

---

# Web Tree View

---

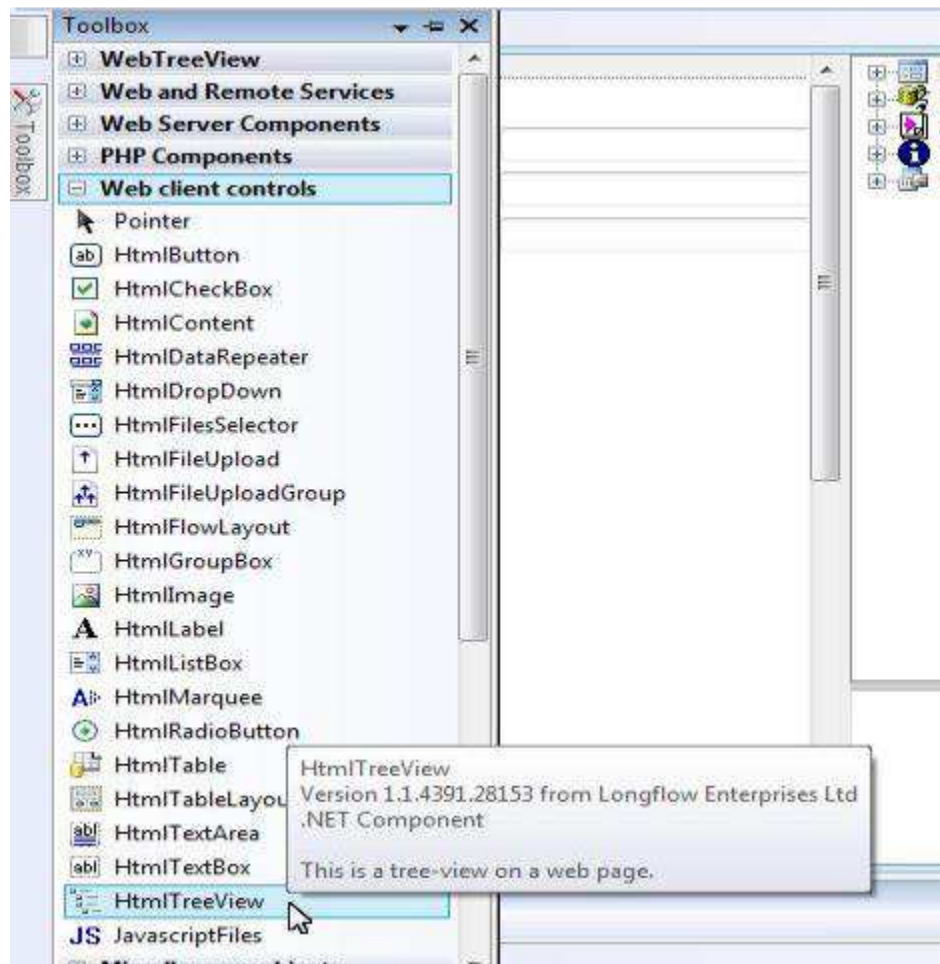
## Contents

Introduction .....	2
Set Tree Nodes at Design Time .....	3
Change Tree Nodes at Runtime .....	7
Add root nodes .....	8
Add child nodes.....	12
Delete selected node .....	13
Modify node text .....	14
Modify node image .....	17
Test.....	20
Add root node .....	20
Add a child node .....	21
Modify node text .....	22
Modify node image .....	23
Delete node.....	24
Data Binding.....	25
Sample database.....	25
Bind tree view to data.....	26
Set DataSource property.....	26
Set Foreign Key .....	27
Set Primary Key .....	27
Bind node properties to data.....	28
Load data.....	29
Test Data Loading.....	30
Data Editing.....	33
Modify nodes .....	33
Add root nodes .....	34
Add child nodes.....	36

Delete nodes .....	38
Save modifications to database .....	40
Use File Upload .....	41
Data-Binding Test.....	46
Recursion for MySQL.....	49
Recursive query problem .....	49
One solution.....	49
Create stored-procedure .....	50
Use stored-procedure in query.....	51
Load data when a tree node is selected .....	53
Test.....	56
Debugging .....	58
Feedbacks .....	60

## Introduction

Hierarchical data are better displayed in a tree view, for example, multi-level categories. HtmlTreeView component can be used on web pages for showing and editing hierarchical data. It may be data-bound to self-linked one-to-many relational data.



## Set Tree Nodes at Design Time

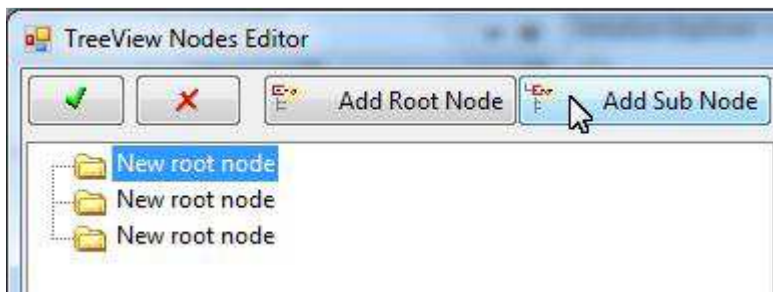
Tree Nodes may be created at design time by setting TreeNodes property:



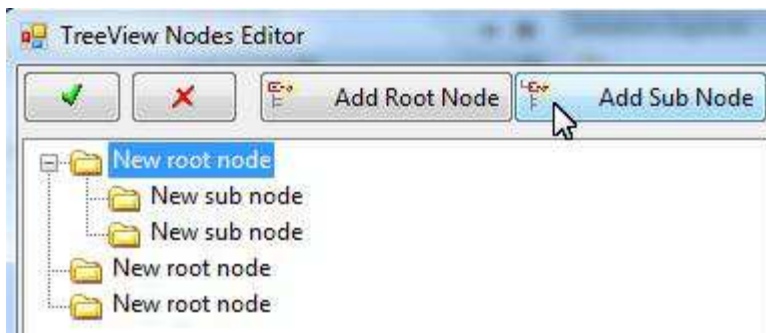
A dialogue box appears for creating and editing tree nodes:



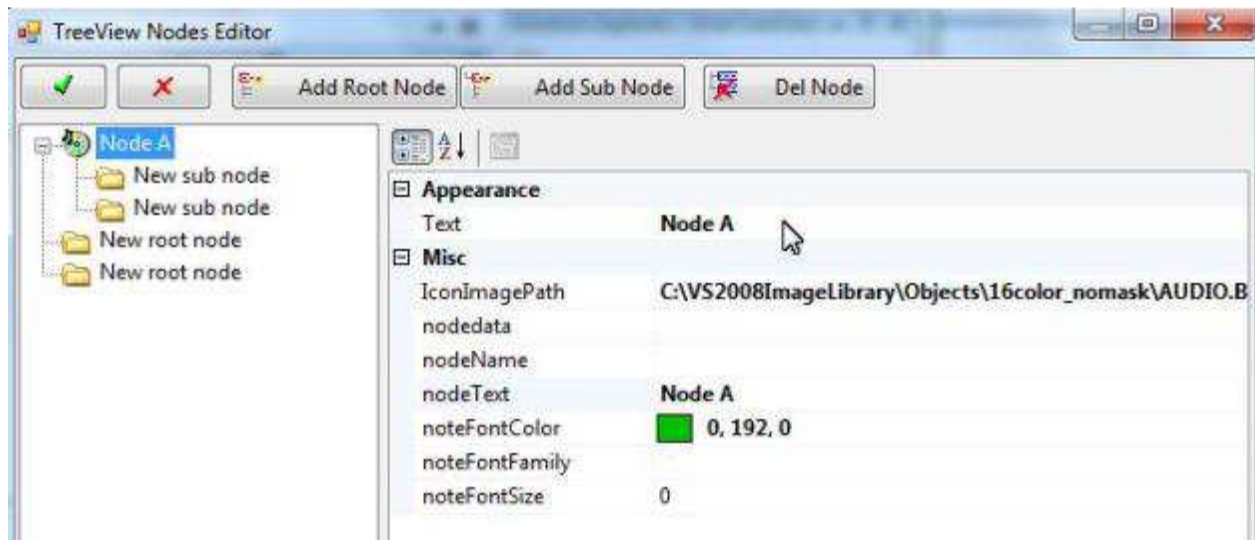
Click "Add Root Node" to add new root tree nodes.




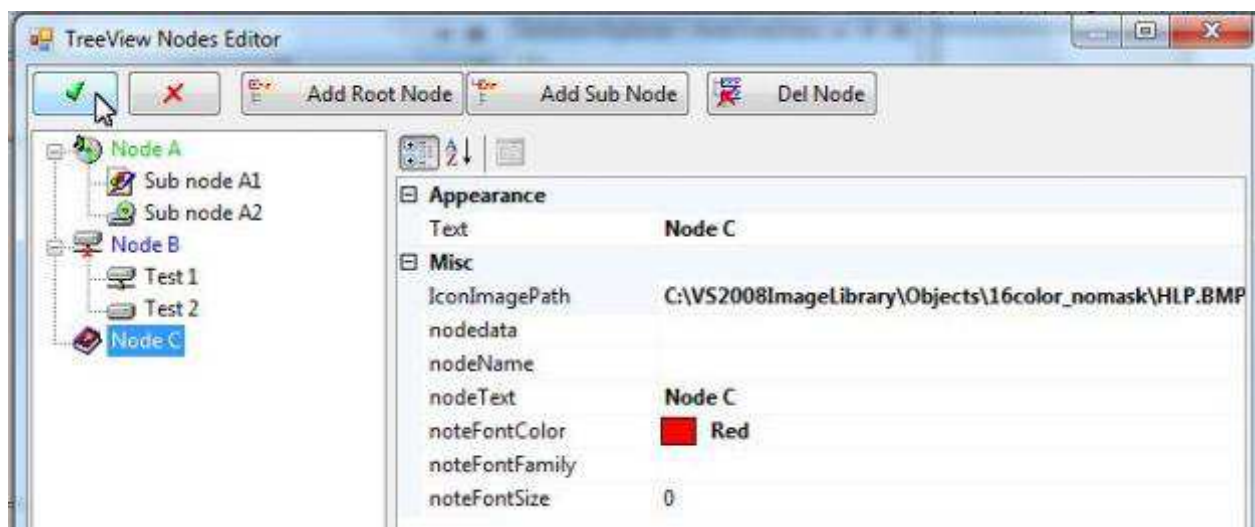
Select a node and click "Add Sub Node" to add child nodes to the selected node.



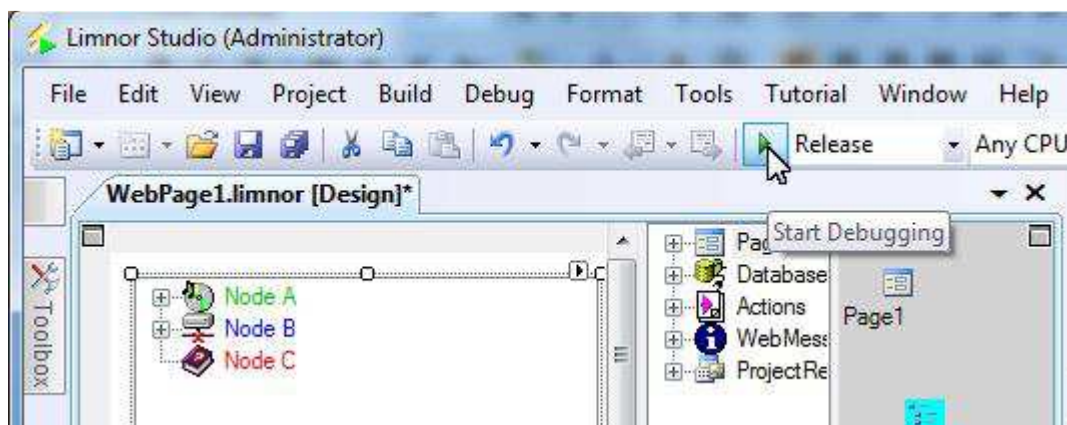
Select a node; the properties of the node are displayed in a Properties Window. We may modify the properties through the Property Window:



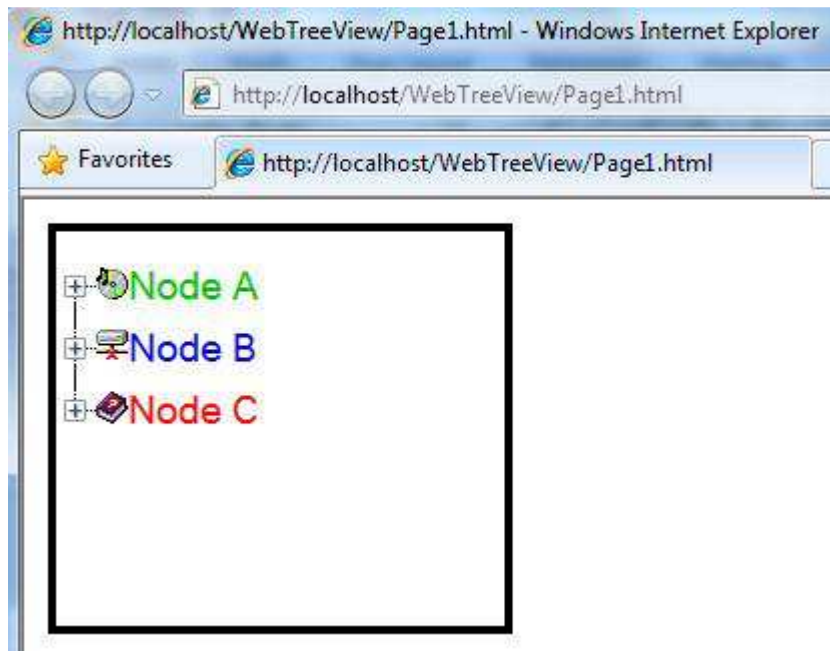
We may set properties for all nodes. Click  to finish editing nodes.



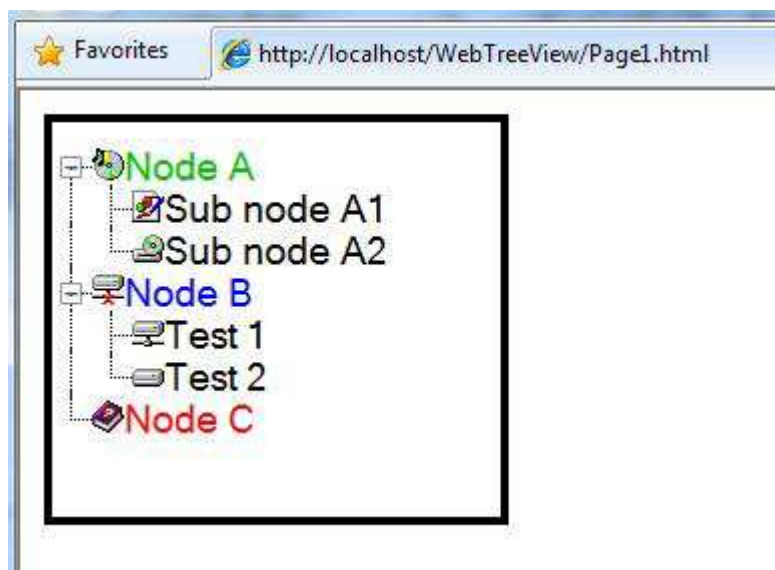
We may test this web page by clicking the Run button.



The web page appears showing the tree view.

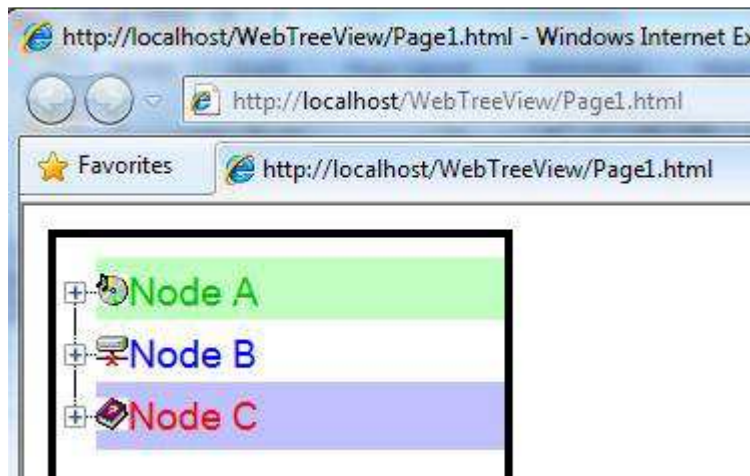


Click + sign to expand tree nodes to show child nodes:

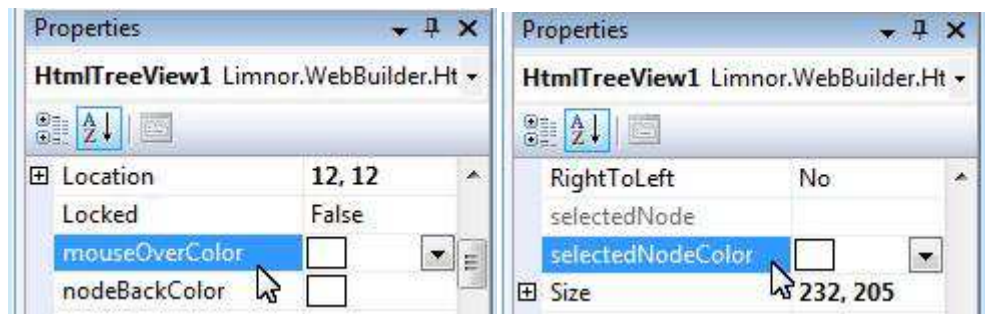


Selected node and the node under mouse pointer are shown with different background color:





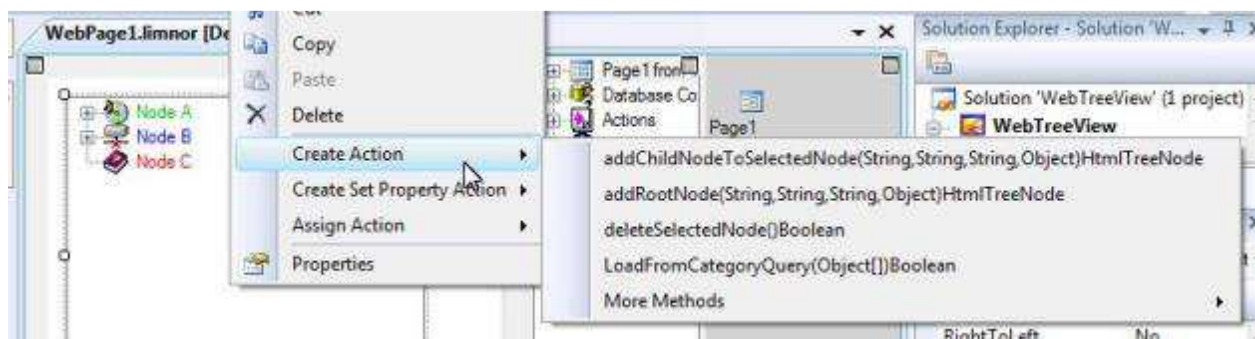
The above screenshot shows the default colors. You may control the colors via properties `mouseOverColor` and `selectedNodeColor`.



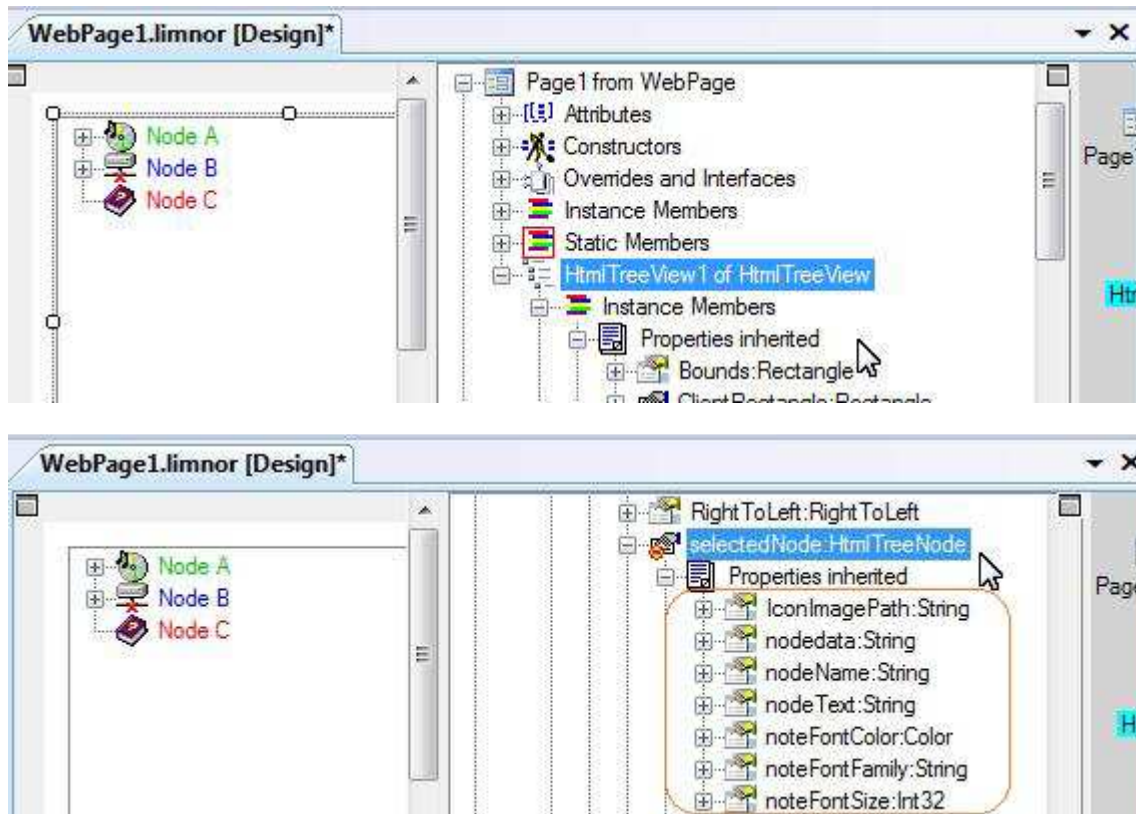
## Change Tree Nodes at Runtime

The `HtmlTreeView` control provides following methods for adding and removing tree nodes at runtime:

- `addRootNode(string nodeName, string text, string imageUrl, object nodedata)`
- `addChildNodeToSelectedNode(string nodeName, string text, string imageUrl, object nodedata)`
- `deleteSelectedNode()`



To modify existing nodes, you may create set-property actions to change the properties of the selected node. This can be done via the selectedNode property of the HtmlTreeView:

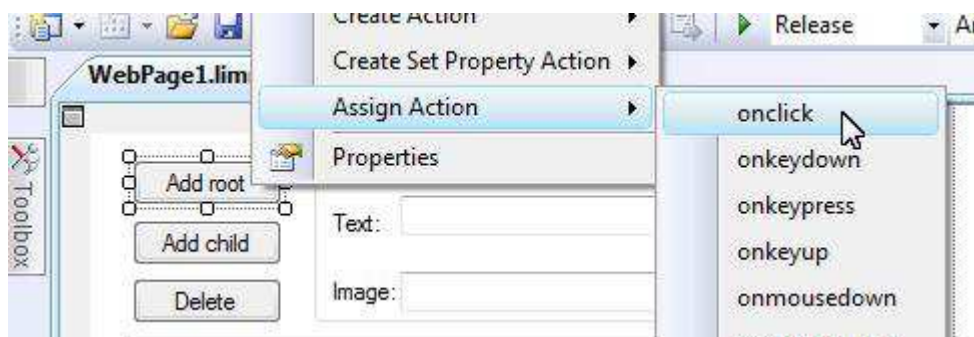


We'll create sample actions and use buttons to execute sample actions.

### Add root nodes

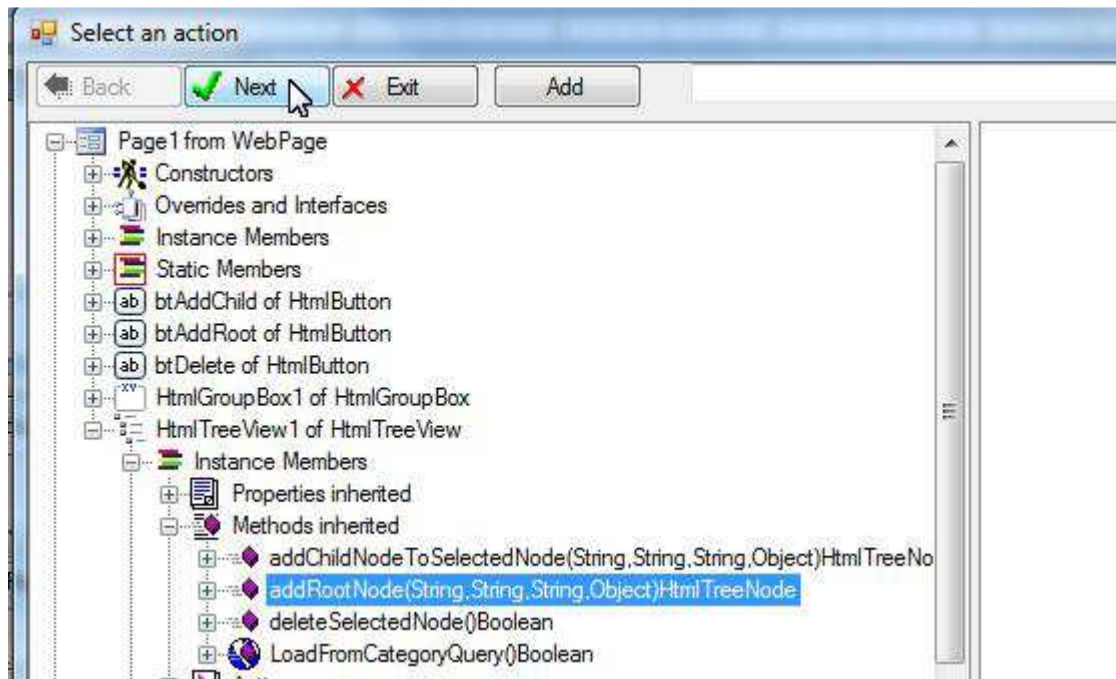
We add some buttons and text boxes to the web page for creating the sample.

Right-click "Add root" button; choose "Assign Action"; choose "onclick" event:

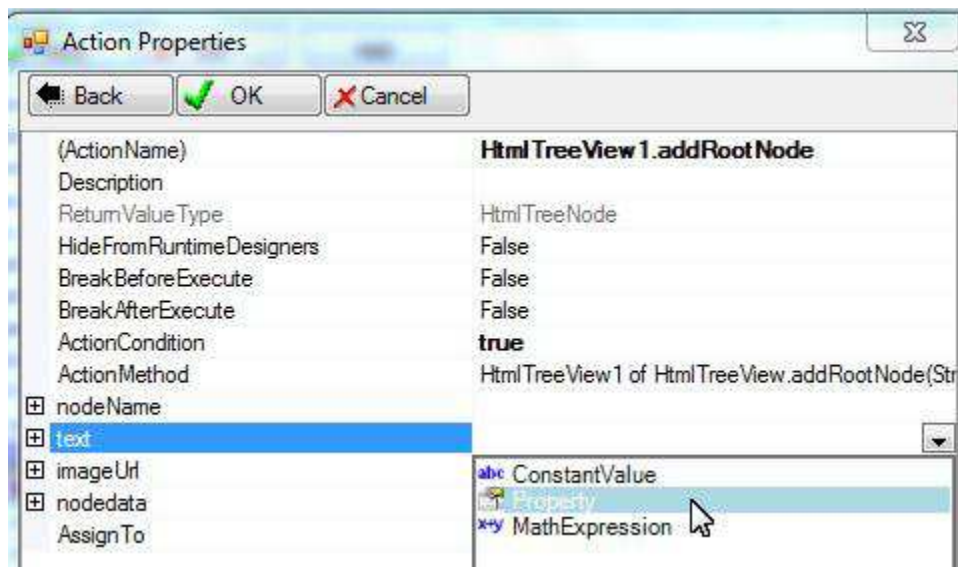


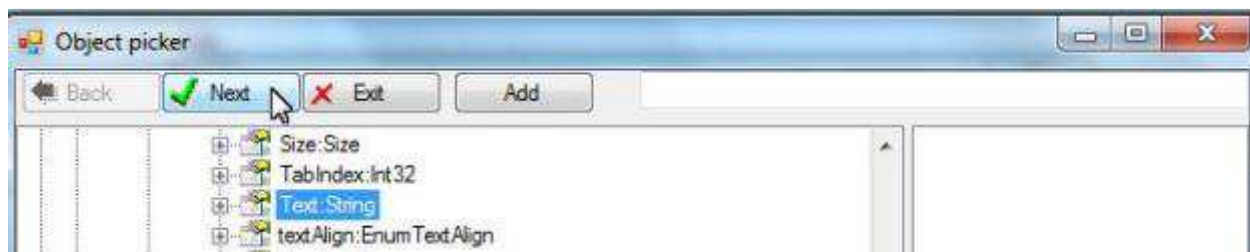
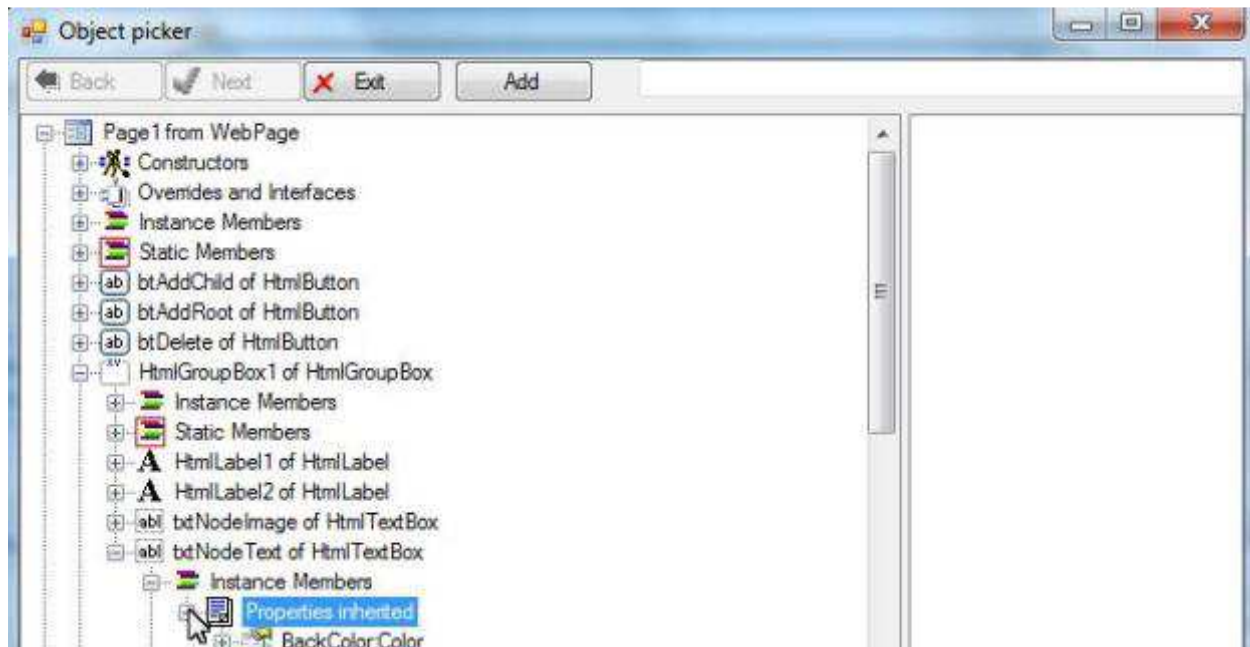
Select the addRootNode method of the HtmlTreeView:



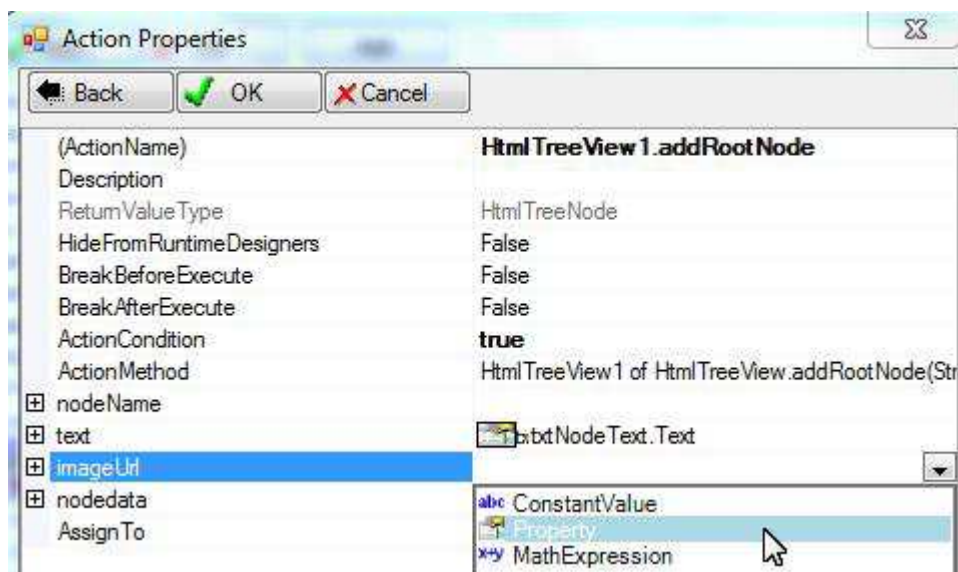


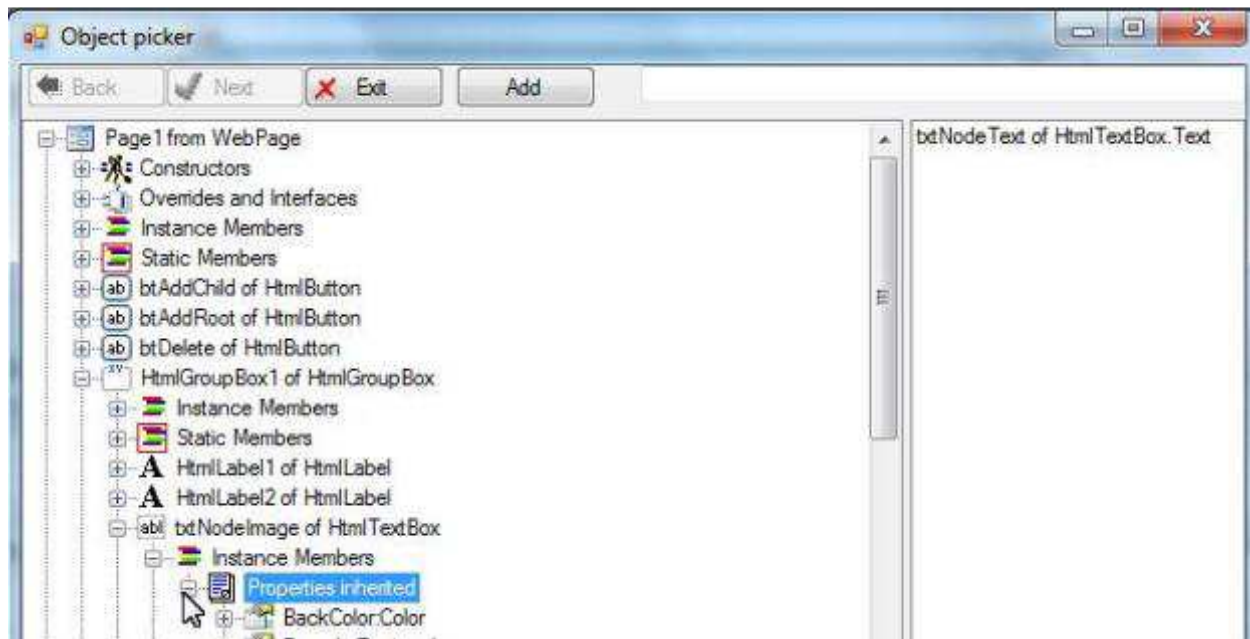
For “text” of the action, choose the Text property of the text box for Node Text:



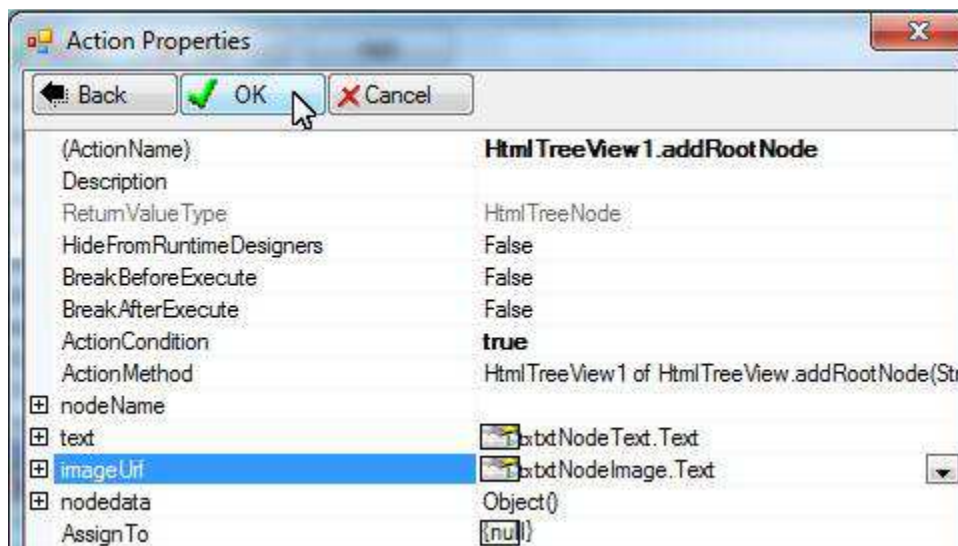


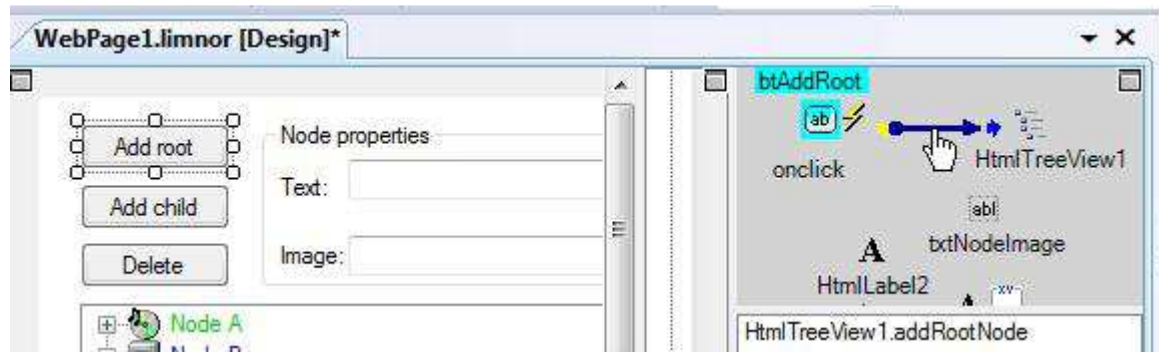
For “imageUrl” of the action, select the Text property of the text box for node image:





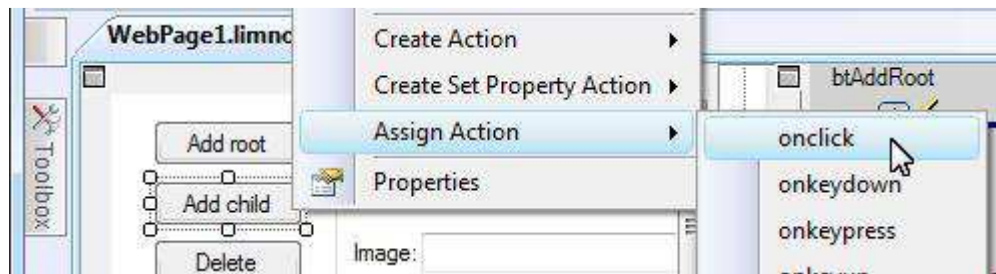
Click OK. An action is created and assigned to the button:



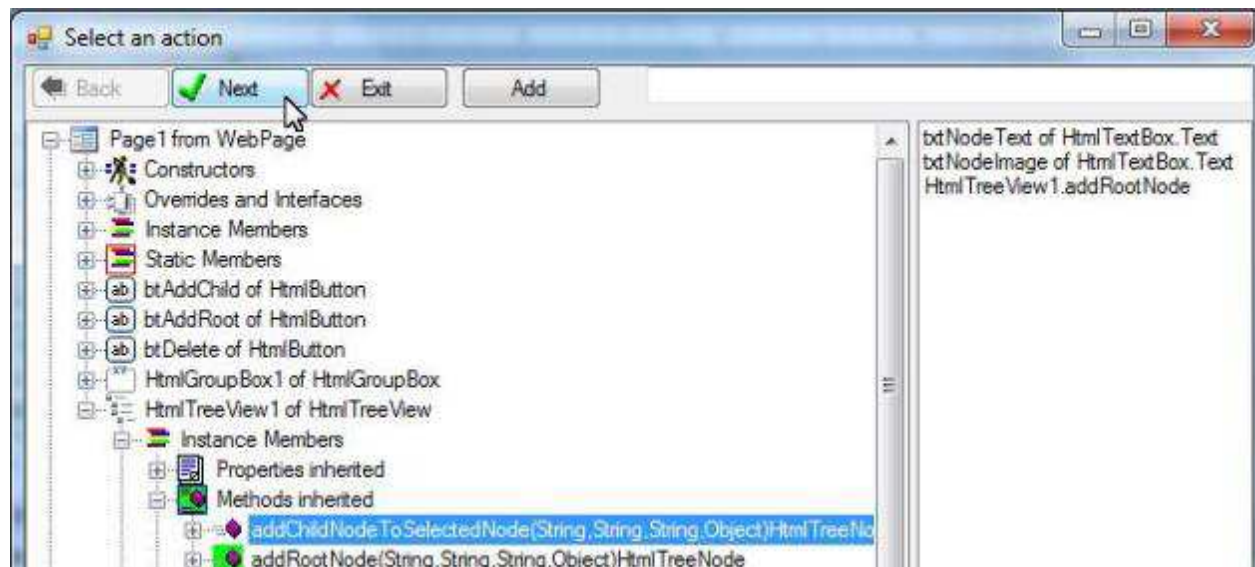


## Add child nodes

Right-click "Add child" button; choose "Assign Action"; choose "onclick" event:

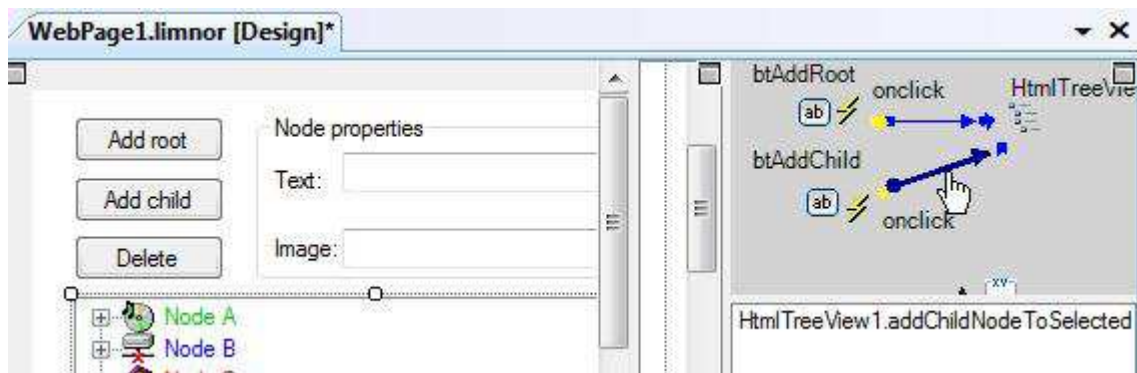
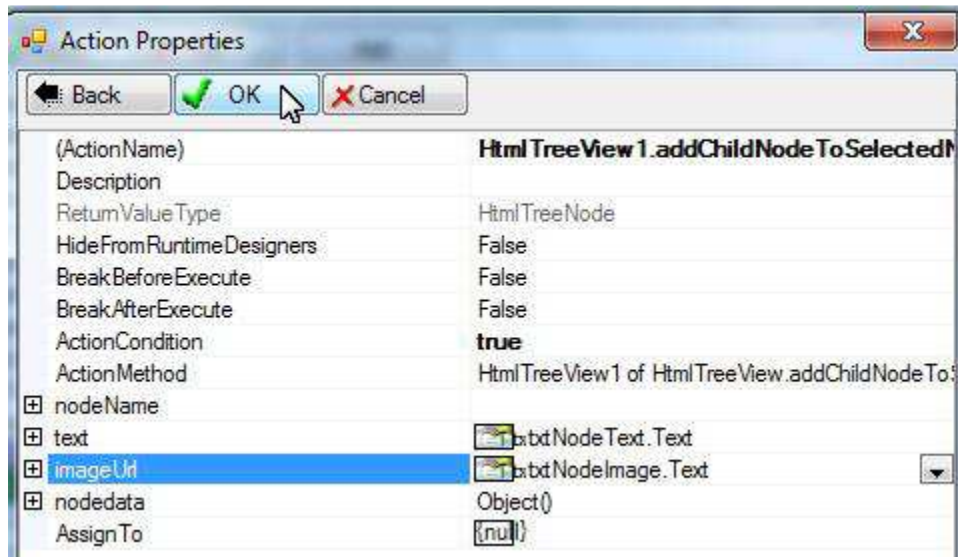


Select addChildNode method of the HtmlTreeView:



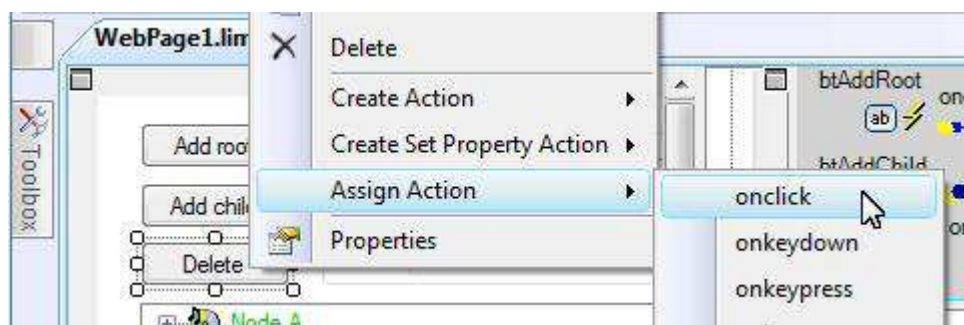
Set "text" and "imageUrl" of the action in the same way as we did for the last action:





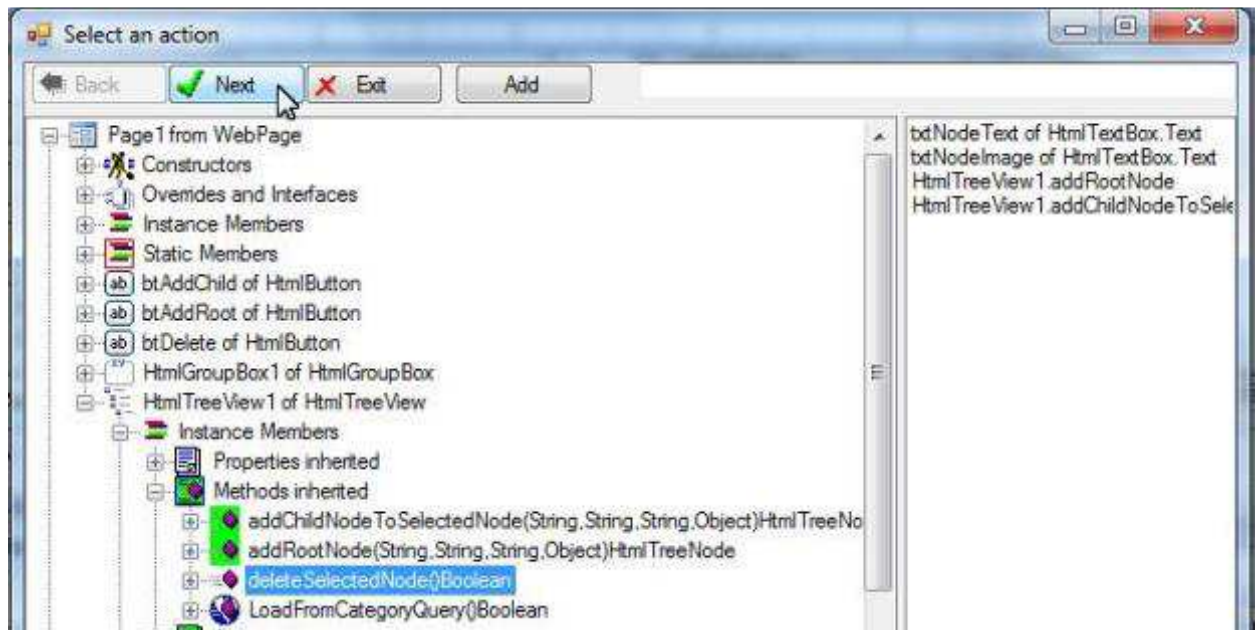
## Delete selected node

Right-click “Delete” button; choose “Assign Action”; choose “onclick” event:

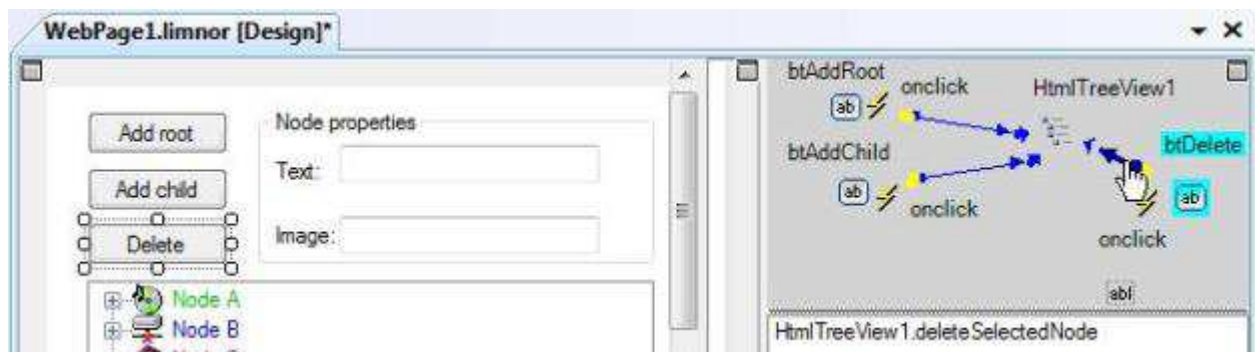
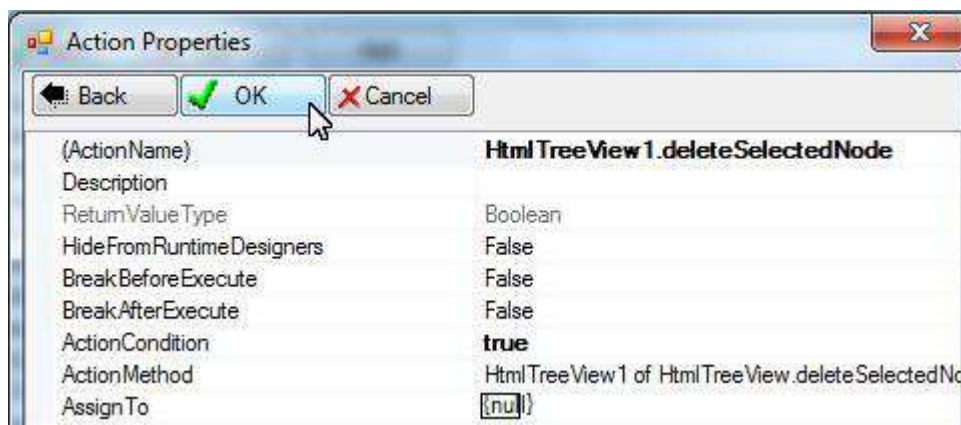


Select “deleteSelectedNode” method of the HtmlTreeView:



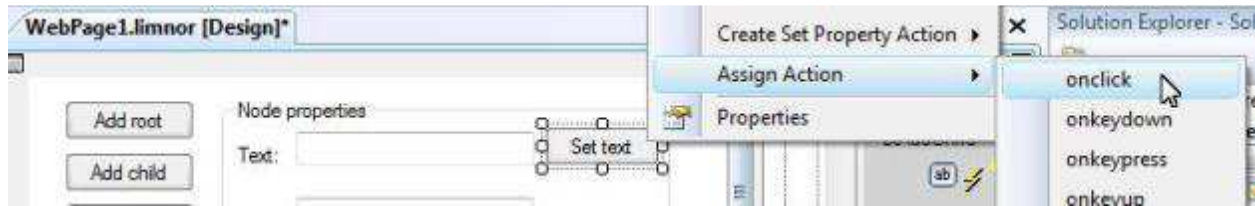


Click OK:

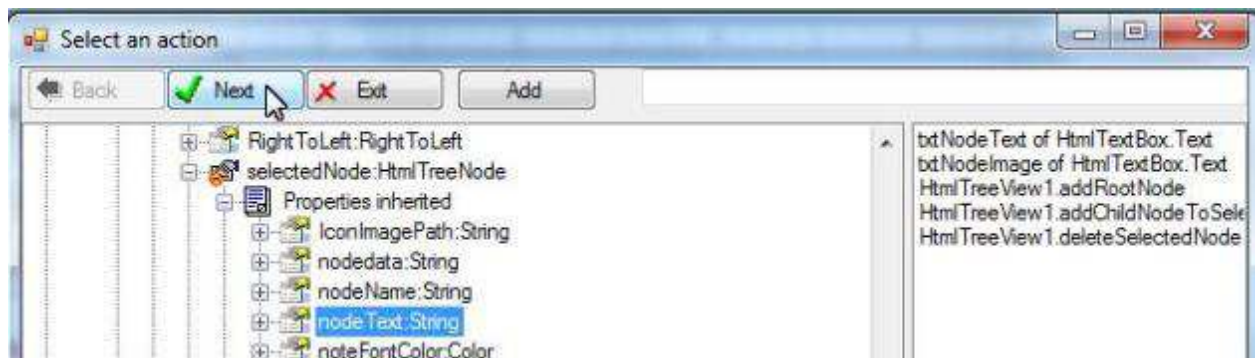
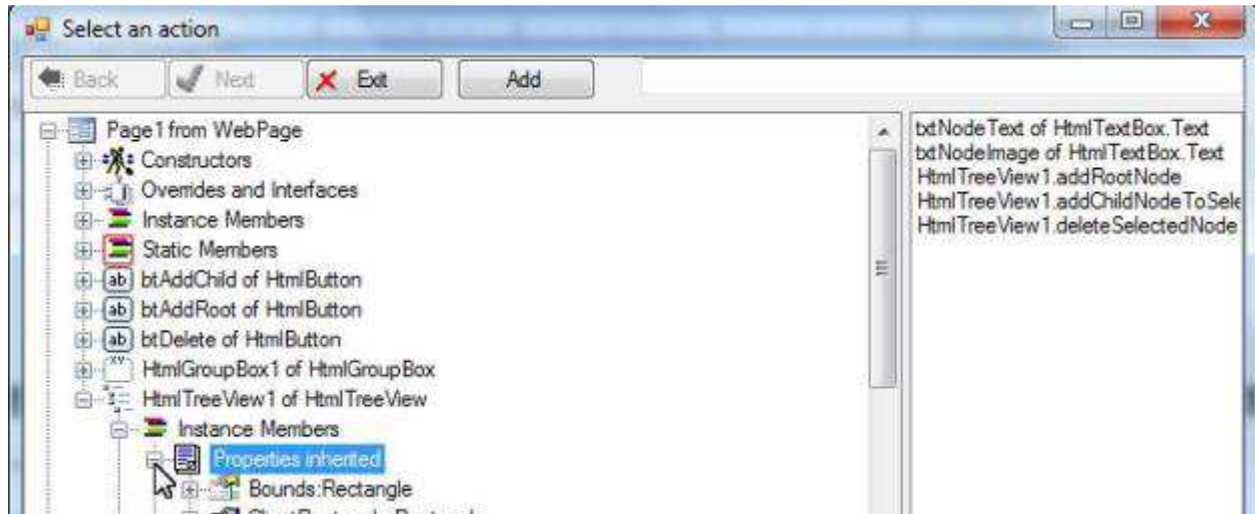


## Modify node text

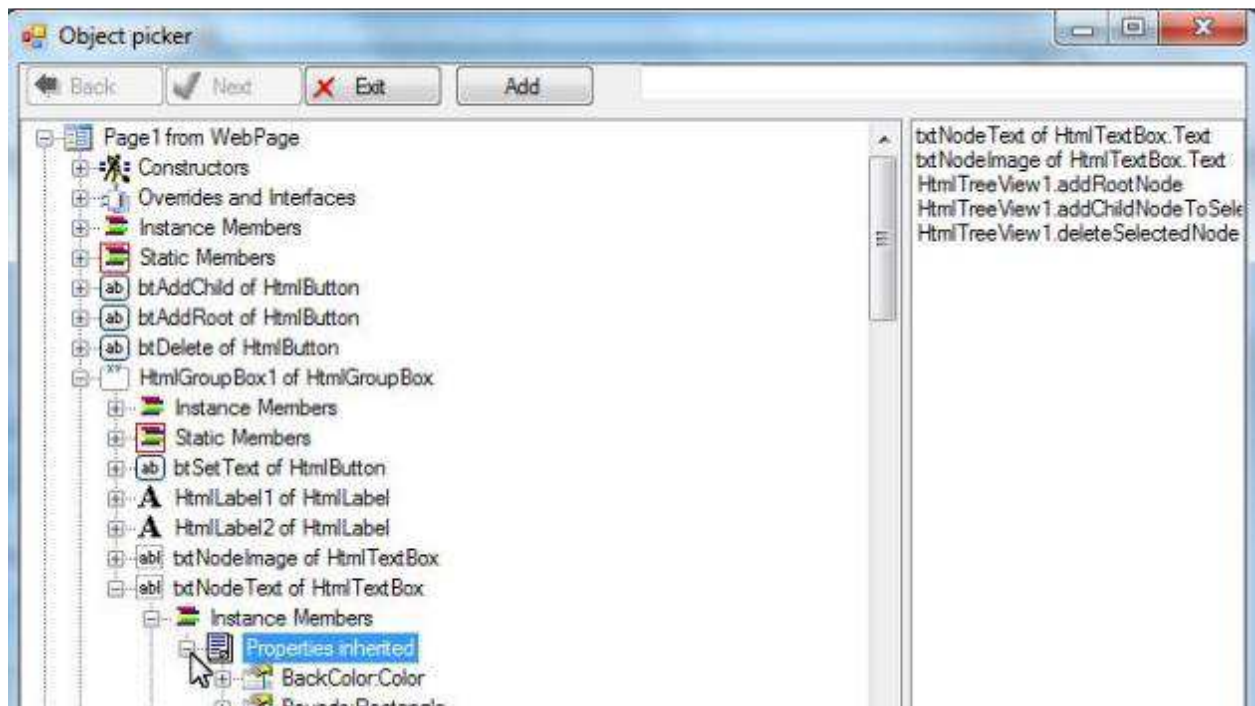
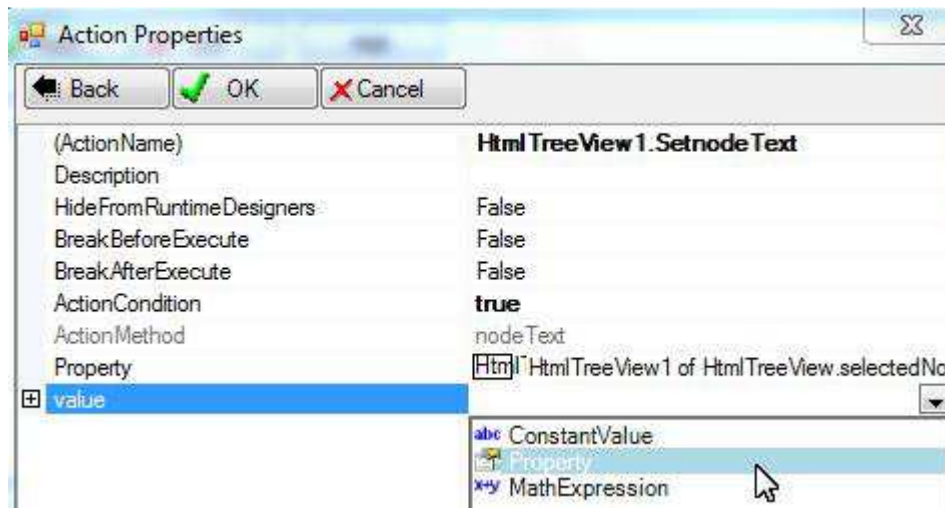
Let's add a button beside the text box for modifying the node text of the selected node. Right-click the button; choose "Assign Action"; choose "onclick" event:



Select nodeText property of the selectedNode property of the HtmlTreeView:

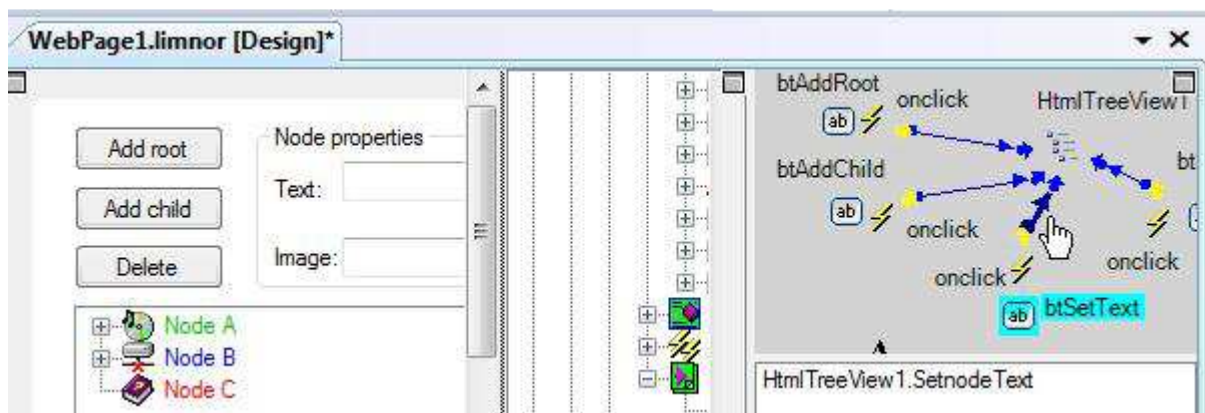
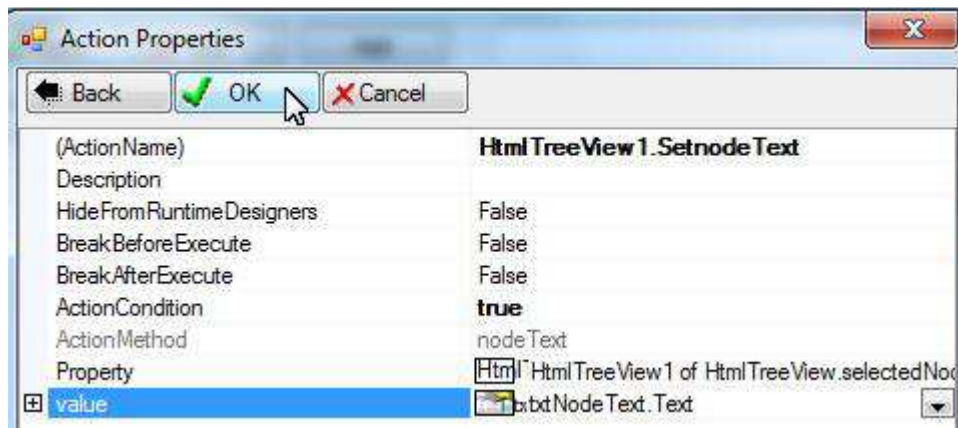


For “value” of the action, choose the Text property of the text box for node text:



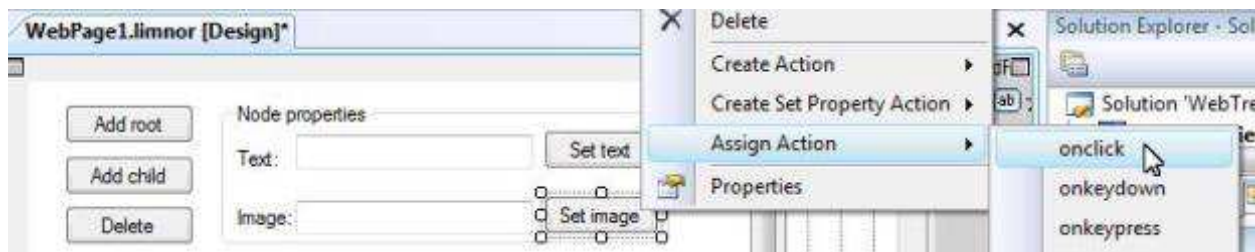
Click OK. The action is created and assigned to the button:



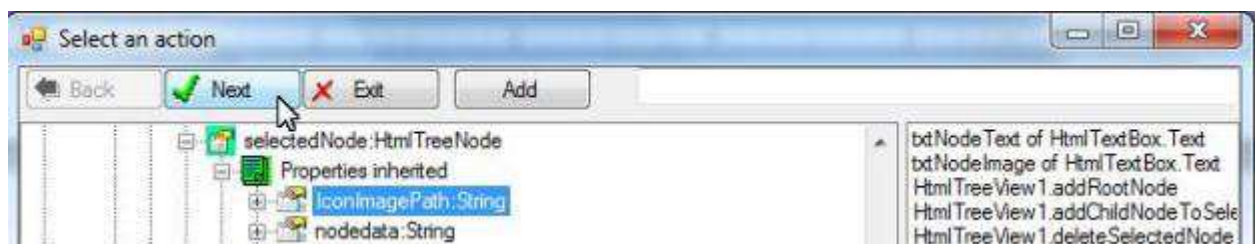
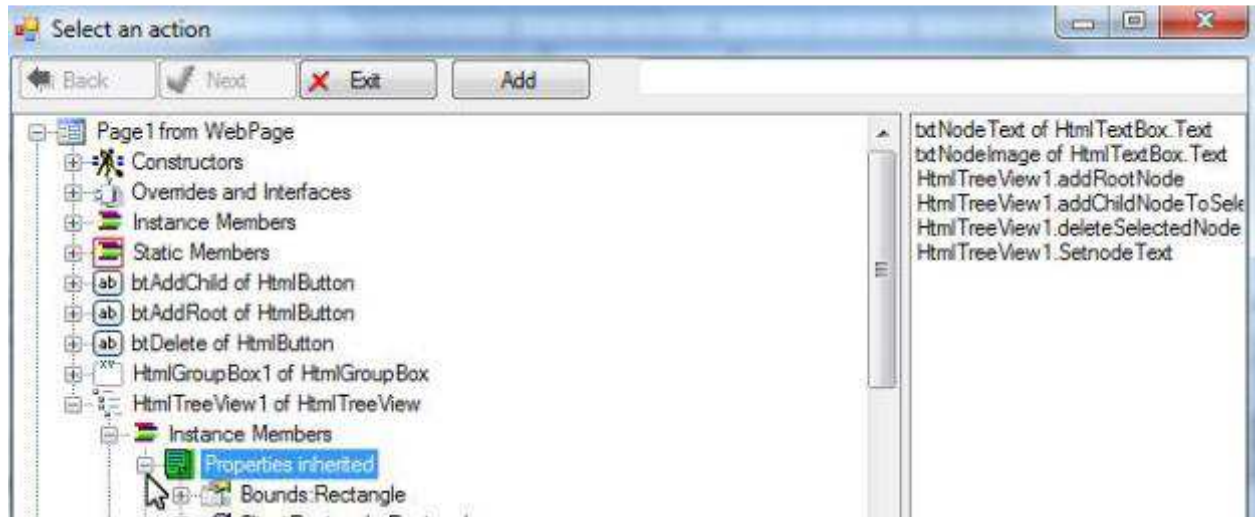


## Modify node image

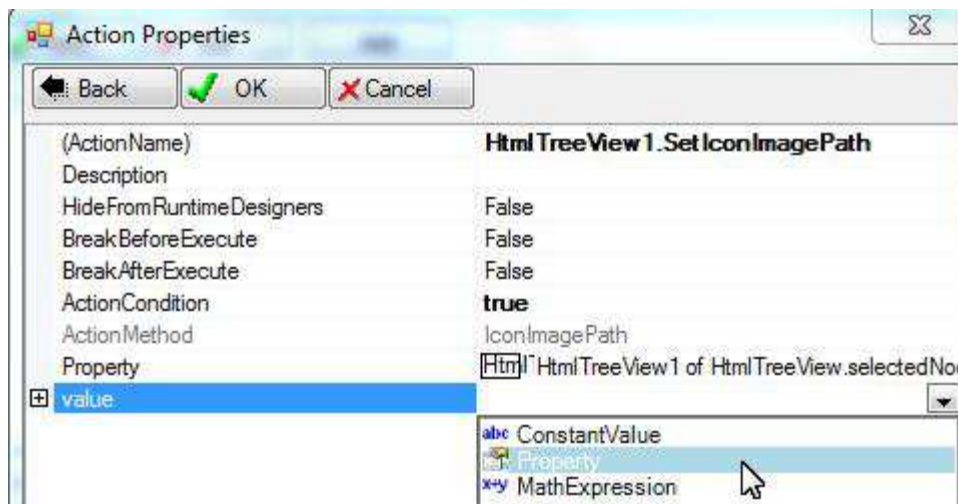
Let's add a button beside the text box for modifying the node image of the selected node. Right-click the button; choose "Assign Action"; choose "onclick" event:



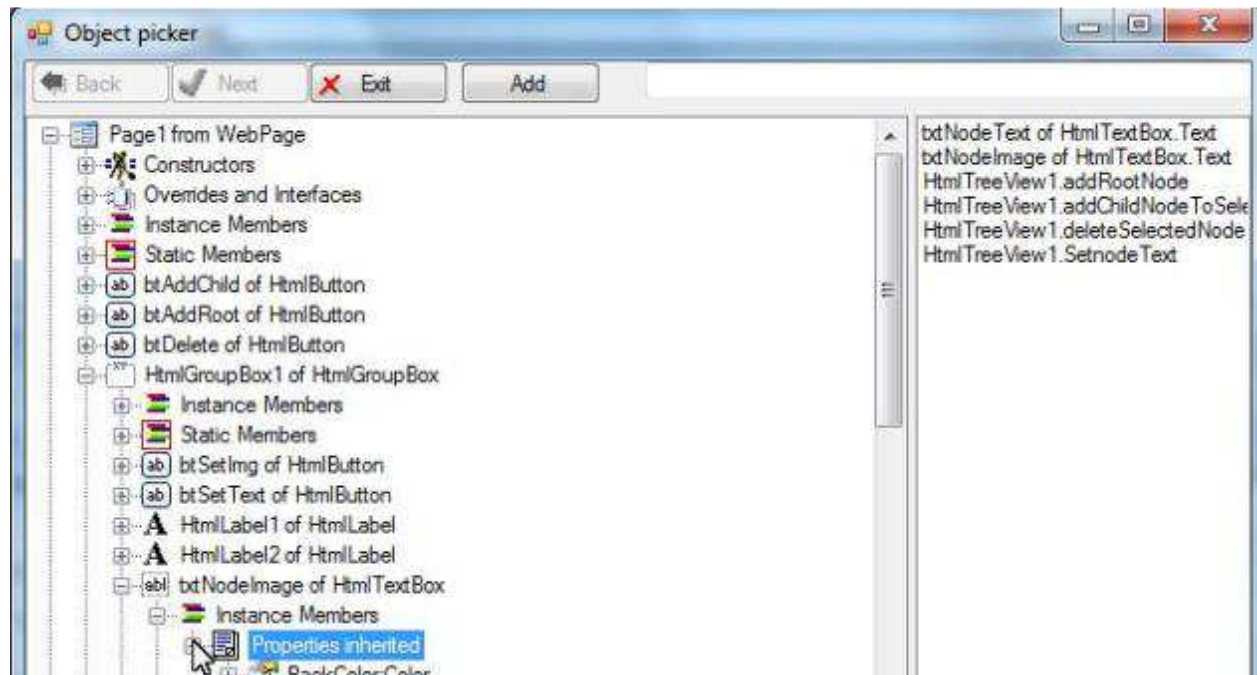
Select iconImagePath property of the selectedNode property of the HtmlTreeView:



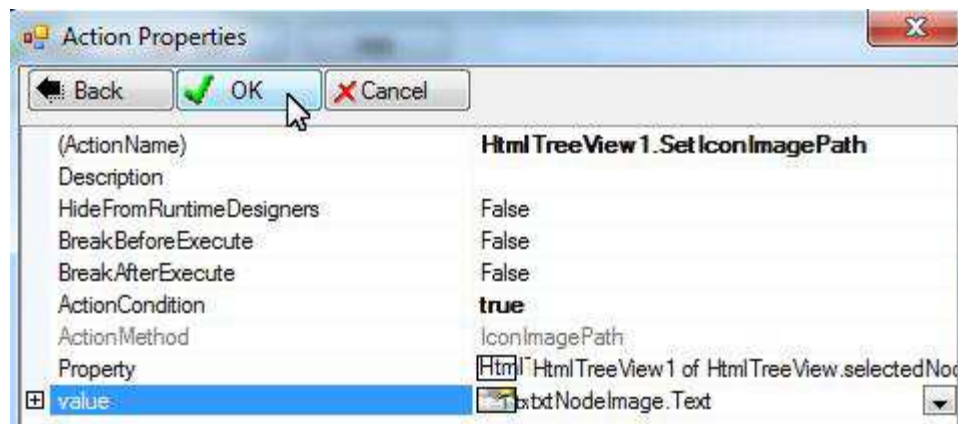
For “value” of the action, choose the Text property of the text box for node image:

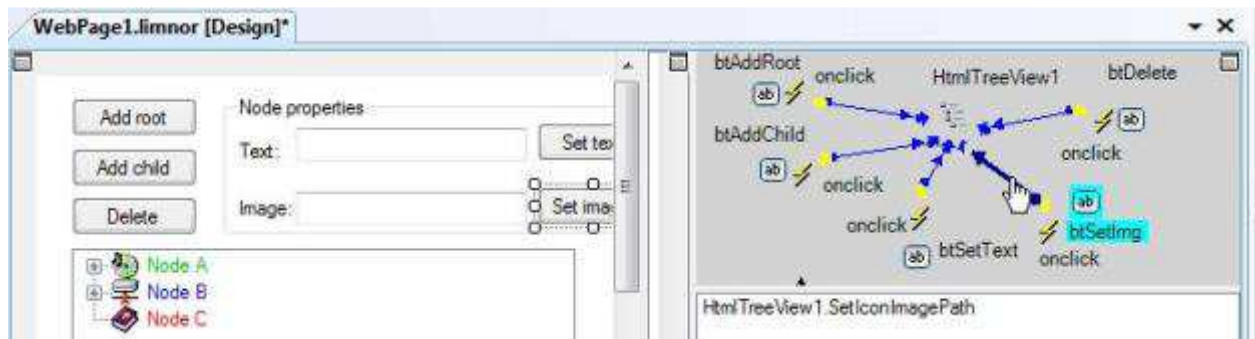






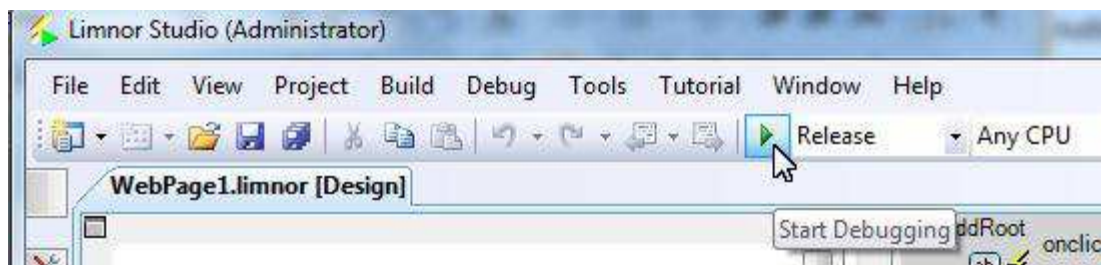
Click OK. The action is created and assigned to the button:



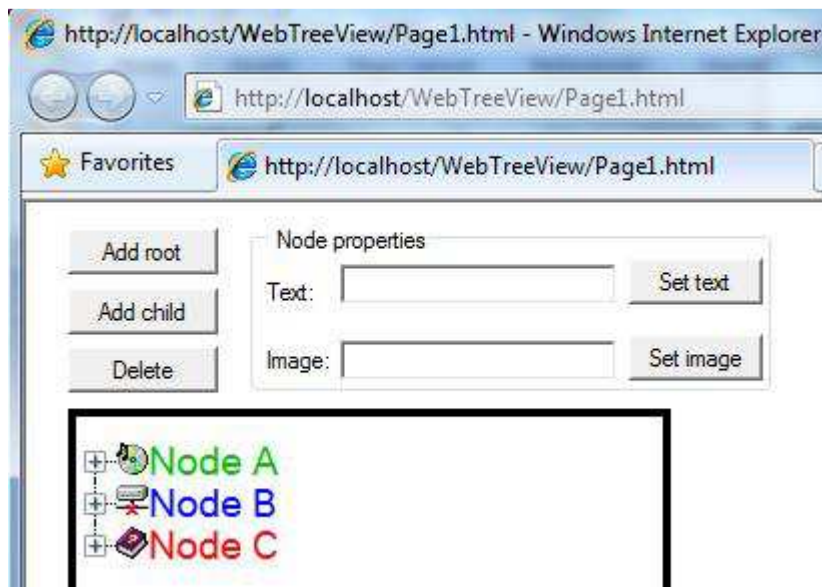


## Test

We may test the web sample now.

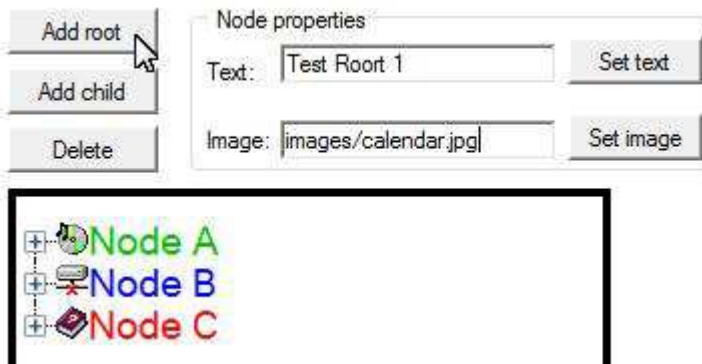


The web page appears:

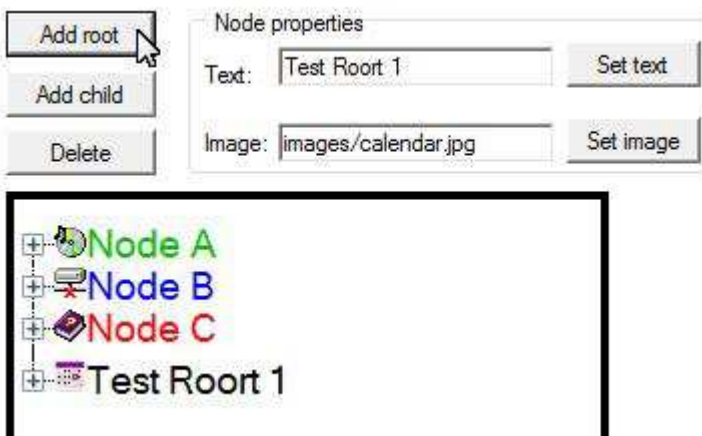


## Add root node

Enter some data in the text boxes and click "Add root":



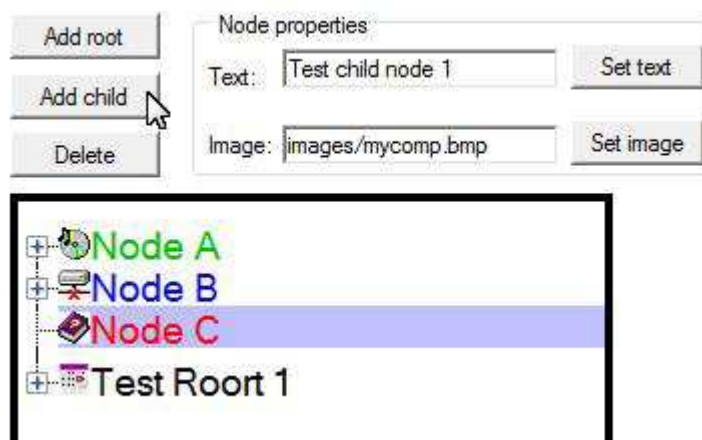
A new root node appears:

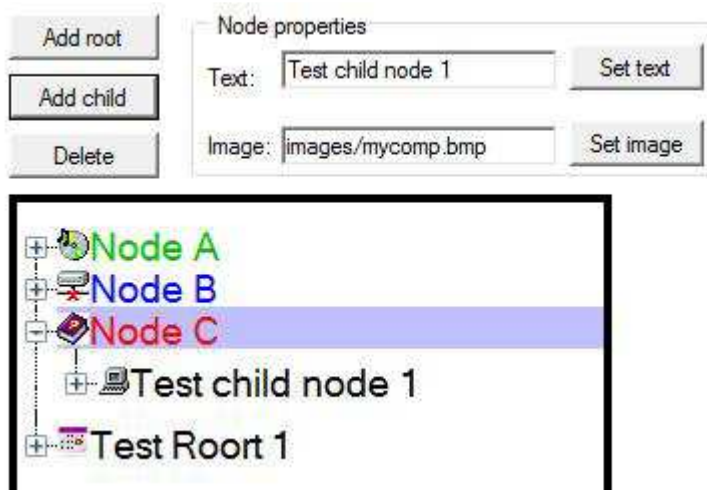


Note that the image path points to a file in the web server. If you want to allow the web visitors to enter image path from the web visitors' local computers then you may use an `HtmlFileUpload` control. We will do it in a sample later.

### Add a child node

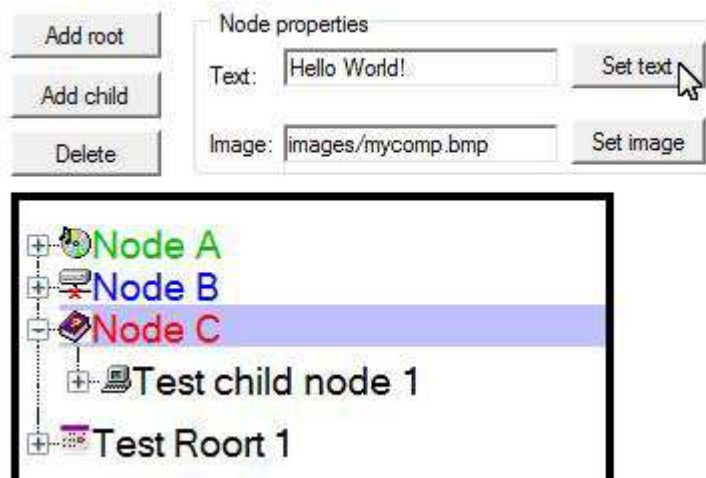
Select a tree node; enter some data in the text boxes; click "Add child". A new child node appears under the selected node.



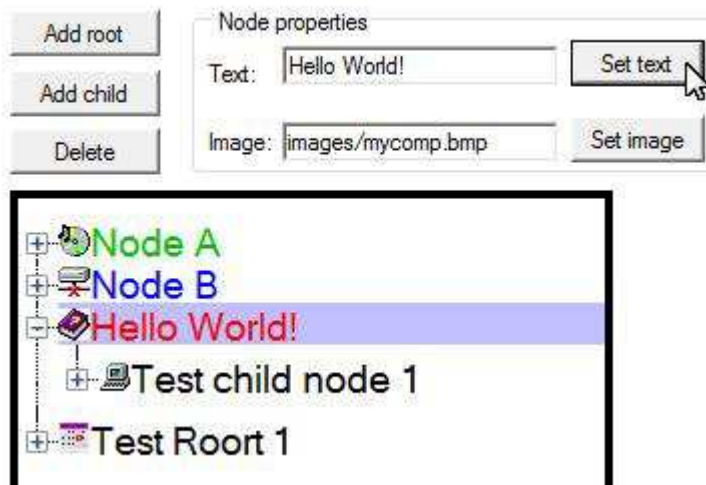


### Modify node text

Select a node; enter text in the text box.

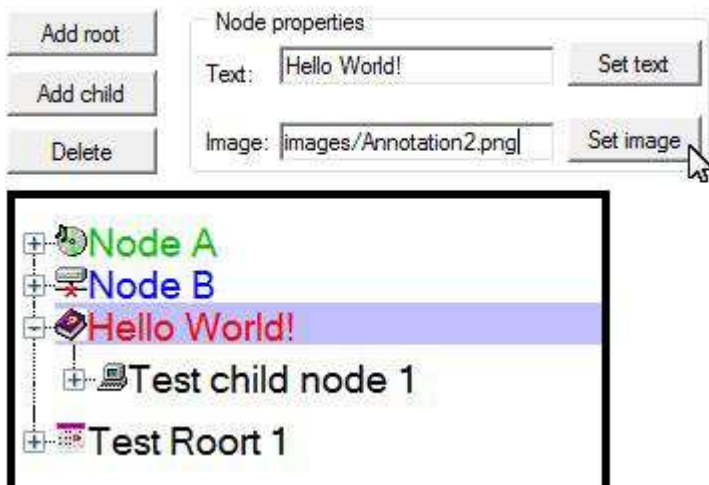


Click "Set text". The text of the selected node is changed:



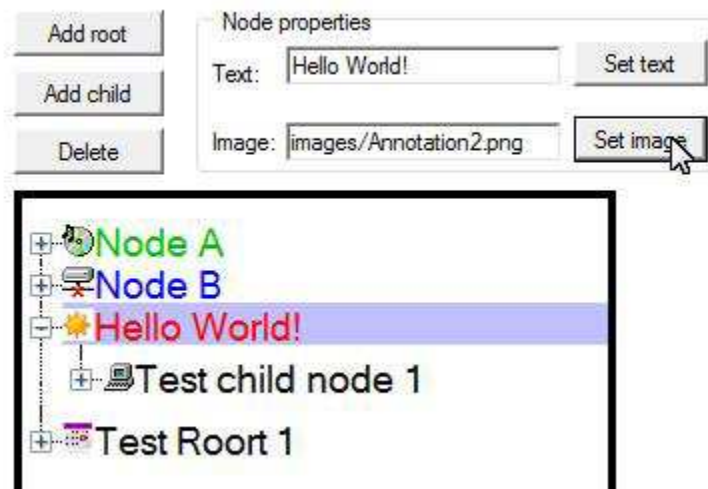
### Modify node image

Select a node. Enter an image URL in the text box.



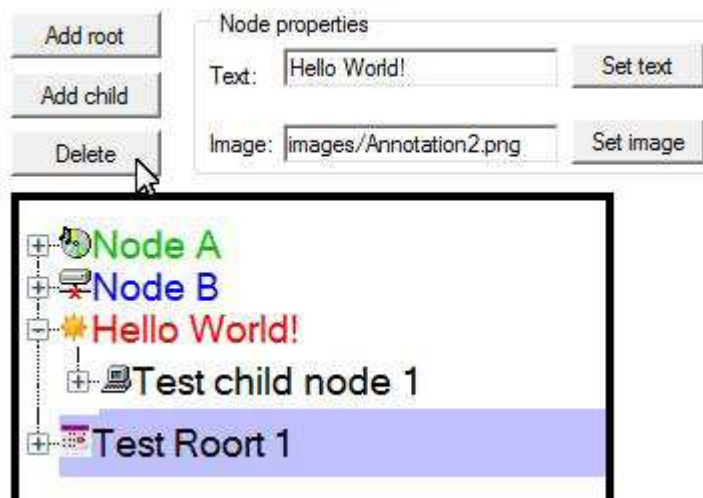
Click "Set image". The node image is changed:



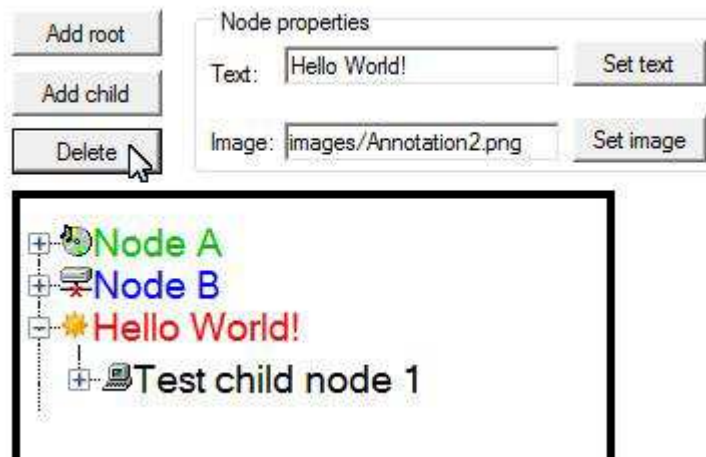


### Delete node

Select a node. Click "Delete".



The selected node disappears:



## Data Binding

The `HtmlTreeView` can be data-bound to data from a relational database. The data is provided by an `EasyDataSet`. The query for the `EasyDataSet` should contain at least a field as primary key and a field as foreign key. The `HtmlTreeView` will use

`{foreign key}` is null

to get records for root nodes. It will use

`{foreign key} = {primary key of the parent node}`

to get records for a parent node's child nodes.

The `HtmlTreeView` will automatically use the above filter. You **do not** insert the above filters in your `EasyDataSet`.

We will use a sample to illustrate the above descriptions.

## Sample database

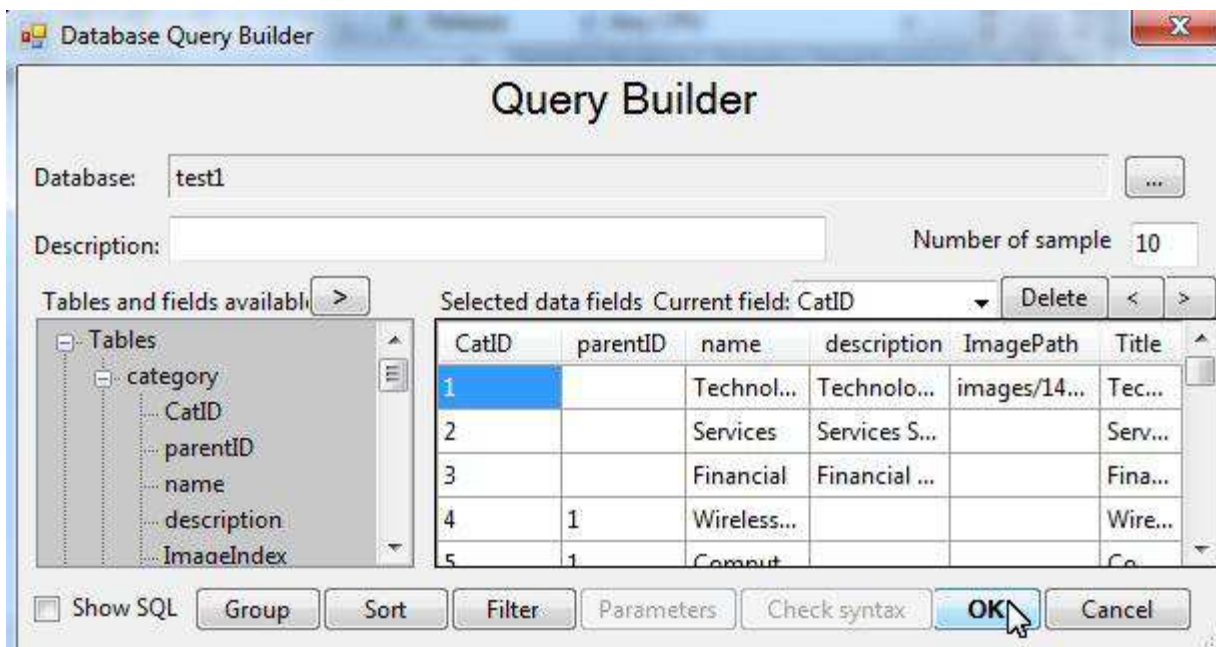
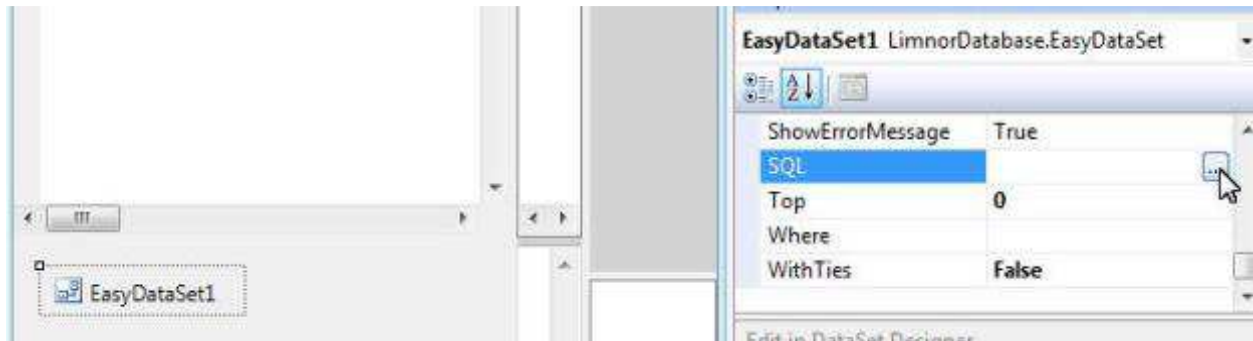
We use a "category" table as a sample:

Field	Type	Null	Key	Default	Extra
CatID	int(11)	NO	PRI	NULL	auto_increment
parentID	int(11)	YES		NULL	
name	varchar(50)	NO		NULL	
description	longtext	YES		NULL	
ImageIndex	int(11)	YES		NULL	
ImagePath	varchar(300)	YES		NULL	
Title	varchar(300)	YES		NULL	

7 rows in set (0.19 sec)

“CatID” is the primary key. “parentID” is a foreign key referencing “CatID”.

Add an EasyDataSet to a web page. Set its SQL property to a query to include needed fields:



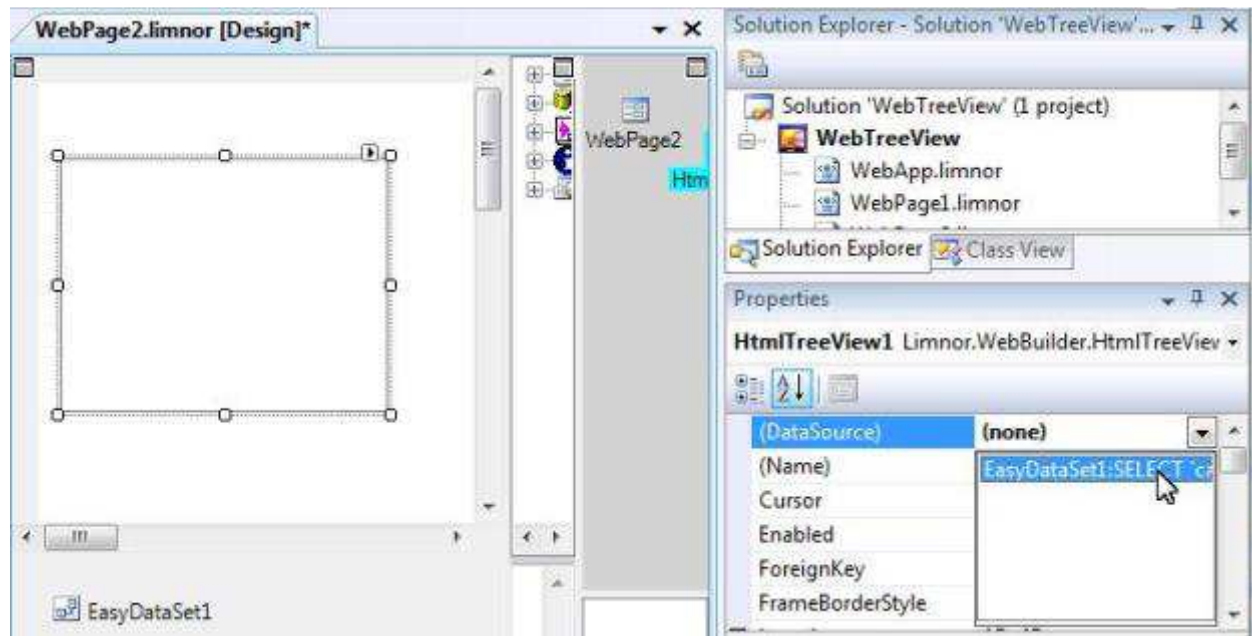
You may use all other query features, for example, add filters and use parameters. For this sample, we just use a simple query.

### Bind tree view to data

Set properties of the HtmlTreeView to bind the HtmlTreeView to the EasyDataSet.

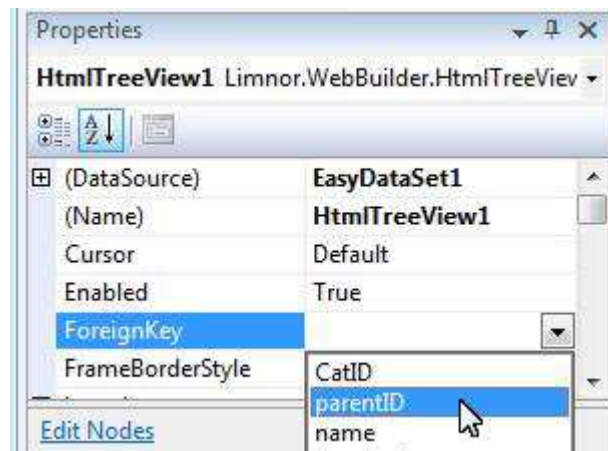
### Set DataSource property

Set DataSource property to the EasyDataSet. It must be done first.



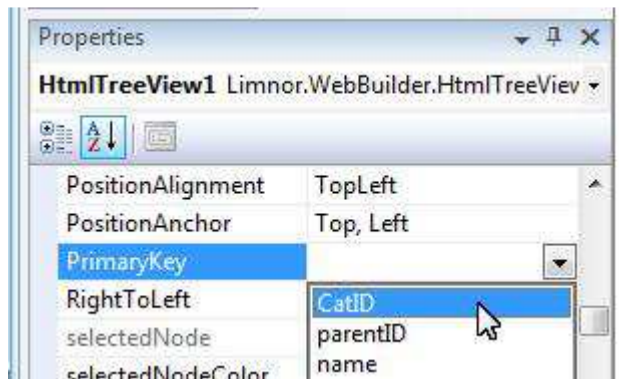
### Set Foreign Key

Set ForeignKey property to "parentID":



### Set Primary Key

Set PrimaryKey property to "CatID":



### Bind node properties to data

The following properties specify which data query fields will be used for which node properties.

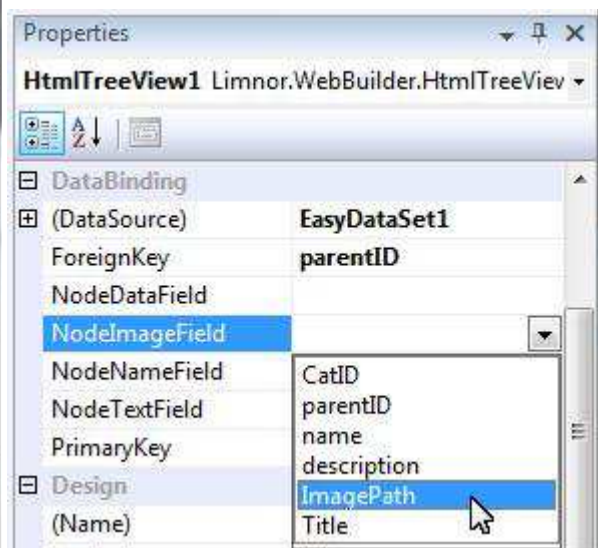
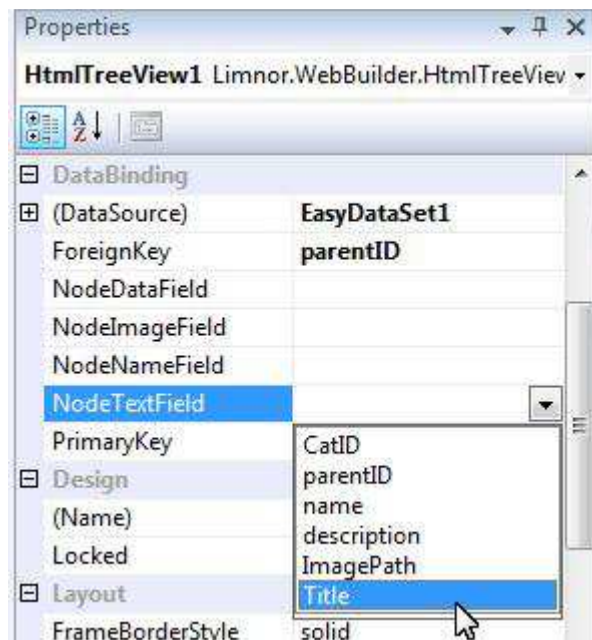
**NodeTextField** – It specifies a field name. The value from the field will be used as node text.

**NodeImageField** – It specifies a field name. The value from the field will be used as node image path.

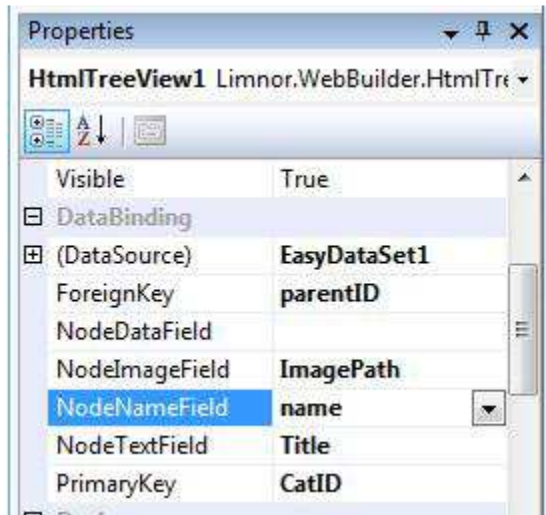
**NodeNameField** – It specifies a field name. The value from the field will be used as node name.

**NodeDataField** – It specifies a field name. The value from the field will be used as node data.

In this sample, we link **NodeTextField** to field “Title”; **NodeImageField** to field “ImagePath”; **NodeNameField** to field “name”:







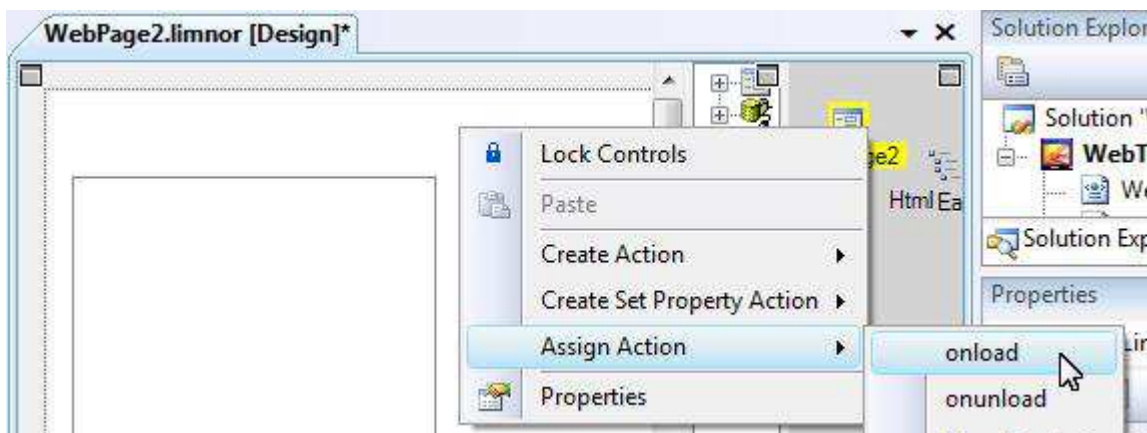
**Note:** if your database has fields not allowing null values then such fields must be linked to node properties: NodeDataField, NodeImageField, NodeNameField, or NodeTextField. If such fields are not linked then adding new tree nodes will fail because null values will be used for new nodes.

## Load data

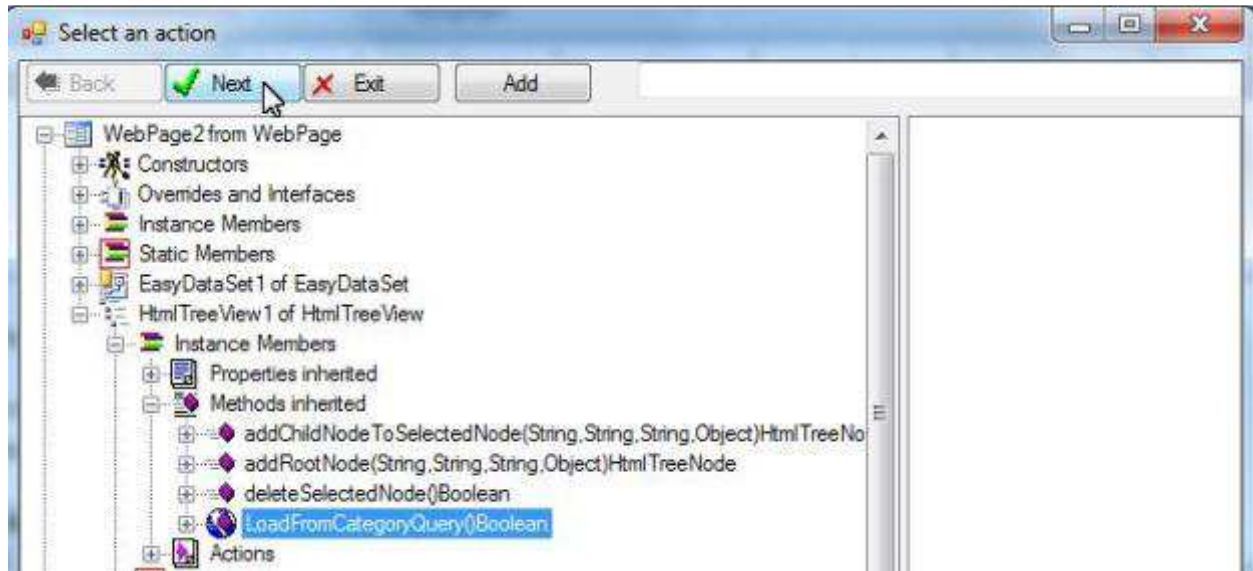
If an EasyDataSet is used as a data source for a tree view then **do not use** the EasyDataSet's following methods: Query, QueryWithParameterValues, QueryWithWhere, QueryWithWhereDynamic, AddNewRecord, and CreateNewRecord. These methods will damage the display of tree view.

To load data for the HtmlTreeView, use the HtmlTreeView's **LoadFromCategoryQuery** method.

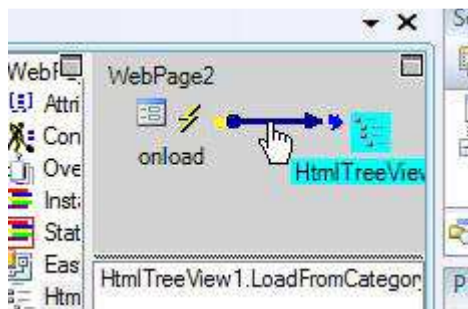
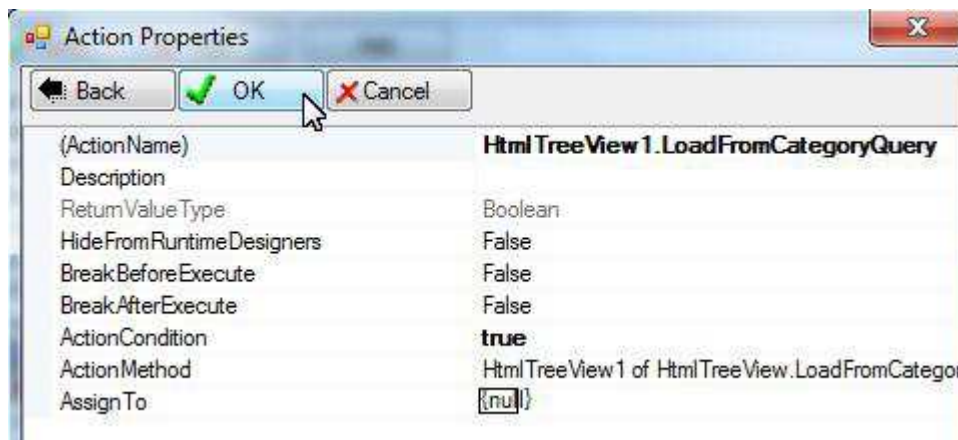
We want to load data on loading the web page. Right-click the web page; choose "Assign Action"; choose "onload" event:



Select the HtmlTreeView's LoadFromCategoryQuery method:

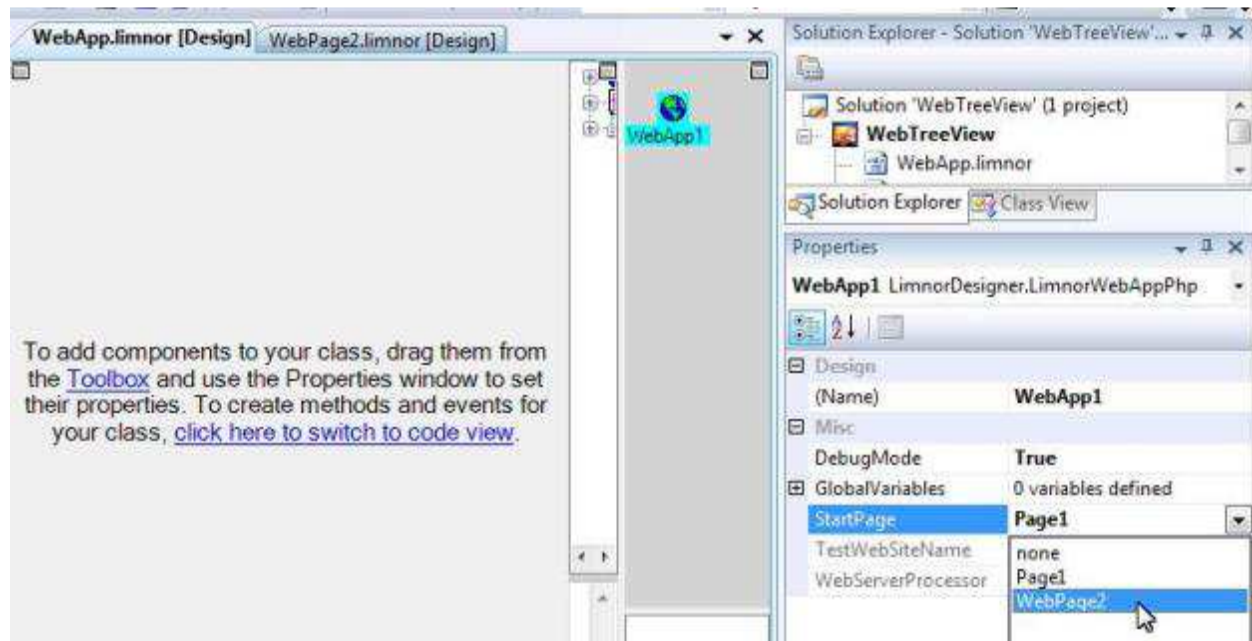


If you used parameters in the query then each parameter will be a property of the action. We do not use parameters in the query. So, click OK to finish creating the action and assigning the action to the “onload” event of the page:

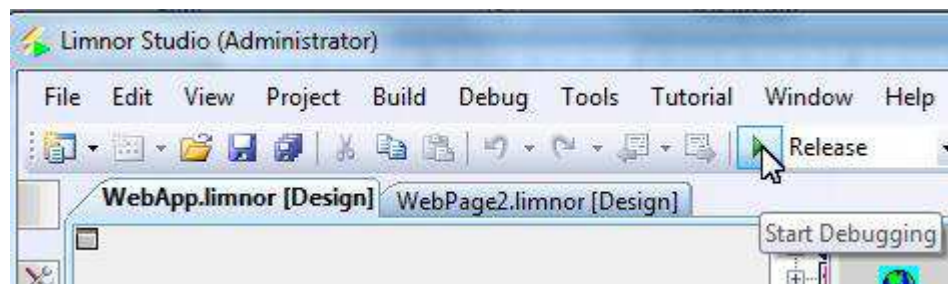


## Test Data Loading

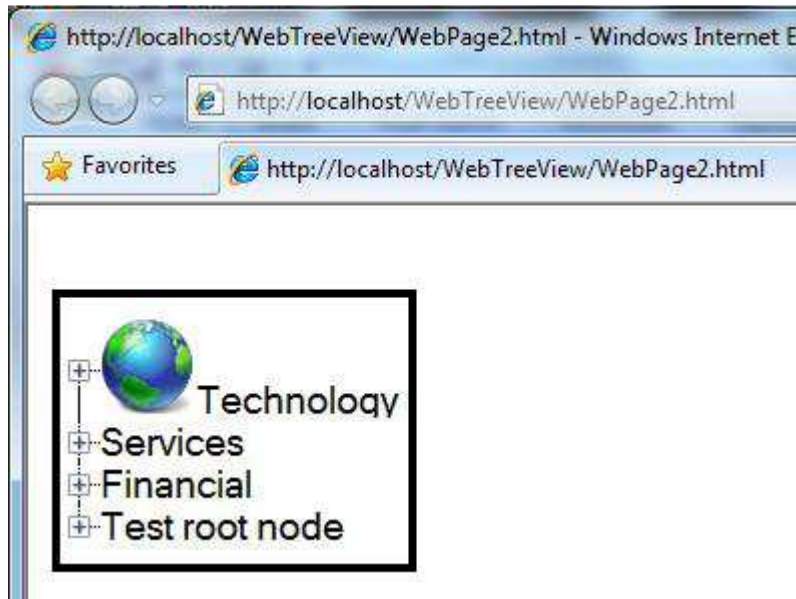
To test the web page, set it as the start page:



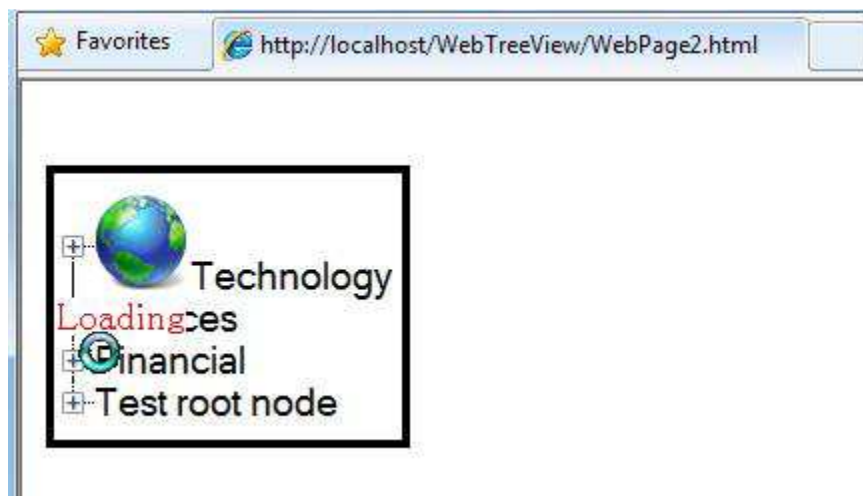
Click Run button to test the web application:



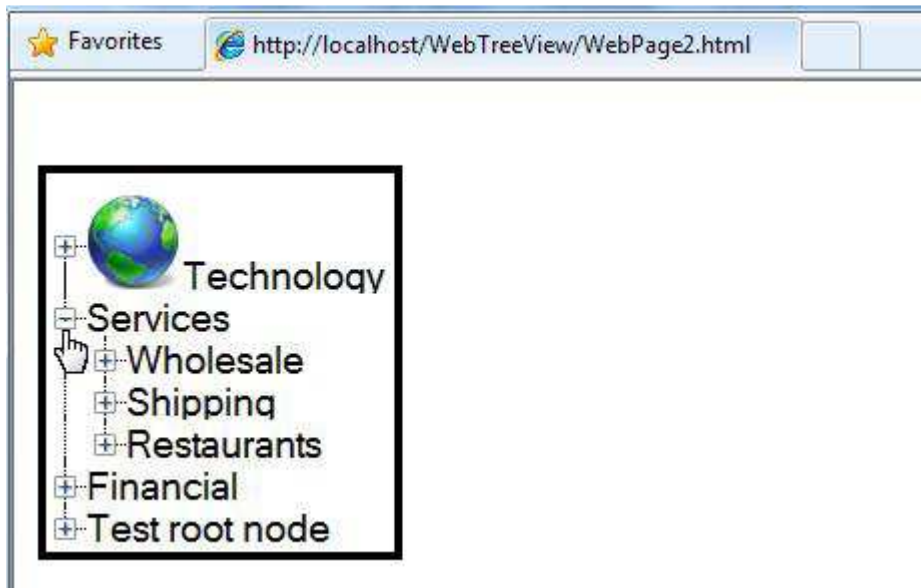
The web page appears. After a while, the data arrive at the web page from the web server, and displayed in the tree view:



Note that initially only the records for the root nodes are loaded from the database. At the first time a node is expanded, the records for the immediate child nodes of the node are loaded from the database:



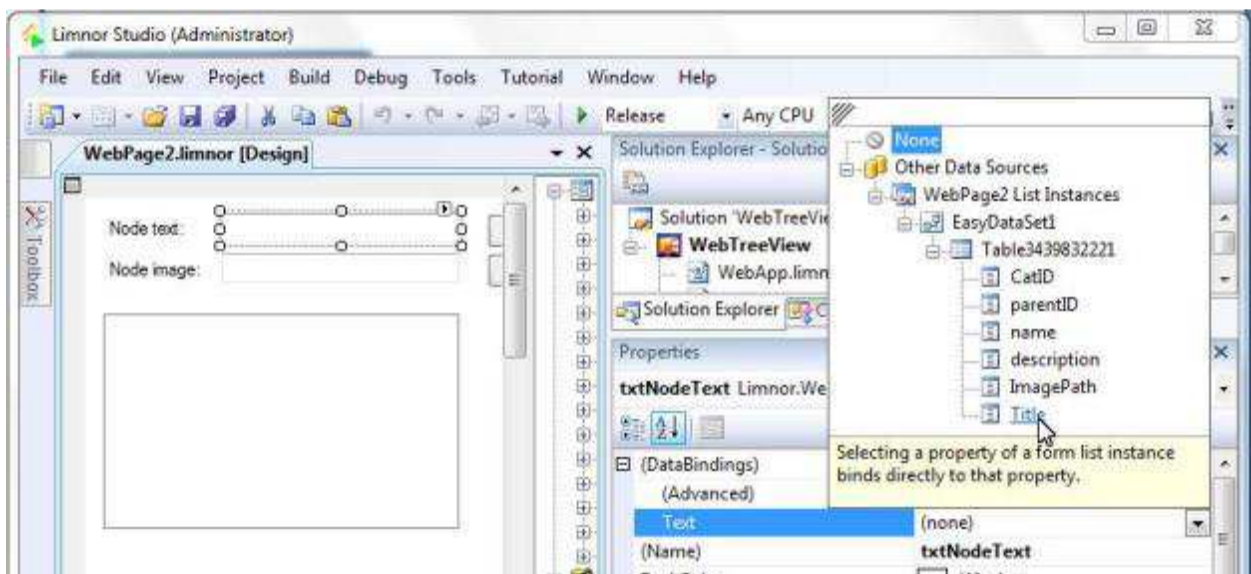


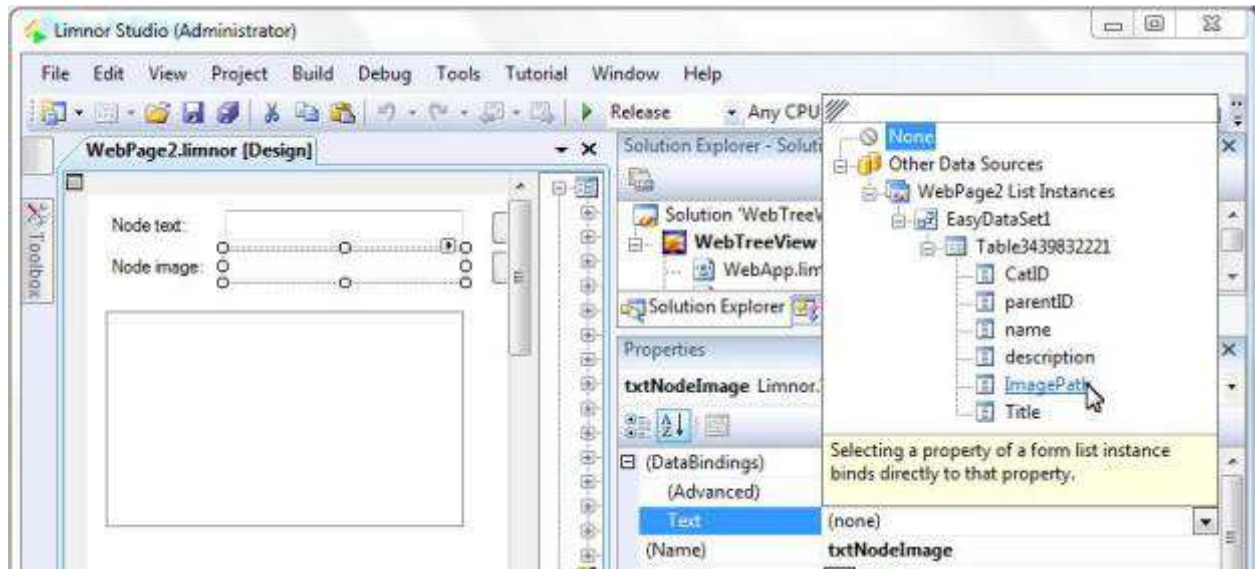


## Data Editing

### Modify nodes

Other controls may also be bound to the EasyDataSet which is used as the data source for the HtmlTreeView. For example, we may use Htmltextbox binding to the same EasyDataSet.

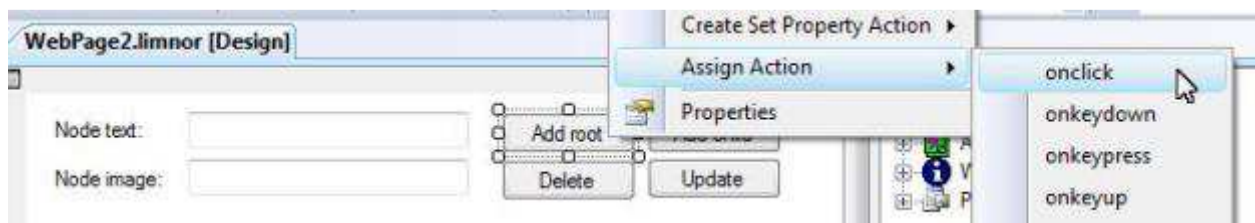




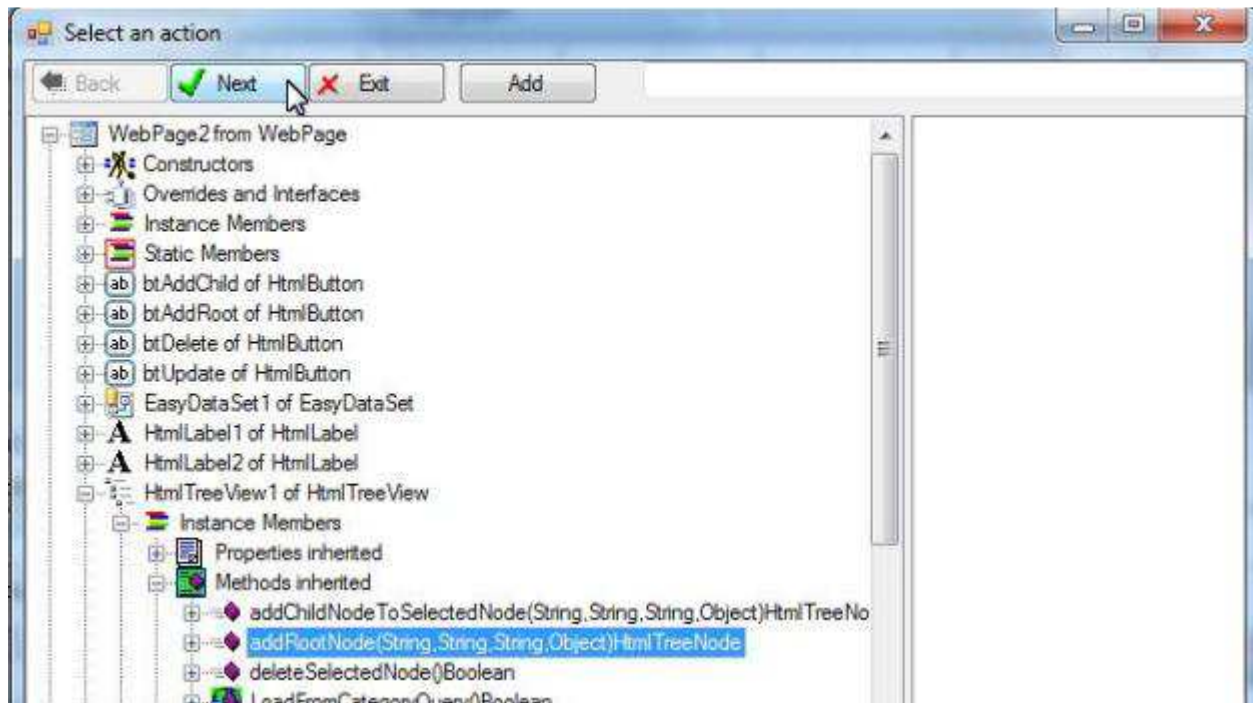
### Add root nodes

`addRootNode` should be used to create a new root node. In this sample, we use a button to execute `addRootNode`.

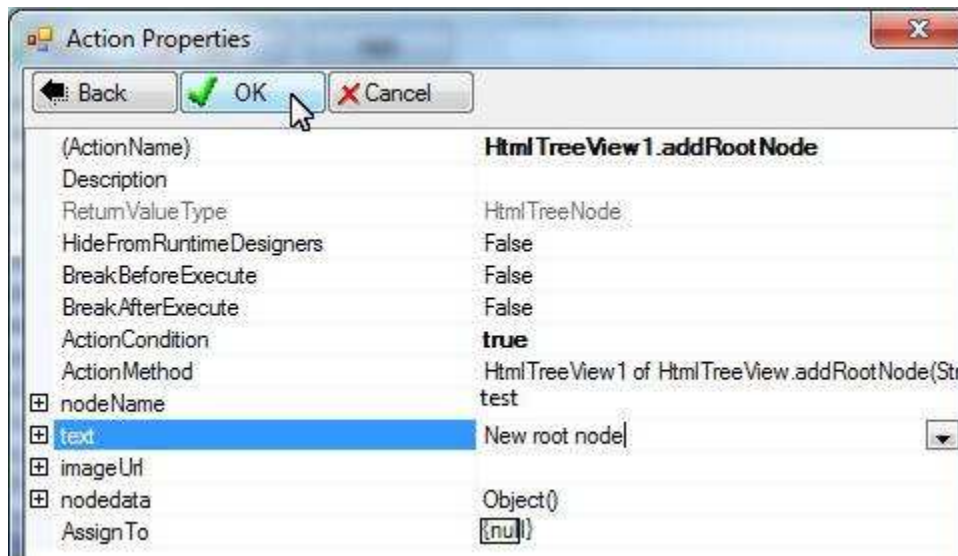
Right-click the button; choose “Assign Action”; choose “onclick” event:



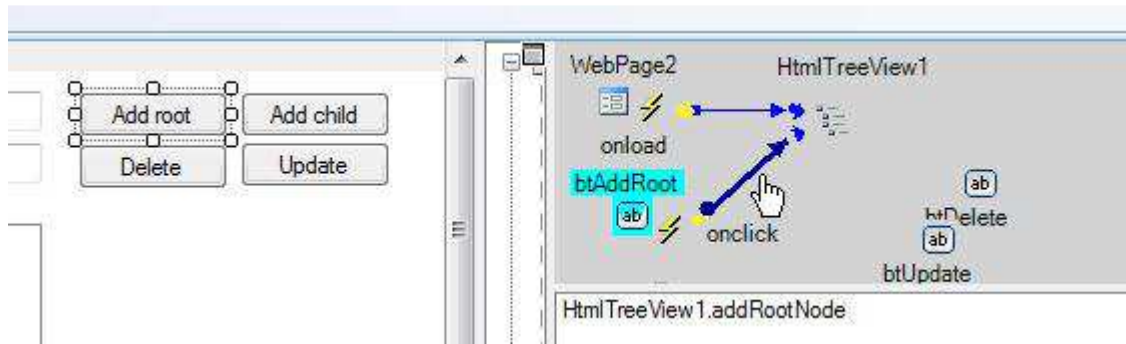
Select `addRootNode`:



Provide default node properties for the new node and click OK:



The action is created and assigned to the button:

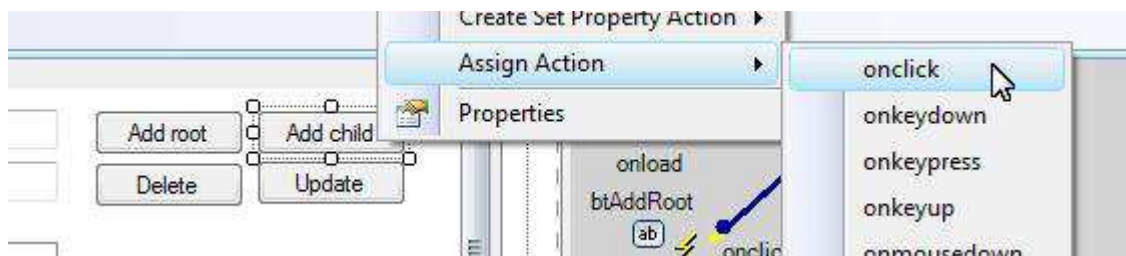


**Note that child nodes cannot be added to a new node because the new node does not have a valid primary key. To make the new node available for adding new child nodes, you may add an Update action of the EasyDataSet to the “onclick” event, following the addRootNode action.**

### Add child nodes

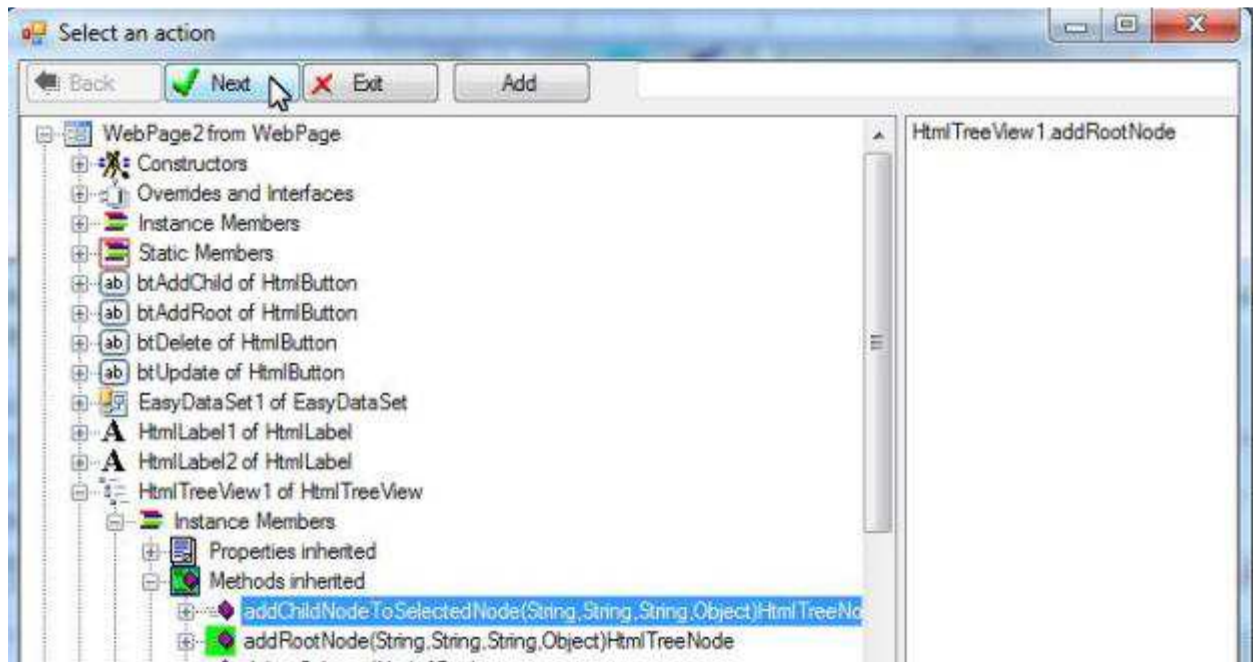
`addChildNodeToSelectedNode` should be used to create a new root node. In this sample, we use a button to execute `addChildNodeToSelectedNode`.

Right-click the button; choose “Assign Action”; choose “onclick” event:

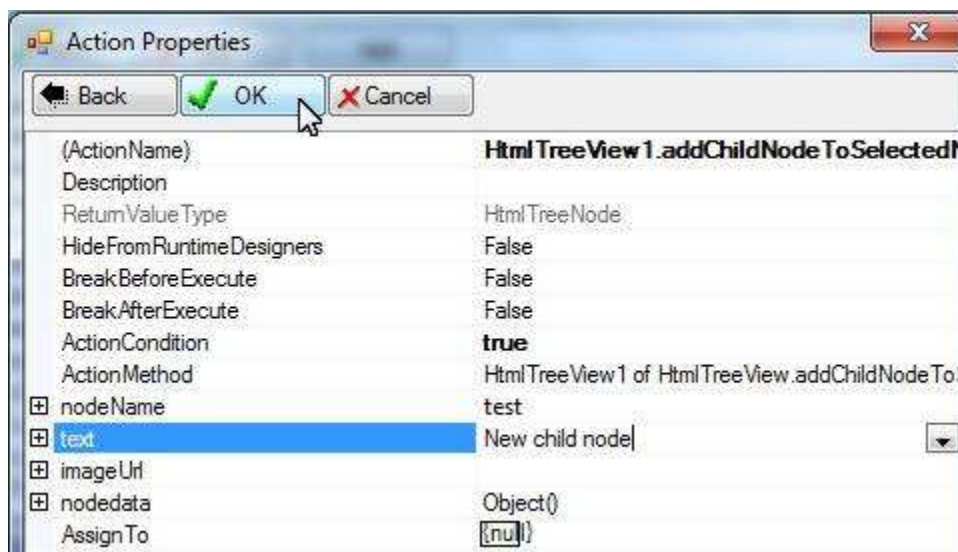


Select `addChildNodeToSelectedNode`:

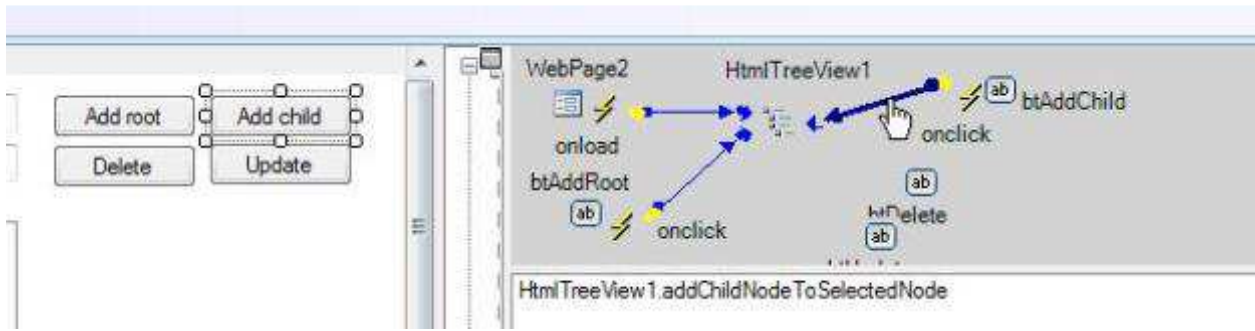




Provide default node properties for the new node and click OK:



The action is created and assigned to the button:

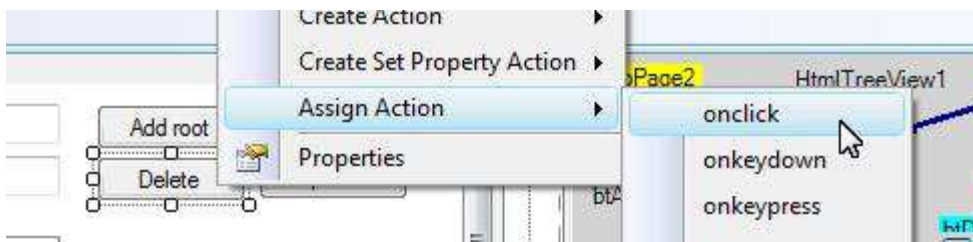


**Note** that child nodes cannot be added to a new node because the new node does not have a valid primary key. To make the new node available for adding new child nodes, you may add an Update action of the EasyDataSet to the “onclick” event, following the `addChildNodeToSelectedNode` action.

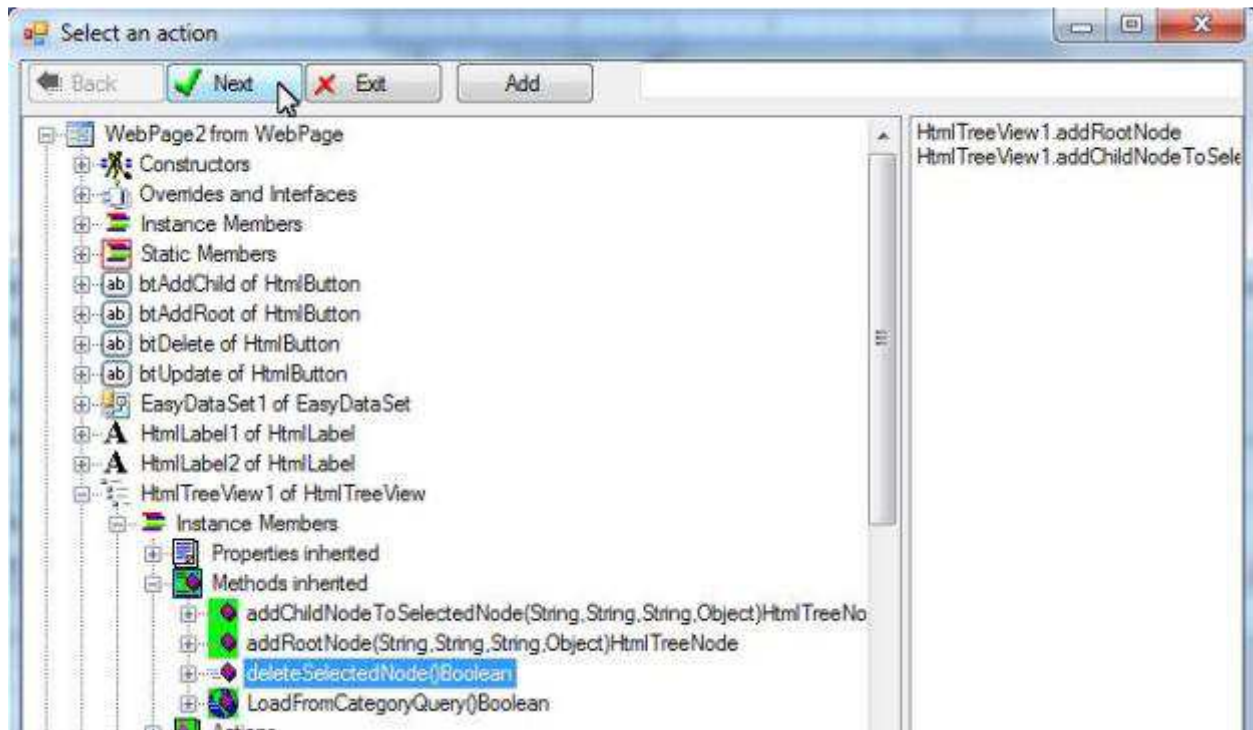
## Delete nodes

`deleteSelectedNode` should be used to delete a node. In this sample, we use a button to execute `deleteSelectedNode`.

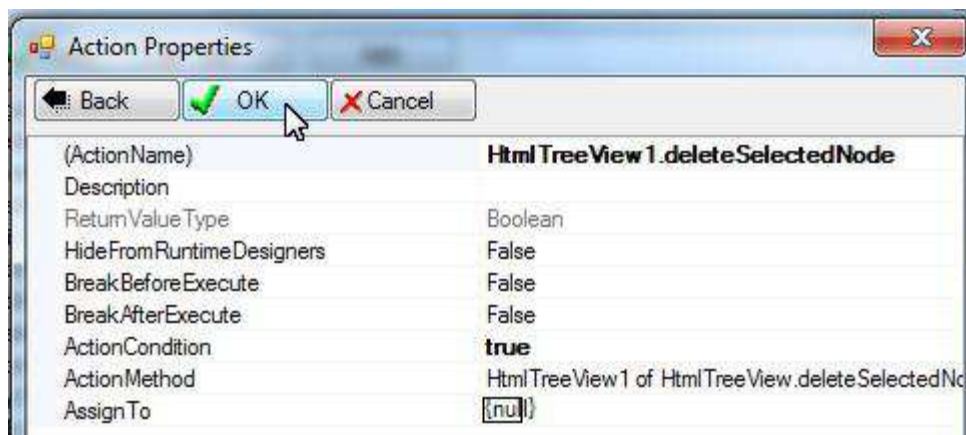
Right-click the button; choose “Assign Action”; choose “onclick” event:



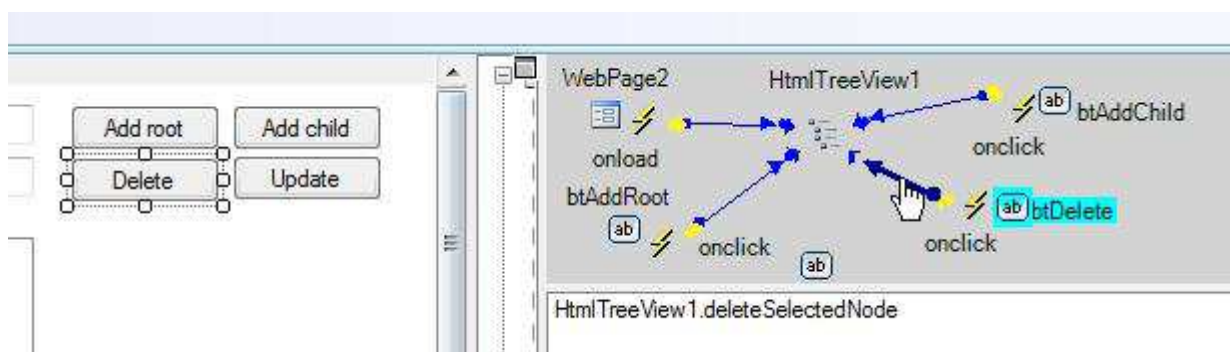
Select deleteSelectedNode:



Click OK:



The action is created and assigned to the button:



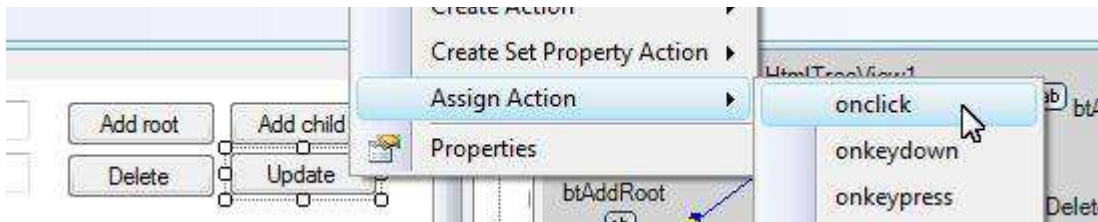
**Notes:** this action will only be executed if the selected node has expended and does not have child nodes.

### Save modifications to database

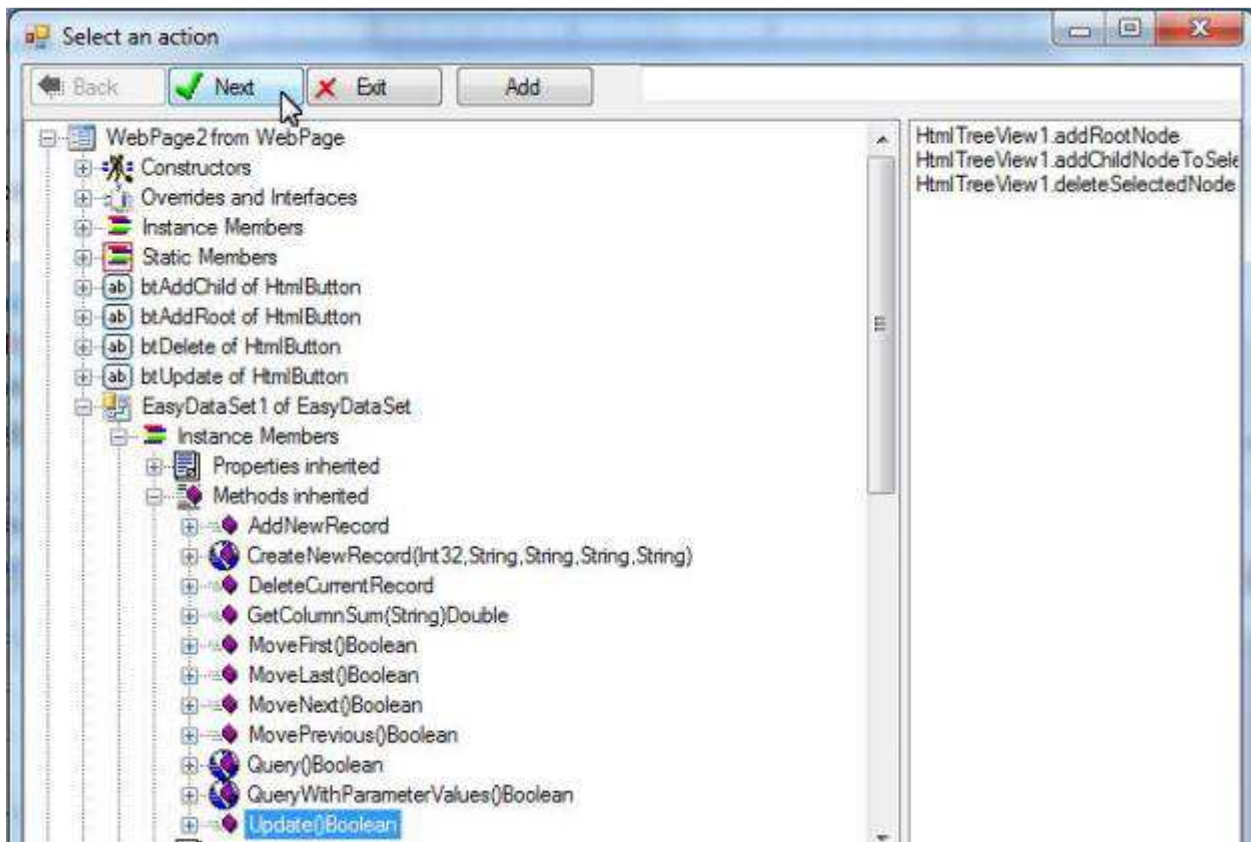
All the above editing actions only modify the tree view in the web page. The data are not saved in the database. An Update action of the EasyDataSet should be used to save all the modifications to the database.

In this sample, we use a button to execute Update.

Right-click the button; choose “Assign Action”; choose “onclick” event:

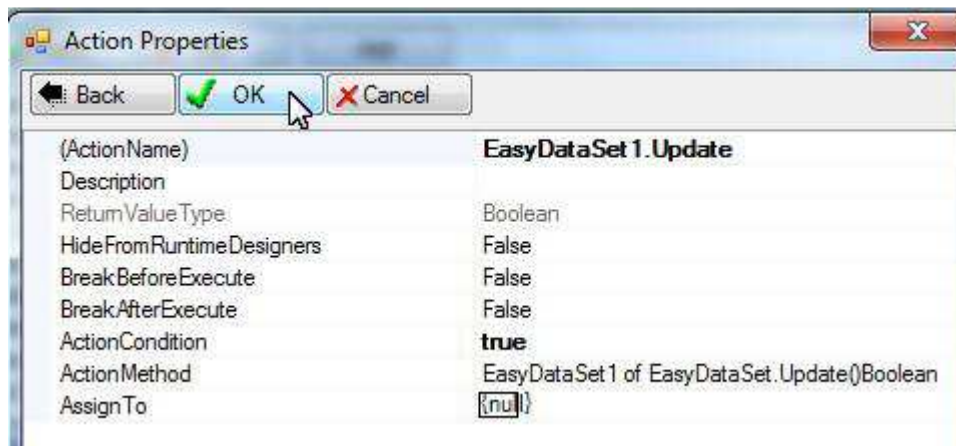


Select Update under EasyDataSet:

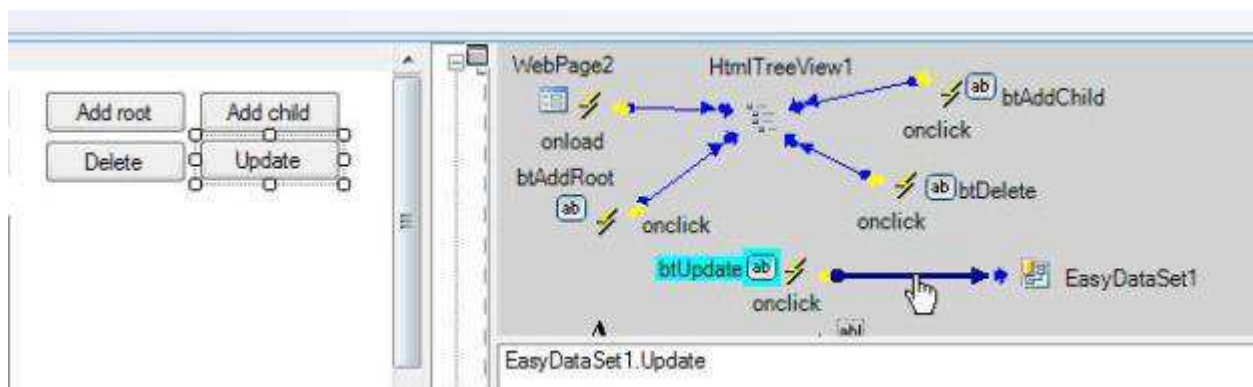


Click OK:





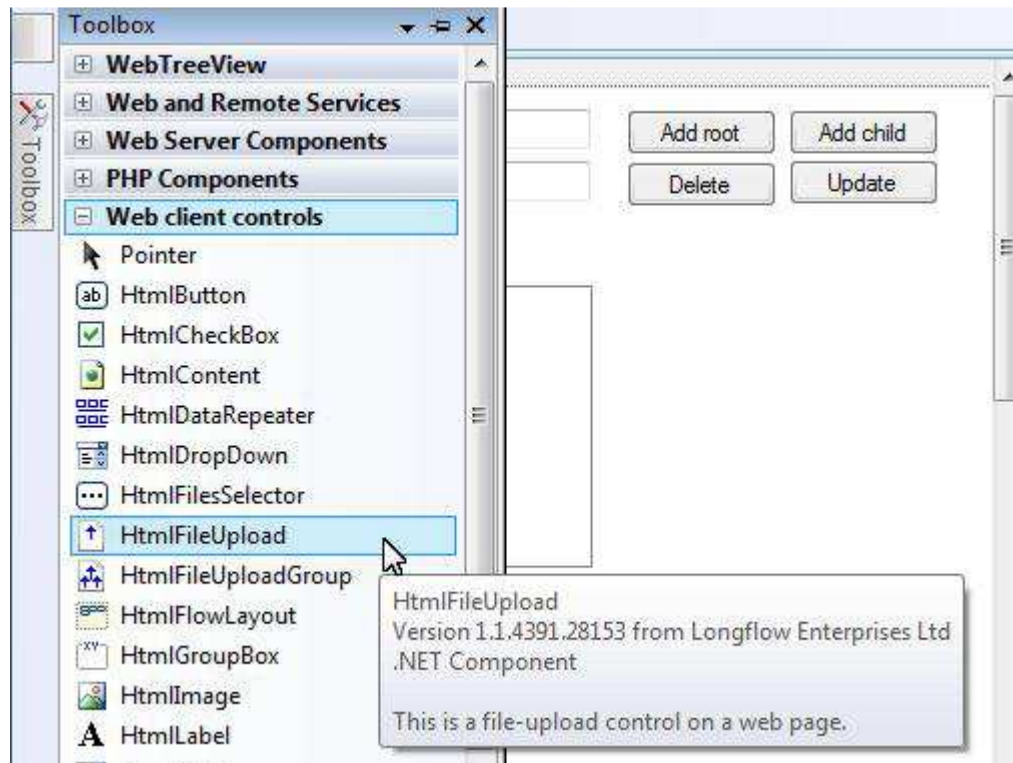
The action is created and assigned to the button:



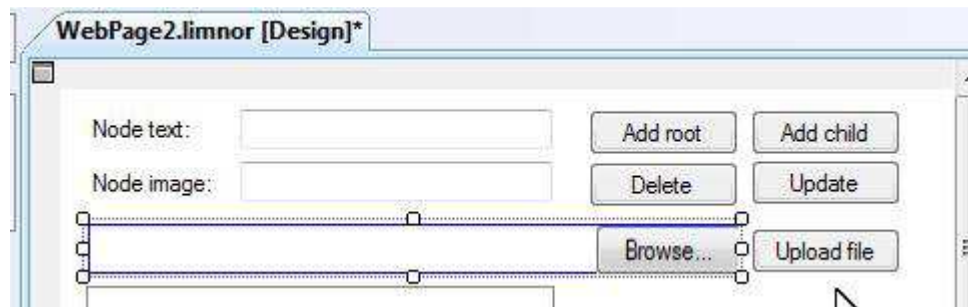
## Use File Upload

We used a text box for the web visitors to enter node image path. The image path should point to an existing file on the web server. If we want to allow the visitors to use image files on their local computer then we need to use a file upload control to upload the image files to the web server. After file uploading we may use an action to set the file path to the text box.

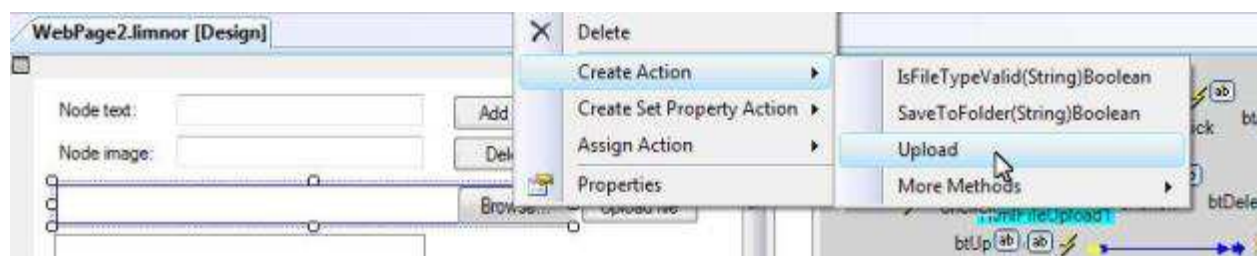
Add a file upload control to the web page:

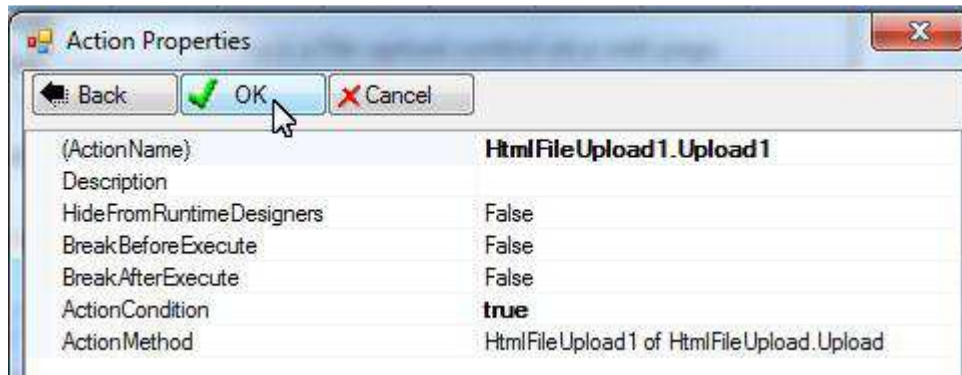


We also add a button to do file upload.

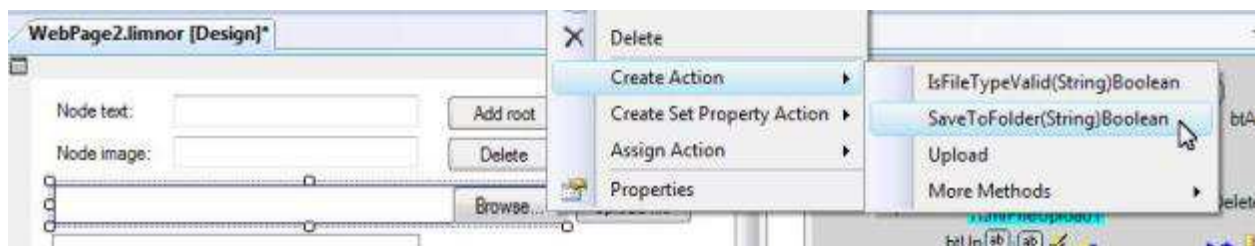


Create an Upload action to send the file to the web server:

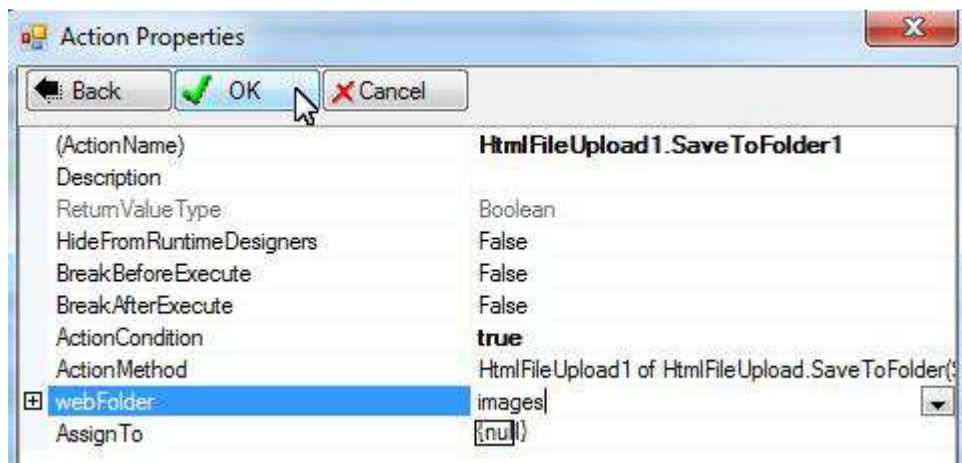




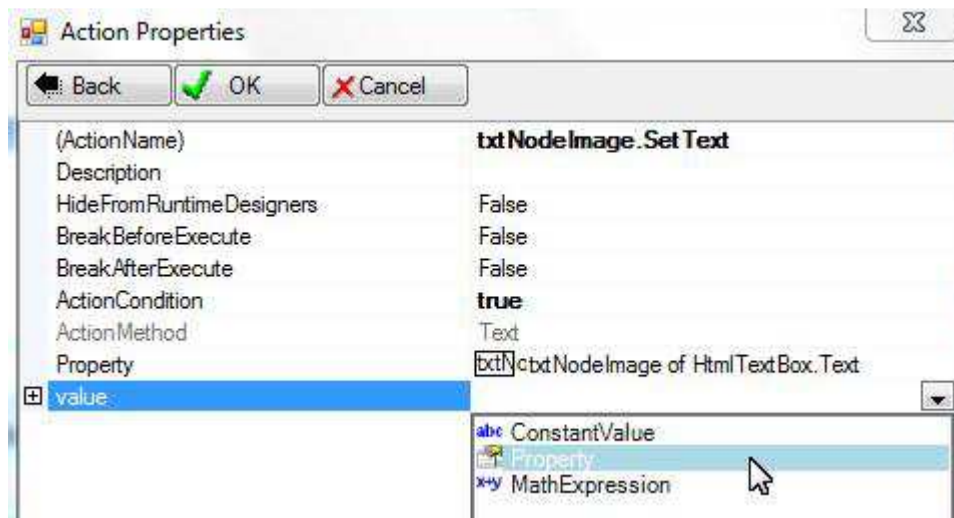
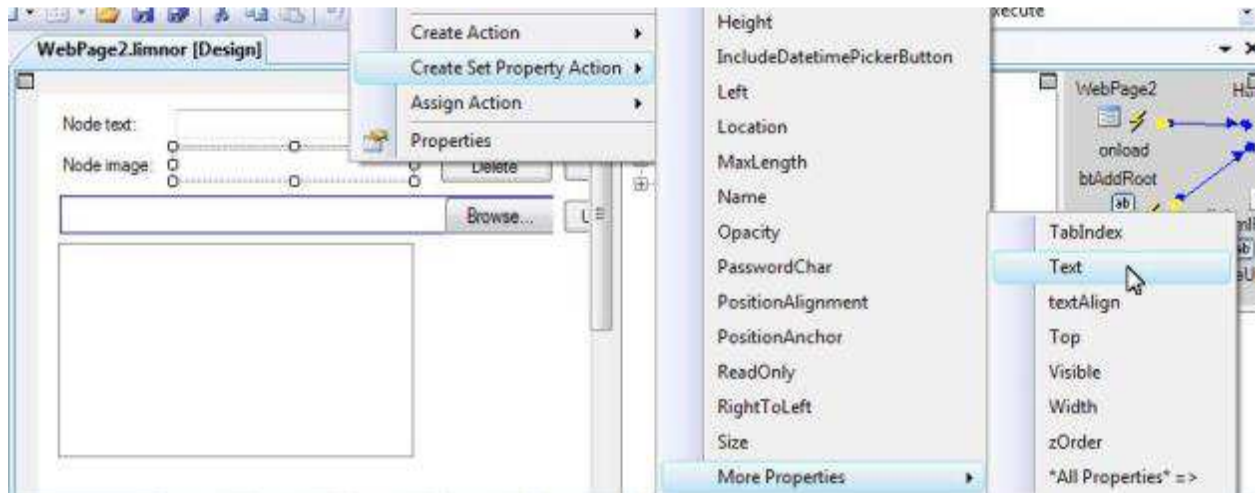
Create a SaveToFolder to save the uploaded file to a folder on the web server:



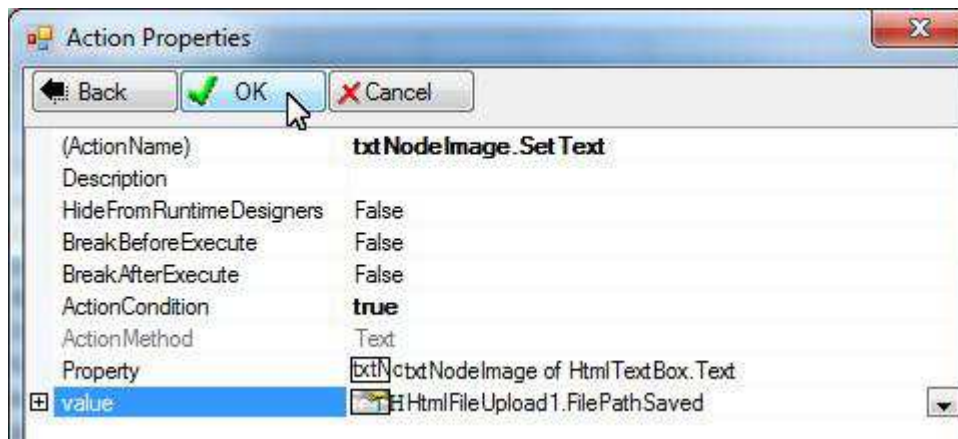
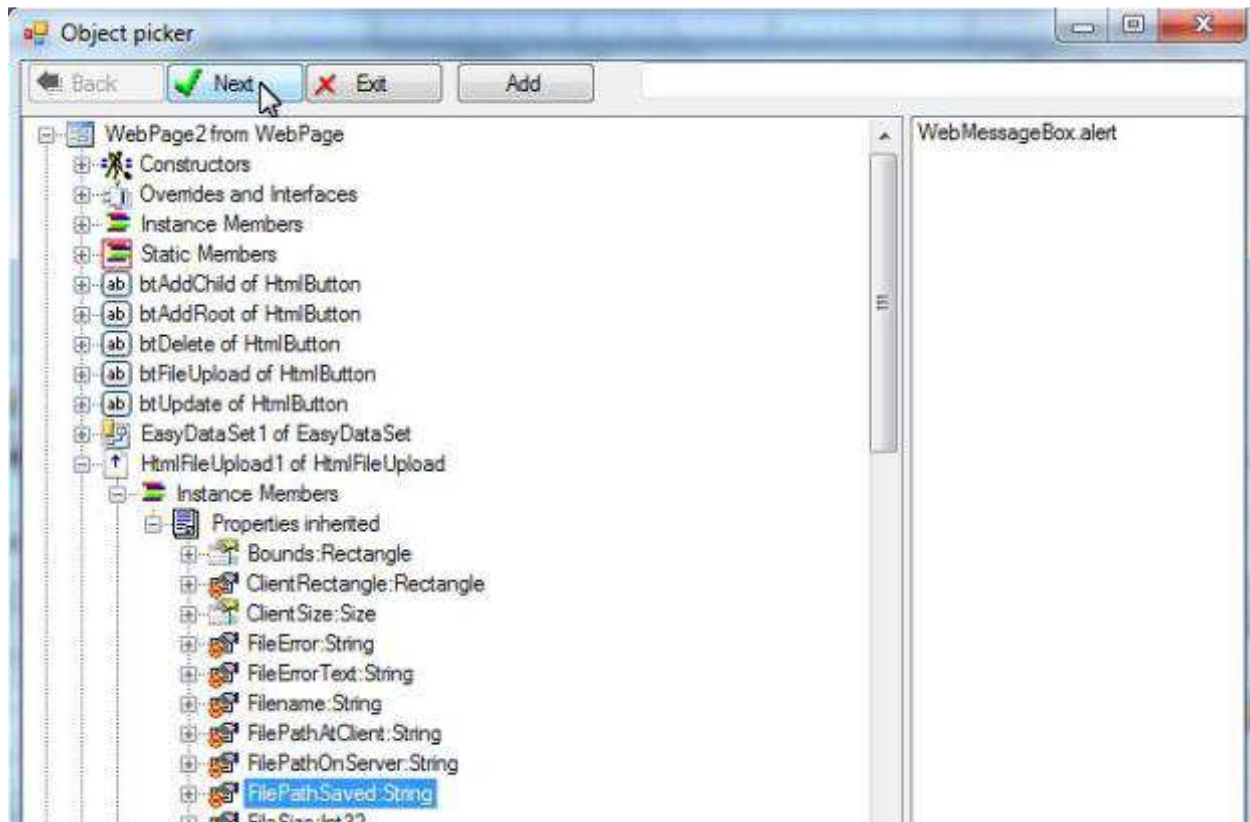
Provide a folder name and click OK:



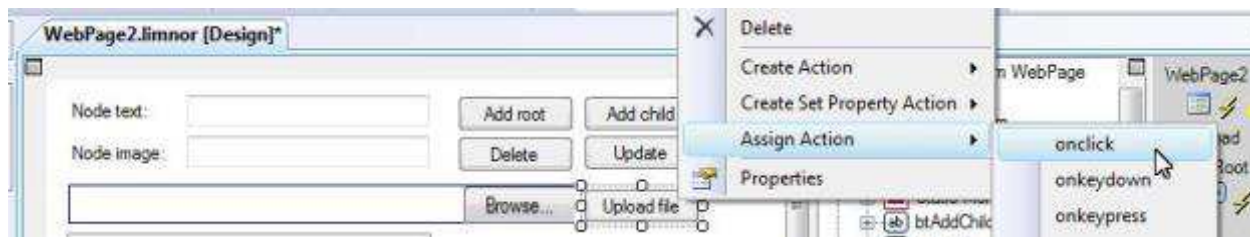
Create an action to set the uploaded file path to the text box for node image:

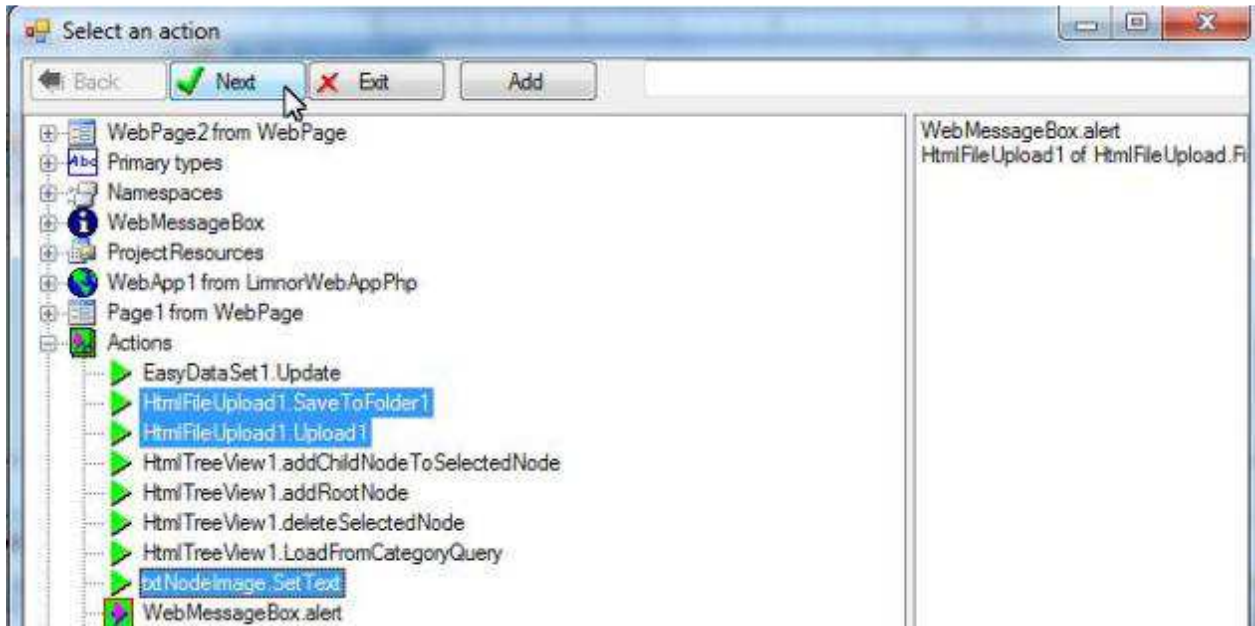




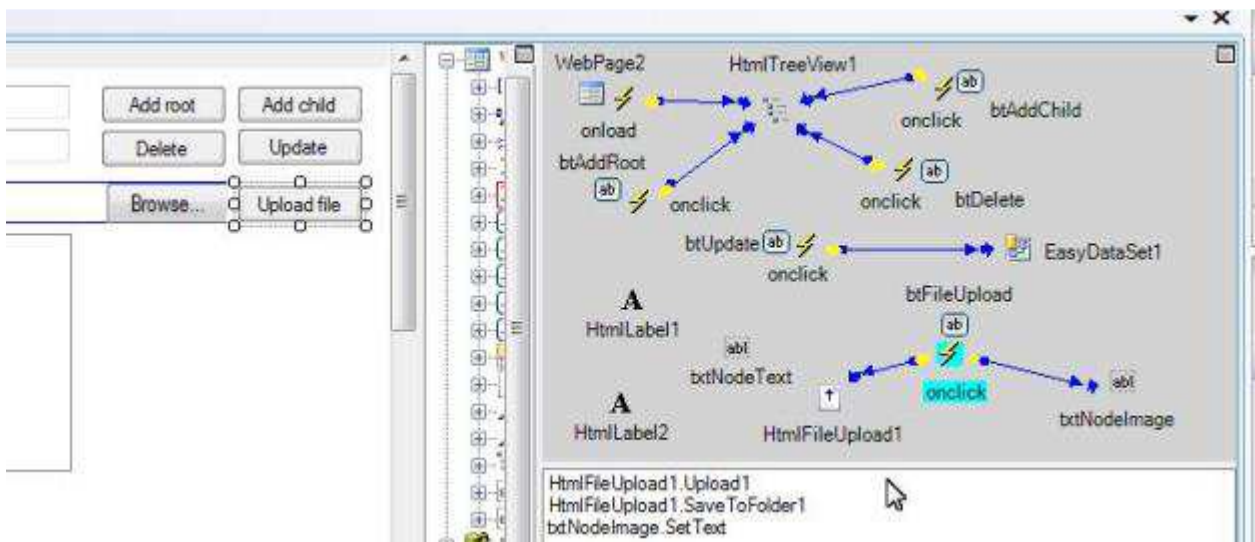


Assign the above 3 actions to the button:



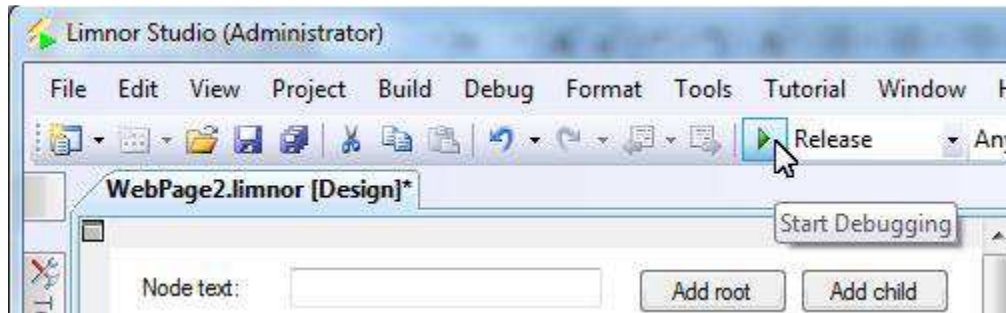


Note that the Upload action must be the first action in the event handling:

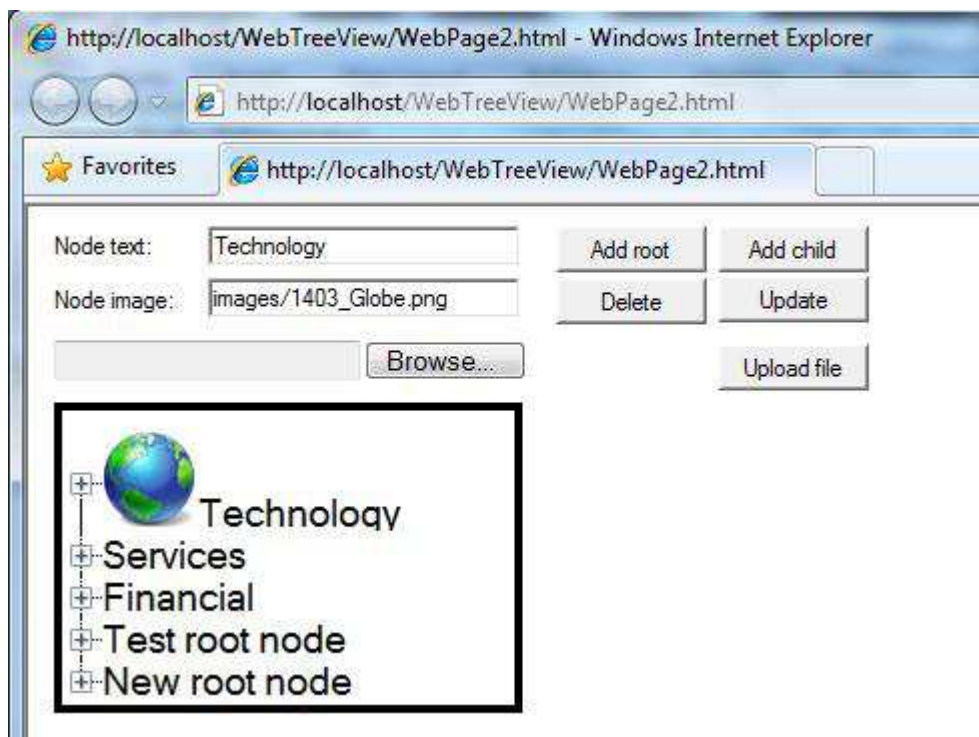


## Data-Binding Test

Click the Run button to test the web application.




The web page appears. After a while and data are displayed in the tree view. We also see the data in the text boxes.



Select different nodes. We will see the data in the text boxes switch to the data of the selected node.

Node text:


Node image:



Browse an image file in the local computer and click “Upload file”:

Node text:

Node image:




The file is uploaded to the web server and the image appears on the node:



Node text:

Node image:

C:\VS2008ImageLibrary\O



Click "Update" button when we finish making all the modifications to tree nodes.

## Recursion for MySQL

### Recursive query problem

Usually the data for the tree is linked to other data. For example, suppose the tree is a category for products. Each product may be linked to a category ID. One common question is that given a category ID, how can we get all products linked to the category ID and all its sub category ID's ?

Following is an example of **incorrect** query:

```
Select * from product where catID in (select catID from category where catID = @catID or
parentID=@catID)
```

The above query only gets products for the given category ID, @catID, and its immediate child category, but missed the grand child categories of the given category.

Some database engines support recursive query syntax for including all levels of child categories. But MySQL does not support it.

### One solution

Here we provide one solution for the problem.

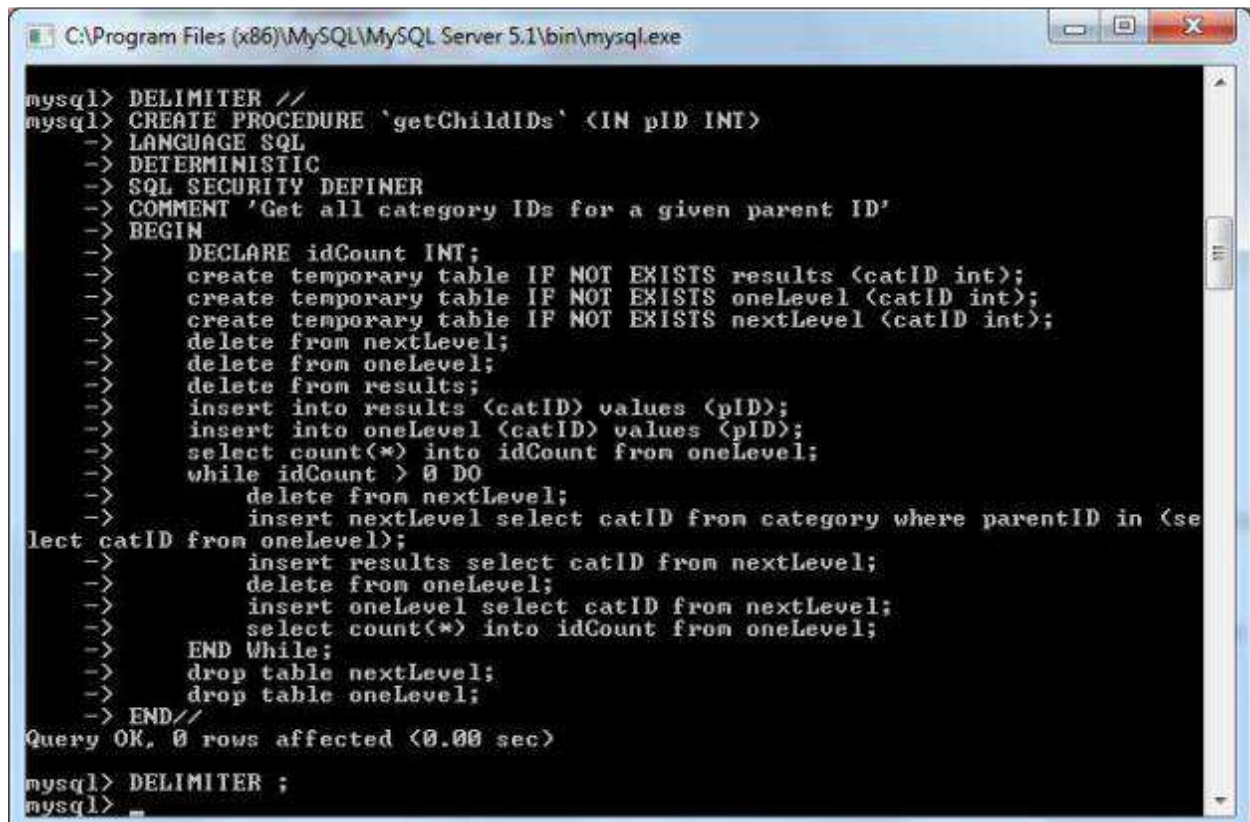
1. Create a stored-procedure in MySQL to get child ID's of all levels for a given parent ID. All ID's are saved in a temporary table. Suppose the temporary table is named Results.
2. Use the temporary table to filter the SELECT query. For example  
Select \* from product where catID in (select catID from Results)

## Create stored-procedure

For the sample we used here, one implementation of the above solution can be as following.

A stored-procedure named getChildIDs is created:

```
DELIMITER //
CREATE PROCEDURE `getChildIDs` (IN pID INT)
LANGUAGE SQL
DETERMINISTIC
SQL SECURITY DEFINER
COMMENT 'Get all category IDs for a given parent ID'
BEGIN
    DECLARE idCount INT;
    create temporary table IF NOT EXISTS results (catID int);
    create temporary table IF NOT EXISTS oneLevel (catID int);
    create temporary table IF NOT EXISTS nextLevel (catID int);
    delete from nextLevel;
    delete from oneLevel;
    delete from results;
    insert into results (catID) values (pID);
    insert into oneLevel (catID) values (pID);
    select count(*) into idCount from oneLevel;
    while idCount > 0 DO
        delete from nextLevel;
        insert nextLevel select catID from category where parentID in (select catID from oneLevel);
        insert results select catID from nextLevel;
        delete from oneLevel;
        insert oneLevel select catID from nextLevel;
        select count(*) into idCount from oneLevel;
    END While;
    drop table nextLevel;
    drop table oneLevel;
END//
DELIMITER ;
```



```

C:\Program Files (x86)\MySQL\MySQL Server 5.1\bin\mysql.exe

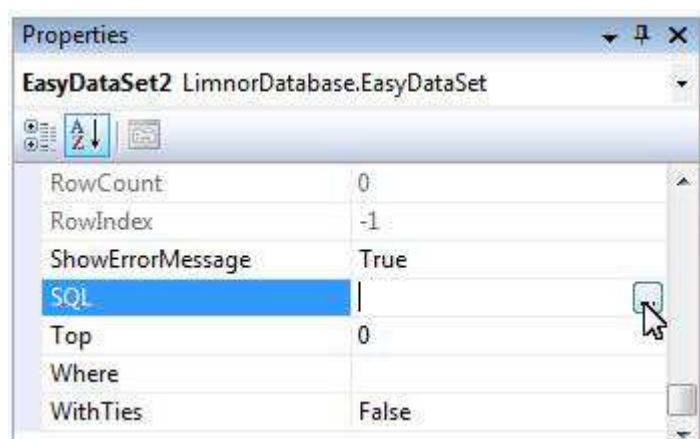
mysql> DELIMITER //
mysql> CREATE PROCEDURE 'getChildIDs' (IN pID INT)
-> LANGUAGE SQL
-> DETERMINISTIC
-> SQL SECURITY DEFINER
-> COMMENT 'Get all category IDs for a given parent ID'
-> BEGIN
->     DECLARE idCount INT;
->     create temporary table IF NOT EXISTS results (catID int);
->     create temporary table IF NOT EXISTS oneLevel (catID int);
->     create temporary table IF NOT EXISTS nextLevel (catID int);
->     delete from nextLevel;
->     delete from oneLevel;
->     delete from results;
->     insert into results (catID) values (pID);
->     insert into oneLevel (catID) values (pID);
->     select count(*) into idCount from oneLevel;
->     while idCount > 0 DO
->         delete from nextLevel;
->         insert nextLevel select catID from category where parentID in (se
lect catID from oneLevel);
->         insert results select catID from nextLevel;
->         delete from oneLevel;
->         insert oneLevel select catID from nextLevel;
->         select count(*) into idCount from oneLevel;
->     END While;
->     drop table nextLevel;
->     drop table oneLevel;
-> END//
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql> _

```

### Use stored-procedure in query

Suppose we want to display all levels of sub categories in a table. Drop a new EasyDataSet to the web page and set its SQL property to get the data:



Use the following query to get data:

```

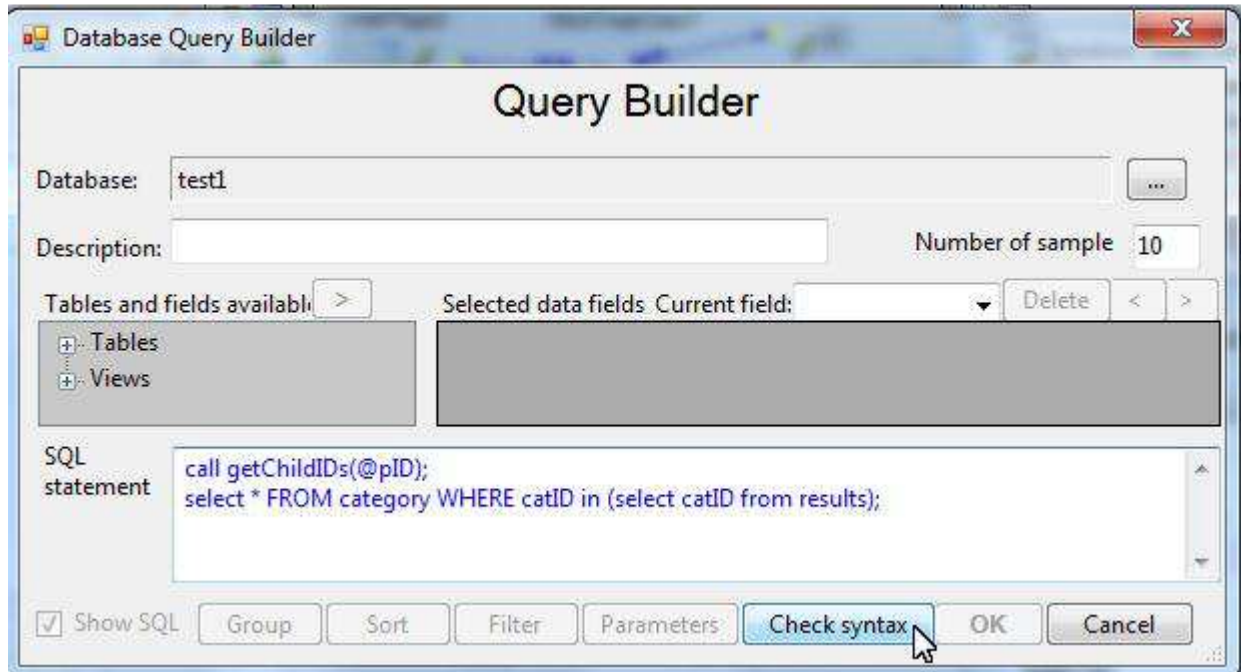
call getChildIDs(@pID);
select * FROM category WHERE catID in (select catID from results);

```

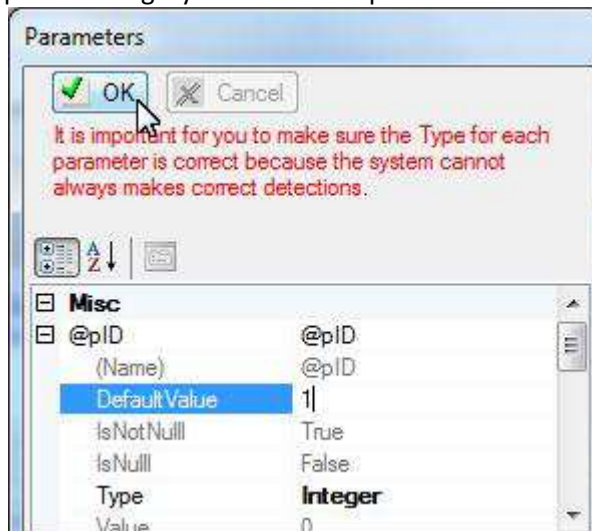
If in your case you want to get product records then it can be

call getChildIDs(@pID);

select \* FROM **product** WHERE catID in (select catID from results);

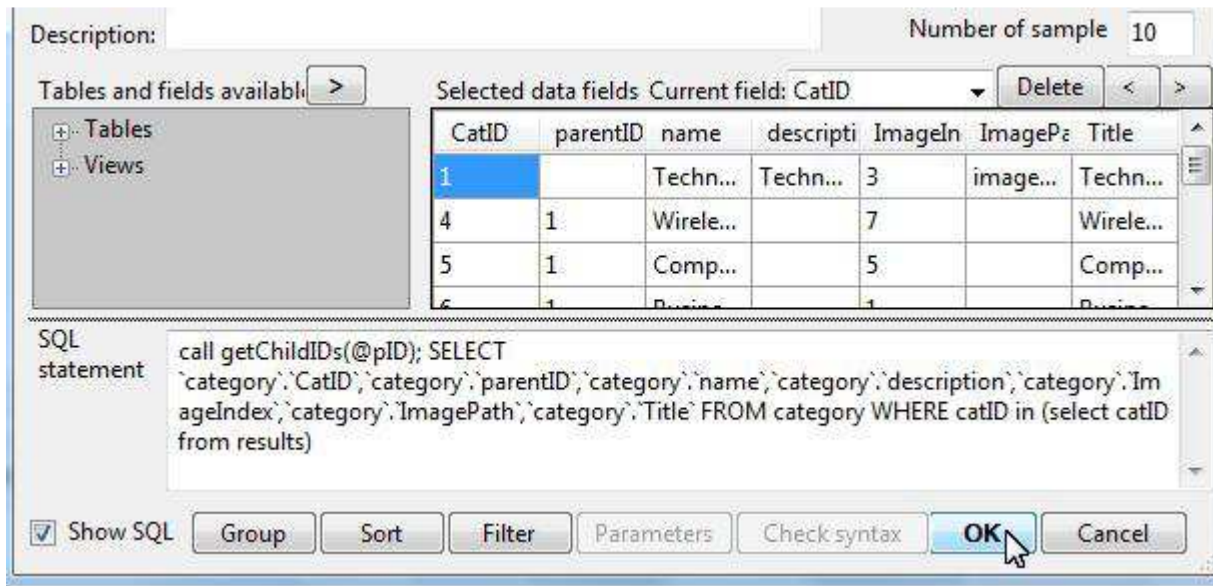


It asks us to give the data type for the parameter, @pID, we used in the query, representing the given parent category ID. In this sample we need to use integer:

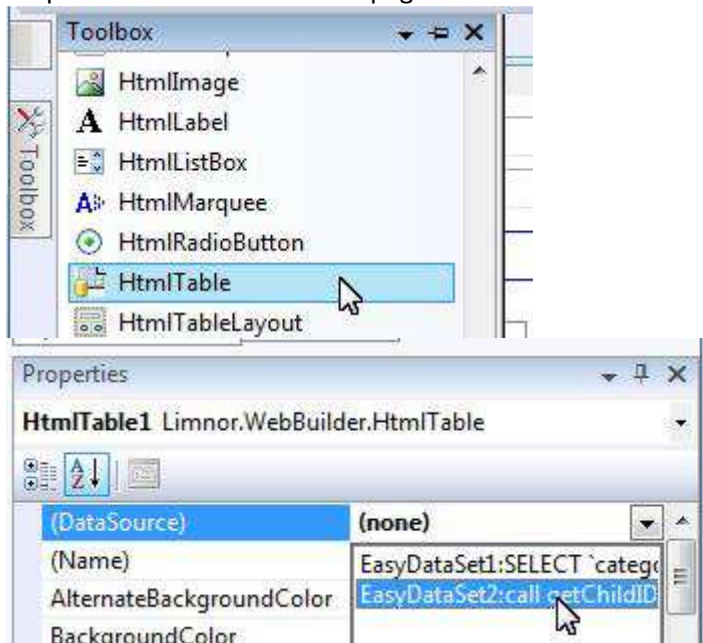


10 sample records appear:





Drop an HtmlTable to the web page and bind it to the new EasyDataSet:

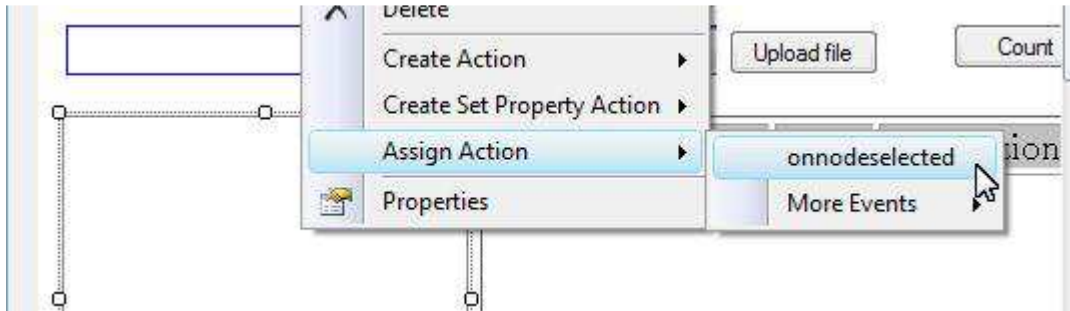


Now the question is when to load the data and how to provide the parent category ID.

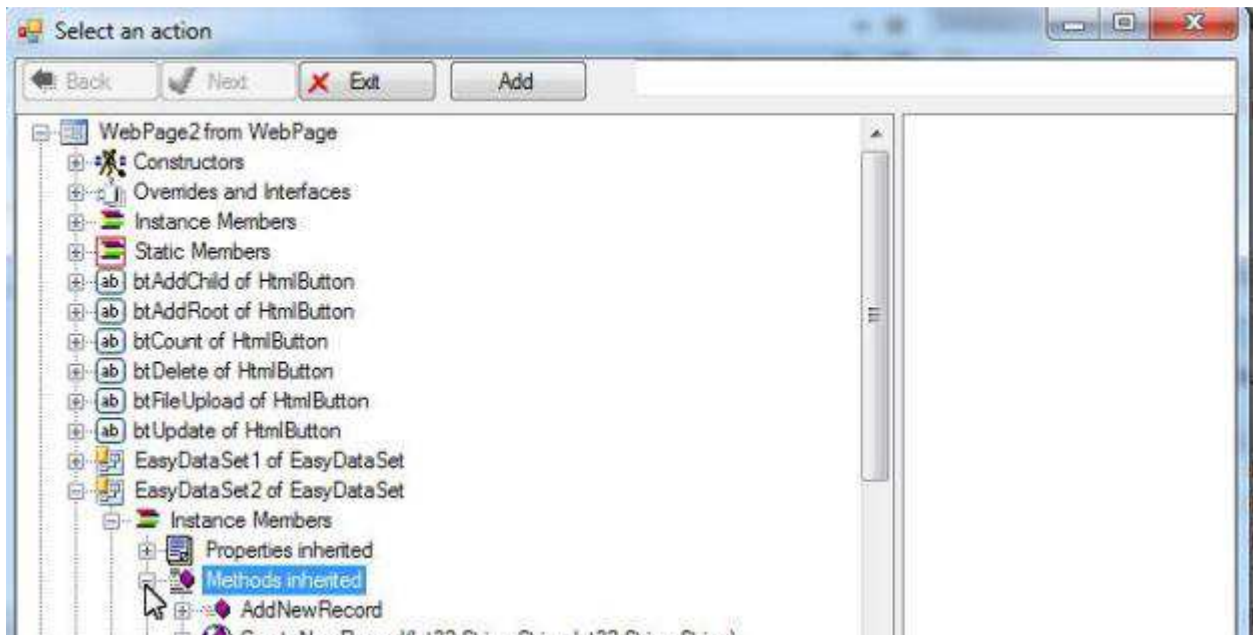
### Load data when a tree node is selected

We want to load the data when a tree node is selected and use the catID value of the selected tree node as the parent category ID for loading the data.

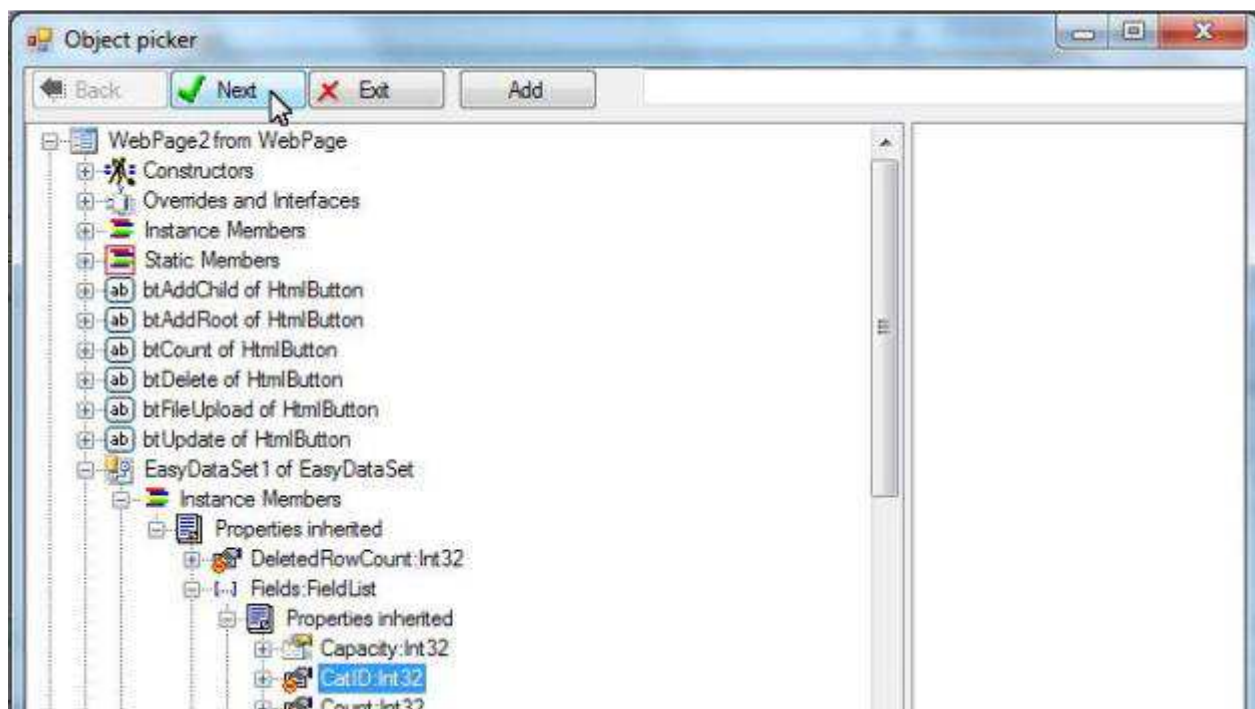
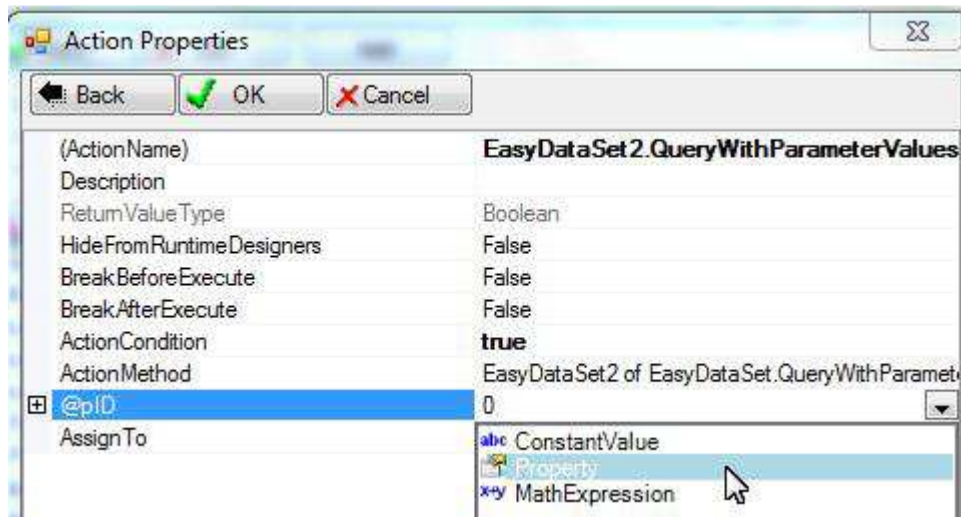
Right-click the tree view; choose "Assign Action"; choose "onnodesselected" event:



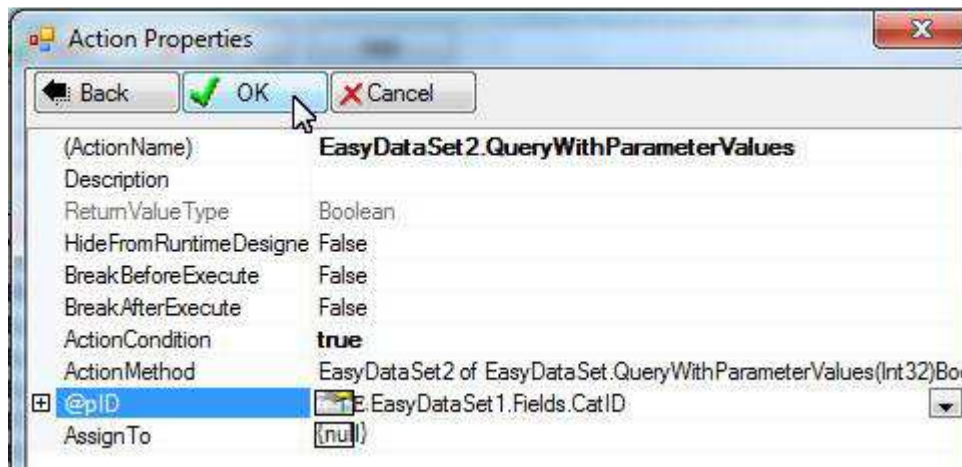
Select the QueryWithParameterValues method of the second EasyDataSet:



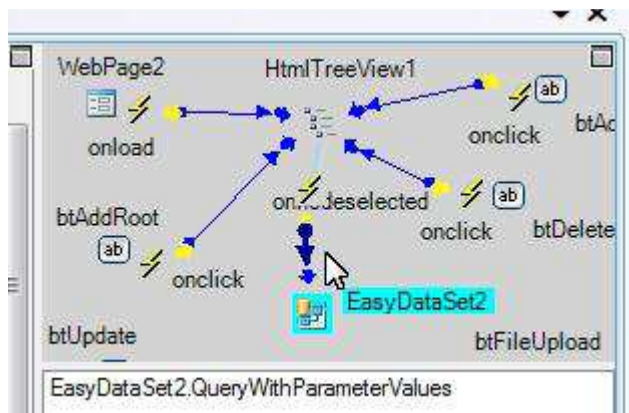
Provide the catID value from the first EasyDataSet to the @pID of the action:



Click OK:

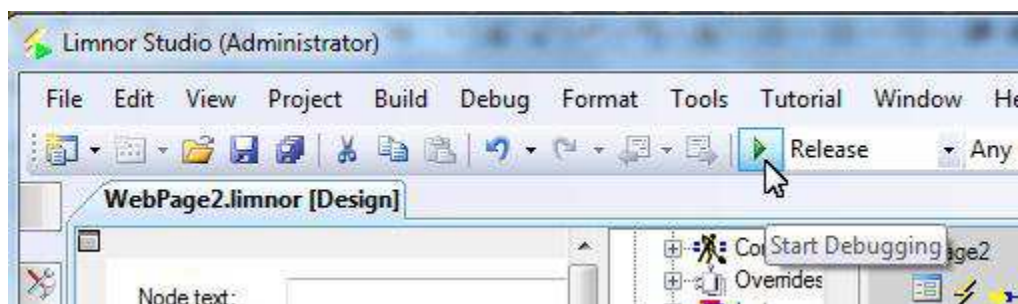


The action is created and assigned to the event:



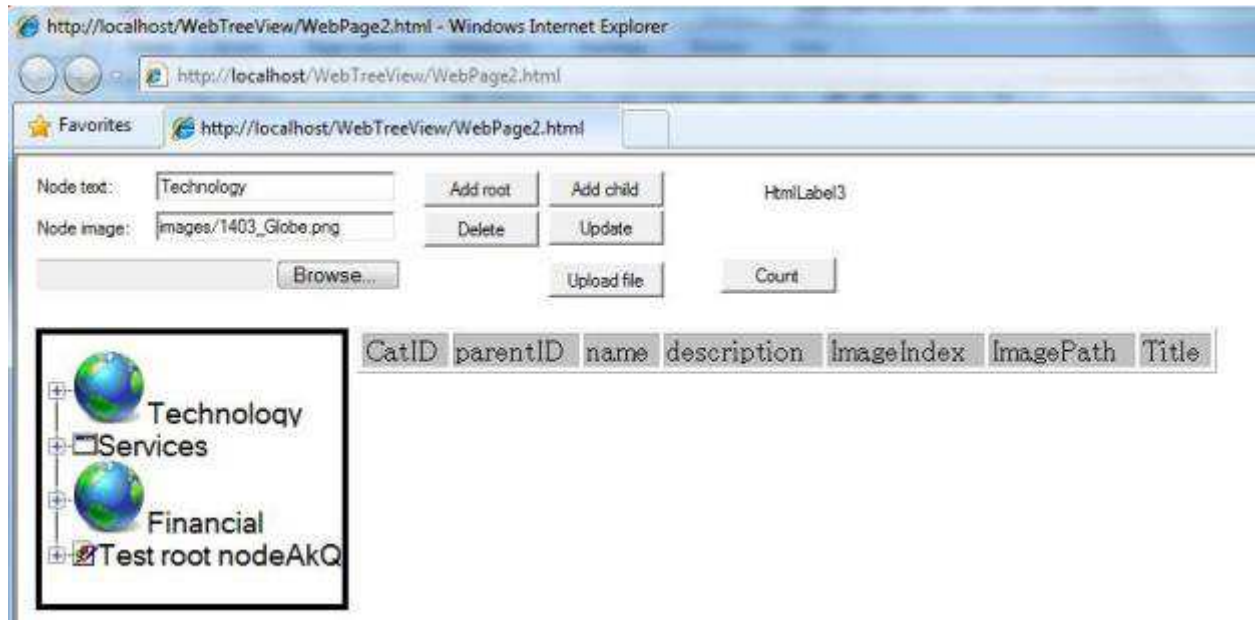
## Test

Run the sample:

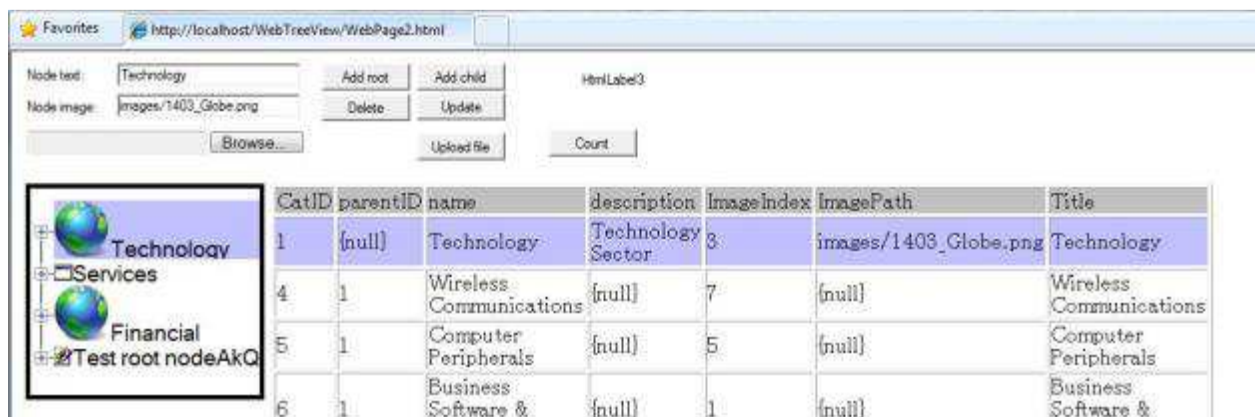


The web page appears and the tree loaded:





Select a tree node; the corresponding categories are displayed in the table:

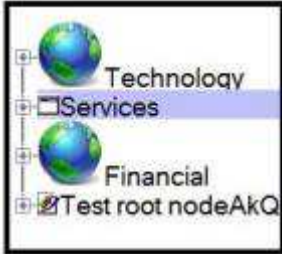


Select another tree node; the corresponding categories are displayed in the table:

Node text:

Node image:

HtmlLabel3

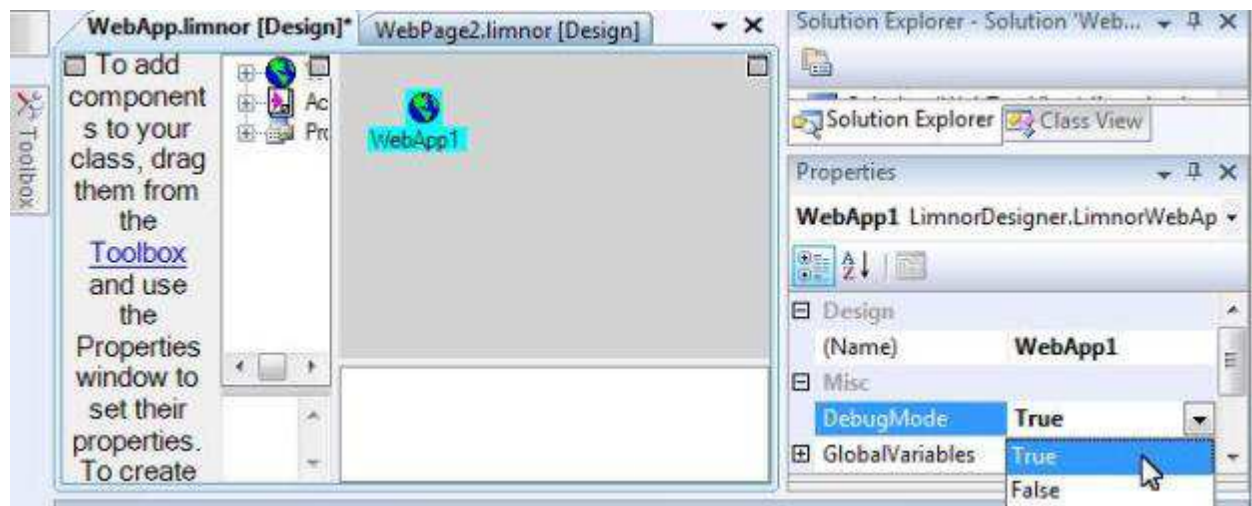


CatID	parentID	name	description	ImageIndex	ImagePath	Title
2	{null}	Services	Services Sector	2	images/EXE.BMP	Services
18	2	Wholesale	{null}	0	{null}	Wholesale
19	2	Shipping	{null}	0	{null}	Shipping
20	2	Restaurants	{null}	0	{null}	Restaurants
24	18	ws-Test1	{null}	0	{null}	ws-Test1
25	18	ws-Test2	{null}	0	{null}	ws-Test2
26	19	By Land	{null}	0	{null}	By Land
27	19	By Air	{null}	0	{null}	By Air
36	2	test	{null}	{null}		New child node

## Debugging

Data-binding makes loading and editing the tree nodes easy. But when your web page does not run as you expected, you want to know what are going on behind the scene.

Turn on Debug mode will show the message exchanges between web pages and the web server.

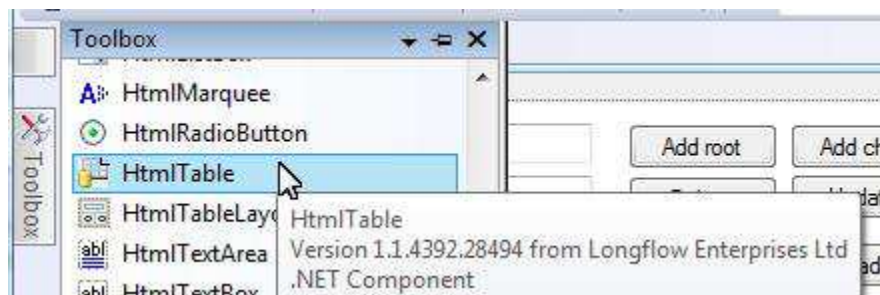


Whenever the web page connects to the web server, the information the web page sends to the web server and the information downloaded from the web server are displayed:

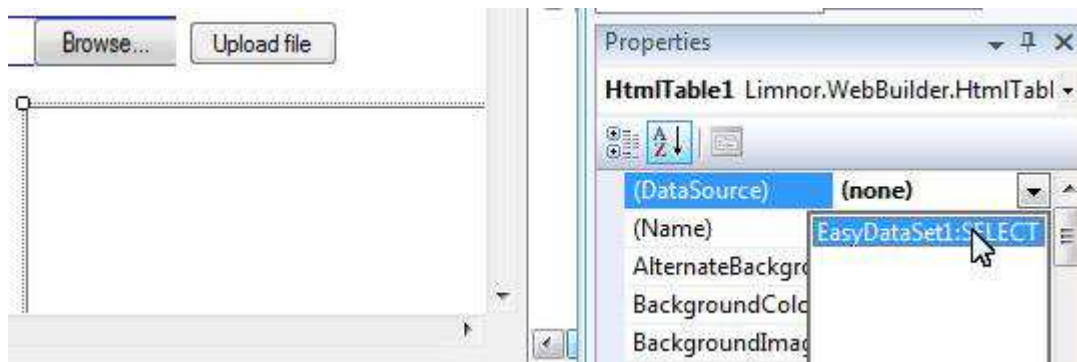


The messages will be displayed in a pop-up window. So, you must turn off popup-blocker in your web browser.

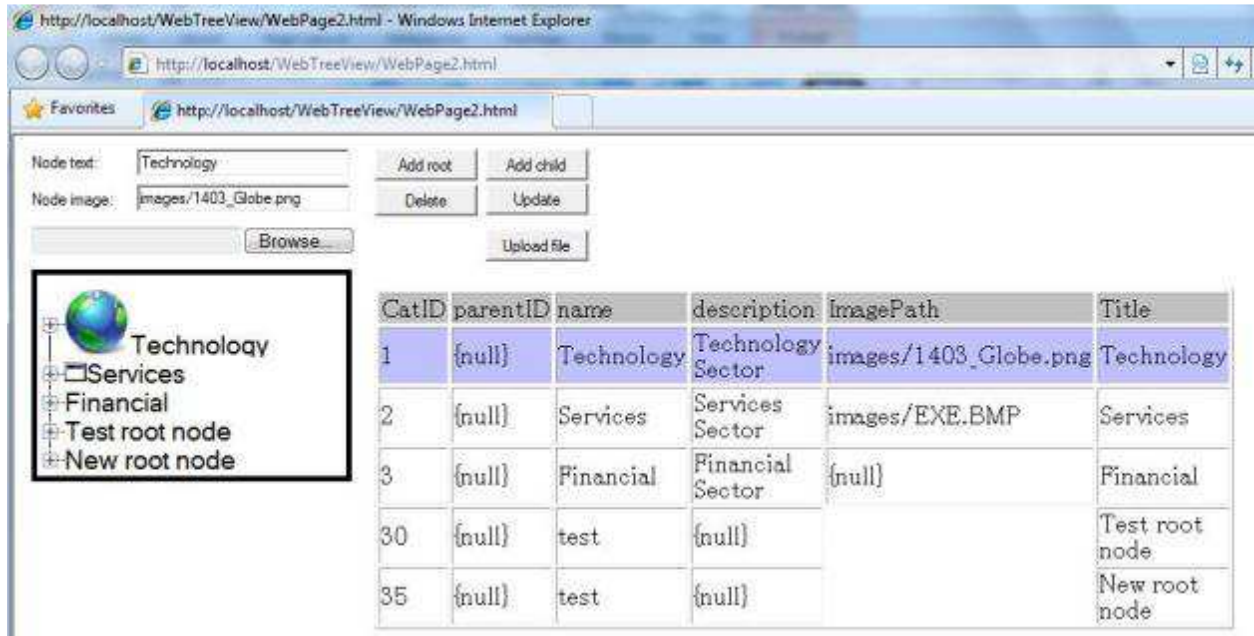
You may also add an HtmlTable temporarily to the web page and bind it to the same EasyDataSet used as the data source for the HtmlTreeView. The HtmlTable will display the data loading.



Set its DataSource to the EasyDataSet:

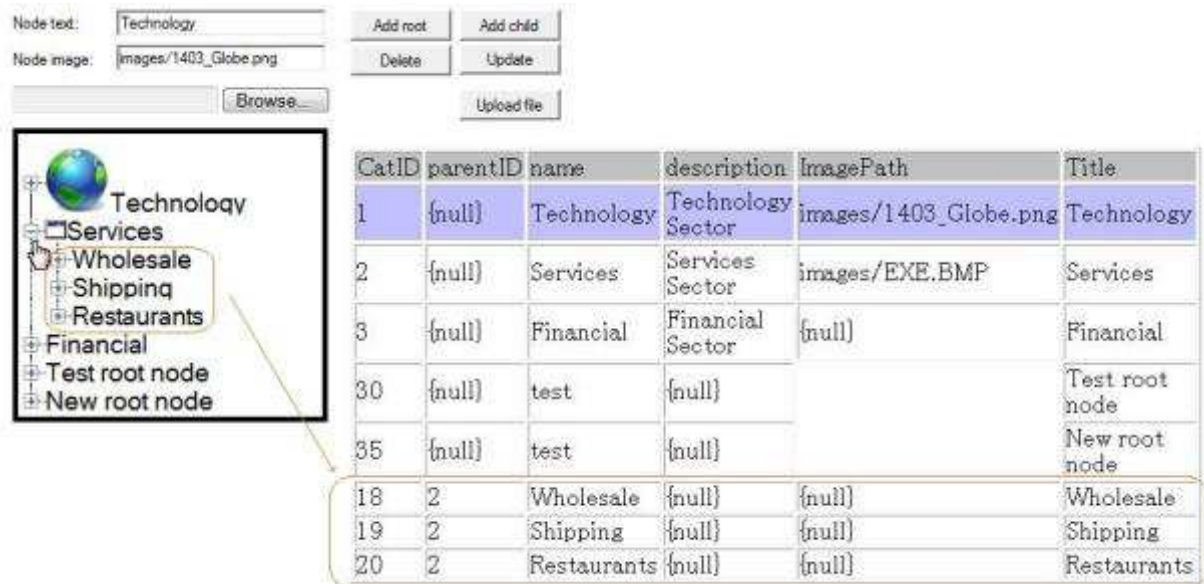


When the web page is loaded, you can see that the records for the root nodes are displayed in the HtmlTable:



CatID	parentID	name	description	ImagePath	Title
1	{null}	Technology	Technology Sector	images/1403_Globe.png	Technology
2	{null}	Services	Services Sector	images/EXE.BMP	Services
3	{null}	Financial	Financial Sector	{null}	Financial
30	{null}	test	{null}		Test root node
35	{null}	test	{null}		New root node

Expand a node; if the node has child nodes then you will see the records for the child nodes appear in the table:



CatID	parentID	name	description	ImagePath	Title
1	{null}	Technology	Technology Sector	images/1403_Globe.png	Technology
2	{null}	Services	Services Sector	images/EXE.BMP	Services
3	{null}	Financial	Financial Sector	{null}	Financial
30	{null}	test	{null}		Test root node
35	{null}	test	{null}		New root node
18	2	Wholesale	{null}	{null}	Wholesale
19	2	Shipping	{null}	{null}	Shipping
20	2	Restaurants	{null}	{null}	Restaurants

## Feedbacks

Please send your feedbacks to [support@limnor.com](mailto:support@limnor.com). Thanks!