunAt="server" />
  <custom:Search Title="Search" ID="Search1" RunAt="server" />
</ZoneTemplate>
```

Setting up Web Parts

- Adding WebPartsManager
- Adding and laying out zones
- Creating some user controls to use as parts
- Setup of the code to change display mode to allow layout changes

Web Parts

- The WebPartManager control contains the implementation for the display modes that are available in the Web Parts control set, and manages all display mode operations for a page.
  - **BrowseDisplayMode Displays Web Parts controls and UI elements in the normal mode in which end users view a page.**
  - **DesignDisplayMode Displays zone UI and enables users to drag Web Parts controls to change the layout of a page.**
  - **EditDisplayMode Displays editing UI elements and enables end users to edit the controls on a page. Allows dragging of controls.**
  - **CatalogDisplayMode Displays catalog UI elements and enables end users to add and remove page controls. Allows dragging of controls.**
  - **ConnectDisplayMode Displays connections UI elements and enables end users to connect Web Parts controls.**

Content Management Systems (CMS)

Provide a Meta-website to built other websites

Summary of ASP.NET

Summary (MVC Pattern)

- Always remember that you have to define three things for your ASP.NET applications:
- **View**: <asp:button id="b1" onclick="bl_click" runat="server"/>
- **Controller (event handlers)**: b1_click(object sender, EventArgs e){textbox1.text ="hello world";}
- **Model**: DataContext Class based on LINQ to SQL

Summary (Maintain State)

- Remember the following important objects that you can use when implementing your controller class (event handlers):
  - Request, Response
  - Page
  - Server
  - **Session, Application, Cache, ViewState**
  - User, Membership, Roles
  - Context.Profile

Summary (Database Driven Apps)

For Database Driven Apps, always follow the following:

1. Create Membership, role and profile database.
2. Change the web.config file “LocalSqlServer” to point to your database.
3. Create LINQ to SQL model to generate the DataContext class.
4. Add a LinqDataSource to your page and bind it to the DataContext class from Step 3.
5. Add DetailsView or GridView control to the page and bind it to the LinqDataSource from step 4.