| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TableRowSorter.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/table/TableModel.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/table/TableStringConverter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/table/TableRowSorter.html)    [**NO FRAMES**](http://docs.google.com/TableRowSorter.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#1ksv4uv) |

## **javax.swing.table**

Class TableRowSorter<M extends [**TableModel**](http://docs.google.com/javax/swing/table/TableModel.html)>

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.swing.RowSorter](http://docs.google.com/javax/swing/RowSorter.html)<M>  
 [javax.swing.DefaultRowSorter](http://docs.google.com/javax/swing/DefaultRowSorter.html)<M,[Integer](http://docs.google.com/java/lang/Integer.html)>  
 **javax.swing.table.TableRowSorter<M>**

**Type Parameters:**M - the type of the model, which must be an implementation of TableModel

public class **TableRowSorter<M extends** [**TableModel**](http://docs.google.com/javax/swing/table/TableModel.html)**>**extends [DefaultRowSorter](http://docs.google.com/javax/swing/DefaultRowSorter.html)<M,[Integer](http://docs.google.com/java/lang/Integer.html)>

An implementation of RowSorter that provides sorting and filtering using a TableModel. The following example shows adding sorting to a JTable:

TableModel myModel = createMyTableModel();  
 JTable table = new JTable(myModel);  
 table.setRowSorter(new TableRowSorter(myModel));

This will do all the wiring such that when the user does the appropriate gesture, such as clicking on the column header, the table will visually sort.

JTable's row-based methods and JTable's selection model refer to the view and not the underlying model. Therefore, it is necessary to convert between the two. For example, to get the selection in terms of myModel you need to convert the indices:

int[] selection = table.getSelectedRows();  
 for (int i = 0; i < selection.length; i++) {  
 selection[i] = table.convertRowIndexToModel(selection[i]);  
 }

Similarly to select a row in JTable based on a coordinate from the underlying model do the inverse:

table.setRowSelectionInterval(table.convertRowIndexToView(row),  
 table.convertRowIndexToView(row));

The previous example assumes you have not enabled filtering. If you have enabled filtering convertRowIndexToView will return -1 for locations that are not visible in the view.

TableRowSorter uses Comparators for doing comparisons. The following defines how a Comparator is chosen for a column:

1. If a Comparator has been specified for the column by the setComparator method, use it.
2. If the column class as returned by getColumnClass is String, use the Comparator returned by Collator.getInstance().
3. If the column class implements Comparable, use a Comparator that invokes the compareTo method.
4. If a TableStringConverter has been specified, use it to convert the values to Strings and then use the Comparator returned by Collator.getInstance().
5. Otherwise use the Comparator returned by Collator.getInstance() on the results from calling toString on the objects.

In addition to sorting TableRowSorter provides the ability to filter. A filter is specified using the setFilter method. The following example will only show rows containing the string "foo":

TableModel myModel = createMyTableModel();  
 TableRowSorter sorter = new TableRowSorter(myModel);  
 sorter.setRowFilter(RowFilter.regexFilter(".\*foo.\*"));  
 JTable table = new JTable(myModel);  
 table.setRowSorter(sorter);

If the underlying model structure changes (the modelStructureChanged method is invoked) the following are reset to their default values: Comparators by column, current sort order, and whether each column is sortable. The default sort order is natural (the same as the model), and columns are sortable by default.

TableRowSorter has one formal type parameter: the type of the model. Passing in a type that corresponds exactly to your model allows you to filter based on your model without casting. Refer to the documentation of RowFilter for an example of this.

**WARNING:** DefaultTableModel returns a column class of Object. As such all comparisons will be done using toString. This may be unnecessarily expensive. If the column only contains one type of value, such as an Integer, you should override getColumnClass and return the appropriate Class. This will dramatically increase the performance of this class.

**Since:** 1.6 **See Also:**[JTable](http://docs.google.com/javax/swing/JTable.html), [RowFilter](http://docs.google.com/javax/swing/RowFilter.html), [DefaultTableModel](http://docs.google.com/javax/swing/table/DefaultTableModel.html), [Collator](http://docs.google.com/java/text/Collator.html), [Comparator](http://docs.google.com/java/util/Comparator.html)

| **Nested Class Summary** | |
| --- | --- |

| **Nested classes/interfaces inherited from class javax.swing.**[**DefaultRowSorter**](http://docs.google.com/javax/swing/DefaultRowSorter.html) |
| --- |
| [DefaultRowSorter.ModelWrapper](http://docs.google.com/javax/swing/DefaultRowSorter.ModelWrapper.html)<[M](http://docs.google.com/javax/swing/DefaultRowSorter.ModelWrapper.html),[I](http://docs.google.com/javax/swing/DefaultRowSorter.ModelWrapper.html)> |

| **Nested classes/interfaces inherited from class javax.swing.**[**RowSorter**](http://docs.google.com/javax/swing/RowSorter.html) |
| --- |
| [RowSorter.SortKey](http://docs.google.com/javax/swing/RowSorter.SortKey.html) |

| **Constructor Summary** | |
| --- | --- |
| [**TableRowSorter**](http://docs.google.com/javax/swing/table/TableRowSorter.html#TableRowSorter())()            Creates a TableRowSorter with an empty model. |
| [**TableRowSorter**](http://docs.google.com/javax/swing/table/TableRowSorter.html#TableRowSorter(M))([M](http://docs.google.com/javax/swing/table/TableRowSorter.html) model)            Creates a TableRowSorter using model as the underlying TableModel. |

| **Method Summary** | |
| --- | --- |
| [Comparator](http://docs.google.com/java/util/Comparator.html)<?> | [**getComparator**](http://docs.google.com/javax/swing/table/TableRowSorter.html#getComparator(int))(int column)            Returns the Comparator for the specified column. |
| [TableStringConverter](http://docs.google.com/javax/swing/table/TableStringConverter.html) | [**getStringConverter**](http://docs.google.com/javax/swing/table/TableRowSorter.html#getStringConverter())()            Returns the object responsible for converting values from the model to strings. |
| void | [**setModel**](http://docs.google.com/javax/swing/table/TableRowSorter.html#setModel(M))([M](http://docs.google.com/javax/swing/table/TableRowSorter.html) model)            Sets the TableModel to use as the underlying model for this TableRowSorter. |
| void | [**setStringConverter**](http://docs.google.com/javax/swing/table/TableRowSorter.html#setStringConverter(javax.swing.table.TableStringConverter))([TableStringConverter](http://docs.google.com/javax/swing/table/TableStringConverter.html) stringConverter)            Sets the object responsible for converting values from the model to strings. |
| protected  boolean | [**useToString**](http://docs.google.com/javax/swing/table/TableRowSorter.html#useToString(int))(int column)            Returns whether or not to convert the value to a string before doing comparisons when sorting. |

| **Methods inherited from class javax.swing.**[**DefaultRowSorter**](http://docs.google.com/javax/swing/DefaultRowSorter.html) |
| --- |
| [allRowsChanged](http://docs.google.com/javax/swing/DefaultRowSorter.html#allRowsChanged()), [convertRowIndexToModel](http://docs.google.com/javax/swing/DefaultRowSorter.html#convertRowIndexToModel(int)), [convertRowIndexToView](http://docs.google.com/javax/swing/DefaultRowSorter.html#convertRowIndexToView(int)), [getMaxSortKeys](http://docs.google.com/javax/swing/DefaultRowSorter.html#getMaxSortKeys()), [getModel](http://docs.google.com/javax/swing/DefaultRowSorter.html#getModel()), [getModelRowCount](http://docs.google.com/javax/swing/DefaultRowSorter.html#getModelRowCount()), [getModelWrapper](http://docs.google.com/javax/swing/DefaultRowSorter.html#getModelWrapper()), [getRowFilter](http://docs.google.com/javax/swing/DefaultRowSorter.html#getRowFilter()), [getSortKeys](http://docs.google.com/javax/swing/DefaultRowSorter.html#getSortKeys()), [getSortsOnUpdates](http://docs.google.com/javax/swing/DefaultRowSorter.html#getSortsOnUpdates()), [getViewRowCount](http://docs.google.com/javax/swing/DefaultRowSorter.html#getViewRowCount()), [isSortable](http://docs.google.com/javax/swing/DefaultRowSorter.html#isSortable(int)), [modelStructureChanged](http://docs.google.com/javax/swing/DefaultRowSorter.html#modelStructureChanged()), [rowsDeleted](http://docs.google.com/javax/swing/DefaultRowSorter.html#rowsDeleted(int,%20int)), [rowsInserted](http://docs.google.com/javax/swing/DefaultRowSorter.html#rowsInserted(int,%20int)), [rowsUpdated](http://docs.google.com/javax/swing/DefaultRowSorter.html#rowsUpdated(int,%20int)), [rowsUpdated](http://docs.google.com/javax/swing/DefaultRowSorter.html#rowsUpdated(int,%20int,%20int)), [setComparator](http://docs.google.com/javax/swing/DefaultRowSorter.html#setComparator(int,%20java.util.Comparator)), [setMaxSortKeys](http://docs.google.com/javax/swing/DefaultRowSorter.html#setMaxSortKeys(int)), [setModelWrapper](http://docs.google.com/javax/swing/DefaultRowSorter.html#setModelWrapper(javax.swing.DefaultRowSorter.ModelWrapper)), [setRowFilter](http://docs.google.com/javax/swing/DefaultRowSorter.html#setRowFilter(javax.swing.RowFilter)), [setSortable](http://docs.google.com/javax/swing/DefaultRowSorter.html#setSortable(int,%20boolean)), [setSortKeys](http://docs.google.com/javax/swing/DefaultRowSorter.html#setSortKeys(java.util.List)), [setSortsOnUpdates](http://docs.google.com/javax/swing/DefaultRowSorter.html#setSortsOnUpdates(boolean)), [sort](http://docs.google.com/javax/swing/DefaultRowSorter.html#sort()), [toggleSortOrder](http://docs.google.com/javax/swing/DefaultRowSorter.html#toggleSortOrder(int)) |

| **Methods inherited from class javax.swing.**[**RowSorter**](http://docs.google.com/javax/swing/RowSorter.html) |
| --- |
| [addRowSorterListener](http://docs.google.com/javax/swing/RowSorter.html#addRowSorterListener(javax.swing.event.RowSorterListener)), [fireRowSorterChanged](http://docs.google.com/javax/swing/RowSorter.html#fireRowSorterChanged(int%5B%5D)), [fireSortOrderChanged](http://docs.google.com/javax/swing/RowSorter.html#fireSortOrderChanged()), [removeRowSorterListener](http://docs.google.com/javax/swing/RowSorter.html#removeRowSorterListener(javax.swing.event.RowSorterListener)) |

| **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)) |

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

### TableRowSorter

public **TableRowSorter**()

Creates a TableRowSorter with an empty model.

### TableRowSorter

public **TableRowSorter**([M](http://docs.google.com/javax/swing/table/TableRowSorter.html) model)

Creates a TableRowSorter using model as the underlying TableModel.

**Parameters:**model - the underlying TableModel to use, null is treated as an empty model

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

### setModel

public void **setModel**([M](http://docs.google.com/javax/swing/table/TableRowSorter.html) model)

Sets the TableModel to use as the underlying model for this TableRowSorter. A value of null can be used to set an empty model.

**Parameters:**model - the underlying model to use, or null

### setStringConverter

public void **setStringConverter**([TableStringConverter](http://docs.google.com/javax/swing/table/TableStringConverter.html) stringConverter)

Sets the object responsible for converting values from the model to strings. If non-null this is used to convert any object values, that do not have a registered Comparator, to strings.

**Parameters:**stringConverter - the object responsible for converting values from the model to strings

### getStringConverter

public [TableStringConverter](http://docs.google.com/javax/swing/table/TableStringConverter.html) **getStringConverter**()

Returns the object responsible for converting values from the model to strings.

**Returns:**object responsible for converting values to strings.

### getComparator

public [Comparator](http://docs.google.com/java/util/Comparator.html)<?> **getComparator**(int column)

Returns the Comparator for the specified column. If a Comparator has not been specified using the setComparator method a Comparator will be returned based on the column class (TableModel.getColumnClass) of the specified column. If the column class is String, Collator.getInstance is returned. If the column class implements Comparable a private Comparator is returned that invokes the compareTo method. Otherwise Collator.getInstance is returned.

**Overrides:**[getComparator](http://docs.google.com/javax/swing/DefaultRowSorter.html#getComparator(int)) in class [DefaultRowSorter](http://docs.google.com/javax/swing/DefaultRowSorter.html)<[M](http://docs.google.com/javax/swing/table/TableRowSorter.html) extends [TableModel](http://docs.google.com/javax/swing/table/TableModel.html),[Integer](http://docs.google.com/java/lang/Integer.html)> **Parameters:**column - the column to fetch the Comparator for, in terms of the underlying model **Returns:**the Comparator for the specified column **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if column is outside the range of the underlying model

### useToString

protected boolean **useToString**(int column)

Returns whether or not to convert the value to a string before doing comparisons when sorting. If true ModelWrapper.getStringValueAt will be used, otherwise ModelWrapper.getValueAt will be used. It is up to subclasses, such as TableRowSorter, to honor this value in their ModelWrapper implementation.

**Overrides:**[useToString](http://docs.google.com/javax/swing/DefaultRowSorter.html#useToString(int)) in class [DefaultRowSorter](http://docs.google.com/javax/swing/DefaultRowSorter.html)<[M](http://docs.google.com/javax/swing/table/TableRowSorter.html) extends [TableModel](http://docs.google.com/javax/swing/table/TableModel.html),[Integer](http://docs.google.com/java/lang/Integer.html)> **Parameters:**column - the index of the column to test, in terms of the underlying model **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if column is not valid

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TableRowSorter.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/table/TableModel.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/table/TableStringConverter.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/table/TableRowSorter.html)    [**NO FRAMES**](http://docs.google.com/TableRowSorter.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#2et92p0) | FIELD | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#3rdcrjn) | [METHOD](#1ksv4uv) |

[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).