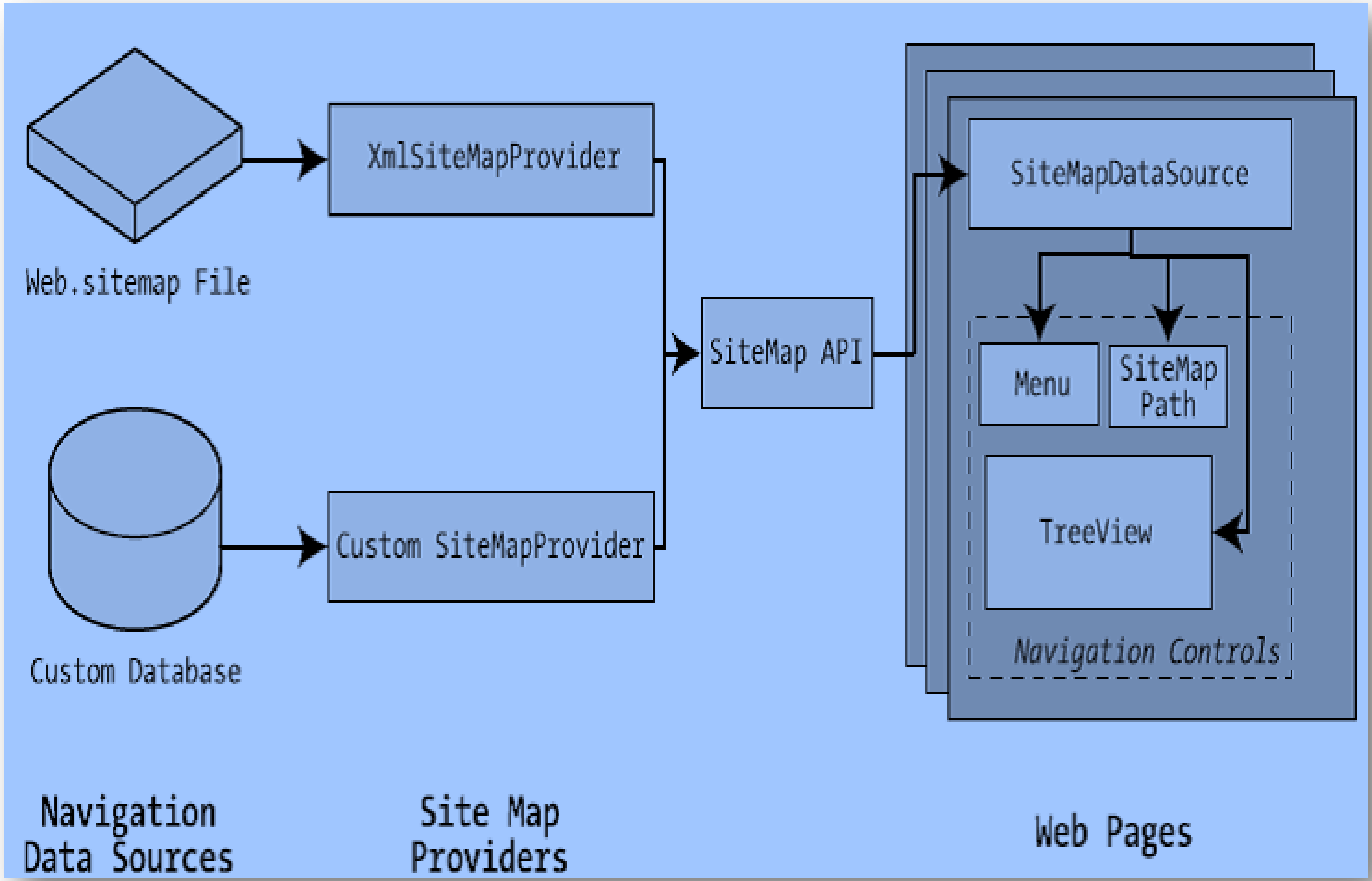
Site Navigation

Site Map

- ASP.NET 2.0 include the site navigation system with the capability to define your entire site in an XML file that is called a *site map*.

- It also has a new series of navigation-based server controls that bind site map files.


Web.sitemap File
XmlSiteMapProvider
Custom Database
Custom SiteMapProvider
SiteMap API
SiteMapDataSource
Menu
SiteMap Path
TreeView
Navigation Controls
Navigation Data Sources
Site Map Providers
Web Pages

XmlSiteMapProvider

- The *XmlSiteMapProvider* looks for a file named **Web.sitemap** in the root of the virtual directory.

- It creates SiteMap object and makes it available to *SiteMapDataSource*.

- *SiteMapDataSource* then should be bound to navigation server control.

Site Map

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="default.aspx" title="Home">
    <siteMapNode title="About iConnect“ url=“abouticonnect.aspx”>
      <siteMapNode url="about.aspx" title="About Us"/>
      <siteMapNode url="management.aspx" title="Management" />
    </siteMapNode>
    <siteMapNode url="software.aspx" title="iConnect Software“
                  description="Software Details" />
  </siteMapNode>
```

siteMapNode Element

- Each Page Is Represented by a <siteMapNode> Element.
- <siteMapNode> Element Can Contain Other <siteMapNode> Elements.
- Every Site Map has at least one <siteMapNode>.
- One cannot create two site map nodes with the same URL.

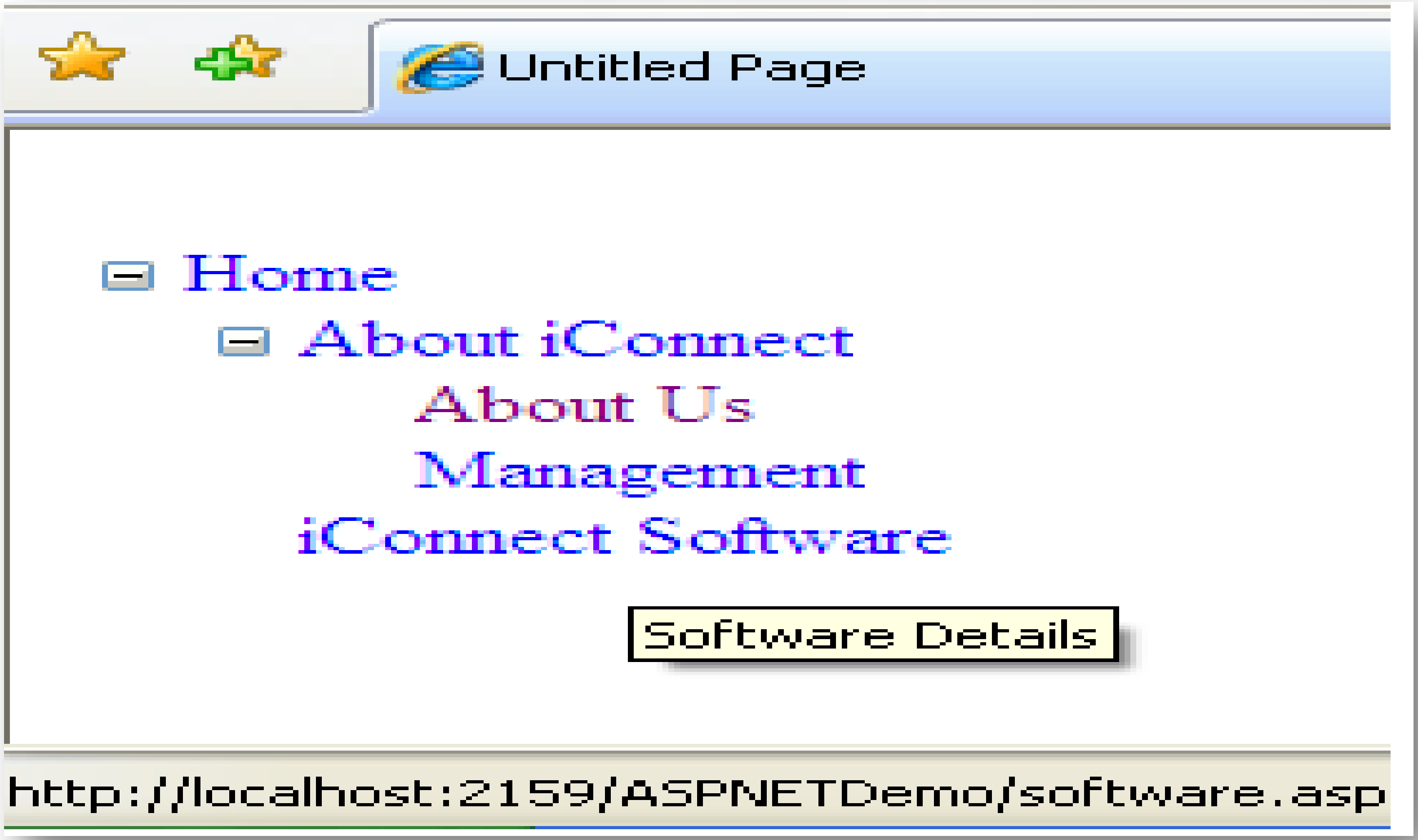
DataSourceID

- All navigation control has the property.
- It binds the site map to the navigation control.

```
<asp:TreeView ID="TreeView1" runat="server“
            DataSourceID="SiteMapDataSource1"> </asp:TreeView>

<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server" />
```

Sitemap in Action



Untitled Page

⊟ Home
  ⊟ About iConnect
      About Us
      Management
  iConnect Software

Software Details

http://localhost:2159/ASPNETDemo/software.asp

SiteMapDataSource Control

- SiteMapDataSource, by default, shows a full tree that starts with the root node.

- SiteMapDataSource Properties lets to configure the navigation tree.

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server“ />
```

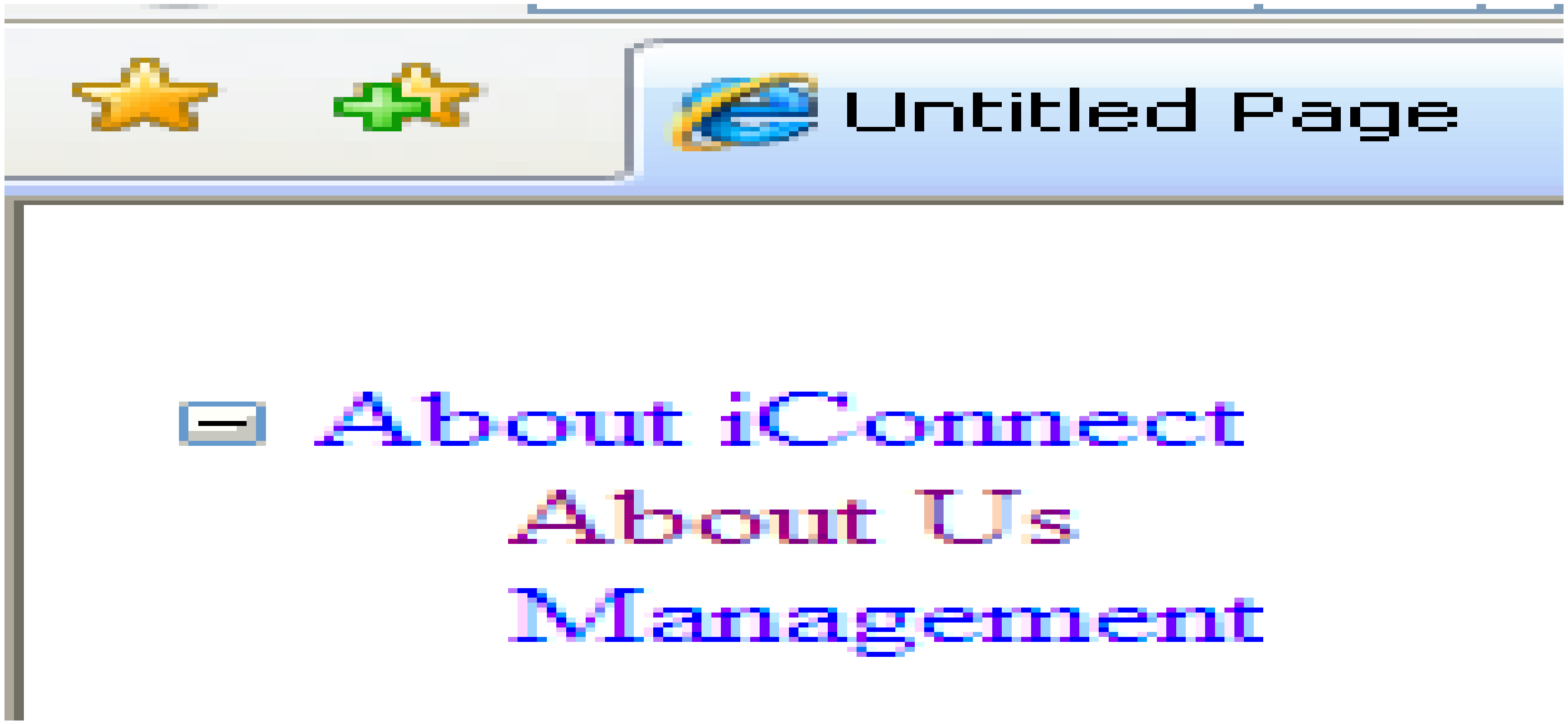
SiteMapDataSource Properties

- ShowStartingNode
  - If false, hides root node. Default is true.
- StartingNodeUrl
  - Set this value to the URL of the node that should be the first node in the navigation tree.

```
<asp:SiteMapDataSource ID="SiteMapDataSource1" runat="server“
StartingNodeUrl=“abouticonnect.aspx” />
```

SiteMapDataSource Properties

- StartingFromCurrentNode
  - If true, sets the current page as the starting node.
- StartingNodeOffset
  - Use this property to shift the starting node up or down the hierarchy. It takes an integer that instructs the SiteMapDataSource to move from the starting node down the tree (if the number is positive) or up the tree (if the number is negative).


Untitled Page

⊟ About iConnect

About Us

Management

TreeView Control

- Every element or entry in the TreeView control is called a *node*.
- The uppermost node in the hierarchy of nodes is the *root node*. It is possible for a TreeView control to have multiple root nodes.
- Each parent node can have one or more child nodes.

TreeView Control

- The TreeView server control is a rich server control for rendering a of data.
- It is used to display hierarchical data.
- A TreeView control is made up of TreeNode objects.
- One can build a TreeView control by declaring TreeNode objects in the TreeView control's Items collection.

Adding TreeNodes

```
<asp:TreeView ID="TreeView1" runat="server">
          <Nodes>
             <asp:TreeNode NavigateUrl="./Default.aspx" Text="Home" >
                <asp:TreeNode NavigateUrl="~/abouticonnect.aspx"
Text="About iConnect“ >
                   <asp:TreeNode NavigateUrl="~/about.aspx" Text="About Us” />
                   <asp:TreeNode NavigateUrl="./management.aspx“
Text="Management“ />
                </asp:TreeNode>
                <asp:TreeNode NavigateUrl="./software.aspx“
Text="iConnect Software“ />
                </asp:TreeNode>
             </Nodes>
      </asp:TreeView>
```

TreeView With CheckBoxes

- One can display check boxes next to each node in a TreeView control by assigning a value to the *ShowCheckBoxes* property.

- One can set *ShowCheckBoxes* to
  - All, Leaf, None
  - Parent, Root.

- Bitwise combination of these values can be used.

```
<asp:TreeView ID="TreeView1" runat="server" ShowCheckBoxes="Leaf">
…
</asp:TreeView>
```

```
TreeView1.ShowCheckBoxes = TreeNodeTypes.Leaf;
```

Accessing Checked Nodes

```
protected void Button1_Click(object sender, System.EventArgs e)
{
        if (TreeView1.CheckedNodes.Count > 0)
        {
                Label1.Text = “We are sending you information on:<p>”;
                foreach (TreeNode node in TreeView1.CheckedNodes)
                {
                Label1.Text += node.Text + “ “ + node.Parent.Text + “<br>”;
                }
        }
        else
        {
        Label1.Text = “You didn’t select anything. Sorry!”;
        }
}
```

Binding TreeView to XML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<movies>
  <action>
    <StarWars />
    <IndependenceDay />
  </action>
  <horror>
    <Jaws />
    <NightmareBeforeChristmas />
  </horror>
</movies>
```

Trial.xml

Binding TreeView to XML

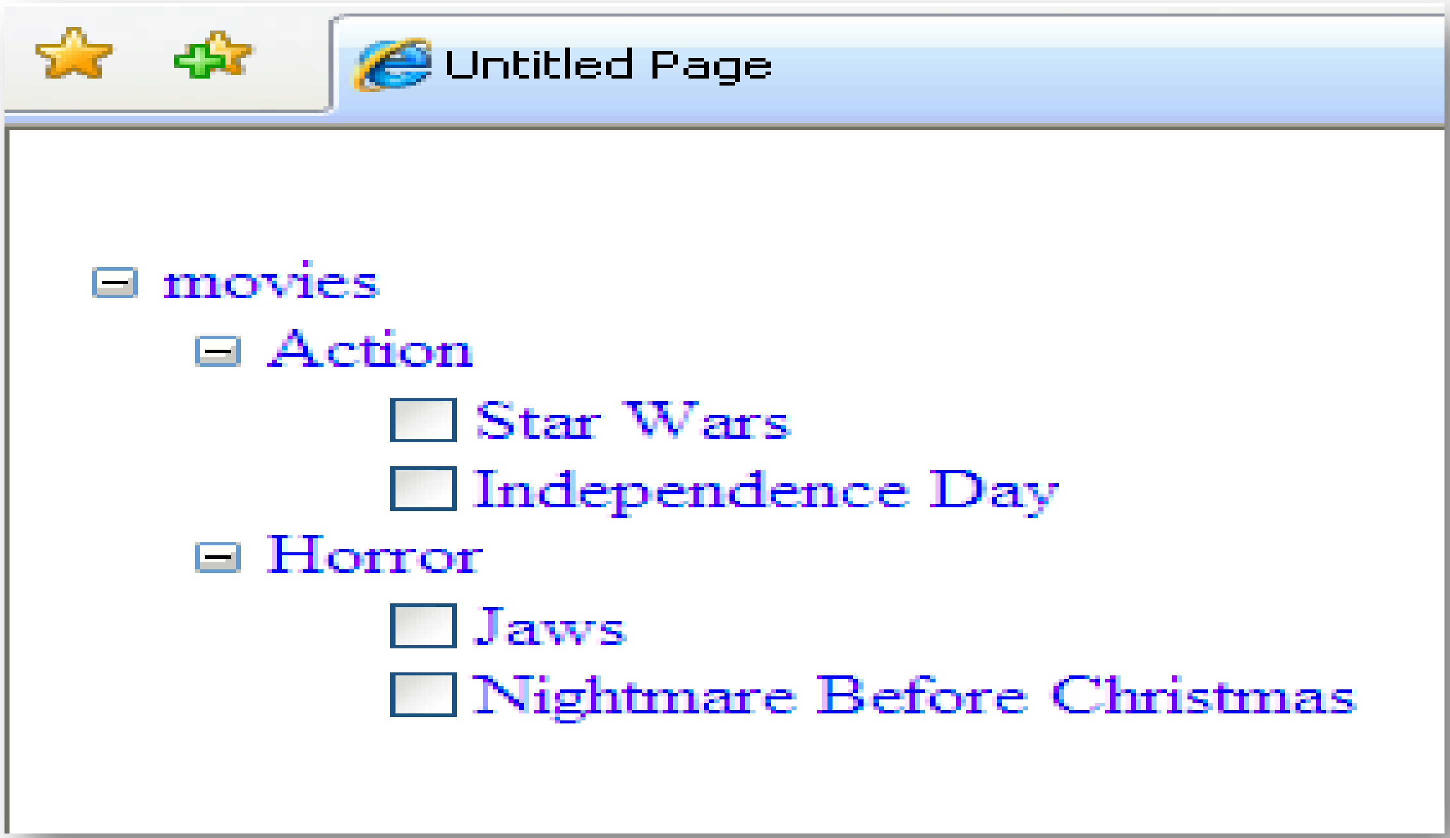
```
<asp:TreeView ID="TreeView1" runat="server" ShowCheckBoxes="Leaf“
                        DataSourceID="XmlDataSource1">
      </asp:TreeView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server“
            DataFile="~/Trial.xml">
</asp:XmlDataSource>
```

TreeView with More Complex XML

```xml
<?xml version="1.0" encoding="utf-8" ?>
<movies>
  <category id="category1" text="Action">
    <movie id="movie1" text="Star Wars" />
    <movie id="movie2" text="Independence Day" />
  </category>
  <category id="category2" text="Horror">
    <movie id="movie3" text="Jaws" />
    <movie id="movie4" text="Nightmare Before Christmas" />
  </category>
</movies>
```

TreeView with More Complex XML

```
<asp:TreeView ID="TreeView1" runat="server" ShowCheckBoxes="Leaf“
                DataSourceID="XmlDataSource1">
           <DataBindings>
               <asp:TreeNodeBinding DataMember="category" TextField="text“
                ValueField="id" />
               <asp:TreeNodeBinding DataMember="movie" TextField="text“
                ValueField="id" />
           </DataBindings>
       </asp:TreeView>
```



Untitled Page

- ⊟ movies
  - ⊟ Action
    - ☐ Star Wars
    - ☐ Independence Day
  - ⊟ Horror
    - ☐ Jaws
    - ☐ Nightmare Before Christmas

Formatting TreeView

- CollapseImageToolTip
  - Specify the title attribute for the collapse image
  - .
- CollapImageUrl
  - Applies a custom image next to nodes that have been expanded to show any of their child nodes
- ExpandDepth
  - Specify the number of tree node levels to display initially.
- ExpandImageToolTip
  - Specify the title attribute for the expand image.

Formatting TreeView

- ExpandImageUrl
  - Applies a custom image next to nodes that have the capability of being expanded to display their child nodes.
- ImageSet
  - Specifies a set of images to use with the TreeView control.
- LineImagesFolder
  - Enables one to specify a folder that contains line images.

Formatting TreeView

- MaxDataBindDepth
  - Determines the maximum levels of nodes to display when binding to a data source.
- NodeIndent
  - Specifies the number of pixels to indent a child Tree node.
- NodeWrap
  - Tells whether or not text is wrapped in a Tree node.

Formatting TreeView

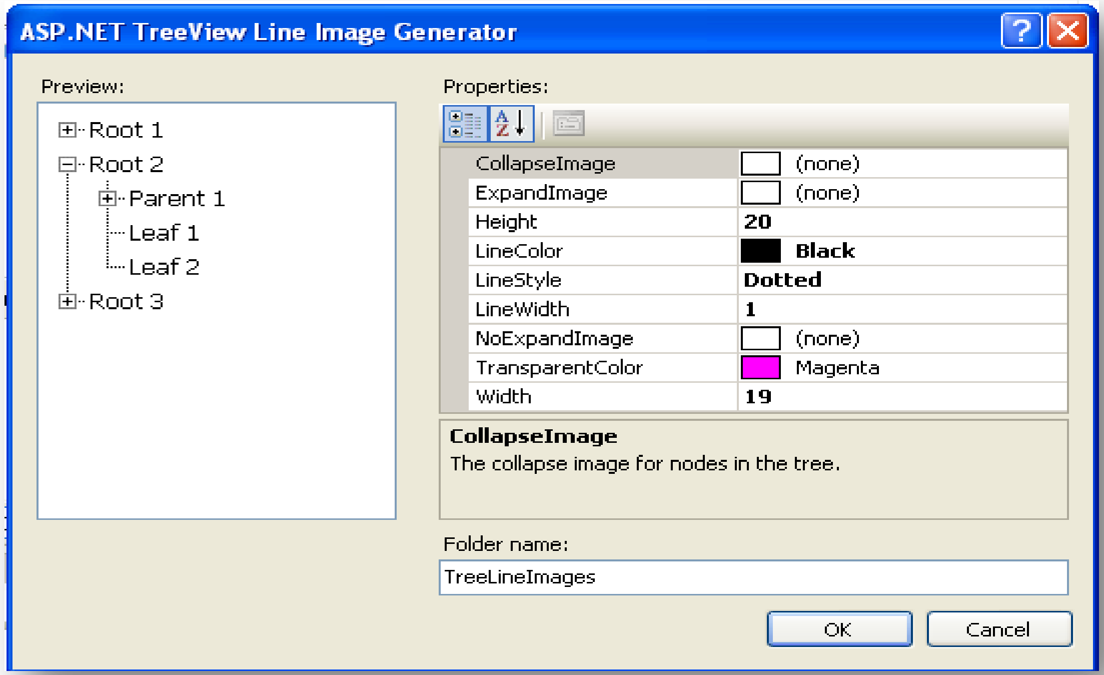
- NoExpandImageUrl
  - Applies a custom image to nodes that, for programmatic reasons, cannot be expanded or to nodes that are leaf nodes.
- ShowExpandCollapse
  - Enables one to disable the expand and collapse icons that appear next to each expandable node.

Formatting TreeView

- ShowLines
  - Enable one to show connecting lines between Tree nodes.
- Target
  - Specifies the name of the window that opens when you navigate to a URL with the TreeView control.
- RootNodeImageUrl/ParentNodeImageUrl
  - Applies a custom image next to only the root/parent nodes.

Formatting TreeView

- TreeView smart tag if clicked and Show Lines option is checked, it has one menu option to customize line images.
- It presents with Line Generator.



CollapseImage	(none)
ExpandImage	(none)
Height	20
LineColor	■ Black
LineStyle	Dotted
LineWidth	1
NoExpandImage	(none)
TransparentColor	Magenta
Width	19

CollapseImage
The collapse image for nodes in the tree.

Folder name: TreeLineImages

Formatting TreeView

- One can style the TreeView control by attaching Cascading Style Sheet classes to the Style objects of TreeView.

- Style Objects:

  - HoverNodeStyle
  - LeafNodeStyle
  - NodeStyle
  - ParentNodeStyle
  - RootNodeStyle
  - SelectedNodeStyle

```
<asp:TreeView id="TreeView1" NodeStyle-CssClass="treeNode"
            RootNodeStyle-CssClass="rootNode"
            LeafNodeStyle-CssClass="leafNode" Runat="server">
```

Programming TreeView

```
TreeView1.ExpandAll();
```

```
TreeView1.CollapseAll();
```

Expand Specific node.

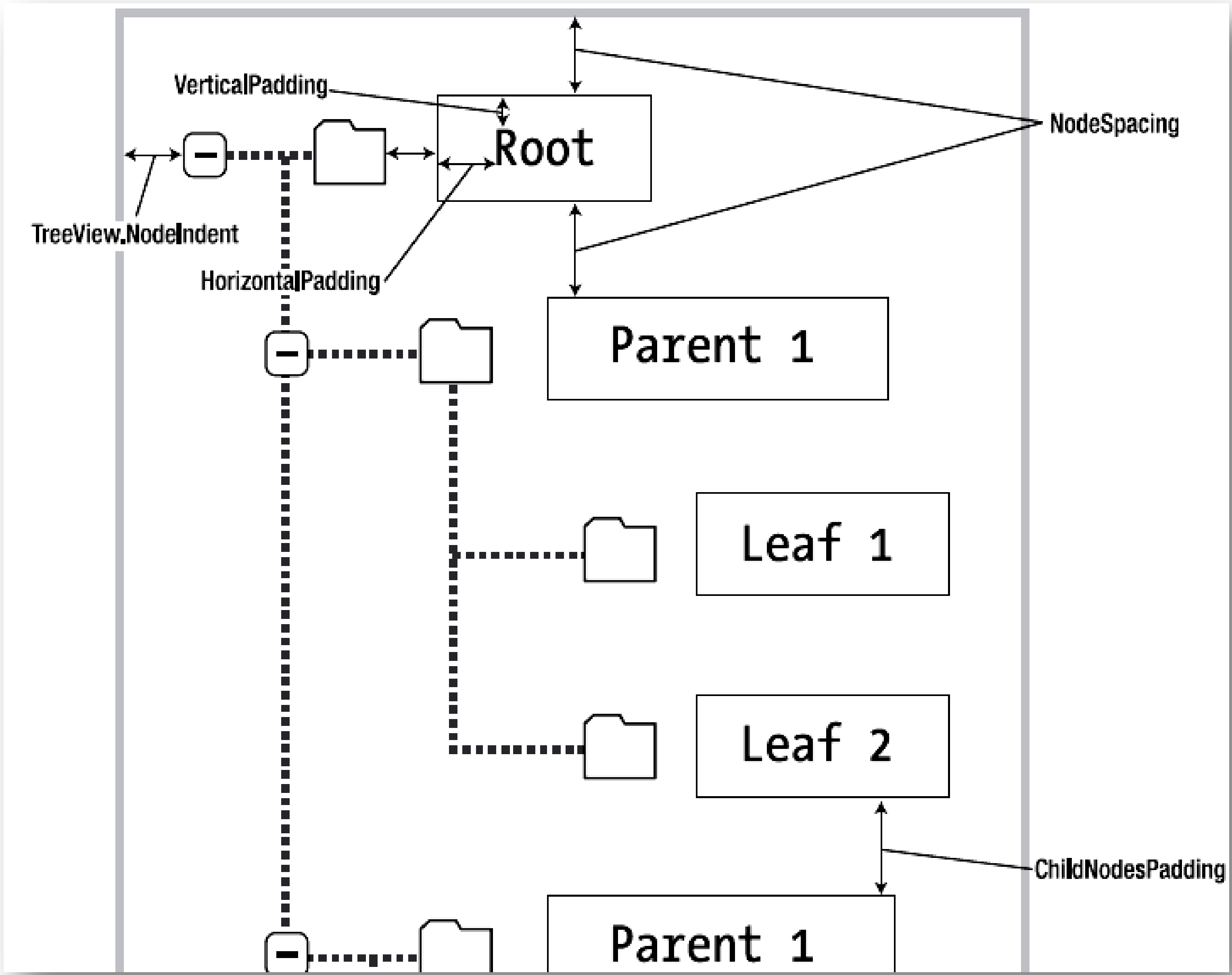
```
TreeNode nd=TreeView1.FindNode(@“Home/About iConnect”);
nd.Expand(); /* or nd.Expanded=true; */
```

Adding Nodes Programmatically

```
protected void Button3_Click(object sender, System.EventArgs e)
{
TreeNode myNode = new TreeNode();
myNode.Text = TextBox1.Text;
myNode.NavigateUrl = TextBox2.Text;
TreeView1.FindNode("Home//iConnect Software").ChildNodes.Add(myNode);
}
```

TreeNode Properties

- Checked
  - Determines if check box is checked or not.
- ImageUrl
  - Specifies the image that appears next to the node.
- NavigateUrl
  - Specifies the URL to which the current Tree node links.
- Selected
  - Determines whether the current Tree node is selected or not.
- ShowCheckBox
  - Display a check box for the current Tree node.
- SelectAction
  - Specifies the action that occurs when you click a Tree node. Possible values are *Expand*, *None*, *Select*, or *SelectExpand*.

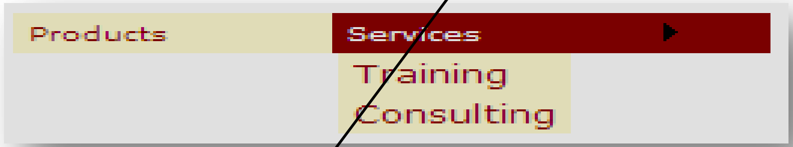

VerticalPadding
NodeSpacing
Root
TreeView.NodeIndent
HorizontalPadding
Parent 1
Leaf 1
Leaf 2
ChildNodesPadding
Parent 1

Menu Server Control

- One can use the Menu control to create the left-column menu to display a vertical list of links.
- One can use the Menu control to create a menu that more closely resembles the drop-down menus.
- Used to display hierarchical data.

Declaratively Adding Menu Items

```
<asp:Menu id="Menu1" Runat="server">
    <Items>
            <asp:MenuItem Text="Products" NavigateUrl="Products.aspx" />
            <asp:MenuItem Text="Services" NavigateUrl="Services.aspx">
            <asp:MenuItem Text="Training" NavigateUrl="Training.aspx" />
            <asp:MenuItem Text="Consulting" NavigateUrl="Consulting.aspx" />
            </asp:MenuItem>
    </Items>
</asp:Menu>
```



Page to be linked

Menu Text

BINDING TO XML

Just like TreeView It can be bound to *SiteMapDataSource* & *XmlDataSource* controls.

```
<asp:Menu ID="Menu1" runat="server" Orientation="Horizontal“
Width="254px"  DataSourceID=“XmlDataSource1">
       </asp:Menu>
<asp:XmlDataSource ID="XmlDataSource1" runat="server“
               DataFile="~/Trial.xml">
</asp:XmlDataSource>
```

Menu orientation Horizontal or Vertical

Binding Complex XML

```
<?xml version="1.0" encoding="utf-8" ?>
<menu>
  <category text="appetizer">
    <item text="soup" price="12.56" />
    <item text="cheese" price="17.23" />
  </category>
  <category text="entree">
    <item text="duck" price="89.21" />
    <item text="chicken" price="34.56" />
  </category>
  <category text="dessert">
    <item text="cake" price="23.43" />
    <item text="pie" price="115.46" />
  </category>
</menu>
```

Binding Complex XML

```
<asp:Menu ID="Menu1" runat="server" Orientation="Horizontal" Width="52px“
        DataSourceID="XmlDataSource1">
    <DataBindings>
    <asp:MenuItemBinding DataMember="category" TextField="text" />
    <asp:MenuItemBinding DataMember="item" TextField="text"
      ValueField="price" />
    </DataBindings>

    </asp:Menu>
    <asp:XmlDataSource ID="XmlDataSource1" runat="server“
        DataFile="~/Trial.xml"></asp:XmlDataSource>
```

Formatting Menu

- DisappearAfter
  - Specifies the amount of time, in milliseconds, that a dynamic menu item is displayed after a user moves the mouse away from the menu item.
- DynamicBottomSeparatorImageUrl (Static)
  - Specifies the URL to an image that appears under each dynamic menu item.
- DynamicEnableDefaultPopOutImage (Static)
  - Disables the image (triangle) that indicates that a dynamic menu item has child menu items.

Formatting Menu

- DynamicHorizontalOffset
  - Specifies the number of pixels that a dynamic menu item is shifted relative to its parent menu item.
- DynamicItemFormatString (Static)
  - Enables one to format the text displayed in a dynamic menu item.
- DynamicPopOutImageUrl (Static)
  - Specifies the URL for the dynamic popout image. By default triangle is displayed.

Formatting Menu

- DynamicTopSeparatorImageUrl (Static)
  - Specifies the URL to an image that appears above each dynamic menu item.
- DynamicVerticalOffset
  - Specifies the number of pixels that a dynamic menu item is shifted relative to its parent menu item.
- ItemWrap
  - Specifies whether the text in menu items should wrap.

Formatting Menu

- MaximumDynamicDisplayLevels
  - Specifies the maximum number of levels of dynamic menu items to display.
- ScrollDownImageUrl
  - Specifies the URL to an image that is displayed and that enables you to scroll down through menu items.
- ScrollUpImageUrl

Formatting Menu

- StaticDisplayLevels
  - Specifies the number of static levels of menu items to display.
- StaticSubMenuIndent
  - Specifies the number of pixels that a static menu item is indented relative to its parent menu item.
- Target
  - Specifies the window in which a new page opens when you click a menu item.

Formatting Menu

- One can style the Menu control by attaching Cascading Style Sheet classes to the Style objects of Menu.
- Style Objects:
  - DynamicHoverStyle / StaticHoverStyle
  - DynamicMenuItemStyle / StaticMenuItemStyle
  - DynamicMenuStyle / StaticMenuStyle
  - DynamicSelectedStyle /StaticSelectedStyle

```
<asp:Menu id=“Menu11" DynamicHoverStyle-CssClass=“MyStyle"
            Runat="server">
```

Formatting Menu

- One can apply styles to menu items based on their level in the menu.

- LevelMenuItemStyles

  - Contains a collection of *MenuItemStyle* controls, which correspond to different menu levels. First style first level

- LevelSelectedStyles

- LevelSubMenuStyles

  - Contains a collection of *MenuItemStyle* controls, which correspond to different menu levels of static menu items.

Formatting MenuItem

- ImageUrl
  - Specifies the URL for an image that is displayed next to a menu item.
- PopOutImageUrl
  - Specifies the URL for an image that is displayed when a menu item contains child menu items.
- SeparatorImageUrl
  - Specifies the URL for an image that appears below a menu item.

Formatting MenuItem

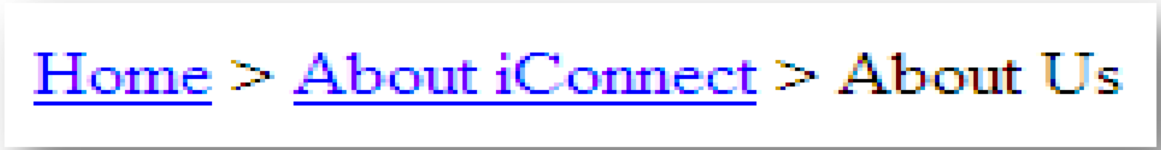
- Selectable
  - Determines whether to prevent users from selecting (clicking) a menu item.
- Selected
  - Decides whether a menu item is selected.
- Target
  - Specifies the name of the window that opens when you click a menu item..

Menu Events

- MenuItemDataBound
- MenuItemClick
- DataBound

SiteMapPath Control

- SiteMapPath control gives a linear path defining where the end user is in the navigation structure.
- The control automatically uses *web.sitemap* file.


Home > About iConnect > About Us

SiteMapPath Properties

- ParentLevelsDisplay
  - Limit the number of parent nodes displayed. By default, a SiteMapPath control displays all the parent nodes.
- PathDirection
  - Allows to reverse the order of the links displayed by the SiteMapPath control. Possible values are *RootToCurrent* and *CurrentToRoot*.

SiteMapPath Properties

- PathSeparator
  - Specifies the character used to separate the nodes displayed by the SiteMapPath control. The default value is >.
- RenderCurrentLinkAsNode
  - Allows to render the SiteMapPath node that represents the current page as a link. By default, the current node is not rendered as a link.

Formatting SiteMapPath

- CurrentNodeStyle
- NodeStyle
- PathSeparatorStyle
- RootNodeStyle

Site Map API

- The *SiteMap* class is an in-memory representation of the site's navigation structure.
- *SiteMap* provides the static properties *CurrentNode* (the site map node representing the current page) and *RootNode* (the root site map node).
- Both return a *SiteMapNode* object from which one can retrieve information from the site map, including the title, description, and URL values.

Site Map Class

- One can search for nodes using the methods of the current *SiteMapProvider* object, which is available through the *SiteMap.Provider* static property.

- *SiteMap.Provider.FindSiteMapNode()* method allows one to search for a node by its URL.

```
protected void Page_Load(Object sender, EventArgs e)
{
		lblHead.Text=SiteMap.CurrentNode.Title;
		lblDescription.Text=SiteMap.CurrentNode.Description;
}
```

SiteMapNode Properties

- ParentNode
  - Returns the node one level up in the navigation hierarchy i.e. node containing current node. On root, it returns *null*.
- ChildNodes
  - Provides a collection of all the child nodes.
- HasChildNodes
  - Determines whether child nodes exist or not.

SiteMapNode Properties

- PreviousSibling
  - Returns the previous node that’s at the same level.
- NextSibling
  - Returns the next node that’s at the same level if present.

```
protected void Page_Load(Object sender, EventArgs e)
{
        if(SiteMap.CurrentNode.NextSibling !=null)
        {
                lnkNext.NavigateUrl=SiteMap.CurrentNode.NextSibling.Url;
                lnkNext.Visible=true;
        }
        else
        {
                lnkNext.Visible=false;
        }
}
```

URL Mapping

- Sometimes one might want to have several URLs lead to the same page.

- ASP.NET includes its own URL mapping feature.

- That maybe, due to, one want to implement the logic in one page and use query string arguments but still provide shorter and friendly URLs to the website users.

- That maybe one have renamed a page, but want to keep the old URL functional to avoid bookmark breaks.

URL Mapping

- Basic Idea behind URL Mapping is to map a request URL to a different URL.
- The mapping rules are stored in the web.config file in the <urlMappings> section.
- ASP.NET doesn't support advanced matching rules, such as wildcards or regular expressions.

URL Mapping

```
<system.web>
		<urlMappings>
			<add url=”~/Category.aspx”
				mappedUrl=”~/Default.aspx?category=default” />
			<add url=“~/Software.aspx”
				mappedUrl=“~/Default.aspx?category=software” />
		</urlMappings>
</system.web>
```