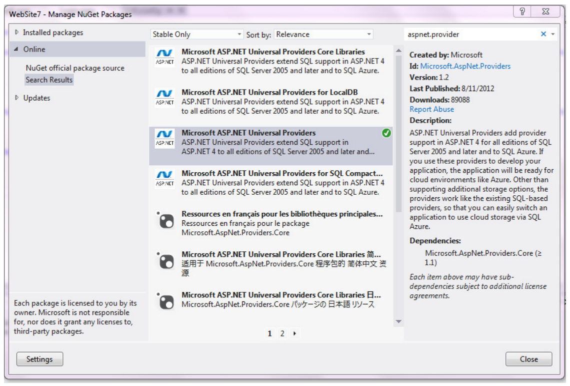
Membership and Role Providers in ASP.NET

Membership and Role Providers

- Membership and role providers exist to provide authentication and authorization services to our applications.
- The provider model in ASP.NET 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 follow the provider pattern by specifying an interface, or contract.

Membership and Role Providers

Use the NuGet Package Manager to add the Membership and Role Providers to your web.config file.



Microsoft ASP.NET Universal Providers Core Libraries
ASP.NET Universal Providers extend SQL support in ASP.NET 4 to all editions of SQL Server 2005 and later and to SQL Azure.

Microsoft ASP.NET Universal Providers for LocalDB
ASP.NET Universal Providers extend SQL support in ASP.NET 4 to all editions of SQL Server 2005 and later and to SQL Azure.

Microsoft ASP.NET Universal Providers ✓
ASP.NET Universal Providers extend SQL support in ASP.NET 4 to all editions of SQL Server 2005 and later and...

Microsoft ASP.NET Universal Providers for SQL Compact...
ASP.NET Universal Providers extend SQL support in ASP.NET 4 to all editions of SQL Server 2005 and later and to SQL Azure.

Ressources en français pour les bibliothèques principales...
Ressources en français pour le package Microsoft.AspNet.Providers.Core

Microsoft ASP.NET Universal Providers Core Libraries 简...
适用于 Microsoft.AspNet.Providers.Core 程序包的 简体中文 资源

Microsoft ASP.NET Universal Providers Core Libraries 日...
Microsoft.AspNet.Providers.Core パッケージの 日本語 リソース

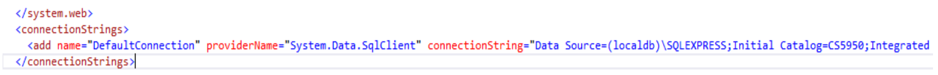
Each package is licensed to you by its owner. Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.

Created by: Microsoft
Id: Microsoft.AspNet.Providers
Version: 1.2
Last Published: 8/11/2012
Downloads: 89088
Report Abuse
Description:
ASP.NET Universal Providers add provider support in ASP.NET 4 for all editions of SQL Server 2005 and later and to SQL Azure. If you use these providers to develop your application, the application will be ready for cloud environments like Azure. Other than supporting additional storage options, the providers work like the existing SQL-based providers, so that you can easily switch an application to use cloud storage via SQL Azure.
Dependencies:
Microsoft.AspNet.Providers.Core (≥ 1.1)

Each item above may have sub-dependencies subject to additional license agreements.

Membership and Role Providers

Update the “DefaultConnection” connectionString in the web.config file to point to your database.



```
</system.web>
<connectionStrings>
  <add name="DefaultConnection" providerName="System.Data.SqlClient" connectionString="Data Source=(localdb)\SQLEXPRESS;Initial Catalog=CS5950;Integrated
</connectionStrings>
```

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.

Use the ASP.NET Sql Server Registration Tool (aspnet_regsql.exe) to create a new "aspnetdb" database.

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

}
```

Roles & Memberships

Login Controls

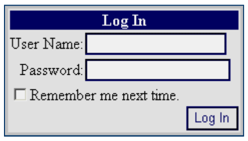
Name	Description
ChangePassword	UI for changing passwords
CreateUserWizard	UI for creating new user accounts
Login	UI for entering and validating user names and passwords
LoginName	Displays authenticated user names
LoginStatus	UI for logging in and logging out
LoginView	Displays different views based on login status and roles
PasswordRecovery	UI for recovering forgotten passwords

The Login Control

- Standard UI for logging in users
- Integrates with Membership service
  - Calls ValidateUser automatically
  - No-code validation and logins
- Also works without Membership service
  (to install this service use aspnet_regsql)
- Incorporates RequiredFieldValidators
- Highly customizable UI and behavior

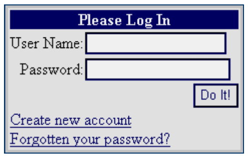
Using the Login Control

```
<html>
  <body>
    <form runat="server">
      <asp:Login RunAt="server" />
    </form>
  </body>
</html>
```



Customizing the Login Control

```
<asp:Login ID="LoginControl" RunAt="server"
  CreateUserText="Create new account"
  CreateUserUrl="CreateUser.aspx"
  DisplayRememberMe="false"
  PasswordRecoveryText="Forgotten your password?"
  PasswordRecoveryUrl="RecoverPassword.aspx"
  LoginButtonText="Do It!"
  TitleText="Please Log In"
/>
```



Login Control Events

Name	Description
LoggingIn	Fired when the user clicks the Log In button. Purpose: to Prevalidate login credentials (e.g., make sure e-mail address is well-formed)
Authenticate	Fired when the user clicks the Log In button. Purpose: to Authenticate the user by validating his or her login credentials
LoggedIn	Fired following a successful login
LoginError	Fired when an attempted login fails

Validating Credential Formats

```
<asp:Login ID="LoginControl" RunAt="server"
  OnLoggingIn="OnValidateCredentials" ... />
    .
    .
    .
<script language="C#" runat="server">
void OnValidateCredentials (Object sender, CancelEventArgs e)
{
    if (!Regex.IsMatch (LoginControl.UserName, "[a-zA-Z0-9]{6,}") ||
        !Regex.IsMatch (LoginControl.Password, "[a-zA-Z0-9]{8,}")) {
        LoginControl.InstructionText = "User names and passwords " +
            "must contain letters and numbers only and must be at " +
            "least 6 and 8 characters long, respectively";
        e.Cancel = true;
    }
}
</script>
```

The LoginView Control

- Displays content differently to different users depending on:
  - Whether user is authenticated
  - If user is authenticated, the role memberships he or she is assigned
- Template-driven
  - <AnonymousTemplate>
  - <LoggedInTemplate>
  - <RoleGroups> and <ContentTemplate>

Using LoginView

```
<asp:LoginView ID="LoginView1" Runat="server">
  <AnonymousTemplate>
    <!-- Content seen by unauthenticated users -->
  </AnonymousTemplate>
  <LoggedInTemplate>
    <!-- Content seen by authenticated users -->
  </LoggedInTemplate>
  <RoleGroups>
    <asp:RoleGroup Roles="Administrators">
      <ContentTemplate>
        <!-- Content seen by authenticated users who are
administrators -->
      </ContentTemplate>
    </asp:RoleGroup>
      ...
  </RoleGroups>
</asp:LoginView>
```

The LoginName Control

- Displays authenticated user names
- Use optional FormatString property to control format of output

```
<asp:LoginView ID="LoginView1" Runat="server">
  <AnonymousTemplate>
    You are not logged in
  </AnonymousTemplate>
  <LoggedInTemplate>
    <asp:LoginName ID="LoginName1" Runat="server"
      FormatString="You are logged in as {0}" />
  </LoggedInTemplate>
</asp:LoginView>
```

The LoginStatus Control

- Displays links for logging in and out
  - "Login" to unauthenticated users
  - "Logout" to authenticated users
- UI and logout behavior are customizable

```
<asp:LoginStatus ID="LoginStatus1" Runat="server"
  LogoutAction="Redirect" LogoutPageUrl="~/Default.aspx" />
```

LoginStatus Properties

Name	Description
LognText	Text displayed for login link (default="Login")
LogoutText	Text displayed for logout link (default="Logout")
LoginImageUrl	URL of image used for login link
LogoutAction	Action to take following logout: Redirect, RedirectToLoginPage, or Refresh (default)
LogOutPageUrl	URL of page to go to following logout if LogoutAction="Redirect"

Enabling the Role Manager

- Role manager is disabled by default
- Enable it via Web.config:

```
<configuration>
  <system.web>
    <roleManager enabled="true" />
  </system.web>
</configuration>
```

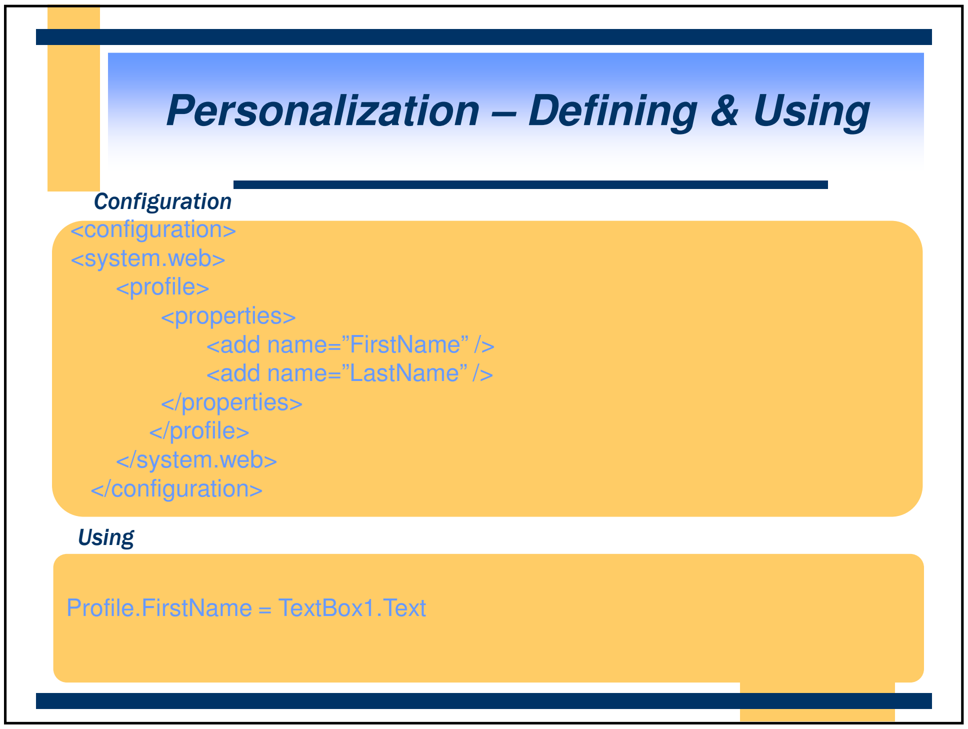
Set the Unobtrusive Validation Mode to None in Web.config

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

Personalization

Personalization - Overview

- Automatic association between the end user viewing the page and any data points stored for that user.
- The personalization properties that are maintained on a per-user basis are stored on the server and not on the client.
- The end user can access these personalization properties on later site visits.
- Ideal way to start creating highly customizable and user-specific sites without messing with all the underlined code.



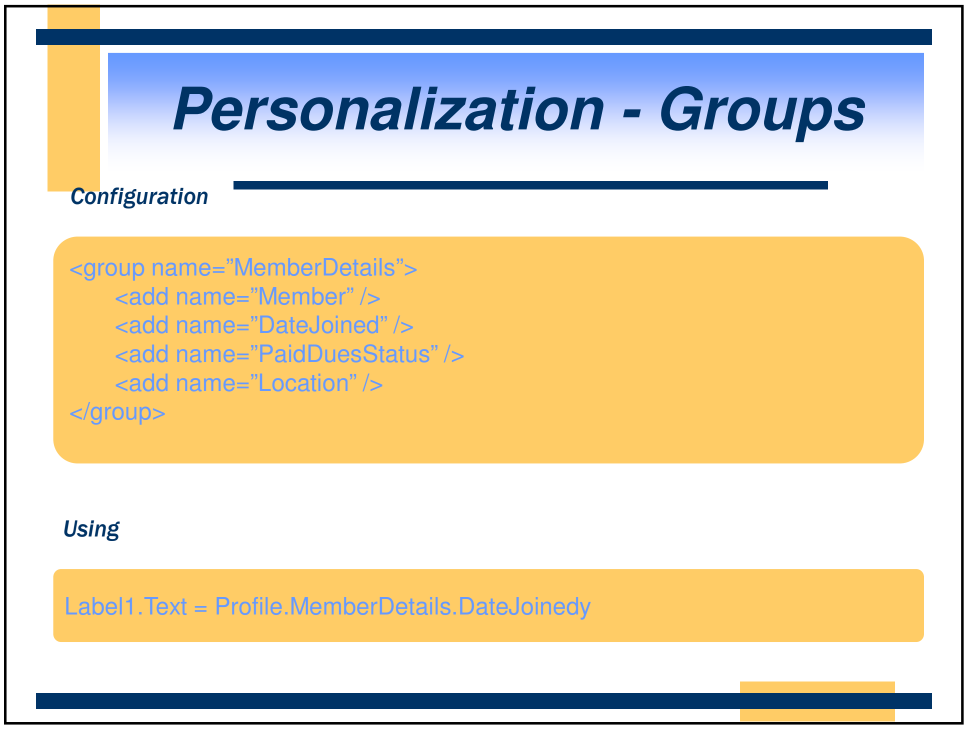
Personalization – Defining & Using

Configuration

```
<configuration>
<system.web>
    <profile>
        <properties>
            <add name=”FirstName” />
            <add name=”LastName” />
        </properties>
      </profile>
    </system.web>
  </configuration>
```

Using

```
Profile.FirstName = TextBox1.Text
```



Personalization - Groups

Configuration

```
<group name=”MemberDetails”>
    <add name=”Member” />
    <add name=”DateJoined” />
    <add name=”PaidDuesStatus” />
    <add name=”Location” />
</group>
```

Using

```
Label1.Text = Profile.MemberDetails.DateJoinedy
```

Personalization - Types

- Define types to the fields
- Use default values to the fields
- Define readonly for fields

```
<add name=”FieldName” type=”FieldType” />
```

Anonymous User Profiles

- By default, profiles aren’t available for anonymous (unauthenticated) users
  - Data keyed by authenticated user IDs
- Anonymous profiles can be enabled
  - Step 1: Enable anonymous identification
  - Step 2: Specify which profile properties are available to anonymous users
- Data keyed by user anonymous IDs

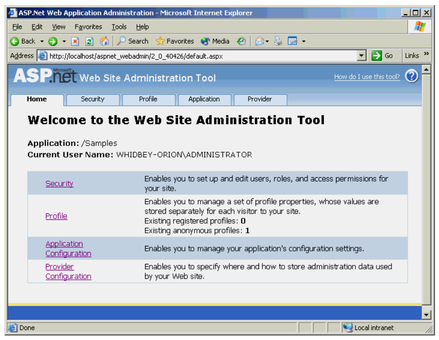
Profiles for Anonymous Users

```
<configuration>
  <system.web>
    <anonymousIdentification enabled="true" />
    <profile>
      <properties>
        <add name="ScreenName" allowAnonymous="true" />
        <add name="Posts" type="System.Int32" defaultValue="0 />
        <add name="LastPost" type="System.DateTime" />
      </properties>
    </profile>
  </system.web>
</configuration>
```

Administration & Management

A&M – Web Site Administration Tool (WAT)

- Browser-based admin GUI


ASP.Net Web Application Administration - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/aspnet_webadmin/2_0_40426/default.aspx
Go Links
ASP.net Web Site Administration Tool
How do I use this tool?
Home Security Profile Application Provider
Done
Local intranet

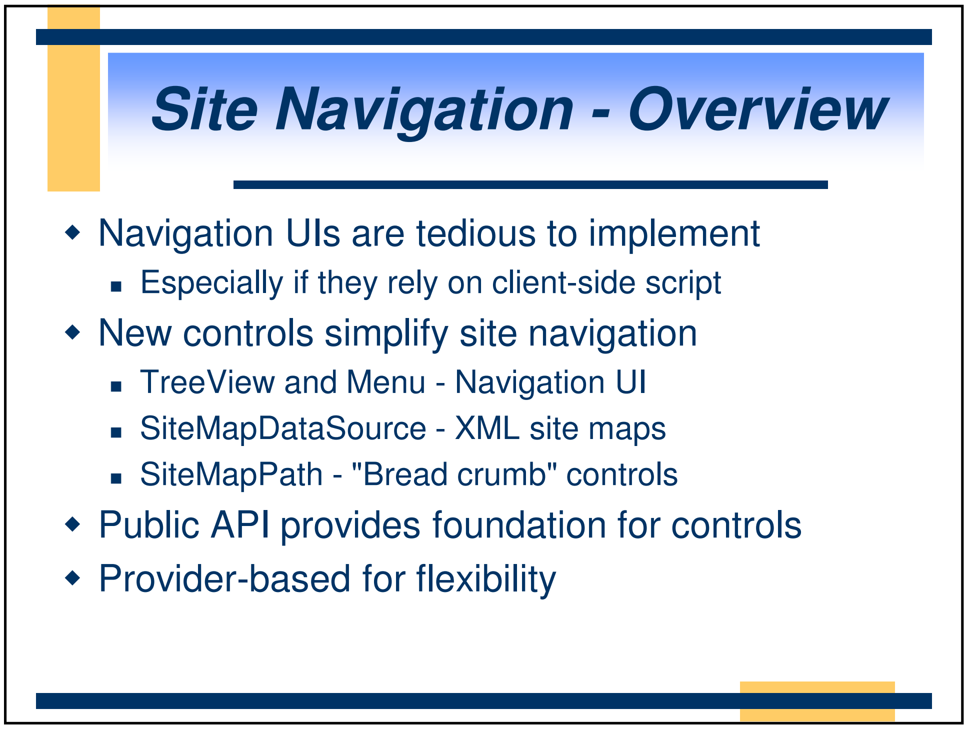
Welcome to the Web Site Administration Tool

Application: /Samples
Current User Name: WHIDBEY-ORION\ADMINISTRATOR

Security	Enables you to set up and edit users, roles, and access permissions for your site.
Profile	Enables you to manage a set of profile properties, whose values are stored separately for each visitor to your site. Existing registered profiles: **0** Existing anonymous profiles: **1**
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.

Invoked by requesting Webadmin.axd or using the "ASP.NET Configuration" command in Visual Studio's Website menu

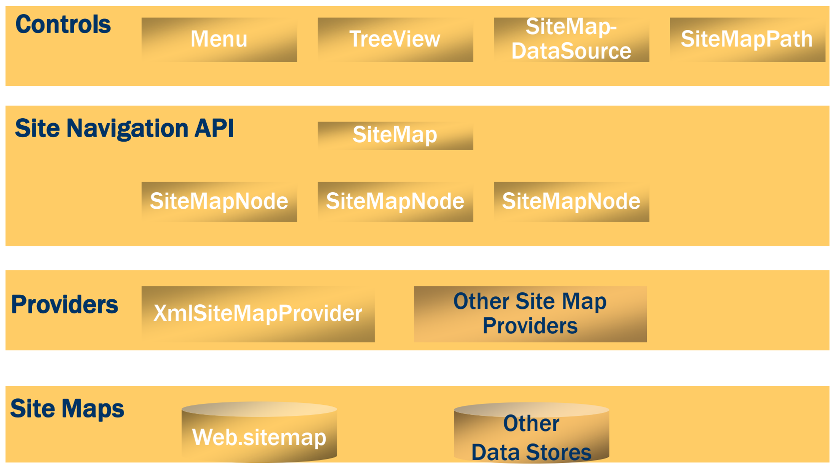
Site Navigation

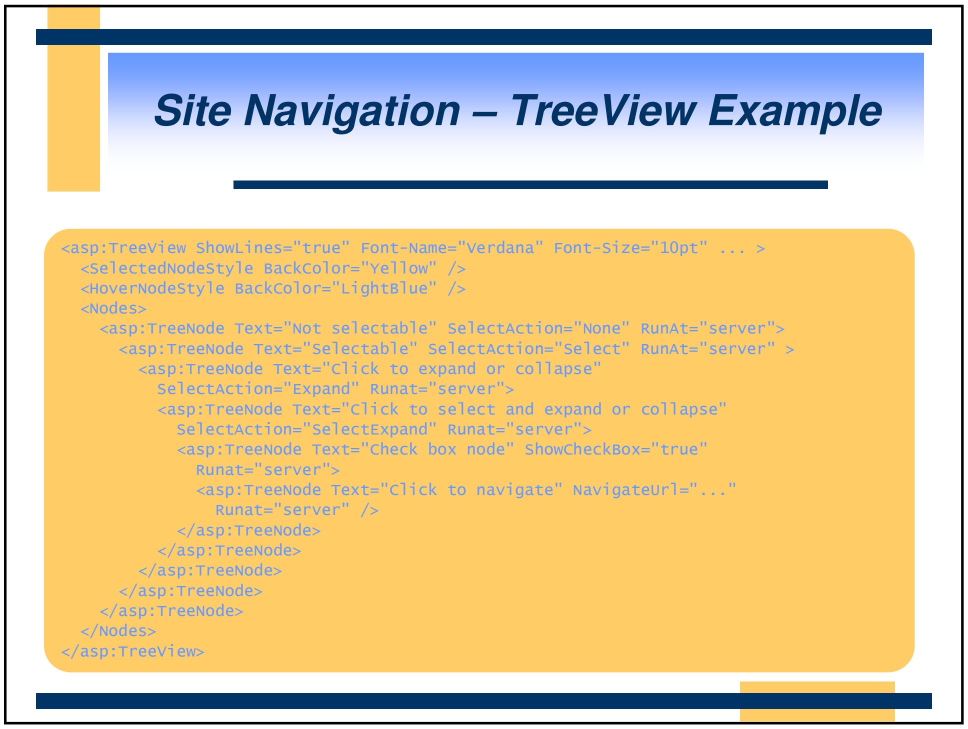


Site Navigation - Overview

- Navigation UIs are tedious to implement
  - Especially if they rely on client-side script
- New controls simplify site navigation
  - TreeView and Menu - Navigation UI
  - SiteMapDataSource - XML site maps
  - SiteMapPath - "Bread crumb" controls
- Public API provides foundation for controls
- Provider-based for flexibility

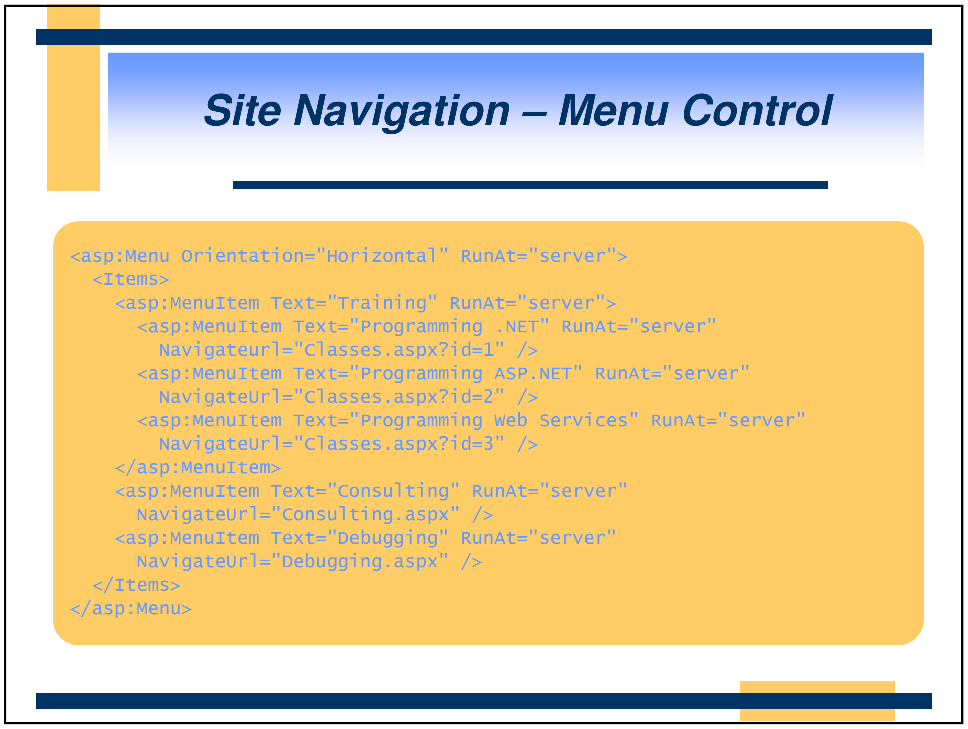
Site Navigation - Schema





Site Navigation – TreeView Example

```
<asp:TreeView ShowLines="true" Font-Name="Verdana" Font-Size="10pt" ... >
  <SelectedNodeStyle BackColor="Yellow" />
  <HoverNodeStyle BackColor="LightBlue" />
  <Nodes>
    <asp:TreeNode Text="Not selectable" SelectAction="None" RunAt="server">
      <asp:TreeNode Text="Selectable" SelectAction="Select" RunAt="server" >
        <asp:TreeNode Text="Click to expand or collapse"
          SelectAction="Expand" Runat="server">
          <asp:TreeNode Text="Click to select and expand or collapse"
            SelectAction="SelectExpand" Runat="server">
            <asp:TreeNode Text="Check box node" ShowCheckBox="true"
              Runat="server">
              <asp:TreeNode Text="Click to navigate" NavigateUrl="..."
                Runat="server" />
            </asp:TreeNode>
          </asp:TreeNode>
        </asp:TreeNode>
      </asp:TreeNode>
    </asp:TreeNode>
  </Nodes>
</asp:TreeView>
```



Site Navigation – Menu Control

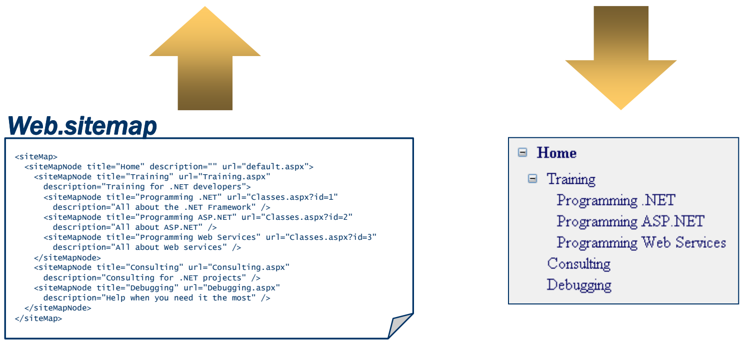
```
<asp:Menu Orientation="Horizontal" RunAt="server">
  <Items>
    <asp:MenuItem Text="Training" RunAt="server">
      <asp:MenuItem Text="Programming .NET" RunAt="server"
        Navigateurl="Classes.aspx?id=1" />
      <asp:MenuItem Text="Programming ASP.NET" RunAt="server"
        NavigateUrl="Classes.aspx?id=2" />
      <asp:MenuItem Text="Programming Web Services" RunAt="server"
        NavigateUrl="Classes.aspx?id=3" />
    </asp:MenuItem>
    <asp:MenuItem Text="Consulting" RunAt="server"
      NavigateUrl="Consulting.aspx" />
    <asp:MenuItem Text="Debugging" RunAt="server"
      NavigateUrl="Debugging.aspx" />
  </Items>
</asp:Menu>
```

Site Navagation - SiteMap

```
<siteMap>
  <siteMapNode title="Home" description="" url="default.aspx">
    <siteMapNode title="Training" url="Training.aspx"
      description="Training for .NET developers">
      <siteMapNode title="Programming .NET" url="Classes.aspx?id=1"
        description="All about the .NET Framework" />
      <siteMapNode title="Programming ASP.NET" url="Classes.aspx?id=2"
        description="All about ASP.NET" />
      <siteMapNode title="Programming Web Services" url="Classes.aspx?id=3"
        description="All about Web services" />
    </siteMapNode>
    <siteMapNode title="Consulting" url="Consulting.aspx"
      description="Consulting for .NET projects" />
    <siteMapNode title="Debugging" url="Debugging.aspx"
      description="Help when you need it the most" />
  </siteMapNode>
</siteMap>
```

Site Navigation – TreeView and SiteMap

```
<asp:SiteMapDataSource ID="SiteMap" RunAt="server" />
<asp:TreeView DataSourceID="SiteMap" RunAt="server" />
```



Web.sitemap

```
<siteMap>
  <siteMapNode title="Home" description="" url="default.aspx">
    <siteMapNode title="Training" url="Training.aspx"
      description="Training for .NET developers">
      <siteMapNode title="Programming .NET" url="Classes.aspx?id=1"
        description="All about the .NET Framework" />
      <siteMapNode title="Programming ASP.NET" url="Classes.aspx?id=2"
        description="All about ASP.NET" />
      <siteMapNode title="Programming Web Services" url="Classes.aspx?id=3"
        description="All about Web services" />
    </siteMapNode>
    <siteMapNode title="Consulting" url="Consulting.aspx"
      description="Consulting for .NET projects" />
    <siteMapNode title="Debugging" url="Debugging.aspx"
      description="Help when you need it the most" />
  </siteMapNode>
</siteMap>
```

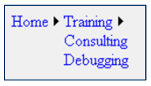
```
⊟ Home
  ⊟ Training
      Programming .NET
      Programming ASP.NET
      Programming Web Services
    Consulting
    Debugging
```

Site Navigation – Menu and SiteMap

```
<asp:SiteMapDataSource ID="SiteMap" RunAt="server" />
<asp:Menu DataSourceID="SiteMap" RunAt="server" />
```

Web.sitemap

```
<siteMap>
  <siteMapNode title="Home" description="" url="default.aspx">
    <siteMapNode title="Training" url="Training.aspx"
      description="Training for .NET developers">
      <siteMapNode title="Programming .NET" url="Classes.aspx?id=1"
        description="All about the .NET Framework" />
      <siteMapNode title="Programming ASP.NET" url="Classes.aspx?id=2"
        description="All about ASP.NET" />
      <siteMapNode title="Programming Web Services" url="Classes.aspx?id=3"
        description="All about Web services" />
    </siteMapNode>
    <siteMapNode title="Consulting" url="Consulting.aspx"
      description="Consulting for .NET projects" />
    <siteMapNode title="Debugging" url="Debugging.aspx"
      description="Help when you need it the most" />
  </siteMapNode>
</siteMap>
```



ASP.NET Data Binding

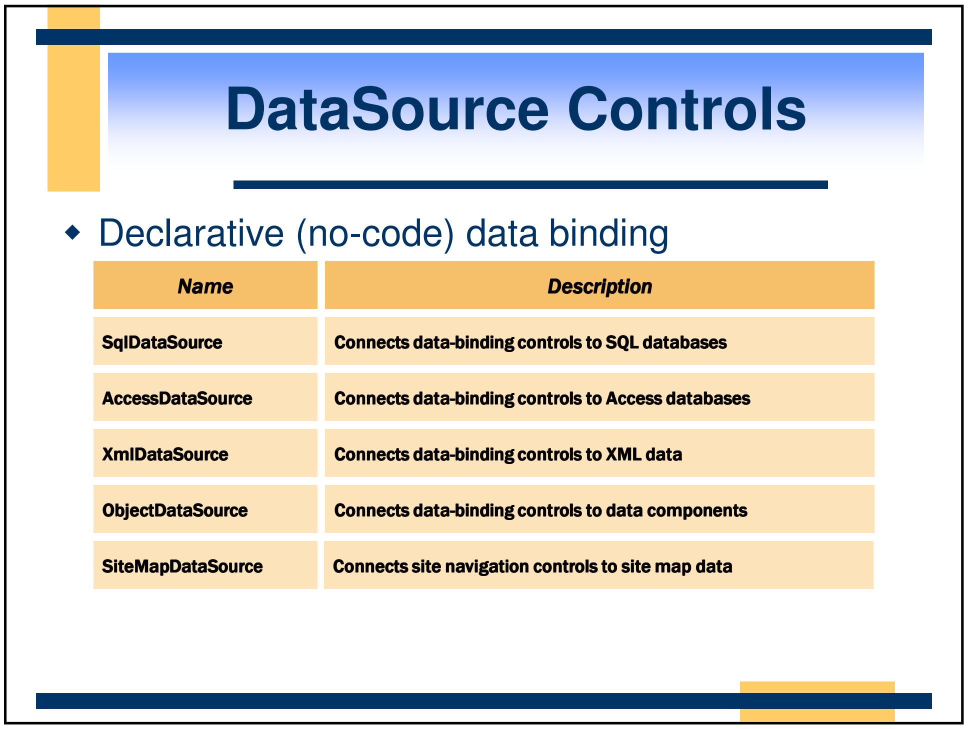
ASP.NET Data Binding

- Data source controls
- Data controls
  - GridView, DetailsView, ListView controls
  - Editing with GridView, DetailsView, ListView

Simplified Data Binding

- Data binding expressions are now simpler

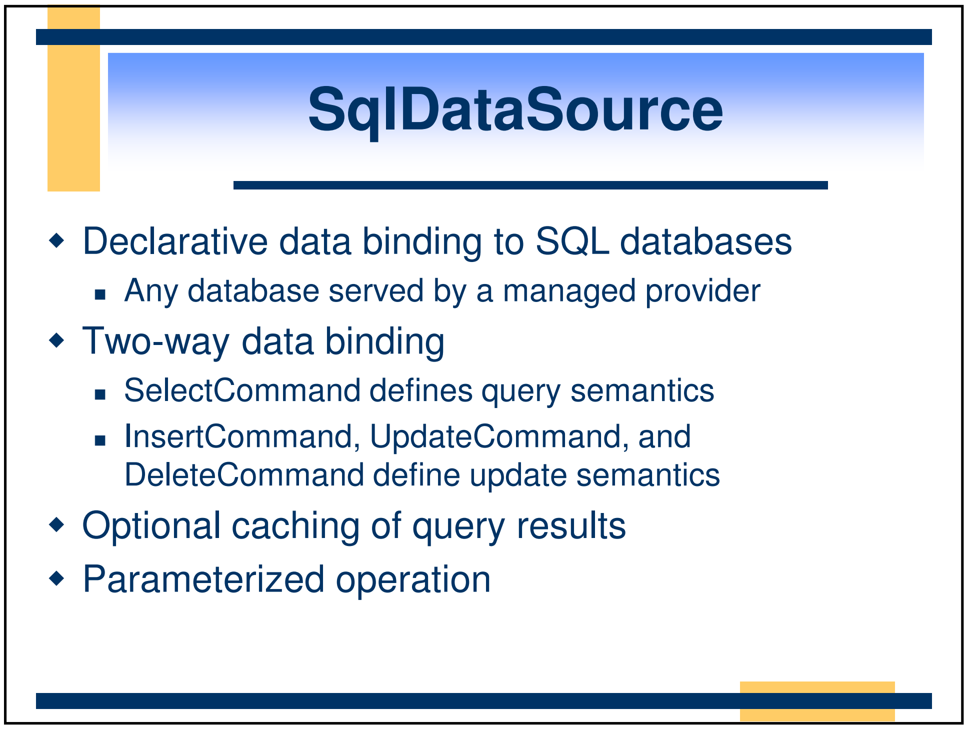
```
<!-- ASP.NET 1.x data binding expression -->
<%# DataBinder.Eval (Container.DataItem, "Price") %>

<!-- 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

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 “DefaultConnection” 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 or ListView control to the page and bind it to the LinqDataSource from step 4.