GWT - TREE WIDGET

Introduction

The **Tree** widget represents a standard hierarchical tree widget. The tree contains a hierarchy of TreeItems that the user can open, close, and select.

Class declaration

Following is the declaration for **com.google.gwt.user.client.ui.Tree** class:

```
public class Tree
   extends Widget
      implements HasWidgets, SourcesTreeEvents, HasFocus,
         HasAnimation, HasAllKeyHandlers, HasAllFocusHandlers,
            HasSelectionHandlers, HasOpenHandlers,
               HasCloseHandlers, SourcesMouseEvents, HasAllMouseHandlers
```

CSS style rules

Following default CSS Style rules will be applied to all the Tree widget. You can override it as per your requirements.

```
.gwt-Tree {}
.gwt-TreeItem {}
.gwt-TreeItem-selected {}
```

Class constructors

S.N.	Constructor & Description
1	**Tree** Constructs an empty tree.
2	**Tree***TreeImagesimages* Constructs a tree that uses the specified image bundle for images.
3	**Tree***TreeImagesimages, booleanuseLeafImages* Constructs a tree that uses the specified image bundle for images.

Class methods

S.N.	Function name & Description
1	**void add***Widgetwidget* Adds the widget as a root tree item.

2

void addFocusListener*FocusListenerlistener*

Adds a listener interface to receive mouse events.

3

TreeItem addItem*java. lang. StringitemText*

Adds a simple tree item containing the specified text.

4

void addItem*TreeItemitem*

Adds an item to the root level of this tree.

5

TreeItem addItem*Widgetwidget*

Adds a new tree item containing the specified widget.

6

void addKeyboardListener*KeyboardListenerlistener*

Adds a listener interface to receive keyboard events.

7

void addMouseListener*MouseListenerlistener*

8

void addTreeListener*TreeListenerlistener*

Adds a listener interface to receive tree events.

9

void clear

Clears all tree items from the current tree.

10

protected void doAttachChildren

If a widget implements HasWidgets, it must override this method and call onAttach for each of its child widgets.

11

protected void doDetachChildren

If a widget implements HasWidgets, it must override this method and call onDetach for each of its child widgets.

12

void ensureSelectedItemVisible

Ensures that the currently-selected item is visible, opening its parents and scrolling the tree as necessary.

13

java.lang.String getImageBase

Deprecated. Use Tree*TreeImages* as it provides a more efficent and manageable way to

supply a set of images to be used within a tree.

14

TreeItem getItem*intindex*

Gets the top-level tree item at the specified index.

15

int getItemCount

Gets the number of items contained at the root of this tree.

16

TreeItem getSelectedItem

Gets the currently selected item.

17

int getTabIndex

Gets the widget's position in the tab index.

18

boolean isAnimationEnabled

19

protected boolean isKeyboardNavigationEnabled*TreeItemcurrentItem*

Indicates if keyboard navigation is enabled for the Tree and for a given TreeItem.

20

java.util.Iterator<Widget> iterator

Gets an iterator for the contained widgets.

21

void onBrowserEvent*Eventevent*

Fired whenever a browser event is received.

22

protected void onEnsureDebugId*java. lang. StringbaseID*

Affected Elements: -root = The root TreeItem.

23

protected void onLoad

This method is called immediately after a widget becomes attached to the browser's document.

24

boolean remove*Widgetw*

Removes a child widget.

25

void removeFocusListener*FocusListenerlistener*

Removes a previously added listener interface.

26

void removeItem*TreeItemitem*

Removes an item from the root level of this tree.

27

void removeItems

Removes all items from the root level of this tree.

28

void removeKeyboardListener*KeyboardListenerlistener*

Removes a previously added listener interface.

29

void removeTreeListener*TreeListenerlistener*

Removes a previously added listener interface.

30

void setAccessKey*charkey*

Sets the widget's 'access key'.

31

void setAnimationEnabled*booleanenable*

Enable or disable animations.

32

void setFocus*booleanfocus*

Explicitly focus/unfocus this widget.

33

void setImageBase*java. lang. StringbaseUrl*

Deprecated. Use Tree*TreeImages* as it provides a more efficent and manageable way to supply a set of images to be used within a tree.

34

void setSelectedItem*TreeItemitem*

Selects a specified item.

35

void setSelectedItem*TreeItemitem, booleanfireEvents*

Selects a specified item.

36

void setTabIndex*intindex*

Sets the widget's position in the tab index.

37

java.util.Iterator<TreeItem> treeItemIterator

Iterator of tree items.

Methods inherited

This class inherits methods from the following classes:

- com.google.gwt.user.client.ui.UIObject
- com.google.gwt.user.client.ui.Widget
- java.lang.Object

Tree Widget Example

This example will take you through simple steps to show usage of a Tree Widget in GWT. Follow the following steps to update the GWT application we created in *GWT - Create Application* chapter:

Step	Description
1	Create a project with a name *HelloWorld* under a package *com.tutorialspoint* as explained in the *GWT - Create Application* chapter.
2	Modify *HelloWorld.gwt.xml*, *HelloWorld.css*, *HelloWorld.html* and *HelloWorld.java* as explained below. Keep rest of the files unchanged.
3	Compile and run the application to verify the result of the implemented logic.

Following is the content of the modified module descriptor **src/com.tutorialspoint/HelloWorld.gwt.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='helloworld'>
  <!-- Inherit the core Web Toolkit stuff.                        -->
  <inherits name='com.google.gwt.user.User'/>

  <!-- Inherit the default GWT style sheet.                       -->
  <inherits name='com.google.gwt.user.theme.clean.Clean'/>

  <!-- Specify the app entry point class.                         -->
  <entry-point class='com.tutorialspoint.client.HelloWorld'/>

  <!-- Specify the paths for translatable code                    -->
  <source path='client'/>
  <source path='shared'/>

</module>
```

Following is the content of the modified Style Sheet file **war/HelloWorld.css**.

```
body{
   text-align: center;
   font-family: verdana, sans-serif;
}
h1{
   font-size: 2em;
   font-weight: bold;
   color: #777777;
   margin: 40px 0px 70px;
   text-align: center;
}
```

```
.gwt-Label {
   font-weight: bold;
   color: maroon;
}

.gwt-Tree .gwt-TreeItem {
  padding: 1px 0px;
  margin: 0px;
  white-space: nowrap;
  cursor: hand;
  cursor: pointer;
}

.gwt-Tree .gwt-TreeItem-selected {
  background: #ebeff9;
}
```

Following is the content of the modified HTML host file **war/HelloWorld.html**.

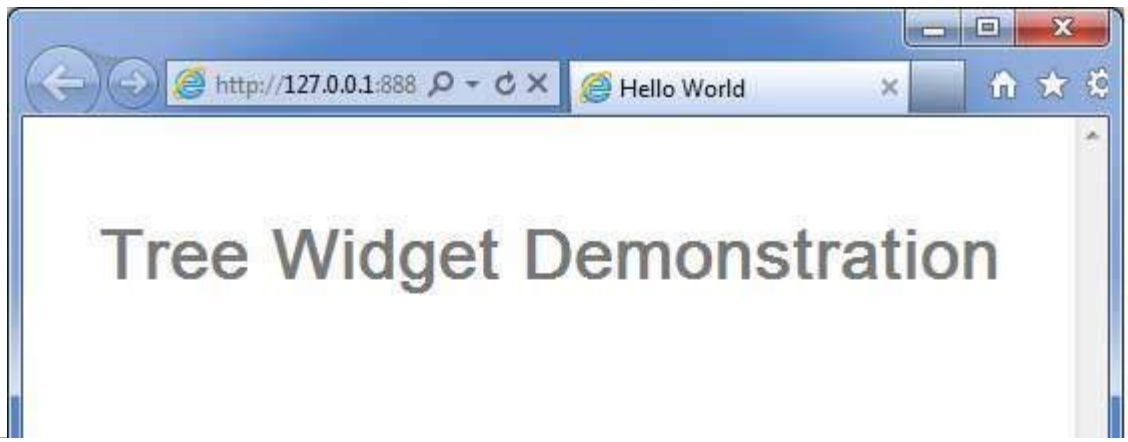
```
<html>
<head>
<title>Hello World</title>
   <link rel="stylesheet" href="HelloWorld.css"/>
   <script language="javascript" src="helloworld/helloworld.nocache.js">
   </script>
</head>
<body>

<h1>Tree Widget Demonstration</h1>
<div ></div>

</body>
</html>
```

Let us have following content of Java file **src/com.tutorialspoint/HelloWorld.java** which will demonstrate use of Tree widget.

```
package com.tutorialspoint.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.logical.shared.SelectionEvent;
import com.google.gwt.event.logical.shared.SelectionHandler;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.Tree;
import com.google.gwt.user.client.ui.TreeItem;
import com.google.gwt.user.client.ui.VerticalPanel;

public class HelloWorld implements EntryPoint {
   public void onModuleLoad() {
      //create a label
      final Label labelMessage = new Label();
      labelMessage.setWidth("300");

      // Create a root tree item as department
      TreeItem department = new TreeItem("Department");

      //create other tree items as department names
      TreeItem salesDepartment = new TreeItem("Sales");
      TreeItem marketingDepartment = new TreeItem("Marketing");
      TreeItem manufacturingDepartment = new TreeItem("Manufacturing");

      //create other tree items as employees
      TreeItem employee1 = new TreeItem("Robert");
      TreeItem employee2 = new TreeItem("Joe");
      TreeItem employee3 = new TreeItem("Chris");
```

```
      //add employees to sales department
      salesDepartment.addItem(employee1);
      salesDepartment.addItem(employee2);
      salesDepartment.addItem(employee3);

      //create other tree items as employees
      TreeItem employee4 = new TreeItem("Mona");
      TreeItem employee5 = new TreeItem("Tena");

      //add employees to marketing department
      marketingDepartment.addItem(employee4);
      marketingDepartment.addItem(employee5);

      //create other tree items as employees
      TreeItem employee6 = new TreeItem("Rener");
      TreeItem employee7 = new TreeItem("Linda");

      //add employees to sales department
      manufacturingDepartment.addItem(employee6);
      manufacturingDepartment.addItem(employee7);

      //add departments to department item
      department.addItem(salesDepartment);
      department.addItem(marketingDepartment);
      department.addItem(manufacturingDepartment);

      //create the tree
      Tree tree = new Tree();

      //add root item to the tree
      tree.addItem(department);

      tree.addSelectionHandler(new SelectionHandlerL<TreeItem>() {
         @Override
         public void onSelection(SelectionEvent<TreeItem> event) {
            labelMessage.setText("Selected Value: "
            + event.getSelectedItem().getText());
         }
      });

      // Add text boxes to the root panel.
      VerticalPanel panel = new VerticalPanel();
      panel.setSpacing(10);
      panel.add(tree);
      panel.add(labelMessage);

      //add the tree to the root panel
      RootPanel.get("gwtContainer").add(panel);
   }
}
```

Once you are ready with all the changes done, let us compile and run the application in development mode as we did in GWT - Create Application chapter. If everything is fine with your application, this will produce following result:



Selecting any value in tree, will update a message below the tree displaying the selected value.