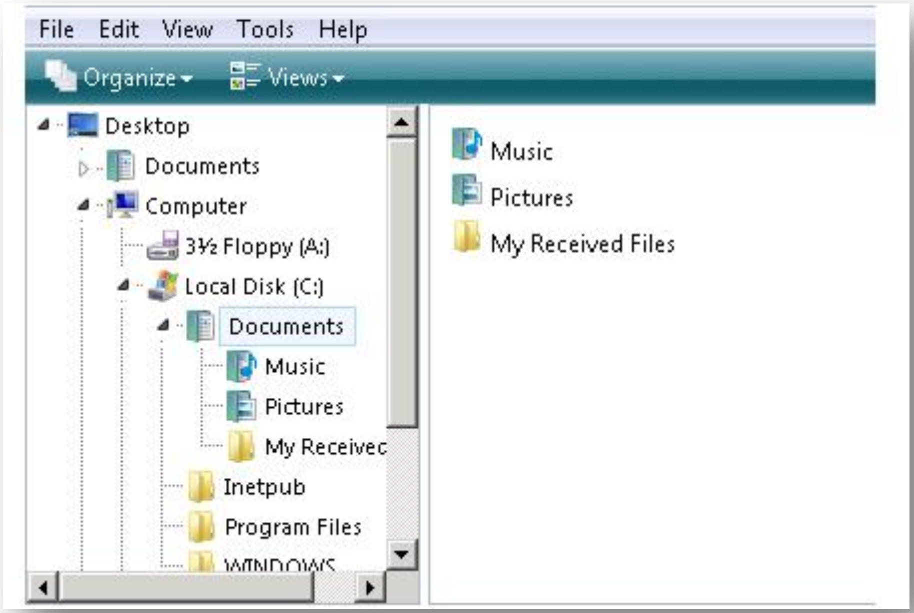
WebTreeView.NET 1.0

WebTreeView.NET® 1.0 is Intersoft's latest ASP.NET server control which enables you to easily create a hierarchical data presentation. This powerful control incorporates numerous unique features which makes WebTreeView.NET® 1.0 flexible and highly customizable to meet your design needs.

WebTreeView.NET® 1.0 is loaded with many features which make it the best tree view control in the industry. These features are as followed:

- Rock-solid client side architecture enables high-performance OOP-based TreeView control.
- Supports databinding to Hierarchical DataSourceControl (such as XmlDataSource and SiteMapDataSource) in both design and runtime.
- The unique Tristate checkbox and of course support the two-state checkbox
- The ability to load the child nodes on demand. This technique, called Load on Demand will give you performance boost while working with large hierarchical data presentation.
- The ability to drag and drop a node to a root node, child node, or sibling node.
- Keyboard navigation provides your end users with the ability to control the node selection by using keyboard
- Find a node based on its path.
- Built-in animation during expanding or collapsing the node.



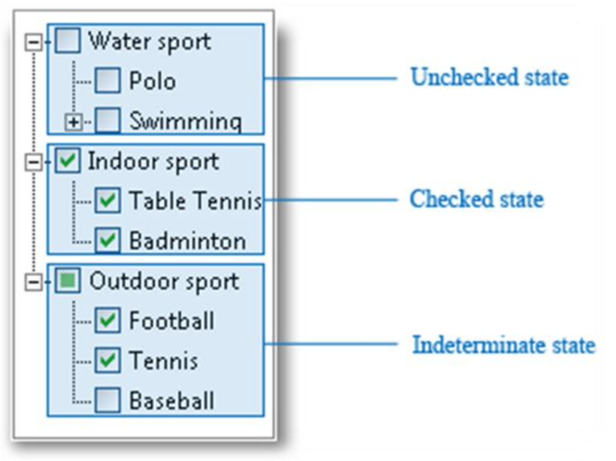
Web-based Vista Explorer's tree view is just a simple sample of our WebTreeView.NET® 1.0

This fully-featured control will benefit the developers in creating a powerful data hierarchical presentation, window-to-window navigation and also structural navigation. This, of course, will help the developers deliver simple, yet powerful tree view control to the customer.

Concepts

TriState CheckBox and Normal CheckBox

WebTreeView.NET® 1.0 introduces you innovative TriState checkbox which allows you to have a better control over the state of checkbox in hierarchical presentation. These three states are: *checked*, *indeterminate*, and *unchecked*.



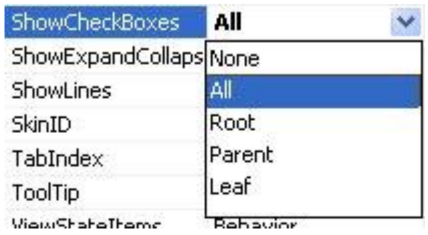
WebTreeView.NET® 1.0 Tristate Checkbox

When to use Tristate checkbox? As an ASP.NET developer, you should have installed Microsoft Visual Studio before and familiar with the setup wizard in Microsoft Visual Studio 2005. In the option page, users are given an option to choose which components they want to install by checking the check box. Notice when you check the root check box, all the child check boxes will be checked. If you check some child check boxes, the root check box will be checked

This Tristate checkbox concept is adopted by WebTreeView.NET® 1.0, implementing the exact behavior of the three-state checkbox concept of Windows-based tree view.

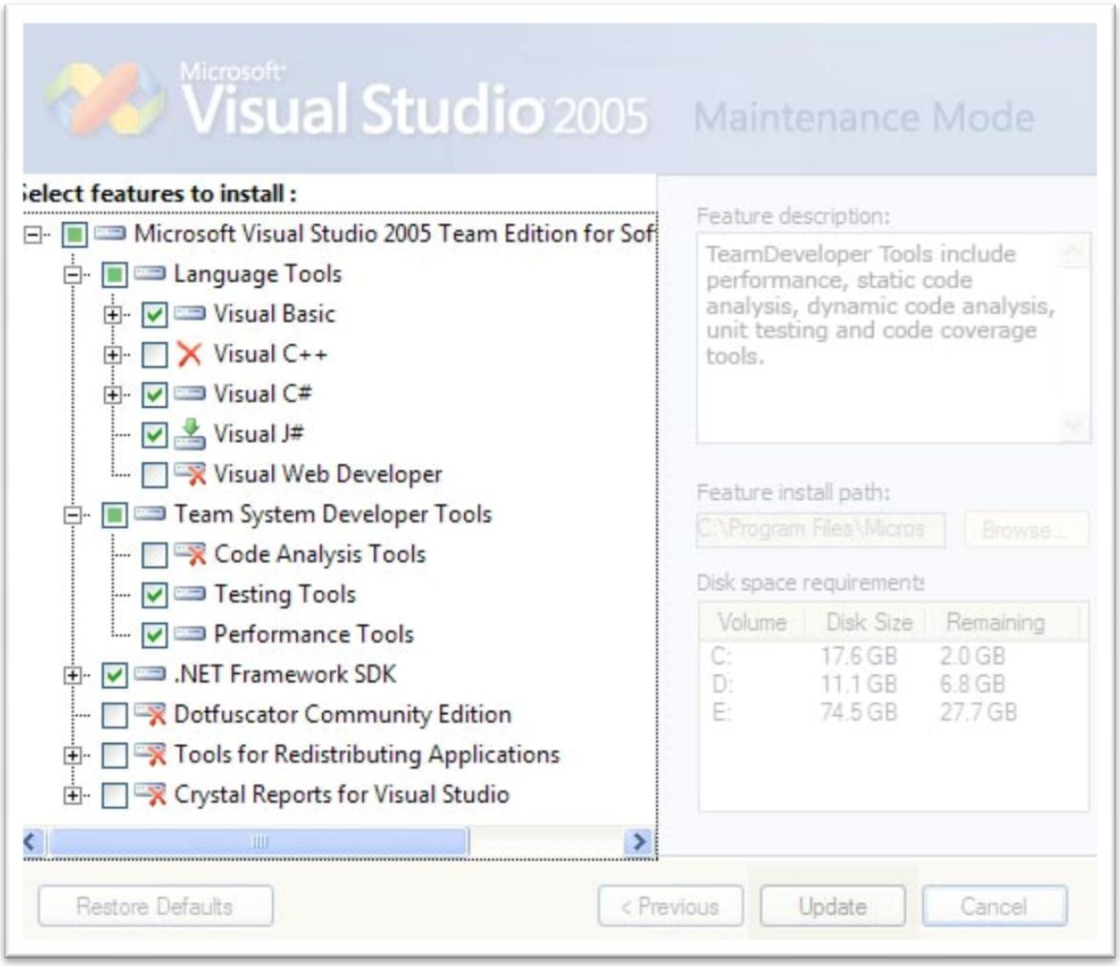
There is no code needed to take advantage of the Tristate checkbox feature, you only need to set the **EnableTriStateCheckbox** property to *True*. Another related property is the **AutoCheckChildNodes**. Set it to *True* and it will automatically check all child nodes when the parent node is checked.

Besides the advanced TriState checkbox, WebTreeView is also equipped with traditional checkbox, the two-state checkbox (*checked* or *unchecked*). Set the **ShowCheckBoxes** property to *All* and you will see a checkbox in every nodes.



The value of ShowCheckBoxes property

Please note that the **AutoCheckBox** property also applies to normal checkbox, two state-state checkbox (*checked* and *unchecked*).

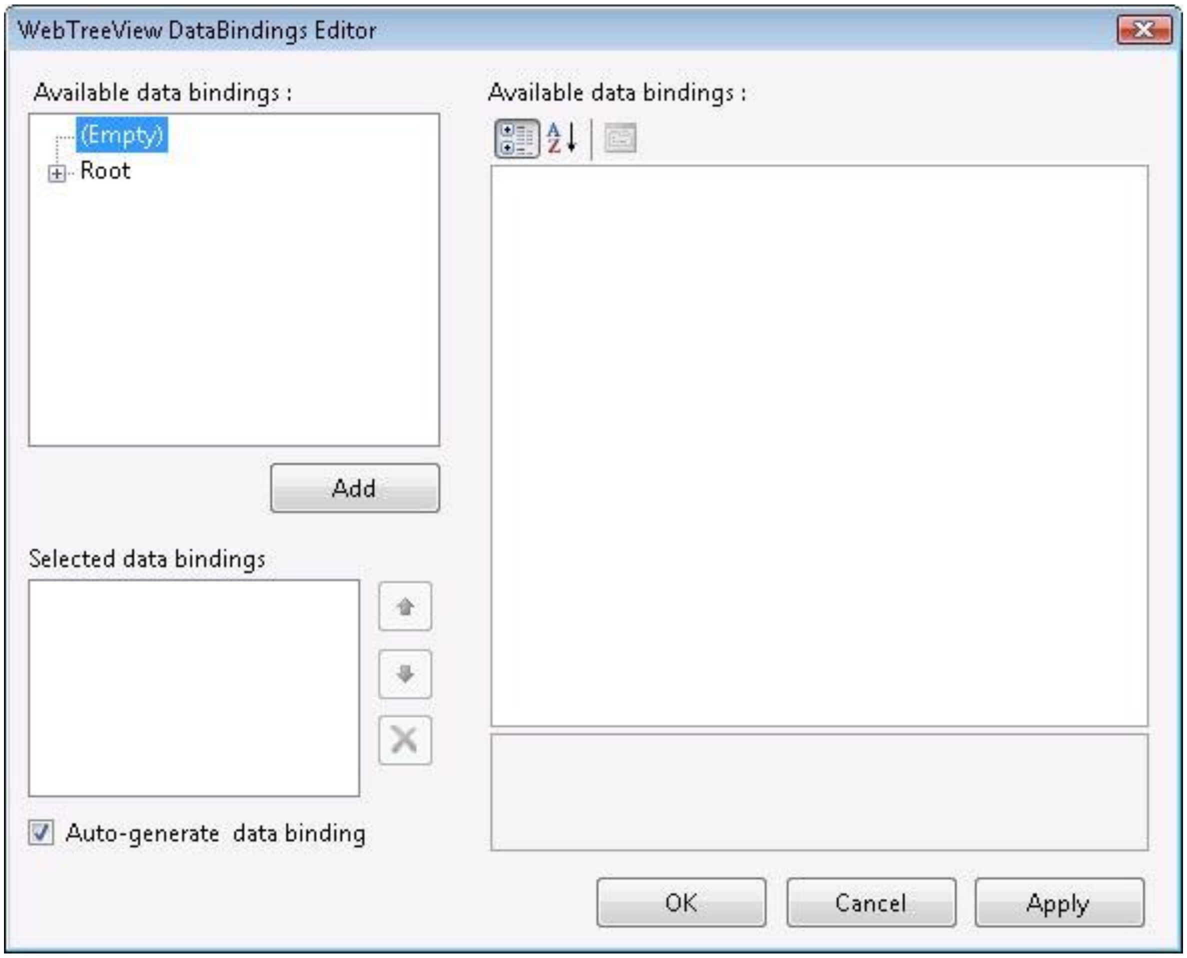


Microsoft Visual Studio Setup Wizard Replica by WebTreeView.NET® 1.0

How-to: Create a simple WebTreeView with Tristate CheckBox

Please follow these steps to create a simple WebTreeView with Tristate checkbox

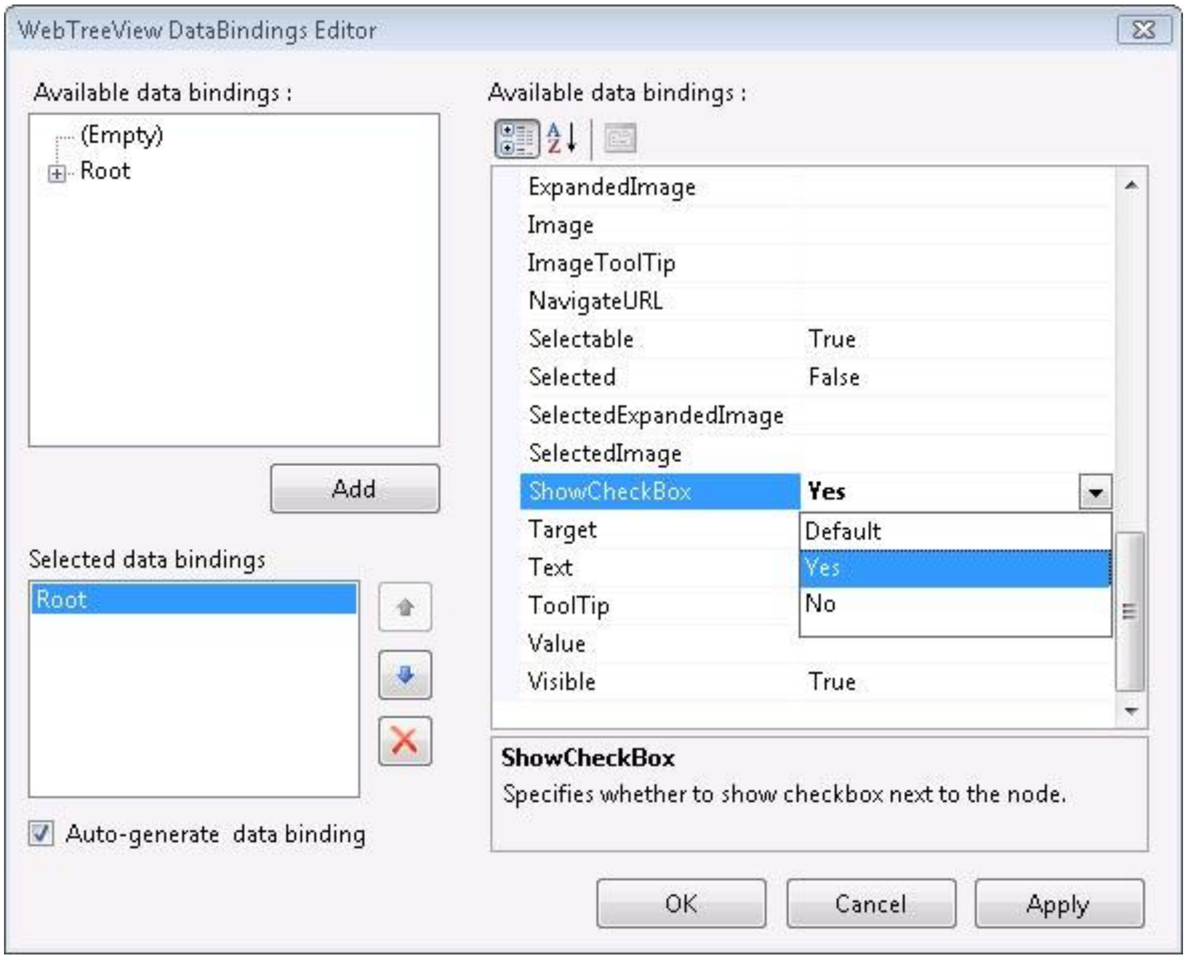
1. Drag a WebTreeView control to the designer and resize it to your desired size.
2. Set the **EnableTriStateCheckBox** to *True* and **AlutoCheckChildNodes** to *True*.
3. Drag an XMLDataSource control to your designer and bind it an XML file. (we use Products.xml for all tutorials in this WhitePaper)
4. Set the **DataSourceID** property to the XMLDataSource controlId, in this case the ID is *XMLDataSource1*
5. Right click on the WebTreeView.NET control and select "**Edit TreeNode DataBindings**".
6. The **WebTreeView DataBindings Editor** will appear and check the Auto-generate data binding.


WebTreeView DataBindings Editor
Available data bindings :
(Empty)
Root
Add
Selected data bindings
Available data bindings :
OK
Cancel
Apply

☑ Auto-generate data binding

WebTreeView's DataBindings Editor Window

7. Please select the **Root** node in the top left box and click **add.**
8. Select the **Root** node in the bottom left box and you will see the list of the properties. Set the **TextField** property to *name* (not shown in the screenshot below) under the **Databindings**. Then, set the **ShowCheckBox** property (under **Default Properties**) to *Yes*. Please refer to the screenshot below.

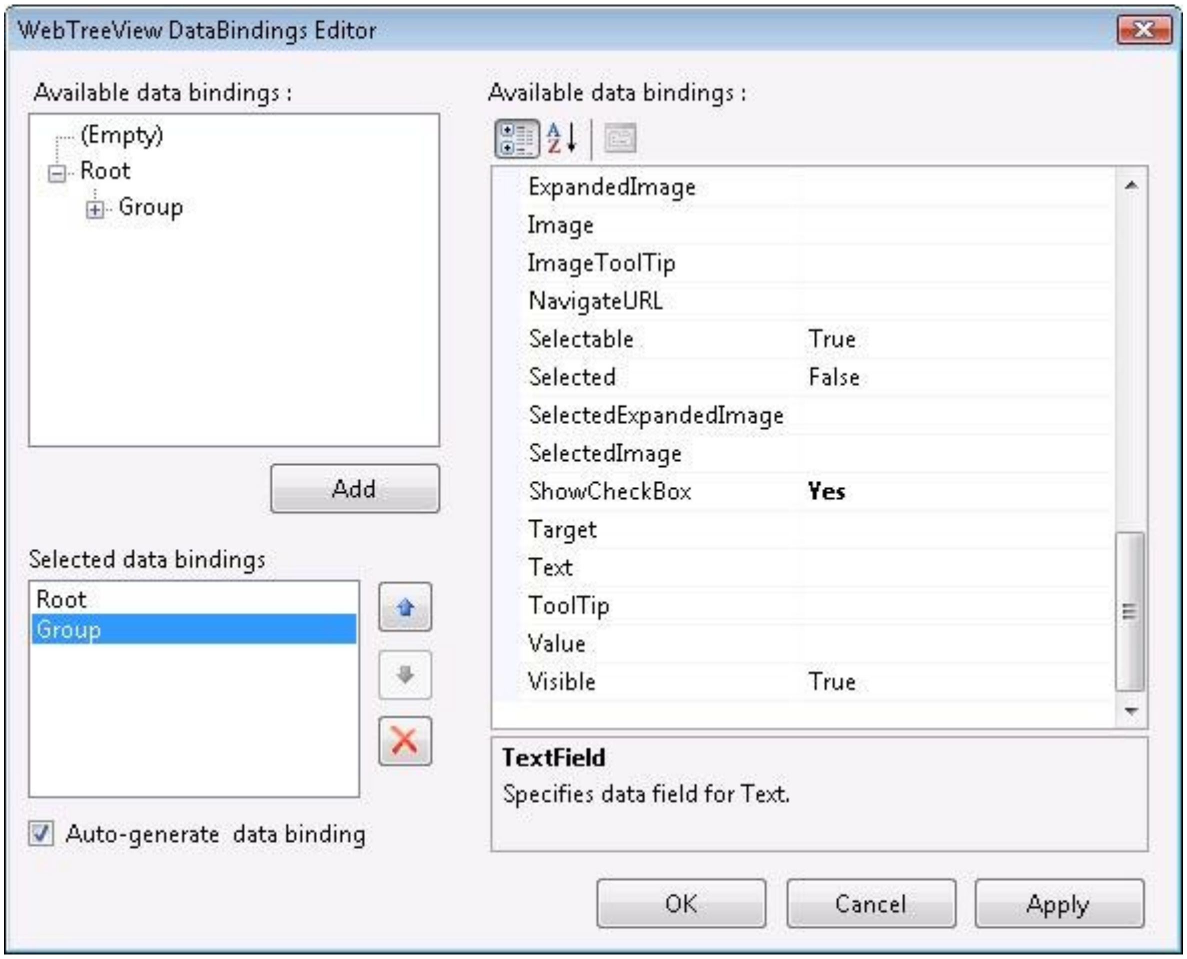

WebTreeView DataBindings Editor
Available data bindings :
(Empty)
Root
Add
Selected data bindings
Root
Auto-generate data binding
Available data bindings :
OK
Cancel
Apply

ExpandedImage	
Image	
ImageToolTip	
NavigateURL	
Selectable	True
Selected	False
SelectedExpandedImage	
SelectedImage	
ShowCheckBox	Yes
Target	Default
Text	Yes
ToolTip	No
Value	
Visible	True

ShowCheckBox
Specifies whether to show checkbox next to the node.

Select the Root Node in the bottom left box and set the ShowCheckBox property to Yes and also set the TextField property to name

9. Please expand the **Root** node and select the **Group** node and click **add.**
10. Select the **Group** node in the bottom left box and set the **TextField** to name and **ShowCheckBox** to Yes.


WebTreeView DataBindings Editor

Available data bindings :

(Empty)
Root
Group

Add

Selected data bindings

Root
Group

[✓] Auto-generate data binding

Available data bindings :

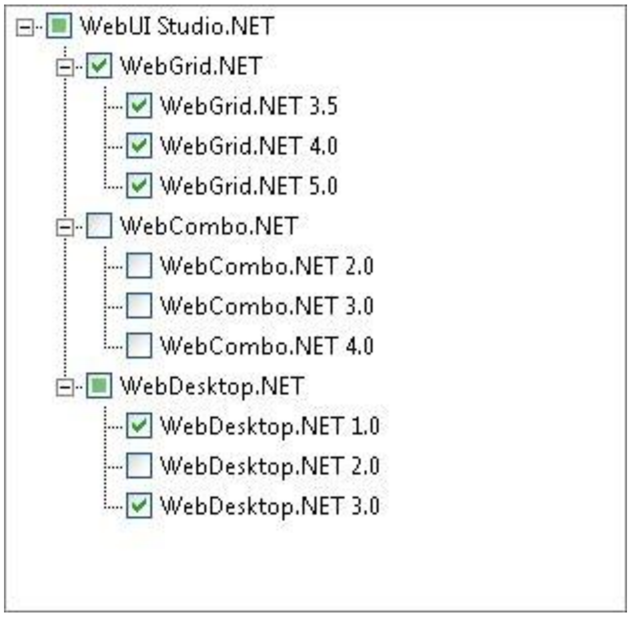
ExpandedImage	
Image	
ImageToolTip	
NavigateURL	
Selectable	True
Selected	False
SelectedExpandedImage	
SelectedImage	
ShowCheckBox	**Yes**
Target	
Text	
ToolTip	
Value	
Visible	True

TextField
Specifies data field for Text.

OK Cancel Apply

Select the Group Node in the bottom left box and set the ShowCheckBox property to Yes and also set the TextField property to name

11. Click ok and run your sample.



Simple WebTreeView.NET with TriState CheckBox

Load on Demand

One important aspect when building a Web-based application using ASP.NET control is that the user interaction should be performed in real time. Real time here means that the information should be available when needed without having to wait for a minute. This will be an issue if a control contains lots of data. This is where the Load on Demand plays important part in WebTreeView.NET. This technique will deliver the content faster with the build-in AJAX capability.

Can you imagine loading one hundred nodes and each node consists of another one hundred child nodes. This will, of course, decrease the performance significantly when the large nodes are delivered at once in first page load. This is the time when the Load on Demand plays important role. It will boost the performance by rendering the parent node first and when you expand a node, it will then render the child node.

With the advanced Load on Demand capability, you can now deliver high performance hierarchical tree view presentation with large data without any performance loss.

The Load on Demand technique comes in handy when you have complex data presentation structure. The concept behind this technique is rendering the child node when needed, which is when the node is expanded. This process will definitely decrease the WebTreeView.NET's workload, thus delivering faster response and better user interaction.

You only need to set the **EnableLoadOnDemand** property to *True* and you are all set. It's just that simple.

Advanced Load on Demand

The concept behind the advanced Load on Demand is that all physical child nodes are initialized in InitializeChildNodes event. This will give you more control on how the child nodes are initialized.

The following code demonstrates you on how to initialize a child node in InitializeChildNodes event

```
protected void WebTreeView2_InitializeChildNodes(object sender,
ISNet.WebUI.WebTreeView.WebTreeViewNodeEventArgs e)
{
   ISNet.WebUI.WebTreeView.WebTreeViewNode a = new ISNet.WebUI.WebTreeView.WebTreeViewNode();
   a.Name = "NewNode" + WebTreeView2.NodeIndex.ToString();
   a.Text = "New Node " + WebTreeView2.NodeIndex.ToString();

   e.Node.Nodes.Add(a);

   ISNet.WebUI.WebTreeView.WebTreeViewNode b = new ISNet.WebUI.WebTreeView.WebTreeViewNode();
   b.Name = "NewNode" + WebTreeView2.NodeIndex.ToString();
   b.Text = "New Node " + WebTreeView2.NodeIndex.ToString();

   e.Node.Nodes.Add(b);
}
```

Keyboard Navigation

WebTreeView.NET® 1.0 supports both mouse navigation and keyboard navigation which will enhance your end-user's interactivity when using WebTreeView.NET.

WebTreeView.NET® 1.0 supports basic desktop-based keyboard navigation such as:

- Cut operation can be executed by pressing the combination of **CTRL + X**.

- Copy operation can be executed by pressing the combination of **CTRL + C**.
- Paste operation can be executed by pressing the combination of **CTRL + V**.
- You can also use the arrow navigation (left, right, up, down, home, delete) to navigate our WebTreeView.NET® 1.0.
- Press **F2** to perform inline-node editing.
- **CTRL + Z** to undo any inline-node editing.

WebTreeView.NET® 1.0 is also equipped with multiple node selection using the **CTRL**+**SHIFT** key. You only need to set the **AllowMultipleSelect** to *True*.

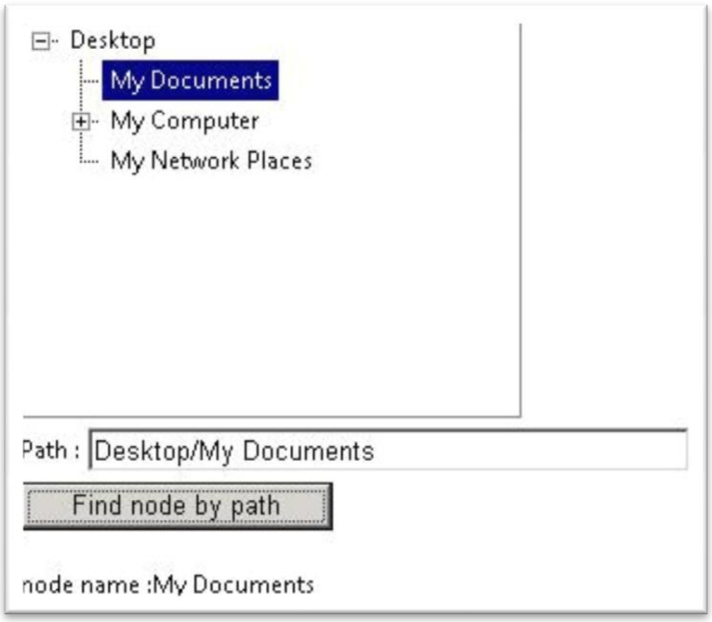
Path

Path in WebTreeView.NET® 1.0 can be defined as a line which connects a node from the root node to the node itself. WebTreeView.NET has three built-in powerful feature related to path such as listed in the following:

- Find node by path

  WebTreeView.NET has a built-in **FindNodeByPath()** method which allows you to find a node by the given path. This means that you can perform any action you desired to the node just by providing the path of a node.

```
function Button1_onclick()
{
  var tv = ISGetObject("WebTreeView1");
  var value = document.getElementById("txtPath").value;
  var node =  tv.FindNodeByPath(value);
  var divStatus = document.getElementById("divWebTreeView1Node");

  divStatus.innerHTML = "";

  if (node!=null)
  {
     node.Select();
     divStatus.innerHTML = "Node name :" + node.Name;
  }
  else
  {
     divStatus.innerHTML = "node is not found";
  }
}
```



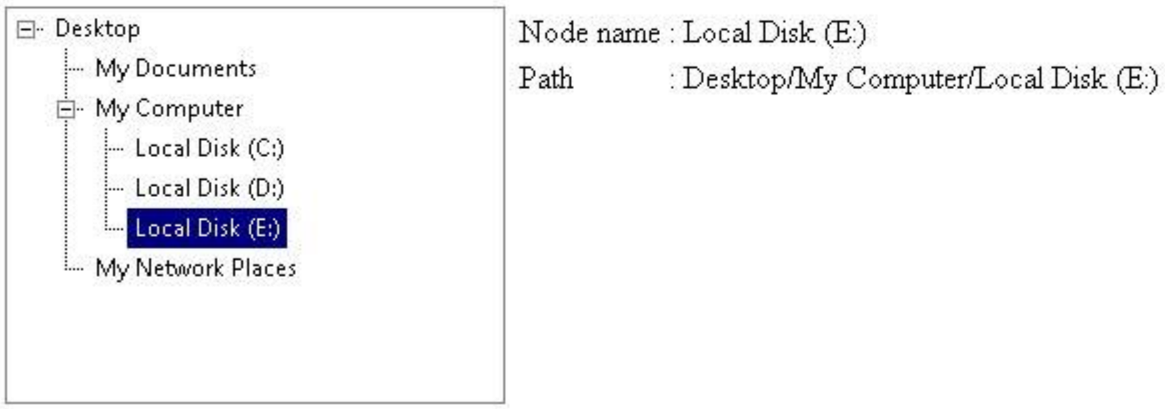
Path : Desktop/My Documents

node name :My Documents

This is a sample of node finding based on its path

When executed, the code above will select a node based on the path typed in the textbox. Please refer to our WebTreeView.NET® 1.0 Sample.

The following sample scenario can be done easily by using our WebTreeView.NET. Please refer to the java script below on how to get the selected node's path.

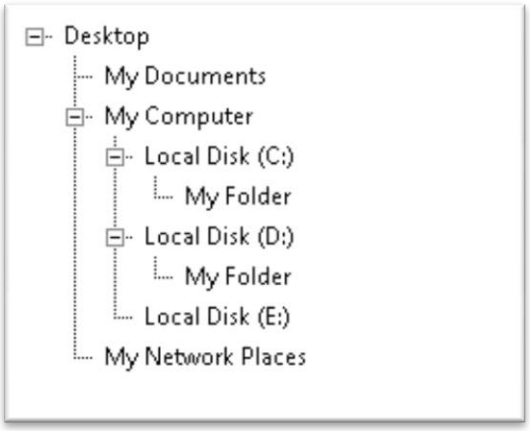


Node name : Local Disk (E:)
Path : Desktop/My Computer/Local Disk (E:)

```
function doWebTreeView1NodeSelect(ctrlId, node)
{
    var divStatus = document.getElementById("divWebTreeViewNodeDepth");
    document.getElementById("tdNode").innerHTML = node.Name;
    document.getElementById("tdPath").innerHTML = node.Get("Path");
}
```

- Identify a node as a unique node
  Path is unique. Every node has its own unique path. This is what differ a node from the others. There is certain scenario that two nodes have the same name. This is possible if both nodes are on the different level or when two nodes are located in two different parents. Path is an important feature when it comes to this kind of scenario.

  The sample below clearly demonstrating this feature. There are two "My Folder" nodes, but both have different path.

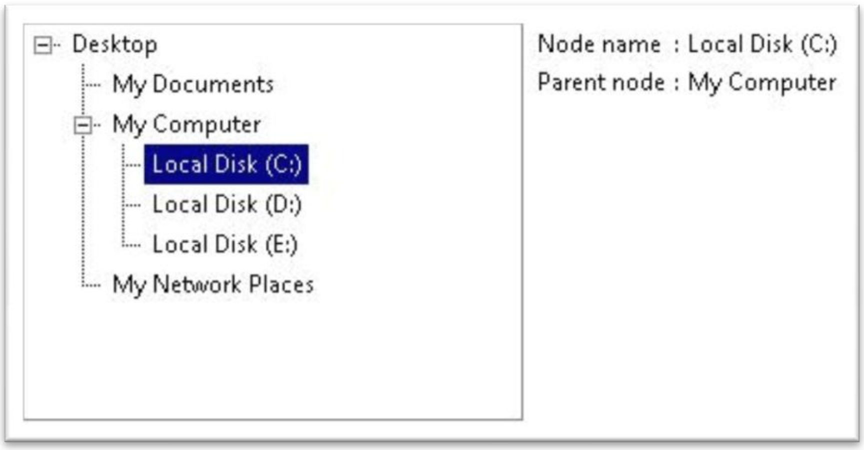


A tree view which have two grandchild nodes, but different path

- Show the hierarchy of a node
  A tree view is a collection of nodes which present a hierarchical data presentation. This node can have items or child nodes and the child nodes can also have child nodes. This is what we call the hierarchy of a node. A node always has a parent node. If it's a single level node, then the single node is the parent node.

  WebTreeView.NET incorporates powerful built-in API which can be used to get the hierarchy of a node. This can be achieved by using **GetParentNode()**. Please refer to the sample below.

```
function doWebTreeView1NodeSelect(ctrlId, node)
{
  var parentNode = node.GetParentNode();
  var parentNodeName = "";
  document.getElementById("tdNode").innerHTML = node.Name;
  if(parentNode!=null)
  {
    parentNodeName = parentNode.Name;
  }
  else
  {
    parentNodeName = "-";
  }
  document.getElementById("tdParent").innerHTML = (parentNodeName);
}
```



Node name : Local Disk (C:)
Parent node : My Computer

This is a sample of showing the hierarchy of a node

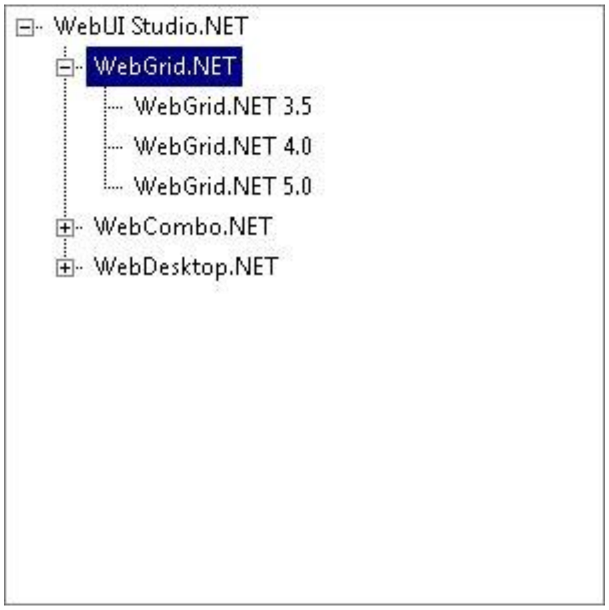
When you select a node, it will show you the node information and its parent node name.

Automatically select a node on expand or collapse

WebTreeView.NET allows you to automatically select a node on expand or collapse. You only need to set **AutoSelectNodeOnExpandCollapse** to *True*. Please refer to the following code.

```
<ISWebTreeView:WebTreeView ID="WebTreeView1" runat="server"
AutoCheckChildNodes="True" AutoGenerateDataBindings="True"
DataSourceID="XmlDataSource1" Height="300px" Width="300px"
```

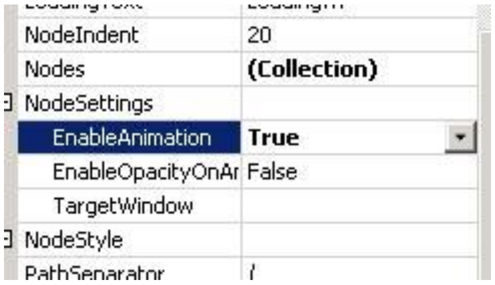
```
AutoSelectOnNodeExpandCollapse="True">
<FrameStyle BorderStyle="Solid" BorderWidth="1px" BorderColor="Gray">
</FrameStyle>
<DataBindings>
  <ISWebTreeView:WebTreeViewNodeBinding DataMember="Root" TextField="name"/>
  <ISWebTreeView:WebTreeViewNodeBinding DataMember="Group" TextField="name"/>
</DataBindings>
<NodeStyle>
  <Active ForeColor="White" BaseStyle="Normal" BackColor="Navy">
  </Active>
  <Over BaseStyle="Normal" Font-Underline="True">
  </Over>
  <Normal Font-Size="9pt" Font-Names="Segoe UI" Overflow="Hidden"
Cursor="Default" OverflowX="Hidden" OverflowY="Hidden">
     <Padding Left="1px" Right="3px">
     </Padding>
  </Normal>
</NodeStyle>
</ISWebTreeView:WebTreeView>
<asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="~/Products.xml">
            </asp:XmlDataSource>
```



If you expand or collapse a node, the node will be selected automatically

Animation during expanding or collapsing a node

This eye candy feature embedded in our WebTreeView.NET® 1.0 will feast your customer's eyes for animated and interactive user interface when expand or collapse a node. If you set the **EnableAnimation** property to *True*, you will be able to enjoy the sliding animation during expanding or collapsing. This proves that our WebTreeView.NET® 1.0 is made to fulfill your customer's need.



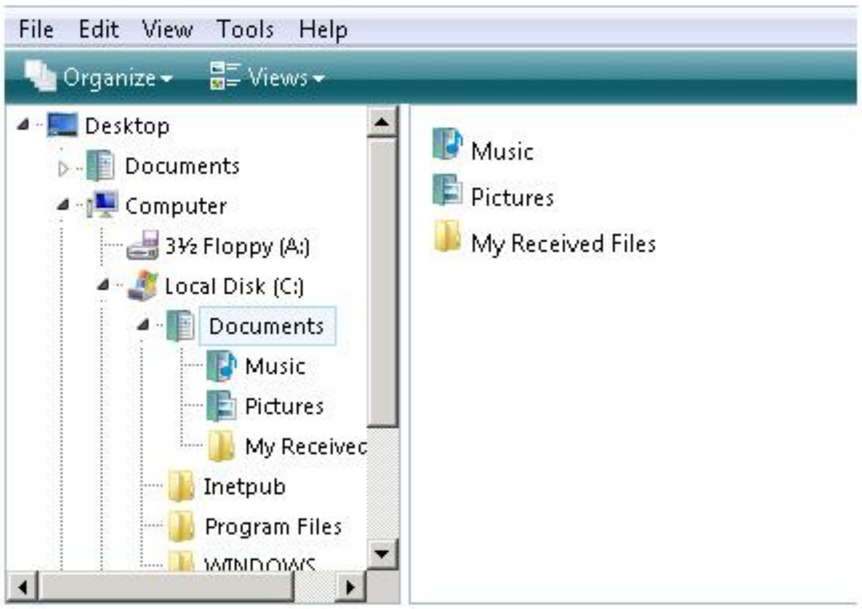
NodeIndent	20
Nodes	(Collection)
NodeSettings	
EnableAnimation	True
EnableOpacityOnAr	False
TargetWindow	
NodeStyle	
PathSeparator	/

WebTreeView's properties window

To enable the collapse and expand animation, open the properties window, expand the **NodeSettings** and set the **EnableAnimation** to *True*.

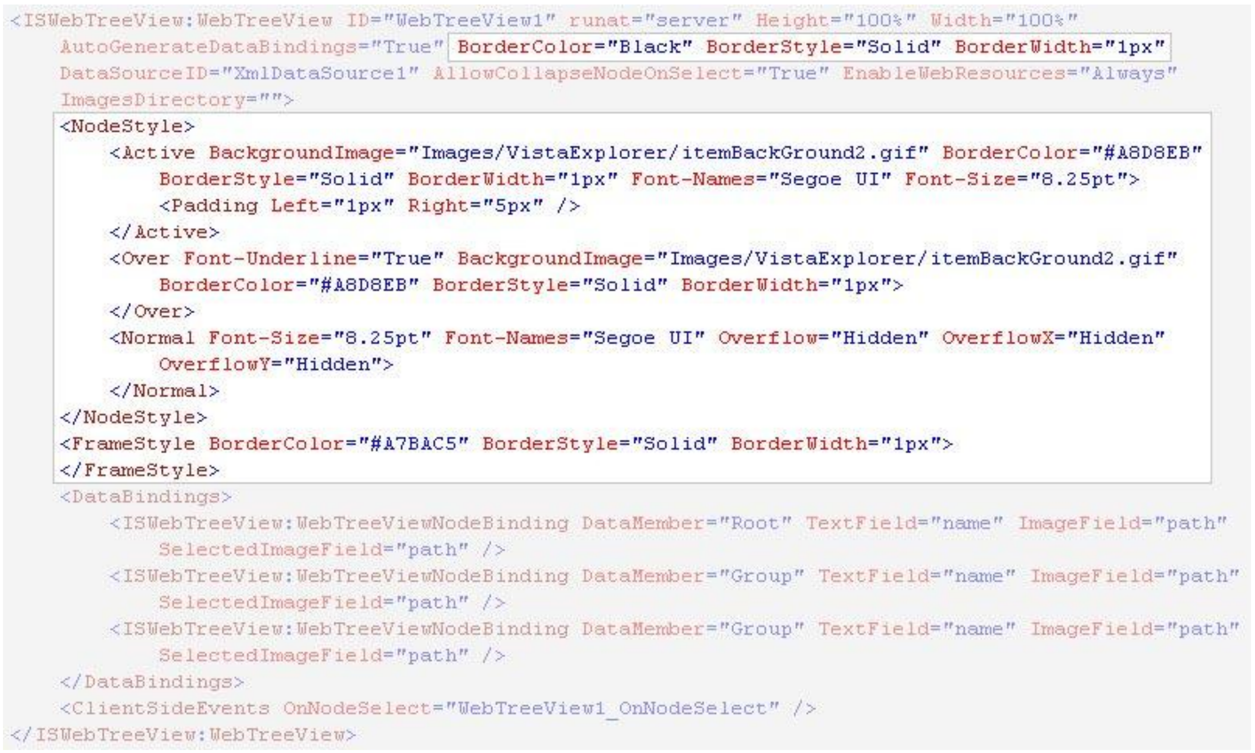
Customizing styles, appearance, and images

Besides incorporating numerous features, WebTreeView is also highly customizable in terms of style, appearance, and images of the node and frame.



Tree view navigation in Web-based Vista Explorer replica made by our WebTreeView.NET

You can easily customize WebTreeView to any style and appearance. The Web-based Vista Explorer replica is a perfect sample describing this scenario. Please take a look at the code below.



```
<ISWebTreeView:WebTreeView ID="WebTreeView1" runat="server" Height="100%" Width="100%"
    AutoGenerateDataBindings="True" BorderColor="Black" BorderStyle="Solid" BorderWidth="1px"
    DataSourceID="XmlDataSource1" AllowCollapseNodeOnSelect="True" EnableWebResources="Always"
    ImagesDirectory="">
    <NodeStyle>
        <Active BackgroundImage="Images/VistaExplorer/itemBackGround2.gif" BorderColor="#A8D8EB"
            BorderStyle="Solid" BorderWidth="1px" Font-Names="Segoe UI" Font-Size="8.25pt">
            <Padding Left="1px" Right="5px" />
        </Active>
        <Over Font-Underline="True" BackgroundImage="Images/VistaExplorer/itemBackGround2.gif"
            BorderColor="#A8D8EB" BorderStyle="Solid" BorderWidth="1px">
        </Over>
        <Normal Font-Size="8.25pt" Font-Names="Segoe UI" Overflow="Hidden" OverflowX="Hidden"
            OverflowY="Hidden">
        </Normal>
    </NodeStyle>
    <FrameStyle BorderColor="#A7BAC5" BorderStyle="Solid" BorderWidth="1px">
    </FrameStyle>
    <DataBindings>
        <ISWebTreeView:WebTreeViewNodeBinding DataMember="Root" TextField="name" ImageField="path"
            SelectedImageField="path" />
        <ISWebTreeView:WebTreeViewNodeBinding DataMember="Group" TextField="name" ImageField="path"
            SelectedImageField="path" />
        <ISWebTreeView:WebTreeViewNodeBinding DataMember="Group" TextField="name" ImageField="path"
            SelectedImageField="path" />
    </DataBindings>
    <ClientSideEvents OnNodeSelect="WebTreeView1_OnNodeSelect" />
</ISWebTreeView:WebTreeView>
```

Above is the Web-based Vista Explorer client-side code.

Notice the above example and you'll see that these following properties are set:

BorderColor	Set the color of the WebTreeView's border
BorderStyle	Set the border's style of the WebTreeView
BorderWidth	Set the border's width of the WebTreeView
BackgroundImage	Set each node's background image
Font-Names	Set the font style to be applied to WebTreeView. This can be set under the

	Font property
Font-Size	Set the size of the font to be applied to WebTreeView. This can also be set under the Font property
Padding left	Determines the amounts of space on the left of WebTreeView
Padding right	Determines the amounts of space on the right of WebTreeView
Font-Underline	Set underline style to each node in WebTreeView
Overflow	Show or hide the WebTreeView's scrollbar
Overflow-X	Show or hide the WebTreeView's horizontal scrollbar
Overflow-Y	Show or hide the WebTreeView's vertical scrollbar

- Frames Appearance
  Frame appearance is an ASP.NET inherited style. This enables you to customize the basic appearance of WebTreeView. You can easily set the background color, font and its color, border color, width, and style.
- Appearance
  Appearance is WebTreeView's own style. It provides you numerous properties to set which will help you create and customize your WebTreeView easily. Under the appearance property, you can set the Frame Style, Node Style (Active state, Normal state, and Over state), the loading image and the text. Please see our documentation for the complete list.
- Images
  Another style you can customize is the images. The images can be set in the **ImageSettings** property.

CheckBoxFalseImage	Gets or sets the images used to render false checkbox
CheckBoxPartialImage	Gets or sets the image used to render partial checkbox
CheckBoxTrueImage	Gets or sets the images used to render false checkbox
CollapseImage	Gets or sets the image used to render node when node is collapsed
ExpandedParentNodeImage	Gets or sets the image used to render expanded parent level node
ExpandedRootNodeImage	Gets or sets the image used to render expanded root level node
ExpandedSelectedParentNode	Gets or sets the image used to render expanded parent level node when node is selected
ExpandedSelectedRootNode	Gets or sets the image used to render expanded root level node when node is selected
ExpandImage	Gets or sets the image used to render node when node is expanded
LeafNodeImage	Gets or sets the image used to render leaf level node
ParentNodeImage	Gets or sets the image used to render parent level node
RootNodeImage	Gets or sets the image used to render root level node
SelectedLeafNodeImage	Gets or sets the image used to render selected leaf level node

SelectedParentNodeImage	Gets or sets the image used to render selected parent level node
SelectedRootNodeImage	Gets or sets the image used to render selected root level node

Node Editing

WebTreeView.NET also comes with the node editing ability. Node editing means that you can add a node, delete a node, and edit a node. These three client-side actions provide you full control over a node. In order to take advantage of this feature, you need to set **AllowAddNode**, **AllowDeleteNode**, **AllowNodeEditing** to *Yes*.

```
function Button1_onclick()
{
        var name = document.getElementById("txtNode");
        var text = document.getElementById("txtText");

        if (name.value==null || name.value=="")
        {
                alert("node name cannot be empty");
                name.focus();
        }
        else
        {
                var tv = ISGetObject("WebTreeView1");
                var selectedNode = tv.GetSelectedNode();
                if (selectedNode==null)
                        tv.AddNode(name.value);
                else
                        selectedNode.AddNode(name.value);
                name.value = "node" + (tv.NodeIndex + 1);
        }
}
```

The above code is a sample on how to implement **AllowAddNode** feature easily. The **AddNode(nodeObject)** method will add a new node which is passed as a parameters to your WebTreeView when called. If the method is called from WebTreeView object, the node will be created as a root node, but if the method is called from an existing WebTreeViewNode, the new node will be treated as a child node.

The **DeleteNode(nodeObject)** method will delete a node when called.

```
var tv = ISGetObject("WebTreeView1")
tv.DeleteNode(node1); //node1 is a node object
```

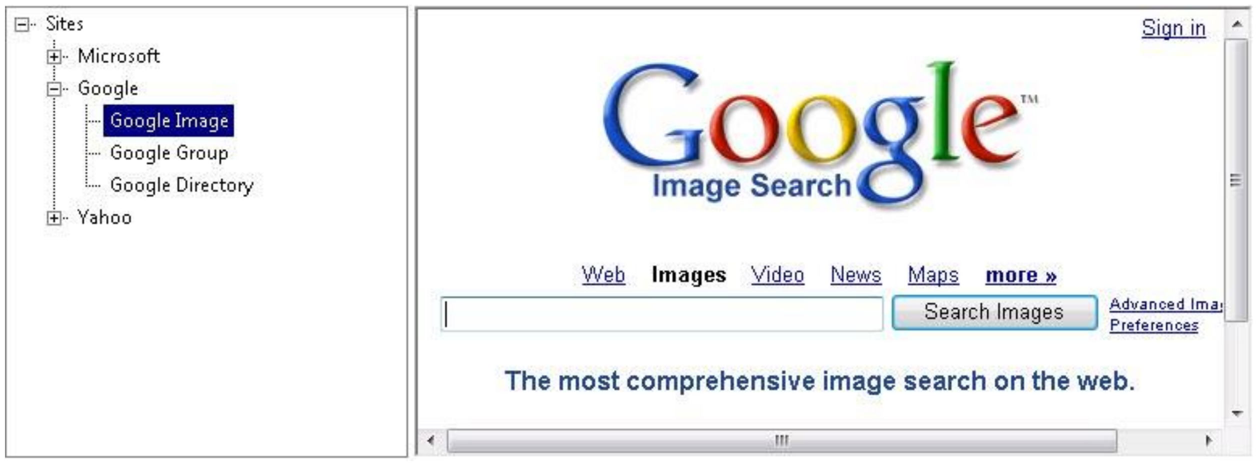
The **DeleteNode()** method works for all node level, root node, child, grand child node. If you delete a parent node, the parent's child node will also be deleted. This is the default behavior.

If you set the **AllowNodeEditing** to *Yes*, your end-users will be able to edit the node's name by using your keyboard and mouse.

Using NavigateURL and Target

NavigateURL is a property which will allow you to set the URL when a node is clicked, while the **TargetWindow** property allows you to set in which window the URL will be displayed. You can direct it to an iFrame or even a new window.

The screenshot below demonstrates the **NavigateURL** and **Target** property



A sample of WebTreeView utilizing NavigateURL and Target property

This is the ASPX code for the above sample. This is a data bound WebTreeView. You can see that **NavigateURLField** value and **TargetField** value are taken from the XML file.

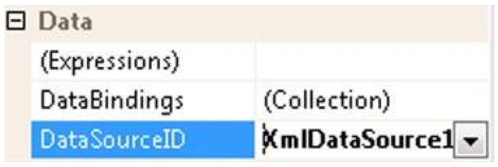
```
<table width="100%">
   <tr>
       <td width="250px">
          <ISWebTreeView:WebTreeView ID="WebTreeView1" runat="server"
Height="280px" Width="250px" AutoGenerateDataBindings="True"
DataSourceID="XmlDataSource1">
          <DataBindings>
             <ISWebTreeView:WebTreeViewNodeBinding DataMember="Root"
TextField="name" />
             <ISWebTreeView:WebTreeViewNodeBinding DataMember="Group"
NavigateURLField="NavigationURL" TextField="name" TargetField="Target" />
          </DataBindings>
          </ISWebTreeView:WebTreeView>

          <asp:XmlDataSource ID="XmlDataSource1" runat="server"
DataFile="~/Sample.xml"></asp:XmlDataSource>
      </td>
      <td valign="top">
         <iframe name="iFrame" id="iFrame" src="about:blank" style="width:
100%; height: 280px;"></iframe>
      </td>
   </tr>
</table>
```

Data Binding

WebTreeView.NET® 1.0 is equipped with data binding capability. You can bind to XMLDataSource and SiteMapDataSource.. This basic feature will help you create a powerful Treeview presentation in the matter of seconds.

Following is the tutorial on how to bind WebTreeView.NET® 1.0 to XMLDataSource.

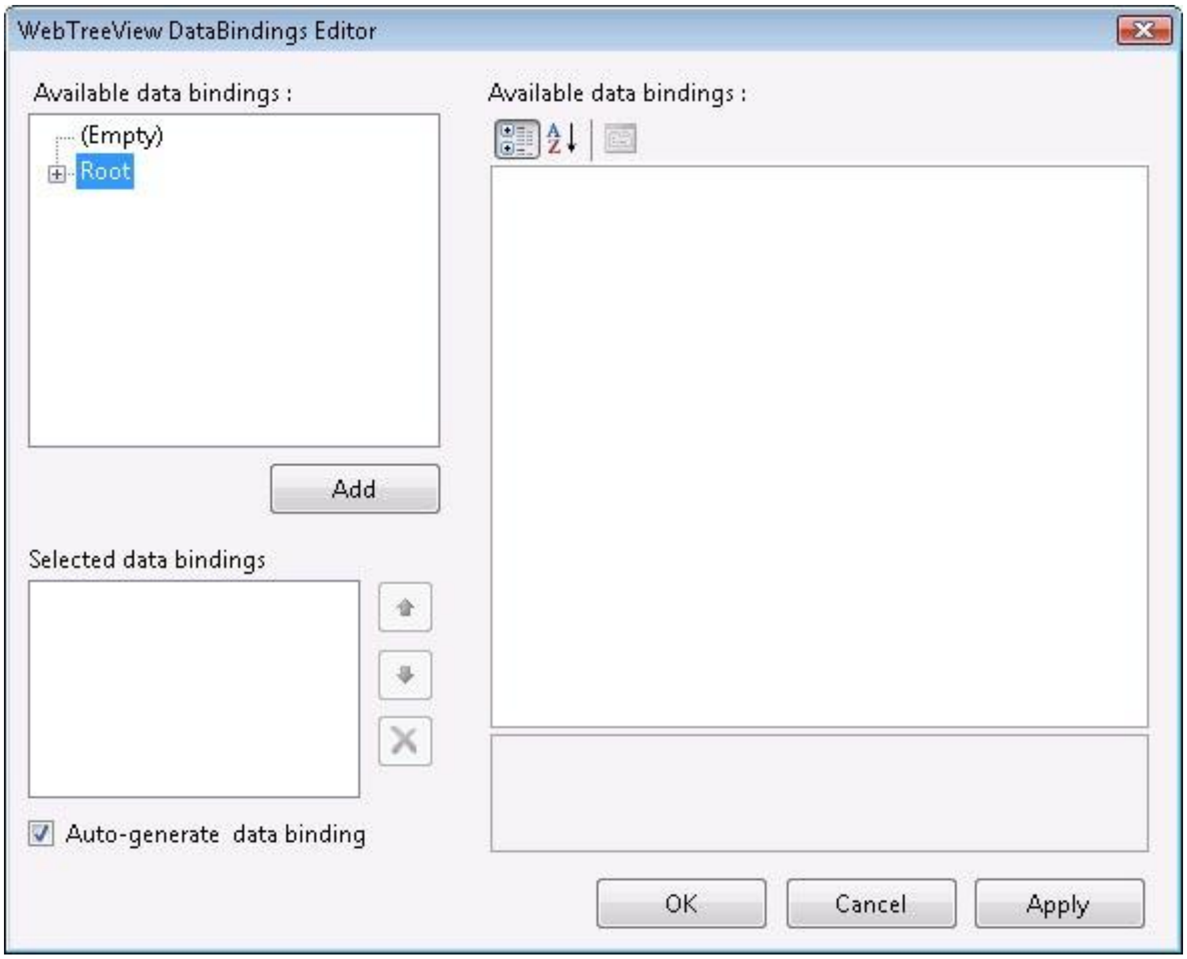
1. You need to create an XMLDataSource. Please set the DataSourceID to the XMLDataSource you've created before.



Data	
(Expressions)	
DataBindings	(Collection)
DataSourceID	XmlDataSource1

Bound a WebTreeView.NET® 1.0 to XMLDataSource

2. Then, click on the DataBindings property and the WebTreeView DataBindings Editor will appear.

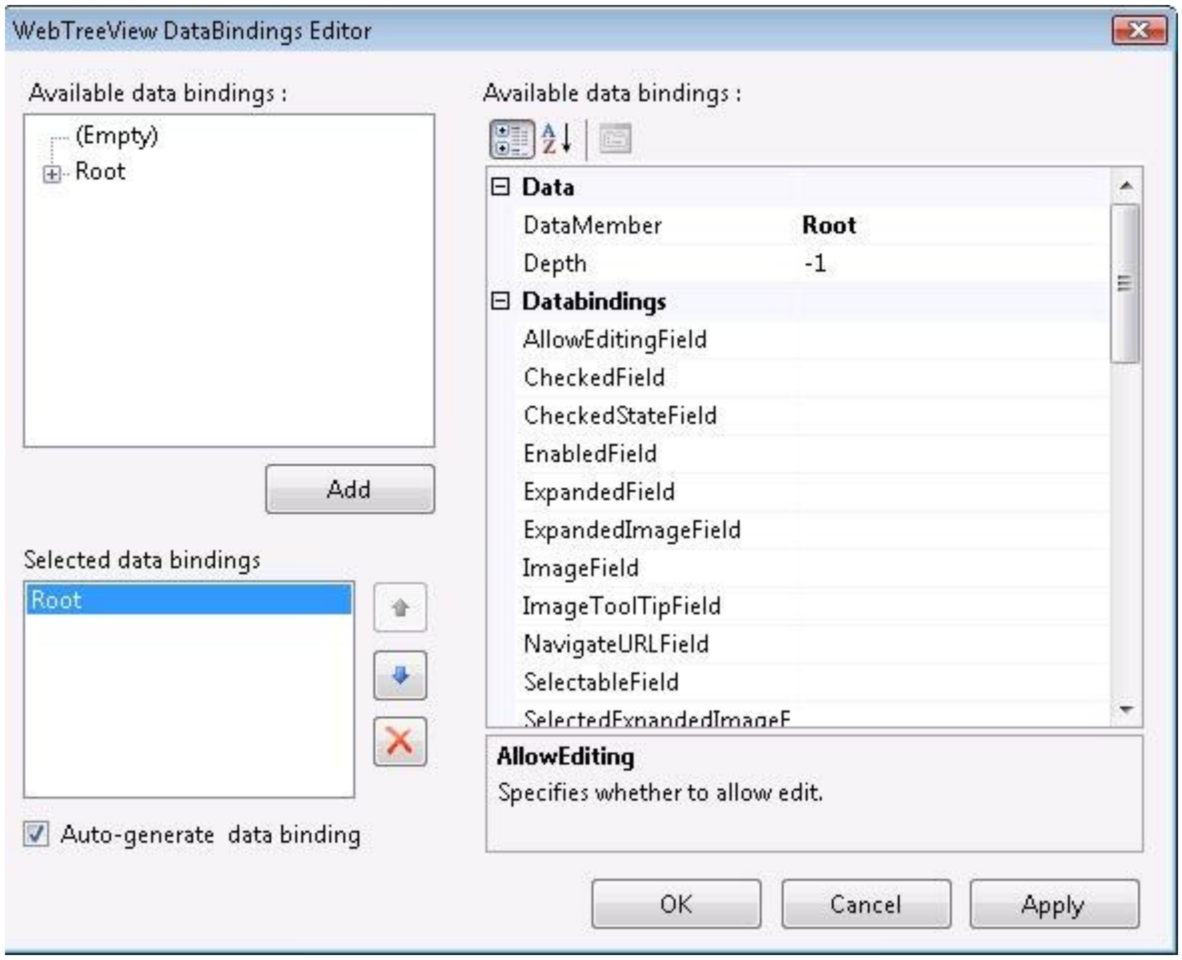

WebTreeView DataBindings Editor
Available data bindings :
(Empty)
Root
Add
Selected data bindings
Auto-generate data binding
Available data bindings :
OK
Cancel
Apply

WebTreeView.NET® 1.0 DataBindings Editor

3. To bind the whole xml, just check the Auto-generate data binding and click ok. The whole XML structure will be bind to the WebTreeView.NET® 1.0.

Advanced Data Binding

Besides the simple, easy binding, WebTreeView.NET® 1.0 is also equipped with a lot of properties you can set. This can be accessible if you add a node to the **Selected data bindings** box and select it. Please see the screenshot below.


WebTreeView DataBindings Editor
Available data bindings :
(Empty)
Root
Add
Selected data bindings
Root
Auto-generate data binding
Available data bindings :
OK
Cancel
Apply

Data
DataMember Root
Depth -1
Databindings
AllowEditingField
CheckedField
CheckedStateField
EnabledField
ExpandedField
ExpandedImageField
ImageField
ImageToolTipField
NavigateURLField
SelectableField
SelectedExpandedImageF

AllowEditing
Specifies whether to allow edit.

WebTreeView.NET® 1.0 DataBindings Editor

Following is the list of available properties you can set.

Data	
DataMember	Gets or sets the Data Member
Depth	Gets or sets the Depth of the databindings
Databindings	
AllowEditingField	Set a node edit ability based on the XML
CheckedField	Initially check or uncheck node when ShowCheckBox property is set to true
CheckedStateField	Initially set the checked state of a node when ShowCheckBox property is set to true
EnabledField	Enable or disable node based on the XML

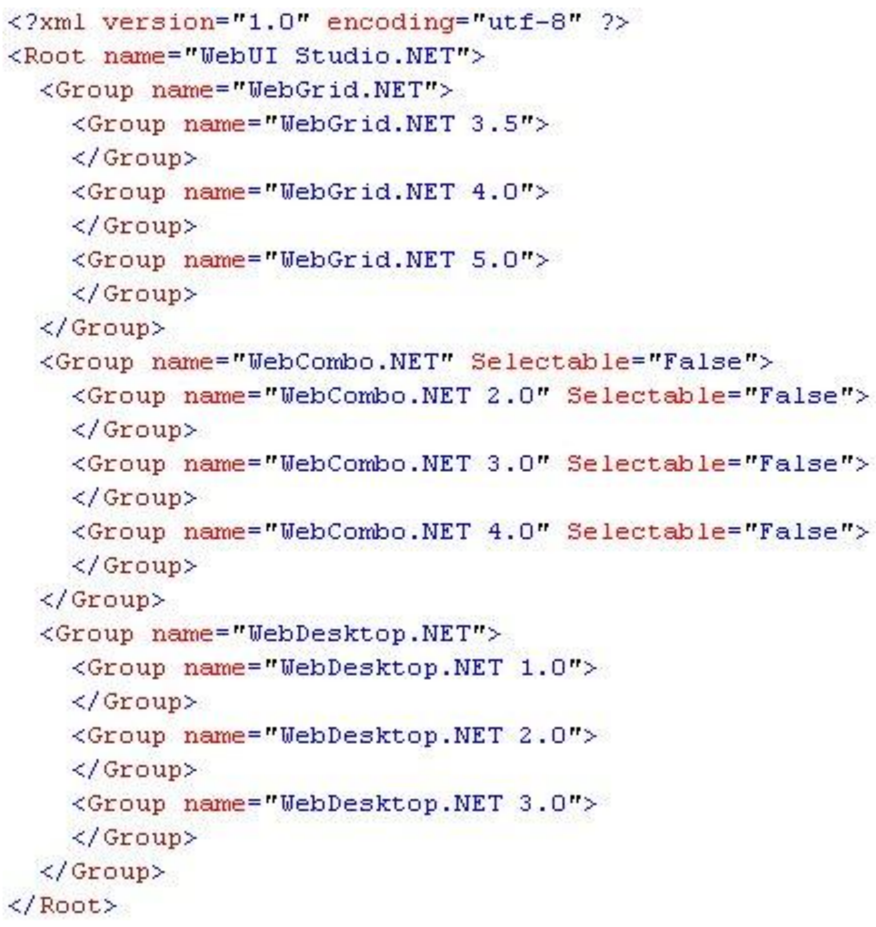
ExpandedField	Expand or collapse node based on the XML
ExpandedImageField	Set the expanded node image icon based on the XML
ImageField	Set the collapsed node image icon based on the XML
ImageToolTipField	Set the tooltip image icon based on the XML
MaxBindingDepthField	Set the node level you want to bind based on the XML
NavigateURLField	Set the URL when a node is selected based on the XML
SelectableField	Specifies if a node is selected or not based on the XML
SelectedExpandedImageFiel	Set the expand icon image of the selected node based on the XML
SelectedField	Specifies if a node is selected or not based on the XML
SelectedImageField	Set the collapse icon image of the selected node based on the XML
ShowCheckBoxField	Show or hide check box based on the XML
TagField	Gets or sets a string which contains data about the WebTreeViewNode
TargetField	Specify the target window when the node is clicked based on the XML
TextField	Set the text of a node that you want to display based on the XML
ToolTipField	Set the tooltip of a node based on the XML
VisibleField	Set the visibility of a node based on the XML

Default Properties

AllowEditing	Set a node edit ability
Checked	Check or uncheck a node
CheckedState	Set the checked state of a node
Enabled	Enable or disable node
Expanded	Expand or collapse node
ExpandedImage	Set the image icon of expanded node
Image	Set the image icon of collapsed node

ImageToolTip	Set the tooltip image icon of a node
MaxBindingDepth	Set the node level you want to bind
NavigateURL	Set the URL when a node is selected
Selectable	Set a node’s selected ability
Selected	Specifies if a node is selected or not
SelectedExpandedImage	Set the expand icon image of the selected node
SelectedImage	Set the collapse icon image of the selected node
ShowCheckBox	Show or hide check box
Tag	Gets or sets a string which contains data about the WebTreeViewNode
Target	Set the target window when a node is clicked
Text	Set the text of a node
ToolTip	Set the tooltip of a node
Value	Set the value of a node
Visible	Set the visibility of a node

Notice that both Databindings properties and Default properties are identical. The difference between them is, you can set each property value under the Databindings based on the value specified on your XML. This approach is designed to provide you with a hassle-free data binding process. You only need to set the structure of your Treeview and its properties in your XML file.

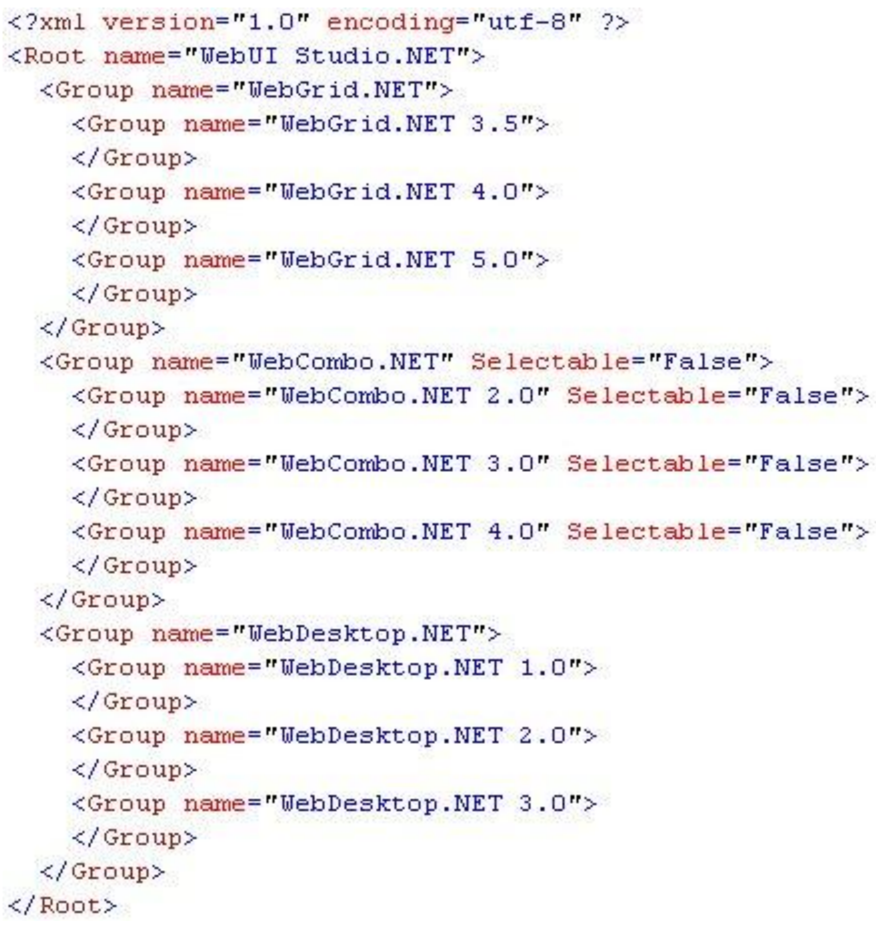


```xml
<?xml version="1.0" encoding="utf-8" ?>
<Root name="WebUI Studio.NET">
  <Group name="WebGrid.NET">
    <Group name="WebGrid.NET 3.5">
    </Group>
    <Group name="WebGrid.NET 4.0">
    </Group>
    <Group name="WebGrid.NET 5.0">
    </Group>
  </Group>
  <Group name="WebCombo.NET" Selectable="False">
    <Group name="WebCombo.NET 2.0" Selectable="False">
    </Group>
    <Group name="WebCombo.NET 3.0" Selectable="False">
    </Group>
    <Group name="WebCombo.NET 4.0" Selectable="False">
    </Group>
  </Group>
  <Group name="WebDesktop.NET">
    <Group name="WebDesktop.NET 1.0">
    </Group>
    <Group name="WebDesktop.NET 2.0">
    </Group>
    <Group name="WebDesktop.NET 3.0">
    </Group>
  </Group>
</Root>
```

A sample of XML file which can be bound to WebTreeView.NET® 1.0

The **Depth** property allows you to controls which node level you want the properties to be applied. One thing you should note is that this only applicable to the same XML name. The default value for this property is **-1** which means all properties will be applied to the same XML name. If you set the value to 0 (the index of a node), the properties will only be applied to all index 0 nodes.

There are some scenarios when you have two different node levels, but both nodes have the same DataMember and you need to set the different properties for each node level. This is the case when you need to change the Depth’s value based on the node’s index that you want to set to. Have a look at the image below and you will see that there are two different node levels, but same DataMember.



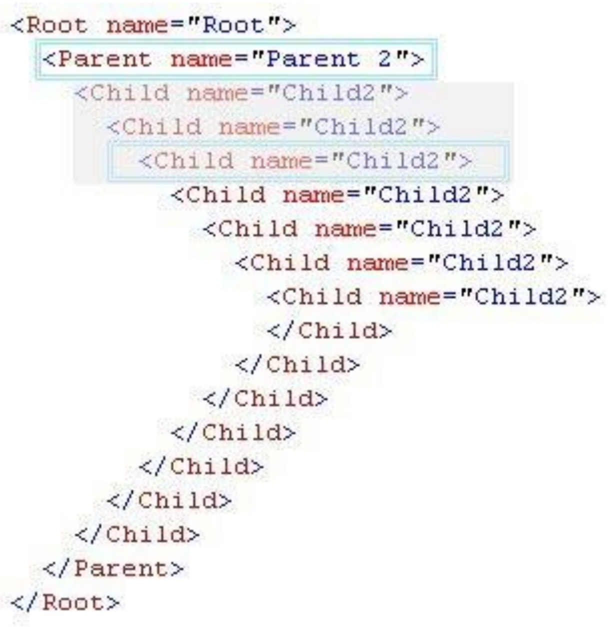
```xml
<?xml version="1.0" encoding="utf-8" ?>
<Root name="WebUI Studio.NET">
  <Group name="WebGrid.NET">
    <Group name="WebGrid.NET 3.5">
    </Group>
    <Group name="WebGrid.NET 4.0">
    </Group>
    <Group name="WebGrid.NET 5.0">
    </Group>
  </Group>
  <Group name="WebCombo.NET" Selectable="False">
    <Group name="WebCombo.NET 2.0" Selectable="False">
    </Group>
    <Group name="WebCombo.NET 3.0" Selectable="False">
    </Group>
    <Group name="WebCombo.NET 4.0" Selectable="False">
    </Group>
  </Group>
  <Group name="WebDesktop.NET">
    <Group name="WebDesktop.NET 1.0">
    </Group>
    <Group name="WebDesktop.NET 2.0">
    </Group>
    <Group name="WebDesktop.NET 3.0">
    </Group>
  </Group>
</Root>
```

Products.xml

If you leave **Depth** to its default value, the Selectable properties will be applied to all nodes. You need to set the Depth to **1**, if you want to apply the **Selectable** to "WebCombo.NET" node only. Set it to 2 if you want to apply the **Selectable** to "WebCombo.NET 2.0" node, "WebCombo.NET 3.0" node, and "WebCombo.NET 4.0" node.

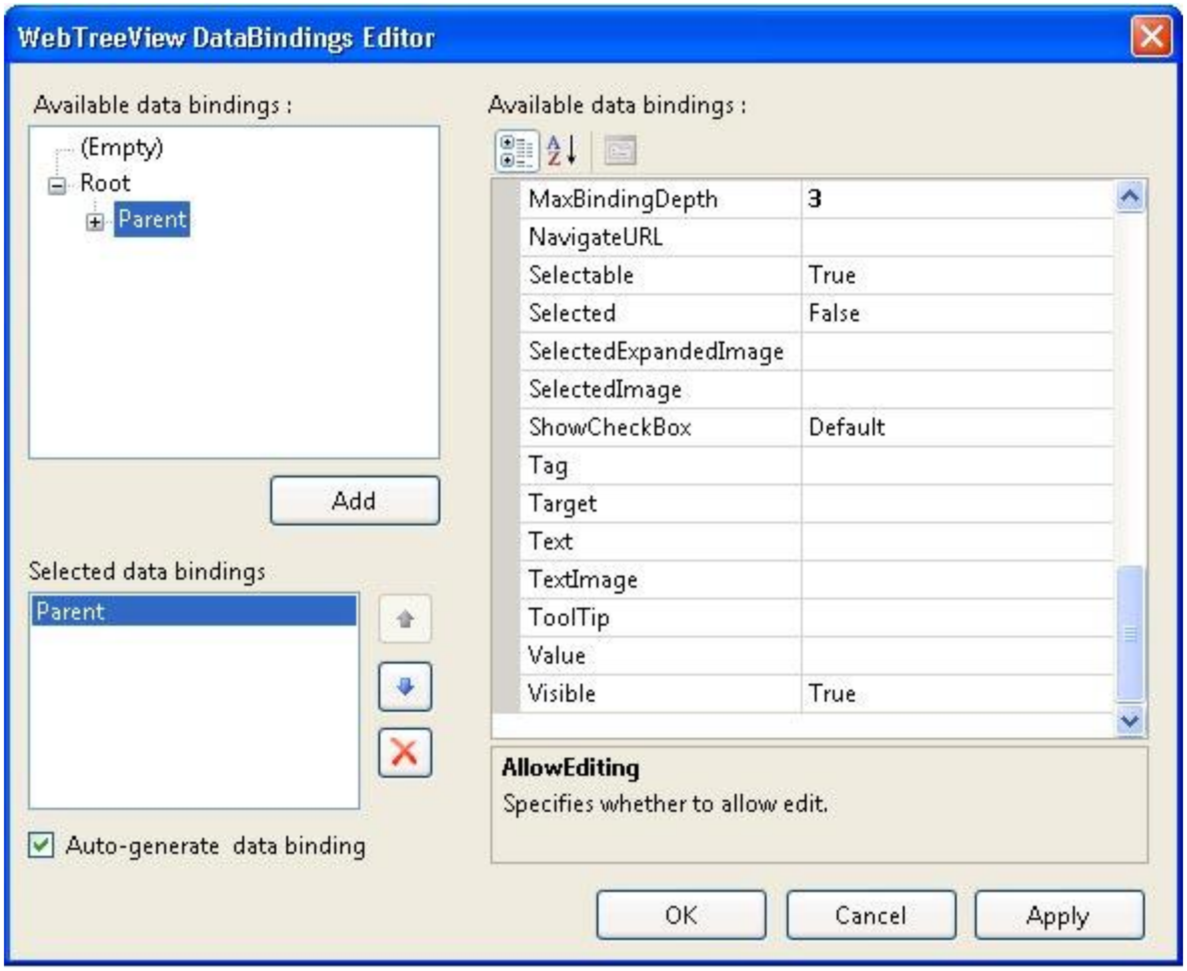
MaxBindingDepth is a property which allows you to select how many child nodes you want to bind. This feature will be a great help when you have deep-structured XML and for example, you want to bind a parent node and the first level child node. WebTreeView provides you **MaxBindingDepth** property to achieve your scenario.

You only need to add a node to the **Selected databindings** box and set the **MaxBindingDepth** value to the desired child node level. Please note that the value of the **MaxBindingDepth** property is based on how many child node level you want to bind, so this is a relative value, not absolute value. For example, you have a deep-structured XML and you want to bind a parent node and level three child nodes only.



```
<Root name="Root">
  <Parent name="Parent 2">
    <Child name="Child2">
      <Child name="Child2">
        <Child name="Child2">
          <Child name="Child2">
            <Child name="Child2">
              <Child name="Child2">
                <Child name="Child2">
                </Child>
              </Child>
            </Child>
          </Child>
        </Child>
      </Child>
    </Child>
  </Parent>
</Root>
```

Notice that the grayed part will be the child nodes of Parent 2 node

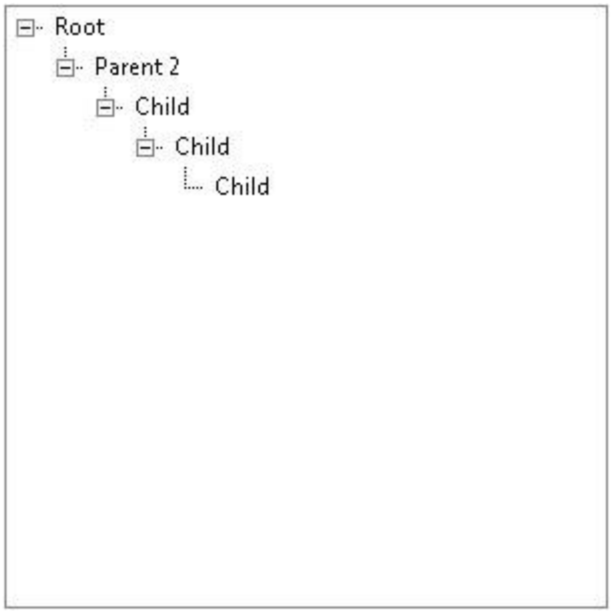

WebTreeView DataBindings Editor
Available data bindings :
(Empty)
Root
Parent
Add
Selected data bindings
Parent
Auto-generate data binding
Available data bindings :
OK
Cancel
Apply

MaxBindingDepth	3
NavigateURL	
Selectable	True
Selected	False
SelectedExpandedImage	
SelectedImage	
ShowCheckBox	Default
Tag	
Target	
Text	
TextImage	
ToolTip	
Value	
Visible	True

AllowEditing
Specifies whether to allow edit.

Select the Parent node and add it to the Selected data bindigs box.
Set the MaxBindingDepth to 3

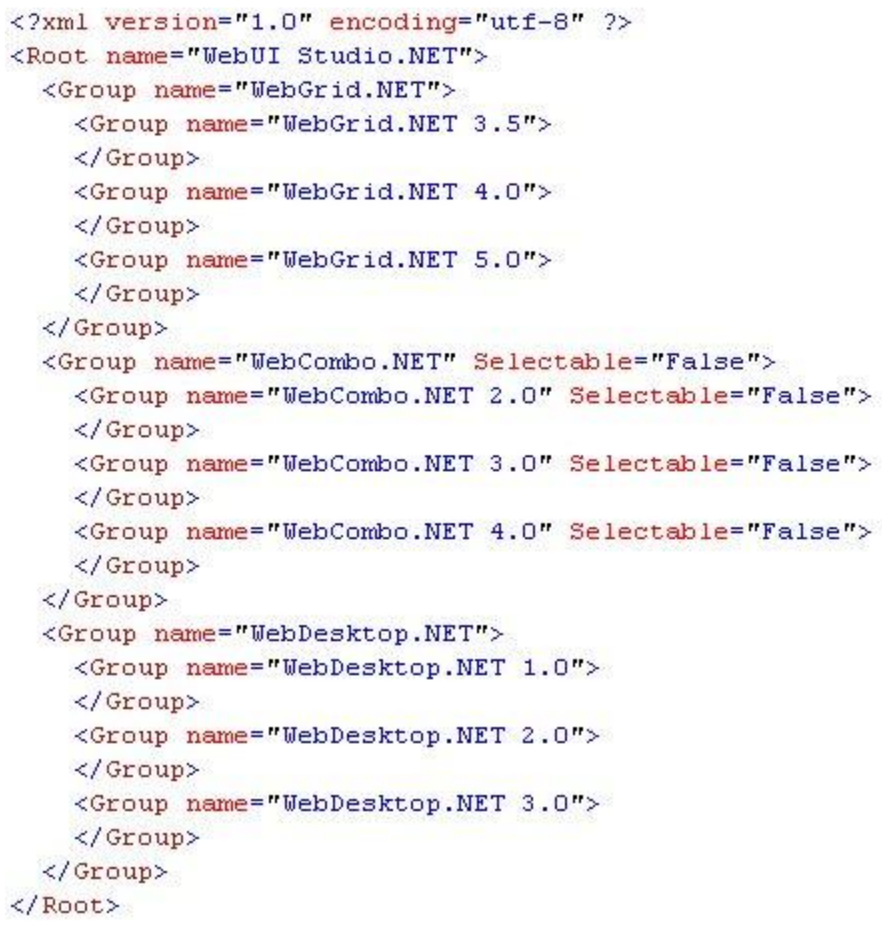
Run your project and it will look like the following screenshot



*A sample utilizing **MaxBindingDepth** property*

How-to: Use Advanced WebTreeView.NET® 1.0 DataBinding

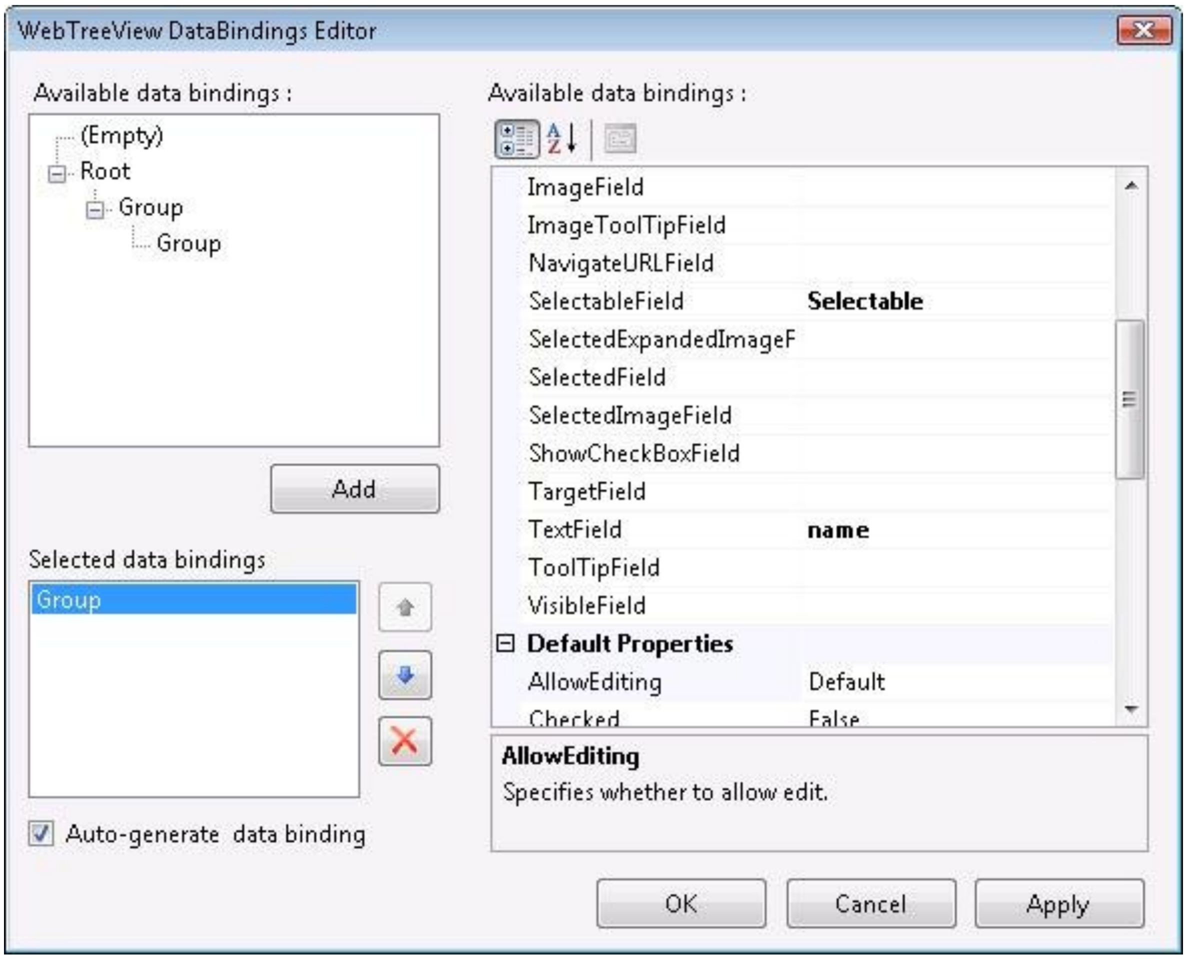
1. Create a WebTreeView.NET® 1.0 instance in your design window.
2. Drag an XMLDataSource and bind it to an XML file, we use Products.xml



```xml
<?xml version="1.0" encoding="utf-8" ?>
<Root name="WebUI Studio.NET">
  <Group name="WebGrid.NET">
    <Group name="WebGrid.NET 3.5">
    </Group>
    <Group name="WebGrid.NET 4.0">
    </Group>
    <Group name="WebGrid.NET 5.0">
    </Group>
  </Group>
  <Group name="WebCombo.NET" Selectable="False">
    <Group name="WebCombo.NET 2.0" Selectable="False">
    </Group>
    <Group name="WebCombo.NET 3.0" Selectable="False">
    </Group>
    <Group name="WebCombo.NET 4.0" Selectable="False">
    </Group>
  </Group>
  <Group name="WebDesktop.NET">
    <Group name="WebDesktop.NET 1.0">
    </Group>
    <Group name="WebDesktop.NET 2.0">
    </Group>
    <Group name="WebDesktop.NET 3.0">
    </Group>
  </Group>
</Root>
```

Products.xml structure, the XML used in this tutorial

3. Click on DataBindings property
4. Select the **Root** and click Add. The **Root** node will be added to **Selected DataBindings**
5. Next, set the SelectableField and TextField as seen on the image below.


WebTreeView DataBindings Editor

Available data bindings :

- (Empty)
- Root
  - Group
    - Group

Selected data bindings

Group

☑ Auto-generate data binding

Available data bindings :

ImageField	
ImageToolTipField	
NavigateURLField	
SelectableField	**Selectable**
SelectedExpandedImageF	
SelectedField	
SelectedImageField	
ShowCheckBoxField	
TargetField	
TextField	**name**
ToolTipField	
VisibleField	
Default Properties	
AllowEditing	Default
Checked	False

AllowEditing
Specifies whether to allow edit.

WebTreeView.NET® 1.0 DataBindings Editor
Set SelectableField and TextField value

Client-side Events

WebTreeView.NET is a highly customizable ASP.NET Treeview control as one of its objective. The following list describes all client-side events in order:

- OnInitialized
  When the WebTreeView is initialized, this event will be invoked. The parameter for this event is **controlId**.
- OnNodeCheckedChanged
  This event will be invoked when you check or uncheck a node. The parameters for this event are **controlId** and **node**.
- OnNodeCollapse
  This event will be invoked when a node is collapsed. The parameters for this event are **controlId** and **node**.

- OnNodeCopy
  This event will be invoked when a node is copied. The parameters for this event are **controlId** and **node**.
- OnNodeExpand
  This event will be invoked when a node is expanded. The parameters for this event are **controlId** and **node**.
- OnNodeMove
  This event will be invoked when a node is moved. The parameters for this event are **controlId** and **node**.
- OnNodeRename
  This event will be invoked when a node is renamed. The parameters for this event are **controlId**, **node, oldName**, and **newName**.
- OnNodeSelect
  This event will be invoked when a node is selected. The parameters for this event are **controlId** and **node**.
- OnUnload
  This event will be invoked when WebTreeView is unloaded. The parameters for this event are **controlId**.