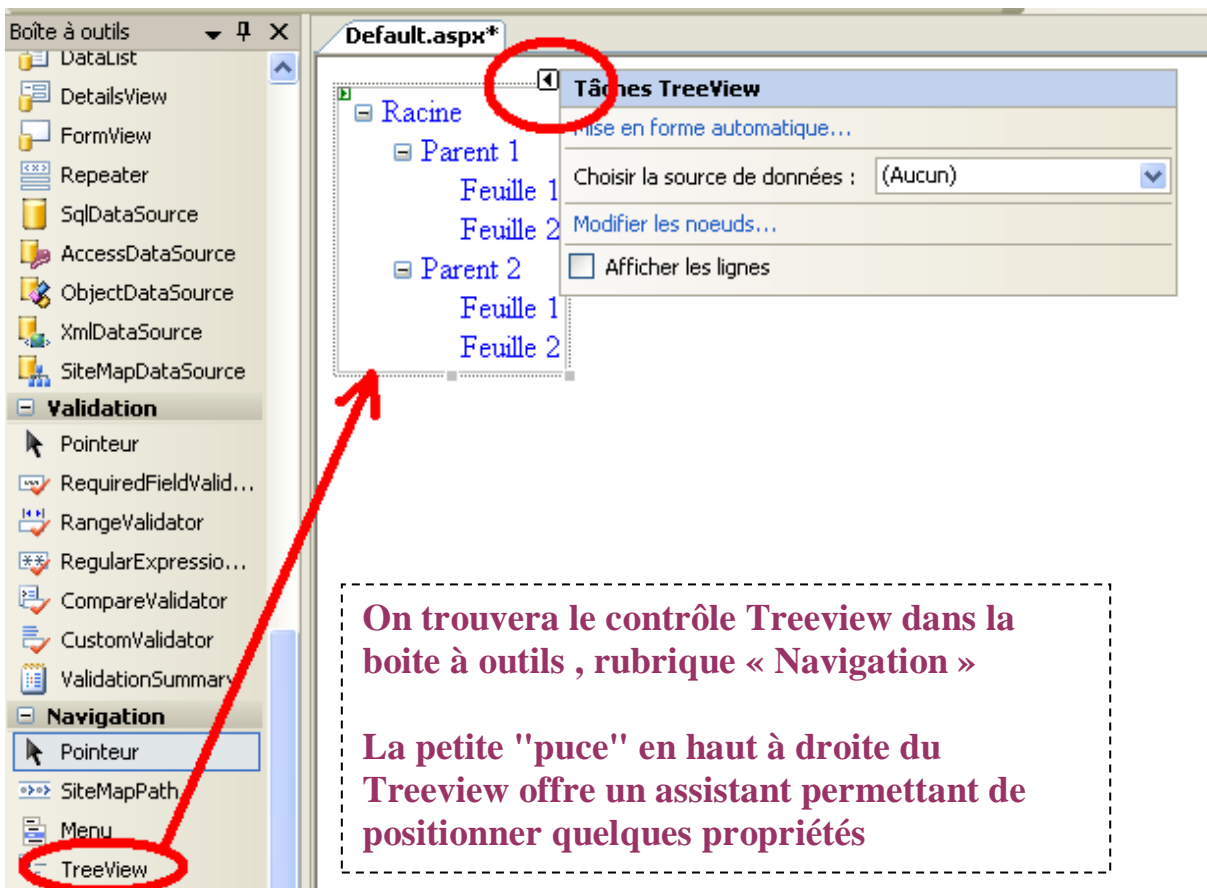


Le contrôle serveur Web TreeView est utilisé pour afficher des données hiérarchiques via une arborescence. Quelques fonctionnalités :

- « Nœud » qui peut être affiché comme texte sélectionnable ou liens hypertexte.
- Apparence personnalisable grâce à des thèmes, des images définies par l'utilisateur et des styles.
- Accès par programme au modèle d'objet TreeView, qui permet de créer dynamiquement des arborescences, de remplir des nœuds, de définir des propriétés...
- Possibilité d'afficher une case à cocher à côté de chaque nœud.
- Liaison de données automatique qui permet aux nœuds du contrôle d'être lié aux données hiérarchiques, telles qu'un document XML, une liste, un tableau...

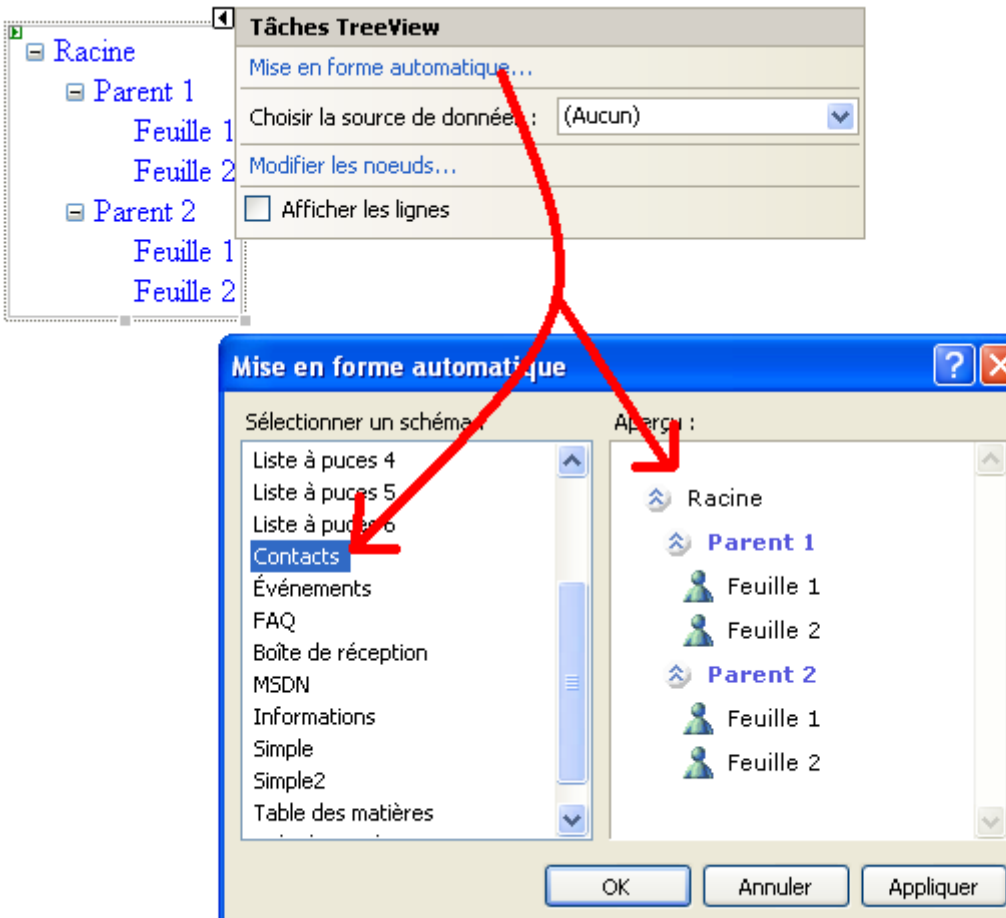
Mise en place du composant

Le contrôle Treeview est un composant visuel qui se trouve dans la boîte à outils

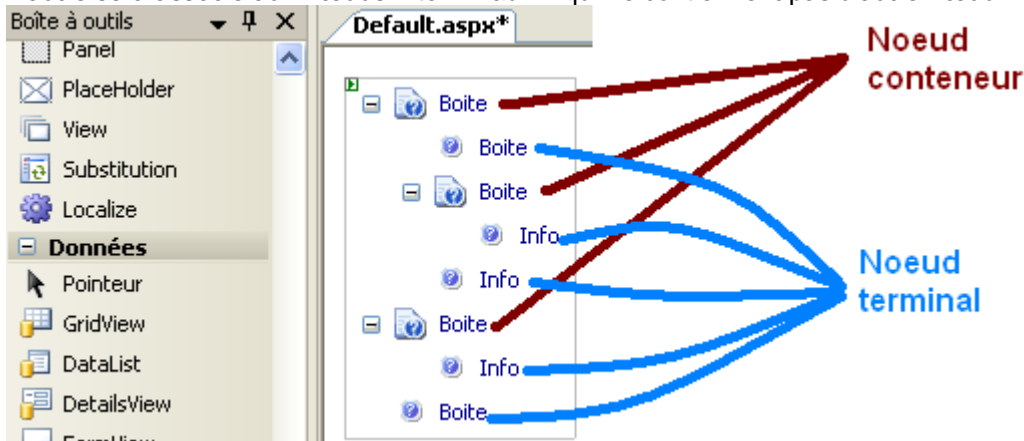


Personnalisation

Les différents nœuds du Treeview peuvent être décorés par un petit symbole.
Un assistant permet d'associer automatiquement un ensemble de symboles aux différents nœuds.



L'assistant propose 2 types de symboles.
L'un sera associé aux conteneurs ou boîtes c'est à dire des nœuds qui contiennent d'autres nœuds.
L'autre sera associé aux nœuds « terminaux » qui ne contiennent pas d'autre nœud.

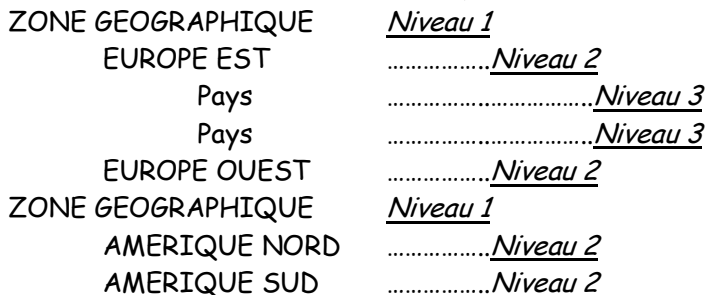


A travers cette structure d'organisation, on reconnaîtra des utilisations célèbres comme l'explorateur de fichiers ou l'organisation des documents XML (et les poupées Russes ;o)

ForeColor	<input type="checkbox"/>
Height	
HoverNodeStyle	
ImageSet	XPFileExplorer
LeafNodeStyle	
LevelStyles	(Collection)
LineImagesFolder	

Pour placer ses propres images, il faut aller positionner finement les propriétés.
Il est possible de déterminer un type d'images par NIVEAU dans l'arborescence.

Par exemple :
Si nous avons l'arborescence suivante on peut déterminer 3 niveaux :



Pour affecter précisément une image à un nœud , par exemple si nous voulons mettre un drapeaux particulier à certains nœuds on pourra utiliser l'assistant :

NB : Attention , après placement des images dans le répertoire du projet, il est parfois nécessaire d'enregistrer, quitter puis revenir pour que l'assistant puisse « voir » les images dans l'arborescence du projet.

Extrait du source aspx

```
...  
<asp:TreeNode ImageUrl="images/flag-fra.gif" Text="France" Value="France"></asp:TreeNode>  
<asp:TreeNode ImageUrl="images/flag-uk.gif" Text="G.B" Value="G.B"></asp:TreeNode>  
...
```

On pourra dénombrer 170 propriétés qui permettent de modifier l'apparence des nœuds.
Le nombre est impressionnant mais la plus grande partie est le reflet des classiques feuilles de style.
Par exemple pour que le nœud sélectionné apparaisse surligné de jaune :

```
<asp:TreeView ID="TreeView1" runat="server" SelectedNodeStyle-BackColor="yellow" ...
```

Evenements

Evènement SelectedNodeChanged

Click sur Italie

- ☐ ZONE EUROPE
 - ☐ OUEST
 - France
 - G.B
 - Italie
 - Allemagne
 - EST
- ☐ ZONE AFRIQUE
 - NORD
- ☐ CONTINENT AMERICAIN
 - NORD
 - SUD



L'évènement le plus important est sans aucun doute celui qui sera invoqué lorsque l'utilisateur va cliquer sur une partie de l'arborescence. Il s'agit de `SelectedNodeChanged`

A gauche un exemple avec un label (Propriété `BorderStyle` à 'Ridge') en haut qui affiche la propriété `Text` du nœud qui a reçu le click.

Voici le contenu de la méthode chargée de traiter l'évènement.

NB : L'objet `sender` passé en argument n'est pas un nœud mais le Treeview complet.

C'est pourquoi il est nécessaire de demander au composant une information sur le nœud sélectionné.

Cela se fait via la propriété `SelectedNode`.

```
protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)  
{  
    Label1.Text = "Click sur " + ((TreeView)sender).SelectedNode.Text;  
}
```

Evènement TreeNodeExpanded

Il peut être utile de savoir quand une branche est "dépliée".

L'évènement `TreeNodeExpanded` nous apportera cette information. On notera le confort apporté par l'argument qui nous envoie directement le nœud incriminé.

```
protected void TreeView1_TreeNodeExpanded(object sender, TreeNodeEventArgs e)  
{  
    TextBox1.Text = "TreeNodeExpanded : " + System.Environment.NewLine;  
    TextBox1.Text += "Node.Text ->" + e.Node.Text;  
}
```

Propriété Value

Il serait plus intéressant d'obtenir une valeur qui a du sens ; dans l'exemple proposé afficher le nom de la capitale sur le click d'un pays n'est pas très difficile.

Il faut simplement positionner la propriété `value` du nœud.

```
<asp:TreeNode ImageUrl="images/flag-ita.gif" Text="Italie" Value="Rome"></asp:TreeNode>  
<asp:TreeNode ImageUrl="images/flag-ger.gif" Text="Allemagne" Value="Berlin"></..
```

Puis dans méthode chargée de traiter l'événement

```
protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
{
    Label1.Text = "Click sur " + ((TreeView)sender).SelectedNode.Value;
}
```



Méthodes : Comment faire / défaire des nœuds ?



Faire

Assez simple puisqu'il suffit d'utiliser la méthode Nodes.Add.

Par exemple, avec une zone de texte pour donner le texte du nœud et un bouton :

```
protected void BtnAjout_Click(object sender, EventArgs e)
{
    TreeView1.Nodes.Add(new TreeNode(TextBox1.Text));
}
```

Les nœuds sont alors ajoutés au niveau racine, après le dernier nœud connu.

Sans information précise de placement, ou pourraient-ils apparaître ?

Cependant, on aura probablement besoin de pouvoir positionner finement un nouveau nœud par rapport à un nœud existant.

La méthode Add nous rendra ce service :

```
TreeNode o = new TreeNode(); // on peut donc ajouter un nœud à un autre nœud
o.ChildNodes.Add(new TreeNode());
```

Encore faut-il pouvoir récupérer un objet nœud pour faire l'ajout.

On pourra pour cela utiliser les information ramenées sur la sélection d'un nœud.

- [-] Renault
 - [-] cabriolet
 - megane II
 - R19
 - [-] berline
 - [-] clio
 - ajout dyn.
- [-] Audi
 - cabriolet
 - berline

```
protected void Page_Load(object sender, EventArgs e)
{
    TreeView1.SelectedNodeStyle.BackColor = System.Drawing.Color.Yellow;
}
protected void BtnAjout_Click(object sender, EventArgs e)
{
    if (TreeView1.SelectedNode != null) // Si un nœud est sélectionné
        TreeView1.SelectedNode.ChildNodes.Add(new TreeNode(TextBox1.Text));
    else // Si aucun nœud n'est sélectionné
        TreeView1.Nodes.Add(new TreeNode(TextBox1.Text));
}
```

Défaire

Pour supprimer un nœud il faut toujours passer par une méthode de la collection ChildNodes.

Donc il faut « remonter » au nœud parent pour lui demander de supprimer un nœud enfant.

Sauf bien entendu si il s'agit d'un nœud racine.

Voici le code :

```
protected void BtnSuppr_Click(object sender, EventArgs e)
{
    if (TreeView1.SelectedNode.Parent != null)
        // Passer par le nœud parent pour supprimer le nœud enfant sélectionné
        TreeView1.SelectedNode.Parent.ChildNodes.Remove(TreeView1.SelectedNode);
    else // Sauf si il s'agit d'un nœud "racine"
        TreeView1.Nodes.Remove(TreeView1.SelectedNode);
}
```

Question : Comment vider un Treeview sans utiliser une routine récursive ?

Réponse : `TreeView1.Nodes.Clear(); // Simplement ...`

Code commenté :

Réalisation d'un explorateur de fichier.

Problématique : il ne faut pas charger le TreeView entièrement (trop lent et puis on a pas forcément besoin de tout) mais uniquement au fur et à mesure de la demande (click sur un nœud).

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        // Au départ l'arbre doit être "replié" sur lui-même
        TreeView1.ExpandDepth = 0;

        // Obtention du repertoire courant
        string nomRepertoire = Server.MapPath("~/");
        TreeNode nouveauNoeud = new TreeNode(nomRepertoire);

        // placer using System.IO
        // Si le noeud a des enfants, il faudra déclencher un évènement pour le remplir
        // La propriété "magique" PopulateOnDemand fait ce lien
        if (Directory.GetFiles(nomRepertoire).Length > 0 ||
            Directory.GetDirectories(nomRepertoire).Length > 0)
            nouveauNoeud.PopulateOnDemand = true;

        TreeView1.Nodes.Add(nouveauNoeud);
    }
}

// la méthode Populate sera appelée une fois pour remplir le noeud courant
// à condition que celui-ci aie la propriété PopulateOnDemand à true.
protected void TreeView1_TreeNodePopulate(object sender, TreeNodeEventArgs e)
{
    TreeNode unNoeud = e.Node;

    // Si le répertoire contient des répertoires il faut les ajouter
    string[] aListeRep = System.IO.Directory.GetDirectories(unNoeud.Value);
    foreach (string unRep in aListeRep)
    {
        TreeNode nouveauNoeud = new TreeNode();
        nouveauNoeud.Text = System.IO.Path.GetFileName(unRep);
        nouveauNoeud.Value = unRep;

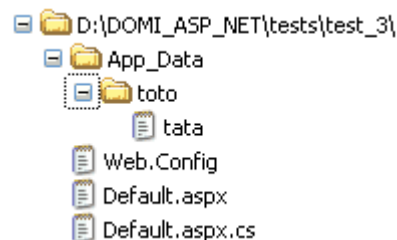
        // Déplier/Replier le nœud simplement en cliquant dessus
        nouveauNoeud.SelectAction = TreeNodeSelectAction.Expand;

        if (System.IO.Directory.GetFiles(unRep).Length > 0 ||
            System.IO.Directory.GetDirectories(unRep).Length > 0)
            nouveauNoeud.PopulateOnDemand = true;

        unNoeud.ChildNodes.Add(nouveauNoeud);
    }

    // Si le répertoire contient des fichiers il faut les ajouter
    string[] aListeFic = System.IO.Directory.GetFiles(unNoeud.Value);
    foreach (string unFic in aListeFic)
    {
        TreeNode nouveauNoeud = new TreeNode();
        nouveauNoeud.Text = System.IO.Path.GetFileName(unFic);
        nouveauNoeud.Value = unFic;
        nouveauNoeud.SelectAction = TreeNodeSelectAction.Select;

        unNoeud.ChildNodes.Add(nouveauNoeud);
    }
}
}
```



Charger des données XML automatiquement

Il est possible de relier le Treeview à une source de données. Le fichier XML, de part son organisation, est la source idéale. Cela peut se faire très facilement à la souris. Faire un click droit sur l'explorateur de solutions pour ajouter un fichier XML.

Copier le source XML suivant :

```
<?xml version="1.0" standalone="yes"?>
<vehicule>
  <voiture marque="jaguar" value="jaguar">
    <modele nom="S-Type V8" value="S-Type V8">
      <caracteristiques km="20000" carburant="essence"
vendeur="9564"></caracteristiques>
      <caracteristiques km="120000" carburant="essence"
vendeur="4554"></caracteristiques>
      <caracteristiques km="90100" carburant="essence" vendeur="A649"></caracteristiques>
    </modele>
    <modele nom="Modena F1" value="Modena F1">
      <caracteristiques km="55000" carburant="essence" vendeur="8864"></caracteristiques>
      <caracteristiques km="145000" carburant="essence" vendeur="4599"></caracteristiques>
      <caracteristiques km="95100" carburant="essence" vendeur="zr89"></caracteristiques>
    </modele>
  </voiture>
  <campingcar marque="volvo" value="volvo">
    <modele nom="capucine" value="capucine">
      <caracteristiques km="300000" carburant="diesel" vendeur="9IP4"></caracteristiques>
    </modele>
  </campingcar>
</vehicule>
```

Configurer la source de données en mode design et c'est terminé.

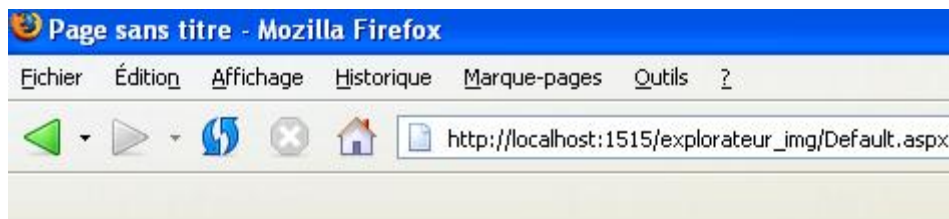
The screenshot illustrates the configuration of an XML data source for a TreeView control in Visual Studio. It is divided into three main sections:

- Top Section:** Shows a file explorer window with a red arrow pointing to the 'monFichierXML.xml' file in the 'D:\...\test_4\' directory. To the right, the 'Modèles Visual Studio installés' (Visual Studio installed templates) pane is visible, with 'Fichier XML' (XML File) selected.
- Middle Section:** Shows the 'Tâches TreeView' (TreeView Tasks) pane. The 'Choisir la source de données' (Choose data source) dropdown is set to 'XmlDataSource1'. The 'Configurer la source de données...' (Configure data source...) option is circled in red.
- Bottom Section:** Shows the 'Configurer la source de données - XmlDataSource1' (Configure data source - XmlDataSource1) dialog box. The 'Fichier de données' (Data file) field contains the path '~\monFichierXML.xml', and the 'Parcourir' (Browse) button is circled in red. Below this, a 'Sélectionner un fichier XML' (Select XML file) dialog is open, showing the project files and 'monFichierXML.xml' selected in the 'Contenu du dossier' (Folder content) pane.

Etude de cas : Réalisation d'un visualisateur d'images :

Le TreeView permettra de visualiser des répertoires.

Un click sur un nœud permettra de visualiser automatiquement les images contenues dans celui-ci.



Pour cette application, il faut être capable de donner au moteur ASP les chemins qu'il est en droit d'attendre pour afficher les nœuds et les images. Or si pour les nœuds du TreeView, les méthodes des classes de System.IO vont être utilisées, il faudra trouver autre chose pour les images qui elles ont besoin d'un chemin de fichier relatif à la racine du site.

- ▼ Images
 - ▷ musique
 - ▷ nature
 - ▷ urbain

belle_ile



Extrait de l'aide Microsoft.

Propriété	Description
ApplicationPath	Obtient le chemin d'accès racine de l'application actuelle, quel que soit l'endroit de l'application où vous le demandez. Dans l'exemple, la propriété retourne les éléments suivants : /
CurrentExecutionFilePath	Obtient le chemin d'accès virtuel de la demande actuelle. Diffère de <code>FilePath</code> en ce que CurrentExecutionFilePath est correct si la demande a été redirigée dans le code serveur. Dans l'exemple, la propriété retourne les éléments suivants : <code>/MyApplication/MyPages/Default.aspx</code> Si vous placez la propriété dans un code qui s'exécute à la suite d'un appel à <code>Transfer</code> ou <code>Execute</code> , le chemin d'accès reflète l'emplacement du code.
FilePath	Obtient le chemin d'accès virtuel de la demande actuelle. Dans l'exemple, la propriété retourne les éléments suivants : <code>/MyApplication/MyPages/Default.aspx</code> Contrairement à CurrentExecutionFilePath , FilePath ne reflète pas de transferts côté serveur.
Path	Obtient le chemin d'accès virtuel de la demande actuelle. Dans l'exemple, la propriété retourne les éléments suivants : <code>/MyApplication/MyPages/default.aspx</code>
PhysicalApplicationPath	Obtient le chemin d'accès physique au système de fichiers du répertoire racine de l'application en cours d'exécution. Dans l'exemple, la propriété retourne les éléments suivants : <code>C:\inetpub\wwwroot\</code>
PhysicalPath	Obtient le chemin d'accès physique au système de fichiers qui correspond à l'URL demandée. Dans l'exemple, la propriété retourne les éléments suivants : <code>C:\inetpub\wwwroot\MyApplication\MyPages\default.aspx</code>

ANNEXE :

Visualisateur d'images : Solution 1

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        // Au départ l'arbre doit être "replié" sur lui-même
        TreeView1.ExpandDepth = 0;

        // Obtention du repertoire courant
        string nomRepertoire = Server.MapPath("~/images");
        TreeNode nouveauNoeud = new TreeNode(nomRepertoire);
        // Si le noeud a des enfants, il faudra déclencher un évènement pour le remplir
        // La propriété "magique" PopulateOnDemand fait ce lien
        if (Directory.GetFiles(nomRepertoire).Length > 0 ||
            Directory.GetDirectories(nomRepertoire).Length > 0)
            nouveauNoeud.PopulateOnDemand = true;

        TreeView1.Nodes.Add(nouveauNoeud);
    }
}

protected void TreeView1_TreeNodePopulate(object sender, TreeNodeEventArgs e)
{
    TreeNode unNoeud = e.Node;

    // Si le répertoire contient des répertoires il faut les ajouter
    string[] aListeRep = System.IO.Directory.GetDirectories(unNoeud.Value);
    foreach (string unRep in aListeRep)
    {
        TreeNode nouveauNoeud = new TreeNode();
        nouveauNoeud.Text = System.IO.Path.GetFileName(unRep);
        nouveauNoeud.Value = unRep;

        // Déplier/Replier le nœud simplement en cliquant dessus
        nouveauNoeud.SelectAction = TreeNodeSelectAction.Expand;

        if (System.IO.Directory.GetFiles(unRep).Length > 0 ||
            System.IO.Directory.GetDirectories(unRep).Length > 0)
            nouveauNoeud.PopulateOnDemand = true;

        unNoeud.ChildNodes.Add(nouveauNoeud);
    }
    // Si le répertoire contient des fichiers il faut les ajouter
    string[] aListeFic = System.IO.Directory.GetFiles(unNoeud.Value);
    foreach (string unFic in aListeFic)
    {
        TreeNode nouveauNoeud = new TreeNode();
        nouveauNoeud.Text = System.IO.Path.GetFileName(unFic);
        nouveauNoeud.SelectAction = TreeNodeSelectAction.Select;
        unNoeud.ChildNodes.Add(nouveauNoeud);
        nouveauNoeud.Value = unFic;
    }
}

protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
{
    string nomImage = ((TreeView)sender).SelectedNode.Value;

    string racine = Server.MapPath("~/images");
    Image1.ImageUrl = @"~/images" + nomImage.Substring(racine.Length);
}

```



Using the TreeView Control and a DataList to Create an Online Image Gallery

By [Scott Mitchell](#)

Introduction

[ASP.NET version 2.0](#) includes a wide array of Web controls not found in previous versions. One such control is the TreeView, which is ideal for displaying hierarchical data. The TreeView control can be bound to a hierarchical data source such as the XmlDataSource or SiteMapDataSource, or can be constructed programmatically. (For an example of using the TreeView to display a site map using the SiteMapDataSource, check out the [Examining ASP.NET 2.0's Site Navigation](#) article series.)

One common source of hierarchical data is the web server's file system. In many scenarios, there may be a folder that contains subfolders and files that the user needs to be able to browse. Using the classes in the [System.IO namespace](#), we can programmatically populate the TreeView with the directory structure of our website. Then, when the user clicks a folder, the selected folder's files can be displayed.

In this article we will examine how to create a simple image gallery web page that's a breeze to maintain. The image gallery lacks the bells and whistles found in more complex and feature-rich image galleries, but this one is a cinch to deploy and maintain. We'll be using two Web controls: a TreeView to list the folders and subfolders in which the images reside; and a DataList control that lists each image in the selected folder. Read on to learn more!

This is one of many image-related articles here on 4Guys. Others include: [A Robust Image Gallery for ASP.NET](#); [Displaying a List of Scaled Images](#); and [True Image Resizing](#)...

Displaying the TreeView

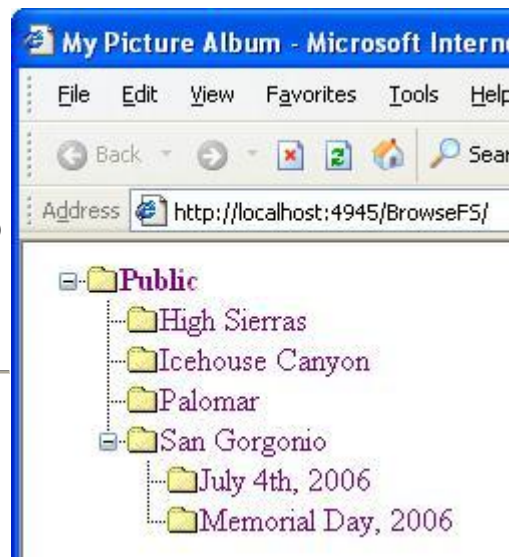
The application we are creating here has two components:

- A TreeView control that lists the folders rooted at some specified folder, and
- A data Web control to list the files in the selected folder. For this article we'll use a DataList, but in other scenarios a [GridView](#), DetailsView, FormView, or Repeater might make more sense.

Let's begin by programmatically creating the TreeView. The ASP.NET 2.0 TreeView's data can be assigned declaratively (through an XmlDataSource or SiteMapDataSource) or programmatically. Since there's no built-in data source control for returning directory information, we'll need to programmatically bind the data to the TreeView. In particular, we need to get information about the root folder and then recurse through the subfolders, adding a TreeNode to the TreeView for each folder encountered.

The .NET Framework contains a plethora of classes for working with the file system; these classes are found in the [System.IO namespace](#). (There are [a bevy of FAQs](#) on this namespace at [ASPFAQs.com](#).) One such class is [DirectoryInfo](#), which has methods for returning a given directory's subdirectories ([GetDirectories\(\)](#)) as well as a method for returning all of the files in the directory ([GetFiles\(\)](#)).

The following code shows how to populate the TreeView [PictureTree](#) with the folder hierarchy rooted at the folder specified by the [VirtualImageRoot](#) constant. The [AddNodeAndDescendents](#) method is



the workhorse here, recursing through the folder structure. (For more information on recursion, check out an old article of mine - [Recursion, Why It's Cool.](#))

```
Private Const VirtualImageRoot = "~/Images/Public/"
Private Sub PopulateTree()
    'Populate the tree based on the subfolders of the specified
VirtualImageRoot
    Dim rootFolder As New
DirectoryInfo(Server.MapPath(VirtualImageRoot))
    Dim root As TreeNode = AddNodeAndDescendents(rootFolder,
Nothing)

    'Add the root to the TreeView
    PictureTree.Nodes.Add(root)
End Sub

Private Function AddNodeAndDescendents(ByVal folder As
DirectoryInfo, ByVal parentNode As TreeNode) As TreeNode
    'Add the TreeNode, displaying the folder's name and storing the
full path to the folder as the value...
    Dim virtualFolderPath As String
    If parentNode Is Nothing Then
        virtualFolderPath = VirtualImageRoot
    Else
        virtualFolderPath = parentNode.Value & folder.Name & "/"
    End If

    Dim node As New TreeNode(folder.Name, virtualFolderPath)

    'Recurse through this folder's subfolders
    Dim subFolders As DirectoryInfo() = folder.GetDirectories()
    For Each subFolder As DirectoryInfo In subFolders
        Dim child As TreeNode = AddNodeAndDescendents(subFolder,
node)
        node.ChildNodes.Add(child)
    Next

    Return node        'Return the new TreeNode
End Function
```

Each node in the tree has an associated display value and hidden value. The display value is what is shown in the tree; the hidden value is persisted across postbacks and used to associate some extra bit of information with each node. These two values are specified in the `TreeNode` constructor - `Dim node As New TreeNode(text, value)`. The folder's name is used as the *text* while the **virtual path** is used as the *value*. The virtual path is the path by which a client would request the image - `/Images/Public/Picture1.jpg`, for example - as compared to the *physical path*, which would be the actual file path on the web server's file system (such as `C:\MySites\ImageGallery\Images\Public\Picture1.jpg`). See [Using Server.MapPath\(\)](#) for more information on the differences between virtual and physical paths and how `Server.MapPath()` can be used to translate from a virtual path to a physical one.

Displaying the Files in the Selected Folder

With the file system structure displayed in the TreeView, all that remains is to display the files in the selected folder. To accomplish this, we can add a data Web control to the page and bind its data to the list of files in the currently selected folder. But how can we tell what folder is selected, and how do we know when the user wants to view the files in a different folder?

The TreeView has a `SelectedValue` [property](#) that returns the *value* of the currently selected node. Recall that we've assigned the folder's virtual path to its corresponding node's *value*. Given the path to the selected folder, we can get the list of files in that folder by creating a `DirectoryInfo` object and using its `GetFiles()` method. This returns an array of `FileInfo` objects that can be bound to the data Web control (a `DataList`, for our image gallery).

When a TreeView node is clicked, a postback ensues and the TreeView's `SelectedNodeChanged` [event](#) fires. When this event fires, we want to rebind the data to the data Web control, using the newly selected folder.

These two tasks are accomplished using the following code. The `DisplayPicturesInFolder(virtualFolderPath)` method gets the files in the specified folder and binds them to `DataList PicturesInFolder`. The `SelectedNodeChanged` event handler simply calls `DisplayPicturesInFolder(virtualFolderPath)`, passing in the *value* of the selected node.

```
Protected Sub PictureTree_SelectedNodeChanged(ByVal sender As
Object, ByVal e As System.EventArgs) Handles
PictureTree.SelectedNodeChanged
    'Refresh the DataList whenever a new node is selected
    DisplayPicturesInFolder(PictureTree.SelectedValue)
End Sub

Private Sub DisplayPicturesInFolder(ByVal virtualFolderPath As
String)
    'Security check: make sure folderPath starts with
VirtualImageRoot and doesn't include any ".."
    If Not virtualFolderPath.StartsWith(VirtualImageRoot) OrElse
virtualFolderPath.IndexOf("..") >= 0 Then
        Throw New ApplicationException("Attempting to view a folder
outside of the public image folder!")
    End If

    'Get information about the files in the specified folder
    Dim folder As New
DirectoryInfo(Server.MapPath(virtualFolderPath))
    Dim fileList As FileInfo() = folder.GetFiles()

    PicturesInFolder.DataSource = fileList
    PicturesInFolder.DataBind()
End Sub
```

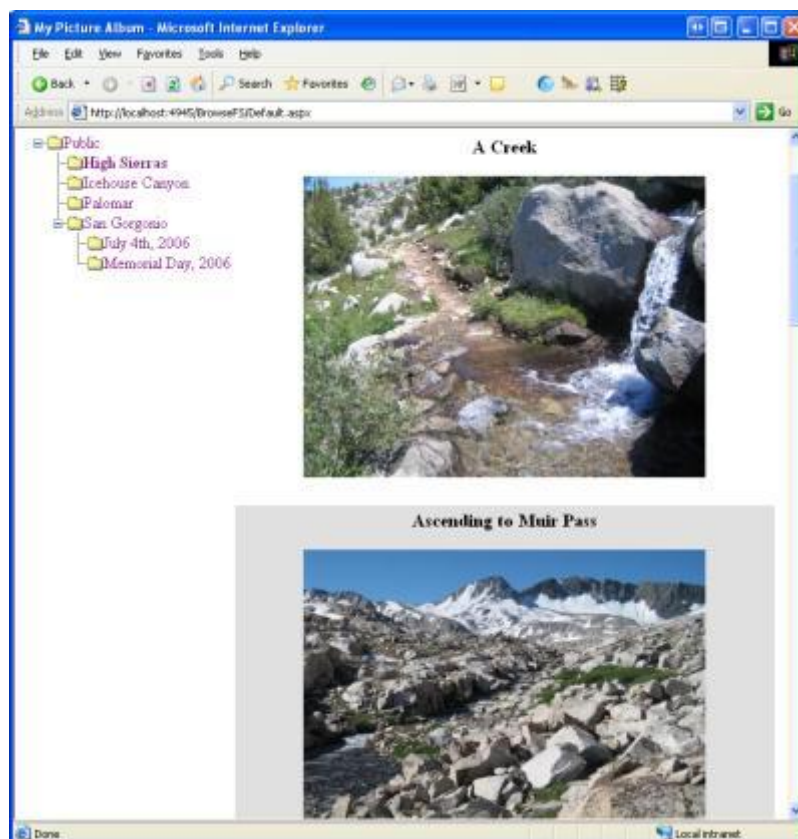
Note: The first line of code in the `DisplayPicturesInFolder(virtualFolderPath)` method does a quick security check to make sure that the path being requested is within the folder hierarchy specified by the `VirtualImageRoot` constant.

The `PicturesInFolder` `DataList` contains a single `ItemTemplate` that displays the name of the image (without the extension) and includes an `Image` Web control for displaying the picture:

```
<asp:DataList ID="PicturesInFolder" runat="server" Width="100%"
CellPadding="5">
  <ItemTemplate>
    <h3><asp:Label runat="server" ID="FileNameLabel" Text='<%#
System.IO.Path.GetFileNameWithoutExtension(Eval("Name"))
%>'></asp:Label></h3>

    <asp:Image runat="server" ID="Picture" ImageUrl='<%#
PictureTree.SelectedValue & Eval("Name").ToString() %>' />
    <br /><br />
  </ItemTemplate>
  <AlternatingItemStyle BackColor="#E0E0E0" />
</asp:DataList>
```

The end result is a simple image gallery application. The `TreeView` lists the folders in the file system rooted from a specified starting folder. Clicking on a particular folder in the `TreeView` displays its images on the right. This image gallery application is a breeze to deploy (as it's just a single ASP.NET page) and to update. To add, edit, or remove images, simply upload, edit, or delete the image files from the web server. The image gallery will immediately reflect the changes!



Happy Programming!

Mini FAQ

Comment désélectionner un nœud ?

Récupérer la sélection est du ressort du composant TreeView qui doit vous indiquer quel est le nœud sélectionné parmi sa liste mais il est logique de considérer que la désélection est du ressort du nœud lui-même. Pour désélectionner un nœud il faut donc jouer sur la propriété « selected » du nœud lui-même.

Comment afficher également les attributs d'un fichier XML relié au TreeView via un DataSource ?

L'assistant du XMLDataSource vous permet, sous la zone concernant le nom du fichier XML, d'indiquer un fichier de transformation XSLT.

Le fruit de la fusion du fichier XSLT et du fichier XML devra donc satisfaire aux conditions.

Cela nécessite évidemment de maîtriser XSLT.

Dans le cas du visualisateur d'images comment associer un texte de description à chaque image ?

Si les images (ou leur chemin) sont stockées dans une base, des champs de la table peuvent fournir l'information. Dans le cas d'une gestion de fichier la solution élégante consiste à avoir pour chaque image un fichier de description. Par exemple ; luminaire.gif sera associé à luminaire.txt.