| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DefaultTreeModel.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/tree/DefaultTreeCellRenderer.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/DefaultTreeSelectionModel.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/DefaultTreeModel.html)    [**NO FRAMES**](http://docs.google.com/DefaultTreeModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

## **javax.swing.tree**

Class DefaultTreeModel

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 **javax.swing.tree.DefaultTreeModel**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html)

public class **DefaultTreeModel**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Serializable](http://docs.google.com/java/io/Serializable.html), [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html)

A simple tree data model that uses TreeNodes. For further information and examples that use DefaultTreeModel, see [How to Use Trees](http://java.sun.com/docs/books/tutorial/uiswing/components/tree.html) in *The Java Tutorial.*

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

| **Field Summary** | |
| --- | --- |
| protected  boolean | [**asksAllowsChildren**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#asksAllowsChildren)            Determines how the isLeaf method figures out if a node is a leaf node. |
| protected  [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) | [**listenerList**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#listenerList)            Listeners. |
| protected  [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) | [**root**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#root)            Root of the tree. |

| **Constructor Summary** | |
| --- | --- |
| [**DefaultTreeModel**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#DefaultTreeModel(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) root)            Creates a tree in which any node can have children. |
| [**DefaultTreeModel**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#DefaultTreeModel(javax.swing.tree.TreeNode,%20boolean))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) root, boolean asksAllowsChildren)            Creates a tree specifying whether any node can have children, or whether only certain nodes can have children. |

| **Method Summary** | |
| --- | --- |
| void | [**addTreeModelListener**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#addTreeModelListener(javax.swing.event.TreeModelListener))([TreeModelListener](http://docs.google.com/javax/swing/event/TreeModelListener.html) l)            Adds a listener for the TreeModelEvent posted after the tree changes. |
| boolean | [**asksAllowsChildren**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#asksAllowsChildren())()            Tells how leaf nodes are determined. |
| protected  void | [**fireTreeNodesChanged**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#fireTreeNodesChanged(java.lang.Object,%20java.lang.Object%5B%5D,%20int%5B%5D,%20java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html) source, [Object](http://docs.google.com/java/lang/Object.html)[] path, int[] childIndices, [Object](http://docs.google.com/java/lang/Object.html)[] children)            Notifies all listeners that have registered interest for notification on this event type. |
| protected  void | [**fireTreeNodesInserted**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#fireTreeNodesInserted(java.lang.Object,%20java.lang.Object%5B%5D,%20int%5B%5D,%20java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html) source, [Object](http://docs.google.com/java/lang/Object.html)[] path, int[] childIndices, [Object](http://docs.google.com/java/lang/Object.html)[] children)            Notifies all listeners that have registered interest for notification on this event type. |
| protected  void | [**fireTreeNodesRemoved**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#fireTreeNodesRemoved(java.lang.Object,%20java.lang.Object%5B%5D,%20int%5B%5D,%20java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html) source, [Object](http://docs.google.com/java/lang/Object.html)[] path, int[] childIndices, [Object](http://docs.google.com/java/lang/Object.html)[] children)            Notifies all listeners that have registered interest for notification on this event type. |
| protected  void | [**fireTreeStructureChanged**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#fireTreeStructureChanged(java.lang.Object,%20java.lang.Object%5B%5D,%20int%5B%5D,%20java.lang.Object%5B%5D))([Object](http://docs.google.com/java/lang/Object.html) source, [Object](http://docs.google.com/java/lang/Object.html)[] path, int[] childIndices, [Object](http://docs.google.com/java/lang/Object.html)[] children)            Notifies all listeners that have registered interest for notification on this event type. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getChild**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getChild(java.lang.Object,%20int))([Object](http://docs.google.com/java/lang/Object.html) parent, int index)            Returns the child of *parent* at index *index* in the parent's child array. |
| int | [**getChildCount**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getChildCount(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) parent)            Returns the number of children of *parent*. |
| int | [**getIndexOfChild**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getIndexOfChild(java.lang.Object,%20java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) parent, [Object](http://docs.google.com/java/lang/Object.html) child)            Returns the index of child in parent. |
| | <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)>  T[] | | --- | | [**getListeners**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getListeners(java.lang.Class))([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)            Returns an array of all the objects currently registered as *Foo*Listeners upon this model. |
| [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] | [**getPathToRoot**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getPathToRoot(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode)            Builds the parents of node up to and including the root node, where the original node is the last element in the returned array. |
| protected  [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] | [**getPathToRoot**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getPathToRoot(javax.swing.tree.TreeNode,%20int))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode, int depth)            Builds the parents of node up to and including the root node, where the original node is the last element in the returned array. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**getRoot**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getRoot())()            Returns the root of the tree. |
| [TreeModelListener](http://docs.google.com/javax/swing/event/TreeModelListener.html)[] | [**getTreeModelListeners**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getTreeModelListeners())()            Returns an array of all the tree model listeners registered on this model. |
| void | [**insertNodeInto**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#insertNodeInto(javax.swing.tree.MutableTreeNode,%20javax.swing.tree.MutableTreeNode,%20int))([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newChild, [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) parent, int index)            Invoked this to insert newChild at location index in parents children. |
| boolean | [**isLeaf**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#isLeaf(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) node)            Returns whether the specified node is a leaf node. |
| void | [**nodeChanged**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#nodeChanged(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node)            Invoke this method after you've changed how node is to be represented in the tree. |
| void | [**nodesChanged**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#nodesChanged(javax.swing.tree.TreeNode,%20int%5B%5D))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node, int[] childIndices)            Invoke this method after you've changed how the children identified by childIndicies are to be represented in the tree. |
| void | [**nodeStructureChanged**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#nodeStructureChanged(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node)            Invoke this method if you've totally changed the children of node and its childrens children... |
| void | [**nodesWereInserted**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#nodesWereInserted(javax.swing.tree.TreeNode,%20int%5B%5D))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node, int[] childIndices)            Invoke this method after you've inserted some TreeNodes into node. |
| void | [**nodesWereRemoved**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#nodesWereRemoved(javax.swing.tree.TreeNode,%20int%5B%5D,%20java.lang.Object%5B%5D))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node, int[] childIndices, [Object](http://docs.google.com/java/lang/Object.html)[] removedChildren)            Invoke this method after you've removed some TreeNodes from node. |
| void | [**reload**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#reload())()            Invoke this method if you've modified the TreeNodes upon which this model depends. |
| void | [**reload**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#reload(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node)            Invoke this method if you've modified the TreeNodes upon which this model depends. |
| void | [**removeNodeFromParent**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#removeNodeFromParent(javax.swing.tree.MutableTreeNode))([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) node)            Message this to remove node from its parent. |
| void | [**removeTreeModelListener**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#removeTreeModelListener(javax.swing.event.TreeModelListener))([TreeModelListener](http://docs.google.com/javax/swing/event/TreeModelListener.html) l)            Removes a listener previously added with **addTreeModelListener()**. |
| void | [**setAsksAllowsChildren**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#setAsksAllowsChildren(boolean))(boolean newValue)            Sets whether or not to test leafness by asking getAllowsChildren() or isLeaf() to the TreeNodes. |
| void | [**setRoot**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#setRoot(javax.swing.tree.TreeNode))([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) root)            Sets the root to root. |
| void | [**valueForPathChanged**](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#valueForPathChanged(javax.swing.tree.TreePath,%20java.lang.Object))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path, [Object](http://docs.google.com/java/lang/Object.html) newValue)            This sets the user object of the TreeNode identified by path and posts a node changed. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### root

protected [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) **root**

Root of the tree.

### listenerList

protected [EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html) **listenerList**

Listeners.

### asksAllowsChildren

protected boolean **asksAllowsChildren**

Determines how the isLeaf method figures out if a node is a leaf node. If true, a node is a leaf node if it does not allow children. (If it allows children, it is not a leaf node, even if no children are present.) That lets you distinguish between *folder* nodes and *file* nodes in a file system, for example.

If this value is false, then any node which has no children is a leaf node, and any node may acquire children.

**See Also:**[TreeNode.getAllowsChildren()](http://docs.google.com/javax/swing/tree/TreeNode.html#getAllowsChildren()), [TreeModel.isLeaf(java.lang.Object)](http://docs.google.com/javax/swing/tree/TreeModel.html#isLeaf(java.lang.Object)), [setAsksAllowsChildren(boolean)](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#setAsksAllowsChildren(boolean))

| **Constructor Detail** |
| --- |

### DefaultTreeModel

public **DefaultTreeModel**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) root)

Creates a tree in which any node can have children.

**Parameters:**root - a TreeNode object that is the root of the tree**See Also:**[DefaultTreeModel(TreeNode, boolean)](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#DefaultTreeModel(javax.swing.tree.TreeNode,%20boolean))

### DefaultTreeModel

public **DefaultTreeModel**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) root,  
 boolean asksAllowsChildren)

Creates a tree specifying whether any node can have children, or whether only certain nodes can have children.

**Parameters:**root - a TreeNode object that is the root of the treeasksAllowsChildren - a boolean, false if any node can have children, true if each node is asked to see if it can have children**See Also:**[asksAllowsChildren](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#asksAllowsChildren)

| **Method Detail** |
| --- |

### setAsksAllowsChildren

public void **setAsksAllowsChildren**(boolean newValue)

Sets whether or not to test leafness by asking getAllowsChildren() or isLeaf() to the TreeNodes. If newvalue is true, getAllowsChildren() is messaged, otherwise isLeaf() is messaged.

### asksAllowsChildren

public boolean **asksAllowsChildren**()

Tells how leaf nodes are determined.

**Returns:**true if only nodes which do not allow children are leaf nodes, false if nodes which have no children (even if allowed) are leaf nodes**See Also:**[asksAllowsChildren](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#asksAllowsChildren)

### setRoot

public void **setRoot**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) root)

Sets the root to root. A null root implies the tree is to display nothing, and is legal.

### getRoot

public [Object](http://docs.google.com/java/lang/Object.html) **getRoot**()

Returns the root of the tree. Returns null only if the tree has no nodes.

**Specified by:**[getRoot](http://docs.google.com/javax/swing/tree/TreeModel.html#getRoot()) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Returns:**the root of the tree

### getIndexOfChild

public int **getIndexOfChild**([Object](http://docs.google.com/java/lang/Object.html) parent,  
 [Object](http://docs.google.com/java/lang/Object.html) child)

Returns the index of child in parent. If either the parent or child is null, returns -1.

**Specified by:**[getIndexOfChild](http://docs.google.com/javax/swing/tree/TreeModel.html#getIndexOfChild(java.lang.Object,%20java.lang.Object)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**parent - a note in the tree, obtained from this data sourcechild - the node we are interested in **Returns:**the index of the child in the parent, or -1 if either the parent or the child is null

### getChild

public [Object](http://docs.google.com/java/lang/Object.html) **getChild**([Object](http://docs.google.com/java/lang/Object.html) parent,  
 int index)

Returns the child of *parent* at index *index* in the parent's child array. *parent* must be a node previously obtained from this data source. This should not return null if *index* is a valid index for *parent* (that is *index* >= 0 && *index* < getChildCount(*parent*)).

**Specified by:**[getChild](http://docs.google.com/javax/swing/tree/TreeModel.html#getChild(java.lang.Object,%20int)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**parent - a node in the tree, obtained from this data source **Returns:**the child of *parent* at index *index*

### getChildCount

public int **getChildCount**([Object](http://docs.google.com/java/lang/Object.html) parent)

Returns the number of children of *parent*. Returns 0 if the node is a leaf or if it has no children. *parent* must be a node previously obtained from this data source.

**Specified by:**[getChildCount](http://docs.google.com/javax/swing/tree/TreeModel.html#getChildCount(java.lang.Object)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**parent - a node in the tree, obtained from this data source **Returns:**the number of children of the node *parent*

### isLeaf

public boolean **isLeaf**([Object](http://docs.google.com/java/lang/Object.html) node)

Returns whether the specified node is a leaf node. The way the test is performed depends on the askAllowsChildren setting.

**Specified by:**[isLeaf](http://docs.google.com/javax/swing/tree/TreeModel.html#isLeaf(java.lang.Object)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**node - the node to check **Returns:**true if the node is a leaf node**See Also:**[asksAllowsChildren](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#asksAllowsChildren), [TreeModel.isLeaf(java.lang.Object)](http://docs.google.com/javax/swing/tree/TreeModel.html#isLeaf(java.lang.Object))

### reload

public void **reload**()

Invoke this method if you've modified the TreeNodes upon which this model depends. The model will notify all of its listeners that the model has changed.

### valueForPathChanged

public void **valueForPathChanged**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path,  
 [Object](http://docs.google.com/java/lang/Object.html) newValue)

This sets the user object of the TreeNode identified by path and posts a node changed. If you use custom user objects in the TreeModel you're going to need to subclass this and set the user object of the changed node to something meaningful.

**Specified by:**[valueForPathChanged](http://docs.google.com/javax/swing/tree/TreeModel.html#valueForPathChanged(javax.swing.tree.TreePath,%20java.lang.Object)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**path - path to the node that the user has alterednewValue - the new value from the TreeCellEditor

### insertNodeInto

public void **insertNodeInto**([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) newChild,  
 [MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) parent,  
 int index)

Invoked this to insert newChild at location index in parents children. This will then message nodesWereInserted to create the appropriate event. This is the preferred way to add children as it will create the appropriate event.

### removeNodeFromParent

public void **removeNodeFromParent**([MutableTreeNode](http://docs.google.com/javax/swing/tree/MutableTreeNode.html) node)

Message this to remove node from its parent. This will message nodesWereRemoved to create the appropriate event. This is the preferred way to remove a node as it handles the event creation for you.

### nodeChanged

public void **nodeChanged**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node)

Invoke this method after you've changed how node is to be represented in the tree.

### reload

public void **reload**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node)

Invoke this method if you've modified the TreeNodes upon which this model depends. The model will notify all of its listeners that the model has changed below the given node.

**Parameters:**node - the node below which the model has changed

### nodesWereInserted

public void **nodesWereInserted**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node,  
 int[] childIndices)

Invoke this method after you've inserted some TreeNodes into node. childIndices should be the index of the new elements and must be sorted in ascending order.

### nodesWereRemoved

public void **nodesWereRemoved**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node,  
 int[] childIndices,  
 [Object](http://docs.google.com/java/lang/Object.html)[] removedChildren)

Invoke this method after you've removed some TreeNodes from node. childIndices should be the index of the removed elements and must be sorted in ascending order. And removedChildren should be the array of the children objects that were removed.

### nodesChanged

public void **nodesChanged**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node,  
 int[] childIndices)

Invoke this method after you've changed how the children identified by childIndicies are to be represented in the tree.

### nodeStructureChanged

public void **nodeStructureChanged**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) node)

Invoke this method if you've totally changed the children of node and its childrens children... This will post a treeStructureChanged event.

### getPathToRoot

public [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] **getPathToRoot**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode)

Builds the parents of node up to and including the root node, where the original node is the last element in the returned array. The length of the returned array gives the node's depth in the tree.

**Parameters:**aNode - the TreeNode to get the path for

### getPathToRoot

protected [TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html)[] **getPathToRoot**([TreeNode](http://docs.google.com/javax/swing/tree/TreeNode.html) aNode,  
 int depth)

Builds the parents of node up to and including the root node, where the original node is the last element in the returned array. The length of the returned array gives the node's depth in the tree.

**Parameters:**aNode - the TreeNode to get the path fordepth - an int giving the number of steps already taken towards the root (on recursive calls), used to size the returned array **Returns:**an array of TreeNodes giving the path from the root to the specified node

### addTreeModelListener

public void **addTreeModelListener**([TreeModelListener](http://docs.google.com/javax/swing/event/TreeModelListener.html) l)

Adds a listener for the TreeModelEvent posted after the tree changes.

**Specified by:**[addTreeModelListener](http://docs.google.com/javax/swing/tree/TreeModel.html#addTreeModelListener(javax.swing.event.TreeModelListener)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**l - the listener to add**See Also:**[removeTreeModelListener(javax.swing.event.TreeModelListener)](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#removeTreeModelListener(javax.swing.event.TreeModelListener))

### removeTreeModelListener

public void **removeTreeModelListener**([TreeModelListener](http://docs.google.com/javax/swing/event/TreeModelListener.html) l)

Removes a listener previously added with **addTreeModelListener()**.

**Specified by:**[removeTreeModelListener](http://docs.google.com/javax/swing/tree/TreeModel.html#removeTreeModelListener(javax.swing.event.TreeModelListener)) in interface [TreeModel](http://docs.google.com/javax/swing/tree/TreeModel.html) **Parameters:**l - the listener to remove**See Also:**[addTreeModelListener(javax.swing.event.TreeModelListener)](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#addTreeModelListener(javax.swing.event.TreeModelListener))

### getTreeModelListeners

public [TreeModelListener](http://docs.google.com/javax/swing/event/TreeModelListener.html)[] **getTreeModelListeners**()

Returns an array of all the tree model listeners registered on this model.

**Returns:**all of this model's TreeModelListeners or an empty array if no tree model listeners are currently registered**Since:** 1.4 **See Also:**[addTreeModelListener(javax.swing.event.TreeModelListener)](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#addTreeModelListener(javax.swing.event.TreeModelListener)), [removeTreeModelListener(javax.swing.event.TreeModelListener)](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#removeTreeModelListener(javax.swing.event.TreeModelListener))

### fireTreeNodesChanged

protected void **fireTreeNodesChanged**([Object](http://docs.google.com/java/lang/Object.html) source,  
 [Object](http://docs.google.com/java/lang/Object.html)[] path,  
 int[] childIndices,  
 [Object](http://docs.google.com/java/lang/Object.html)[] children)

Notifies all listeners that have registered interest for notification on this event type. The event instance is lazily created using the parameters passed into the fire method.

**Parameters:**source - the node being changedpath - the path to the root nodechildIndices - the indices of the changed elementschildren - the changed elements**See Also:**[EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### fireTreeNodesInserted

protected void **fireTreeNodesInserted**([Object](http://docs.google.com/java/lang/Object.html) source,  
 [Object](http://docs.google.com/java/lang/Object.html)[] path,  
 int[] childIndices,  
 [Object](http://docs.google.com/java/lang/Object.html)[] children)

Notifies all listeners that have registered interest for notification on this event type. The event instance is lazily created using the parameters passed into the fire method.

**Parameters:**source - the node where new elements are being insertedpath - the path to the root nodechildIndices - the indices of the new elementschildren - the new elements**See Also:**[EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### fireTreeNodesRemoved

protected void **fireTreeNodesRemoved**([Object](http://docs.google.com/java/lang/Object.html) source,  
 [Object](http://docs.google.com/java/lang/Object.html)[] path,  
 int[] childIndices,  
 [Object](http://docs.google.com/java/lang/Object.html)[] children)

Notifies all listeners that have registered interest for notification on this event type. The event instance is lazily created using the parameters passed into the fire method.

**Parameters:**source - the node where elements are being removedpath - the path to the root nodechildIndices - the indices of the removed elementschildren - the removed elements**See Also:**[EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### fireTreeStructureChanged

protected void **fireTreeStructureChanged**([Object](http://docs.google.com/java/lang/Object.html) source,  
 [Object](http://docs.google.com/java/lang/Object.html)[] path,  
 int[] childIndices,  
 [Object](http://docs.google.com/java/lang/Object.html)[] children)

Notifies all listeners that have registered interest for notification on this event type. The event instance is lazily created using the parameters passed into the fire method.

**Parameters:**source - the node where the tree model has changedpath - the path to the root nodechildIndices - the indices of the affected elementschildren - the affected elements**See Also:**[EventListenerList](http://docs.google.com/javax/swing/event/EventListenerList.html)

### getListeners

public <T extends [EventListener](http://docs.google.com/java/util/EventListener.html)> T[] **getListeners**([Class](http://docs.google.com/java/lang/Class.html)<T> listenerType)

Returns an array of all the objects currently registered as *Foo*Listeners upon this model. *Foo*Listeners are registered using the add*Foo*Listener method.

You can specify the listenerType argument with a class literal, such as *Foo*Listener.class. For example, you can query a DefaultTreeModel m for its tree model listeners with the following code:

TreeModelListener[] tmls = (TreeModelListener[])(m.getListeners(TreeModelListener.class));

If no such listeners exist, this method returns an empty array.

**Parameters:**listenerType - the type of listeners requested; this parameter should specify an interface that descends from java.util.EventListener **Returns:**an array of all objects registered as *Foo*Listeners on this component, or an empty array if no such listeners have been added **Throws:** [ClassCastException](http://docs.google.com/java/lang/ClassCastException.html) - if listenerType doesn't specify a class or interface that implements java.util.EventListener**Since:** 1.3 **See Also:**[getTreeModelListeners()](http://docs.google.com/javax/swing/tree/DefaultTreeModel.html#getTreeModelListeners())

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/DefaultTreeModel.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/tree/DefaultTreeCellRenderer.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/DefaultTreeSelectionModel.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/DefaultTreeModel.html)    [**NO FRAMES**](http://docs.google.com/DefaultTreeModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#3rdcrjn) | [METHOD](#35nkun2) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).