| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TreeSelectionModel.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/TreePath.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/VariableHeightLayoutCache.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/TreeSelectionModel.html)    [**NO FRAMES**](http://docs.google.com/TreeSelectionModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#2s8eyo1) |

## **javax.swing.tree**

Interface TreeSelectionModel

**All Known Implementing Classes:** [DefaultTreeSelectionModel](http://docs.google.com/javax/swing/tree/DefaultTreeSelectionModel.html), [JTree.EmptySelectionModel](http://docs.google.com/javax/swing/JTree.EmptySelectionModel.html)

public interface **TreeSelectionModel**

This interface represents the current state of the selection for the tree component. For information and examples of using tree selection models, see [How to Use Trees](http://java.sun.com/docs/books/tutorial/uiswing/components/tree.html) in *The Java Tutorial.*

The state of the tree selection is characterized by a set of TreePaths, and optionally a set of integers. The mapping from TreePath to integer is done by way of an instance of RowMapper. It is not necessary for a TreeSelectionModel to have a RowMapper to correctly operate, but without a RowMapper getSelectionRows will return null.

A TreeSelectionModel can be configured to allow only one path (SINGLE\_TREE\_SELECTION) a number of continguous paths (CONTIGUOUS\_TREE\_SELECTION) or a number of discontiguous paths (DISCONTIGUOUS\_TREE\_SELECTION). A RowMapper is used to determine if TreePaths are contiguous. In the absence of a RowMapper CONTIGUOUS\_TREE\_SELECTION and DISCONTIGUOUS\_TREE\_SELECTION behave the same, that is they allow any number of paths to be contained in the TreeSelectionModel.

For a selection model of CONTIGUOUS\_TREE\_SELECTION any time the paths are changed (setSelectionPath, addSelectionPath ...) the TreePaths are again checked to make they are contiguous. A check of the TreePaths can also be forced by invoking resetRowSelection. How a set of discontiguous TreePaths is mapped to a contiguous set is left to implementors of this interface to enforce a particular policy.

Implementations should combine duplicate TreePaths that are added to the selection. For example, the following code

TreePath[] paths = new TreePath[] { treePath, treePath };  
 treeSelectionModel.setSelectionPaths(paths);

should result in only one path being selected: treePath, and not two copies of treePath.

The lead TreePath is the last path that was added (or set). The lead row is then the row that corresponds to the TreePath as determined from the RowMapper.

| **Field Summary** | |
| --- | --- |
| static int | [**CONTIGUOUS\_TREE\_SELECTION**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#CONTIGUOUS_TREE_SELECTION)            Selection can only be contiguous. |
| static int | [**DISCONTIGUOUS\_TREE\_SELECTION**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#DISCONTIGUOUS_TREE_SELECTION)            Selection can contain any number of items that are not necessarily contiguous. |
| static int | [**SINGLE\_TREE\_SELECTION**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#SINGLE_TREE_SELECTION)            Selection can only contain one path at a time. |

| **Method Summary** | |
| --- | --- |
| void | [**addPropertyChangeListener**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#addPropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Adds a PropertyChangeListener to the listener list. |
| void | [**addSelectionPath**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#addSelectionPath(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Adds path to the current selection. |
| void | [**addSelectionPaths**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#addSelectionPaths(javax.swing.tree.TreePath%5B%5D))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)            Adds paths to the current selection. |
| void | [**addTreeSelectionListener**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#addTreeSelectionListener(javax.swing.event.TreeSelectionListener))([TreeSelectionListener](http://docs.google.com/javax/swing/event/TreeSelectionListener.html) x)            Adds x to the list of listeners that are notified each time the set of selected TreePaths changes. |
| void | [**clearSelection**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#clearSelection())()            Empties the current selection. |
| [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) | [**getLeadSelectionPath**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getLeadSelectionPath())()            Returns the last path that was added. |
| int | [**getLeadSelectionRow**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getLeadSelectionRow())()            Returns the lead selection index. |
| int | [**getMaxSelectionRow**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getMaxSelectionRow())()            Returns the largest value obtained from the RowMapper for the current set of selected TreePaths. |
| int | [**getMinSelectionRow**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getMinSelectionRow())()            Returns the smallest value obtained from the RowMapper for the current set of selected TreePaths. |
| [RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html) | [**getRowMapper**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getRowMapper())()            Returns the RowMapper instance that is able to map a TreePath to a row. |
| int | [**getSelectionCount**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getSelectionCount())()            Returns the number of paths that are selected. |
| int | [**getSelectionMode**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getSelectionMode())()            Returns the current selection mode, one of SINGLE\_TREE\_SELECTION, CONTIGUOUS\_TREE\_SELECTION or DISCONTIGUOUS\_TREE\_SELECTION. |
| [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) | [**getSelectionPath**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getSelectionPath())()            Returns the first path in the selection. |
| [TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] | [**getSelectionPaths**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getSelectionPaths())()            Returns the paths in the selection. |
| int[] | [**getSelectionRows**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#getSelectionRows())()            Returns all of the currently selected rows. |
| boolean | [**isPathSelected**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#isPathSelected(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Returns true if the path, path, is in the current selection. |
| boolean | [**isRowSelected**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#isRowSelected(int))(int row)            Returns true if the row identified by row is selected. |
| boolean | [**isSelectionEmpty**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#isSelectionEmpty())()            Returns true if the selection is currently empty. |
| void | [**removePropertyChangeListener**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#removePropertyChangeListener(java.beans.PropertyChangeListener))([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)            Removes a PropertyChangeListener from the listener list. |
| void | [**removeSelectionPath**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#removeSelectionPath(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Removes path from the selection. |
| void | [**removeSelectionPaths**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#removeSelectionPaths(javax.swing.tree.TreePath%5B%5D))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)            Removes paths from the selection. |
| void | [**removeTreeSelectionListener**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#removeTreeSelectionListener(javax.swing.event.TreeSelectionListener))([TreeSelectionListener](http://docs.google.com/javax/swing/event/TreeSelectionListener.html) x)            Removes x from the list of listeners that are notified each time the set of selected TreePaths changes. |
| void | [**resetRowSelection**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#resetRowSelection())()            Updates this object's mapping from TreePaths to rows. |
| void | [**setRowMapper**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#setRowMapper(javax.swing.tree.RowMapper))([RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html) newMapper)            Sets the RowMapper instance. |
| void | [**setSelectionMode**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#setSelectionMode(int))(int mode)            Sets the selection model, which must be one of SINGLE\_TREE\_SELECTION, CONTIGUOUS\_TREE\_SELECTION or DISCONTIGUOUS\_TREE\_SELECTION. |
| void | [**setSelectionPath**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#setSelectionPath(javax.swing.tree.TreePath))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)            Sets the selection to path. |
| void | [**setSelectionPaths**](http://docs.google.com/javax/swing/tree/TreeSelectionModel.html#setSelectionPaths(javax.swing.tree.TreePath%5B%5D))([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)            Sets the selection to path. |

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

### SINGLE\_TREE\_SELECTION

static final int **SINGLE\_TREE\_SELECTION**

Selection can only contain one path at a time.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.tree.TreeSelectionModel.SINGLE_TREE_SELECTION)

### CONTIGUOUS\_TREE\_SELECTION

static final int **CONTIGUOUS\_TREE\_SELECTION**

Selection can only be contiguous. This will only be enforced if a RowMapper instance is provided. That is, if no RowMapper is set this behaves the same as DISCONTIGUOUS\_TREE\_SELECTION.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.tree.TreeSelectionModel.CONTIGUOUS_TREE_SELECTION)

### DISCONTIGUOUS\_TREE\_SELECTION

static final int **DISCONTIGUOUS\_TREE\_SELECTION**

Selection can contain any number of items that are not necessarily contiguous.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.tree.TreeSelectionModel.DISCONTIGUOUS_TREE_SELECTION)

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

### setSelectionMode

void **setSelectionMode**(int mode)

Sets the selection model, which must be one of SINGLE\_TREE\_SELECTION, CONTIGUOUS\_TREE\_SELECTION or DISCONTIGUOUS\_TREE\_SELECTION.

This may change the selection if the current selection is not valid for the new mode. For example, if three TreePaths are selected when the mode is changed to SINGLE\_TREE\_SELECTION, only one TreePath will remain selected. It is up to the particular implementation to decide what TreePath remains selected.

### getSelectionMode

int **getSelectionMode**()

Returns the current selection mode, one of SINGLE\_TREE\_SELECTION, CONTIGUOUS\_TREE\_SELECTION or DISCONTIGUOUS\_TREE\_SELECTION.

### setSelectionPath

void **setSelectionPath**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Sets the selection to path. If this represents a change, then the TreeSelectionListeners are notified. If path is null, this has the same effect as invoking clearSelection.

**Parameters:**path - new path to select

### setSelectionPaths

void **setSelectionPaths**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)

Sets the selection to path. If this represents a change, then the TreeSelectionListeners are notified. If paths is null, this has the same effect as invoking clearSelection.

**Parameters:**paths - new selection

### addSelectionPath

void **addSelectionPath**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Adds path to the current selection. If path is not currently in the selection the TreeSelectionListeners are notified. This has no effect if path is null.

**Parameters:**path - the new path to add to the current selection

### addSelectionPaths

void **addSelectionPaths**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)

Adds paths to the current selection. If any of the paths in paths are not currently in the selection the TreeSelectionListeners are notified. This has no effect if paths is null.

**Parameters:**paths - the new paths to add to the current selection

### removeSelectionPath

void **removeSelectionPath**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Removes path from the selection. If path is in the selection The TreeSelectionListeners are notified. This has no effect if path is null.

**Parameters:**path - the path to remove from the selection

### removeSelectionPaths

void **removeSelectionPaths**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] paths)

Removes paths from the selection. If any of the paths in paths are in the selection, the TreeSelectionListeners are notified. This method has no effect if paths is null.

**Parameters:**paths - the path to remove from the selection

### getSelectionPath

[TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) **getSelectionPath**()

Returns the first path in the selection. How first is defined is up to implementors, and may not necessarily be the TreePath with the smallest integer value as determined from the RowMapper.

### getSelectionPaths

[TreePath](http://docs.google.com/javax/swing/tree/TreePath.html)[] **getSelectionPaths**()

Returns the paths in the selection. This will return null (or an empty array) if nothing is currently selected.

### getSelectionCount

int **getSelectionCount**()

Returns the number of paths that are selected.

### isPathSelected

boolean **isPathSelected**([TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) path)

Returns true if the path, path, is in the current selection.

### isSelectionEmpty

boolean **isSelectionEmpty**()

Returns true if the selection is currently empty.

### clearSelection

void **clearSelection**()

Empties the current selection. If this represents a change in the current selection, the selection listeners are notified.

### setRowMapper

void **setRowMapper**([RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html) newMapper)

Sets the RowMapper instance. This instance is used to determine the row for a particular TreePath.

### getRowMapper

[RowMapper](http://docs.google.com/javax/swing/tree/RowMapper.html) **getRowMapper**()

Returns the RowMapper instance that is able to map a TreePath to a row.

### getSelectionRows

int[] **getSelectionRows**()

Returns all of the currently selected rows. This will return null (or an empty array) if there are no selected TreePaths or a RowMapper has not been set.

### getMinSelectionRow

int **getMinSelectionRow**()

Returns the smallest value obtained from the RowMapper for the current set of selected TreePaths. If nothing is selected, or there is no RowMapper, this will return -1.

### getMaxSelectionRow

int **getMaxSelectionRow**()

Returns the largest value obtained from the RowMapper for the current set of selected TreePaths. If nothing is selected, or there is no RowMapper, this will return -1.

### isRowSelected

boolean **isRowSelected**(int row)

Returns true if the row identified by row is selected.

### resetRowSelection

void **resetRowSelection**()

Updates this object's mapping from TreePaths to rows. This should be invoked when the mapping from TreePaths to integers has changed (for example, a node has been expanded).

You do not normally have to call this; JTree and its associated listeners will invoke this for you. If you are implementing your own view class, then you will have to invoke this.

### getLeadSelectionRow

int **getLeadSelectionRow**()

Returns the lead selection index. That is the last index that was added.

### getLeadSelectionPath

[TreePath](http://docs.google.com/javax/swing/tree/TreePath.html) **getLeadSelectionPath**()

Returns the last path that was added. This may differ from the leadSelectionPath property maintained by the JTree.

### addPropertyChangeListener

void **addPropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)

Adds a PropertyChangeListener to the listener list. The listener is registered for all properties.

A PropertyChangeEvent will get fired when the selection mode changes.

**Parameters:**listener - the PropertyChangeListener to be added

### removePropertyChangeListener

void **removePropertyChangeListener**([PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) listener)

Removes a PropertyChangeListener from the listener list. This removes a PropertyChangeListener that was registered for all properties.

**Parameters:**listener - the PropertyChangeListener to be removed

### addTreeSelectionListener

void **addTreeSelectionListener**([TreeSelectionListener](http://docs.google.com/javax/swing/event/TreeSelectionListener.html) x)

Adds x to the list of listeners that are notified each time the set of selected TreePaths changes.

**Parameters:**x - the new listener to be added

### removeTreeSelectionListener

void **removeTreeSelectionListener**([TreeSelectionListener](http://docs.google.com/javax/swing/event/TreeSelectionListener.html) x)

Removes x from the list of listeners that are notified each time the set of selected TreePaths changes.

**Parameters:**x - the listener to remove

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TreeSelectionModel.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/TreePath.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/tree/VariableHeightLayoutCache.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/tree/TreeSelectionModel.html)    [**NO FRAMES**](http://docs.google.com/TreeSelectionModel.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | CONSTR | [METHOD](#2et92p0) | DETAIL: [FIELD](#tyjcwt) | CONSTR | [METHOD](#2s8eyo1) |

[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).