



Soft Aspects

RIAs & Beyond



New light of Rich Internet Applications

Developing Drag-N-Drop enabled Shopping Cart

Table of Contents

1. Introduction.....	3
2. Overview.....	3
3. Step by step implementation.....	4
4. Full sample source code.....	10

1. Introduction

In this tutorial we'll guide you through step-by-step instructions of creating shopping cart application with drag-n-drop functionality.

There will be the main tree in the left hand area with available goods , table with selected items and Recycled Bin area to remove the ones. The items will be moving from the tree to the table and to the Bin using drag-n-drop operations. In addition , different Strategies will be developed allowing to customize drag-n-drop behavior.

2. Overview

WebGalileo Faces drag-n-drop enabled components allow to use standard drag-n-drop operations when moving items between two or more components or inside the ones.

Drag and drop components have two boolean attributes to enable drag and drop functionality:

- **dragEnabled** - allows to drag the component or it's part to other component(s)
- **dropEnabled** - allows to drop other drag and drop components to the given one

These attributes can be used in any combinations. By default both attributes' values are set to false.

Drag and drop components have client and server side event listeners:

- **userFunctionOnDrop** - client side JavaScript function to be used onces drop event occurs
- **onDropListener** - server side listener, must implement **com.softaspects.faces.galileo.support.draganddrop.event.OnDropEventListener** interface

To setup drag and drop operation it's required to set appropriate drag and drop Strategy.

When user makes drag and drop operation client side JavaScript function is called. Then **AJAX** request is sent to the component's server side. Target component Strategy is called to perform exact drag and drop operation. Then the server side **onDropListener** is called for the components involved in the drag and drop operations. Optionally some components may be refreshed by **AJAX automatically**.

It is possible to use **default or custom drag and drop Strategy** by setting component "**strategy**" property.

For custom Strategies the corresponding property should refer to full class name or JSF value binding. In the case of default Strategies it is possible to use the following pre-defined short names:

- **copy** - copy data from the source component to the target one
- **move** - move data from the source component to the target one
- **remove** - remove data from the source component
- **create** - create data at the target component based on the source component data
- **copySelfMove** - combination of copy and move operations - move inside component, otherwise perform copy operation

To create custom drag and drop Strategy **com.softaspects.faces.galileo.support.draganddrop.strategy.DnDStrategy** interface should be implemented. This interface contains the following method:

```
public DnDOperationResult doDragOperation(DnDMetaData from, DnDMetaData to);
```

The method should perform drag and drop operation for source and target components. Parameters are Drag and Drop components meta data which can contain additional data for particular components (like tree path, row, cell, etc.). It returns the resulting **DnDOperationResult** object indicating what exact components should be refreshed (one, both or none).

3. Step by step implementation

First of all , we'll be using standard strategy to implement the functionality of the application. We'll create one managed bean for all data models and JSP page, but before this we'll modify **faces-config.xml** file to add the managed bean:

```
<managed-bean>
  <managed-bean-name>CustomerBacKetBean</managed-bean-name>
  <managed-bean-class>samples.dragndrop.CustomerBacKet</managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

Now we'll create simple models for this page in **samples\dragndrop\CustomerBasket.java** file:

```

/**
 * Prefix for images path
 */
public static final String baseImagesPath =
    "{ApplicationRootContext}/images/tree/";

/**
 * Tree data model
 */
private BaseTreeDataModelImpl treeDataModel = null;

/**
 * Tree selection model
 */
private BaseTreeSelectionModelImpl treeSelectionModel = null;

/**
 * Table data model
 */
private DataModel tableDataModel = null;

/**
 * Table column model
 */
private ColumnModel tableColumnModel = null;

/**
 * Tree data model getter
 *
 * @return tree data model
 */
public BaseTreeDataModelImpl getTreeDataModel() {
    if (treeDataModel == null) {
        treeDataModel = new BaseTreeDataModelImpl();
        TreeItem root = TreeRendererUtil.createFolder(null,
            "Root", baseImagesPath + "user1.gif");
        treeDataModel.addElement(root);
        TreeItem video = TreeRendererUtil.createFolder(root,
            "Video", baseImagesPath + "monitor.gif");
        TreeRendererUtil.createDoc(video, "Monitor",
            baseImagesPath + "monitor2.gif");
        TreeRendererUtil.createDoc(video, "Plazma",
            baseImagesPath + "plasma-tv.gif");
        TreeRendererUtil.createDoc(video, "TV",
            baseImagesPath + "tv.gif");

        TreeRendererUtil.createFolder(root, "Empty Video Folder",
            baseImagesPath + "monitor.gif");

        TreeItem storages = TreeRendererUtil.createFolder(root,
            "Storages", baseImagesPath + "package.gif");
        TreeItem storagess = TreeRendererUtil.createFolder(storages,
            "Storages", baseImagesPath + "package.gif");
    }
}

```

```

TreeRendererUtil.createDoc(storagess, "CD",
    baseImagesPath + "cd.gif");
TreeRendererUtil.createDoc(storagess, "CDR",
    baseImagesPath + "cd_gold.gif");
TreeItem other = TreeRendererUtil.createFolder(storagess, "Other",
    baseImagesPath + "cube_yellow.gif");
TreeRendererUtil.createDoc(other, "Tape",
    baseImagesPath + "tape.gif");
TreeRendererUtil.createDoc(other, "Record",
    baseImagesPath + "record.gif");

TreeItem videophoto = TreeRendererUtil.createFolder(root,
    "Video and Photography", baseImagesPath + "videocamera.gif");
TreeRendererUtil.createDoc(videophoto, "Camera",
    baseImagesPath + "camera.gif");
TreeRendererUtil.createDoc(videophoto, "Super Camera",
    baseImagesPath + "camera2.gif");

TreeItem sound = TreeRendererUtil.createFolder(root, "Sound",
    baseImagesPath + "loudspeaker.gif");
TreeRendererUtil.createDoc(sound, "Speaker",
    baseImagesPath + "loudspeaker2.gif");
TreeRendererUtil.createDoc(sound, "Megaphone",
    baseImagesPath + "megaphone.gif");
TreeRendererUtil.createDoc(sound, "Headphones",
    baseImagesPath + "headphones.gif");
TreeRendererUtil.createDoc(sound, "Radio",
    baseImagesPath + "radio.gif");
}
return treeDataModel;
}

/**
 * Tree selection model getter
 *
 * @return tree selection model
 */
public BaseTreeSelectionModelImpl getTreeSelectionModel() {
    if (treeSelectionModel == null)
        treeSelectionModel = new BaseTreeSelectionModelImpl();
    return treeSelectionModel;
}

/**
 * Table data model getter
 *
 * @return table data model
 */
public DataModel getTableDataModel() {
    if (tableDataModel == null) {
        tableDataModel = new DataModelImpl();
        List vTable = new ArrayList();
        tableDataModel.setData(vTable,
            getTableColumnModel().getColumnCount());
    }
    return tableDataModel;
}

```

```

/**
 * Table column model getter
 *
 * @return table column model
 */
public ColumnModel getTableColumnModel() {
    if (tableColumnModel == null) {
        tableColumnModel = new ColumnModelImpl();
        int column = 0;
        Column rowColumn = new ColumnImpl(TableDataModelUtil.ROW_COLUMN,
            "Row #", column++);
        rowColumn.setPreferredWidth("100px");
        rowColumn.setColumnClass(TableDataModelUtil.INT);
        rowColumn.setAlignMode(ComponentDefinitions.ALIGN_CENTER);
        Column titleColumn = new ColumnImpl("description",
            "Description", column++);
        titleColumn.setPreferredWidth("200px");
        titleColumn.setAlignMode(ComponentDefinitions.ALIGN_LEFT);
        Column priceColumn = new ColumnImpl("price", "Price", column++);
        priceColumn.setPreferredWidth("100px");
        priceColumn.setColumnClass(TableDataModelUtil.CURRENCY);
        Column countColumn = new ColumnImpl("count", "Count", column);
        countColumn.setPreferredWidth("100px");
        countColumn.setColumnClass(TableDataModelUtil.INT);
        tableColumnModel.addColumn(rowColumn);
        tableColumnModel.addColumn(titleColumn);
        tableColumnModel.addColumn(priceColumn);
        tableColumnModel.addColumn(countColumn);
    }
    return tableColumnModel;
}

```

Next step is to create **customerBasket.jsp** file:

```

<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
<%@ taglib uri="http://www.softaspects.com/wgf/tree" prefix="tree" %>
<%@ taglib uri="http://www.softaspects.com/wgf/table" prefix="table" %>
<%@ taglib uri="http://www.softaspects.com/wgf/dragArea"
    prefix="dragArea" %>
<f:view>
    <html>
    <head>
        <title>Customer basket sample</title>
    <body>
    <h:form id="customerBasketSampleForm">
        <h:panelGrid columns="2">
            <tree:tree id="goodsTree"
                varName="goodsTree" dragEnabled="true">
                <tree:dataModel
                    beanName="#{CustomerBasketBean.treeDataModel}"
                    scope="session"/>
                <tree:selectionModel
                    beanName="#{CustomerBasketBean.treeSelectionModel}"
                    scope="session"/>
            </tree:tree>
            <table:table id="basketTable"

```

```

        pageSize="10" styleClass="tableStyleClass"
        rowsPerPageValues="5,10,20,50"
        ajaxRefreshEnabled="true"
        dragEnabled="true" dropEnabled="true"
        dnDStrategy="copy"
        cellSpacing="0" cellPadding="0" border="0">
<table:dataModel
    beanName="{CustomerBasketBean.tableDataModel}"
    scope="session"/>
<table:columnModel
    beanName="{CustomerBasketBean.tableColumnModel}"
    scope="session"/>
<table:interfaceManager scope="session"/>
<table:header scope="session"/>
</table:table>
<dragArea:dragArea id="garbageArea" dropEnabled="true"
    dnDStrategy="remove">
    <h:outputText
        value="To remove any item from the table drag it to this
box"/>
    </dragArea:dragArea>
</h:panelGrid>
</h:form>
</body>
</html>
</f:view>

```

In the JSP page above we set drag operation for the tree to allow to drag items from the tree to the table. Then we setup the table component to allow to add and remove items from the table. Also we set table's **dnDStrategy** property to **copySelfMove** strategy allowing to copy data to table by drag and drop and change rows' order inside the table. For Recycle Bin area (dragArea component) we'll enable drop operation by **dropEnable** property and setup **dnDStrategy** to remove. This will allow to remove data from the table. Now we can start this sample and try to use it:

Row #	Description	Price	Count
0	Monitor	\$0.00	0
0	Tape	\$0.00	0
0	Radio	\$0.00	0
0	CD	\$0.00	0
0	Camera	\$0.00	0
0	Plazma	\$0.00	0
0	TV	\$0.00	0

Rows Per Page : 10 Total : 7

To remove any item from the table drag it to this box

It works fine, but we can see some unexpected functionality: it is possible to drop items from the tree by dragging it to Recycled Bin area; columns count, price and row id columns values are always 0.

To implement the correct business logic, we have to **create custom drag and drop strategy**. For this sample we can use one strategy for table and dragArea. We'll implement DnDStrategy interface:

```
public DnDOperationResult doDragOperation(DnDMetaData from,
    DnDMetaData to) {
    if (from.getComponent().getId().equals("goodsTree")) {
        if (to.getComponent().getId().equals("garbageArea"))
            return DnDOperationResult.NO_REFRESH;
        if (to.getComponent().getId().equals("basketTable")) {
            TreeItem selectedItem = (TreeItem)
                getTreeDataModel().getElementAt(from.getSelectionPath());
            if ((selectedItem != null) &&
                (selectedItem.getType().equalsIgnoreCase("folder"))) {
                StringBuffer result = new StringBuffer();
                result.append(RenderingUtils.createJScriptBeginTag());
                result.append(AjaxUtils.getAlertFunction(
                    "It is possible to drag only items without childs.));
                result.append(RenderingUtils.createJScriptEndTag());
                try {
                    FacesContext.getCurrentInstance().getResponseWriter().
                        write(result.toString());
                } catch (IOException e) {
                    e.printStackTrace();
                }
                return DnDOperationResult.NO_REFRESH;
            }
        }
    }
    defaultStrategy.doDragOperation(from, to);
}
```

```

int row = to.getRow() >= 0 ? to.getRow() : 0;
getTableDataModel().setValueAt(new Integer(
    getTableDataModel().getRowCount()), row, 0);
if (selectedItem != null)
    getTableDataModel().setValueAt(new Double(
        selectedItem.getData()), row, 2);
getTableDataModel().setValueAt(new Integer(1), row, 3);
return DnDOperationResult.REFRESH_TARGET;
}
}
if (from.getComponent().getId().equals("basketTable")) {
    if (to.getComponent().getId().equals("garbageArea")) {
        getTableDataModel().removeRow(
            from.getRow() >= 0 ? from.getRow() : 0);
        return DnDOperationResult.REFRESH_SOURCE;
    }
    return defaultStrategy.doDragOperation(from, to);
}
return DnDOperationResult.NO_REFRESH;
}
}

```

Now we store price in the tree data model and set it in corresponding table column. “Count” row value is always 1 and “Row #” value is the number of row in the table.

Also we’ll apply the following **custom business rule**: customer can drag-n-drop only tree leafs, when customer tries to drag-n-drop folder - warning message will be shown.

We have to change **dnDStrategy** property to “#{CustomerBasketBean}” in JSP file to allow new strategy to be used (for table and dragArea components).

The screenshot shows a web application interface. On the left is a tree view with a 'Root' node containing folders like 'Video', 'Empty Video Folder', 'Storages', 'Video and Photography', and 'Sound'. The 'Video' folder is expanded to show 'Monitor', 'Plazma', and 'TV'. In the center is a table with columns 'Row #', 'Description', 'Price', and 'Count'. Below the table are 'Rows Per Page' (set to 10) and 'Total : 7'. At the bottom is a blue warning box with a trash icon and the text: 'To remove any item from the table drag it to this box'.

Row #	Description	Price	Count
1	CD	\$1.00	1
2	Monitor	\$200.00	1
3	TV	\$300.00	1
4	Tape	\$20.00	1
5	CDR	\$1.00	1
6	Camera	\$600.00	1
7	Super Camera	\$1,500.00	1

Once the easy steps above completed , we're able to run the shopping cart application with custom drag-n-drop strategies for the operations.

4. Full sample source code

File styles.css

```
.top {
    vertical-align: top;
}

.tableStyleClass {
    border-collapse: collapse;
    font-family: Tahoma, Verdana, Arial, Helvetica, serif, sans-serif;
    font-size: 7pt;
    border-top: 1px #72A6D7 solid;
    border-left: 1px #72A6D7 solid;
    border-right: 1px #72A6D7 solid;
    border-bottom: 1px #72A6D7 solid;
}

.cellDefaultStyleClass, .cellSelectedStyleClass, .cellFocusedStyleClass {
    font-family: Tahoma, Verdana, Arial, Helvetica, serif, sans-serif;
    font-size: 7pt;
    color: black;
}

.cellDefaultStyleClass {
    border-style: inset;
    border-width: 1px;
    padding: 1px 5px 1px 5px;
    border: 1px #72A6D7 solid;
    border-left: 1px #72A6D7 dotted;
    border-right: 1px #72A6D7 dotted;
    background-color: #FFFFFF;
}

.cellSelectedStyleClass {
    background-color: #DDE6F3;
}

.headerFocusedStyleClass {
}

.headerSortedStyleClass {
}

.headerDefaultStyleClass {
    font-family: Tahoma, Verdana, Arial, Helvetica, serif, sans-serif;
    font-size: 8pt;
    font-weight: bold;
    color: #FFFFFF;
    text-decoration: none;
    cursor: pointer;
    border-left: 1px #72A6D7 dotted;
    border-right: 1px #72A6D7 dotted;
}
```

```

.headerSelectedStyleClass {
}

.headerStyleClass {
  font-family: Tahoma, Verdana, Arial, Helvetica, serif, sans-serif;
  font-size: 8pt;
  font-weight: bold;
  color: #FFFFFF;
  text-decoration: none;
  cursor: pointer;
  background-color: #72A6D7;
}

.simpleSelectedStyleClass, .simpleFocusedStyleClass {
  background-color: #DDE6F3;
}

.simpleDefaultStyleClass {
  font-family: Tahoma, Verdana, Arial, Helvetica, serif, sans-serif;
  font-size: 8pt;
  font-weight: bold;
  color: #FFFFFF;
  text-decoration: none;
  cursor: pointer;
}

.dd {
  background-color: #DDE6F3;
  color: #72A6D7;
  border-bottom: gray 1px solid;
  border-left: gray 1px solid;
  border-right: gray 1px solid;
  border-top: gray 1px solid;
  font-family: Verdana, Arial, Helvetica, sans-serif;
  font-size: 7pt;
  font-weight: bold;
  width: 170px;
  height: 50px;
  text-align: left;
  padding: 2px;
}

.ddtext {
  padding-left: 24px;
  display: block;
  width: 140px;
  background: url( '../..//images/tree/delete.gif' ) no-repeat;
}

```

File customerBasket.jsp

```

<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>

```

```

<%@ taglib uri="http://www.softaspects.com/wgf/tree" prefix="tree" %>
<%@ taglib uri="http://www.softaspects.com/wgf/table" prefix="table" %>
<%@ taglib uri="http://www.softaspects.com/wgf/dragArea"
    prefix="dragArea" %>
<f:view>
  <html>
  <head>
    <title>Customer basket sample</title>
    <link href="styles.css" rel="stylesheet" type="text/css">
  <body>
  <h:form id="customerBasketSampleForm">
    <h:panelGrid columns="2" rowClasses="top">
      <tree:tree id="goodsTree"
        varName="goodsTree" dragEnabled="true">
        <tree:dataModel beanName="#{CustomerBasketBean.treeDataModel}"
          scope="session"/>
        <tree:selectionModel
          beanName="#{CustomerBasketBean.treeSelectionModel}"
          scope="session"/>
      </tree:tree>
      <table:table id="basketTable" pageSize="10"
        styleClass="tableStyleClass"
        cellSpacing="0" cellPadding="0" border="0"
        rowsPerPageValues="5,10,20,50" ajaxRefreshEnabled="true"
        dragEnabled="true" dropEnabled="true"
        dnDStrategy="#{CustomerBasketBean}">
        <table:dataModel
          beanName="#{CustomerBasketBean.tableDataModel}"
          scope="session"/>
        <table:columnModel
          beanName="#{CustomerBasketBean.tableColumnModel}"
          scope="session"/>
        <table:interfaceManager scope="session"
          cellSelectedStyleClass="cellSelectedStyleClass"
          cellFocusedStyleClass="cellFocusedStyleClass"
          cellDefaultStyleClass="cellDefaultStyleClass"
        />
        <table:header scope="session"/>
      </table:table>
      <h:outputText value=""/>
      <dragArea:dragArea id="garbageArea"
        styleClass="dd" dropEnabled="true"
        dnDStrategy="#{CustomerBasketBean}">
        <h:outputText styleClass="ddtext" value=
          "To remove any item from the table drag it to this box"/>
      </dragArea:dragArea>
    </h:panelGrid>
  </h:form>
</body>
</html>
</f:view>

```

File CustomerBasket.java

```

package samples.dragndrop;

import com.softaspects.faces.galileo.support.ajax.AjaxUtils;

```

```

import com.softaspects.faces.galileo.support.draganddrop.DnDMetaData;
import
com.softaspects.faces.galileo.support.draganddrop.strategy.DnDOperationResult;
import com.softaspects.faces.galileo.support.draganddrop.strategy.DnDStrategy;
import
com.softaspects.faces.galileo.support.draganddrop.strategy.base.CopySelfMoveStra
tegy;
import com.softaspects.framework.galileo.components.base.ComponentDefinitions;
import com.softaspects.framework.galileo.components.table.*;
import com.softaspects.framework.galileo.components.tree.TreeItem;
import
com.softaspects.framework.galileo.components.treemodel.BaseTreeDataModelImpl;
import
com.softaspects.framework.galileo.components.treemodel.BaseTreeSelectionModelImp
l;
import com.softaspects.framework.galileo.renderers.html.tree.TreeRendererUtil;
import com.softaspects.framework.galileo.support.renderers.RenderingUtils;
import testingapplication.table.TableDataModelUtil;

import javax.faces.context.FacesContext;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * Customer basket sample managed bean
 */
public class CustomerBasket implements DnDStrategy {
    /**
     * Prefix for images path
     */
    public static final String baseImagesPath =
        "{ApplicationRootContext}/images/tree/";

    /**
     * Tree data model
     */
    private BaseTreeDataModelImpl treeDataModel = null;
    /**
     * Tree selection model
     */
    private BaseTreeSelectionModelImpl treeSelectionModel = null;
    /**
     * Table data model
     */
    private DataModel tableDataModel = null;
    /**
     * Table column model
     */
    private ColumnModel tableColumnModel = null;

    /**
     * Default drag and drop strategy
     */
    private DnDStrategy defaultStrategy = new CopySelfMoveStrategy();

    /**
     * Tree data model getter

```

```

*
* @return tree data model
*/
public BaseTreeDataModelImpl getTreeDataModel() {
    if (treeDataModel == null) {
        treeDataModel = new BaseTreeDataModelImpl();
        TreeItem root = TreeRendererUtil.createFolder(null,
            "Root", baseImagesPath + "user1.gif");
        treeDataModel.addElement(root);

        TreeItem video = TreeRendererUtil.createFolder(root,
            "Video", baseImagesPath + "monitor.gif");
        TreeRendererUtil.createDoc(video, "Monitor",
            baseImagesPath + "monitor2.gif").setData("200");
        TreeRendererUtil.createDoc(video, "Plazma",
            baseImagesPath + "plasma-tv.gif").setData("2000");
        TreeRendererUtil.createDoc(video, "TV",
            baseImagesPath + "tv.gif").setData("300");

        TreeRendererUtil.createFolder(root, "Empty Video Folder",
            baseImagesPath + "monitor.gif");

        TreeItem storages = TreeRendererUtil.createFolder(root,
            "Storages", baseImagesPath + "package.gif");
        TreeItem storagess = TreeRendererUtil.createFolder(storages,
            "Storages", baseImagesPath + "package.gif");
        TreeRendererUtil.createDoc(storagess, "CD", baseImagesPath +
            "cd.gif").setData("1");
        TreeRendererUtil.createDoc(storagess, "CDR", baseImagesPath +
            "cd_gold.gif").setData("1");
        TreeItem other = TreeRendererUtil.createFolder(storages,
            "Other", baseImagesPath + "cube_yellow.gif");
        TreeRendererUtil.createDoc(other, "Tape", baseImagesPath +
            "tape.gif").setData("20");
        TreeRendererUtil.createDoc(other, "Record", baseImagesPath +
            "record.gif").setData("50");

        TreeItem videophoto = TreeRendererUtil.createFolder(root,
            "Video and Photography", baseImagesPath + "videocamera.gif");
        TreeRendererUtil.createDoc(videophoto, "Camera",
            baseImagesPath + "camera.gif").setData("600");
        TreeRendererUtil.createDoc(videophoto, "Super Camera",
            baseImagesPath + "camera2.gif").setData("1500");

        TreeItem sound = TreeRendererUtil.createFolder(root, "Sound",
            baseImagesPath + "loudspeaker.gif");
        TreeRendererUtil.createDoc(sound, "Speaker",
            baseImagesPath + "loudspeaker2.gif").setData("40");
        TreeRendererUtil.createDoc(sound, "Megaphone",
            baseImagesPath + "megaphone.gif").setData("60");
        TreeRendererUtil.createDoc(sound, "Headphones",
            baseImagesPath + "headphones.gif").setData("10");
        TreeRendererUtil.createDoc(sound, "Radio",
            baseImagesPath + "radio.gif").setData("5");

    }
    return treeDataModel;
}

```

```

/**
 * Tree selection model getter
 *
 * @return tree selection model
 */
public BaseTreeSelectionModelImpl getTreeSelectionModel() {
    if (treeSelectionModel == null)
        treeSelectionModel = new BaseTreeSelectionModelImpl();
    return treeSelectionModel;
}

/**
 * Table data model getter
 *
 * @return table data model
 */
public DataModel getTableDataModel() {
    if (tableDataModel == null) {
        tableDataModel = new DataModelImpl();
        List vTable = new ArrayList();
        tableDataModel.setData(
            vTable, getTableColumnModel().getColumnCount());
    }
    return tableDataModel;
}

/**
 * Table column model getter
 *
 * @return table column model
 */
public ColumnModel getTableColumnModel() {
    if (tableColumnModel == null) {
        tableColumnModel = new ColumnModelImpl();
        int column = 0;
        Column rowColumn = new ColumnImpl(
            TableDataModelUtil.ROW_COLUMN, "Row #", column++);
        rowColumn.setPreferredWidth("100px");
        rowColumn.setColumnClass(TableDataModelUtil.INT);
        rowColumn.setAlignMode(ComponentDefinitions.ALIGN_CENTER);
        Column titleColumn = new ColumnImpl(
            "description", "Description", column++);
        titleColumn.setPreferredWidth("200px");
        titleColumn.setAlignMode(ComponentDefinitions.ALIGN_LEFT);
        Column priceColumn = new ColumnImpl("price", "Price", column++);
        priceColumn.setPreferredWidth("100px");
        priceColumn.setColumnClass(TableDataModelUtil.CURRENCY);
        Column countColumn = new ColumnImpl("count", "Count", column);
        countColumn.setPreferredWidth("100px");
        countColumn.setColumnClass(TableDataModelUtil.INT);
        tableColumnModel.addColumn(rowColumn);
        tableColumnModel.addColumn(titleColumn);
        tableColumnModel.addColumn(priceColumn);
        tableColumnModel.addColumn(countColumn);
    }
    return tableColumnModel;
}

```

```

/**
 * Do drag and drop operation
 * @param from source component
 * @param to target component
 * @return operation result
 */
public DnDOperationException doDragOperation(
    DnDOperationException from, DnDOperationException to) {
    if (from.getComponent().getId().equals("goodsTree")) {
        if (to.getComponent().getId().equals("garbageArea"))
            return DnDOperationException.NO_REFRESH;
        if (to.getComponent().getId().equals("basketTable")) {
            TreeItem selectedItem =
                (TreeItem) getTreeDataModel().getElementAt(
                    from.getSelectionPath());
            if ((selectedItem != null) &&
                (selectedItem.getType().equalsIgnoreCase("folder"))) {
                StringBuffer result = new StringBuffer();
                result.append(RenderingUtils.createJScriptBeginTag());
                result.append(AjaxUtils.getAlertFunction(
                    "It is possible to drag only items without childs."));
                result.append(RenderingUtils.createJScriptEndTag());
                try {
                    FacesContext.getCurrentInstance().
                        getResponseWriter().write(result.toString());
                } catch (IOException e) {
                    e.printStackTrace();
                }
                return DnDOperationException.NO_REFRESH;
            }
            defaultStrategy.doDragOperation(from, to);
            int row = to.getRow() >= 0 ? to.getRow() : 0;
            getTableDataModel().setValueAt(
                new Integer(getTableDataModel().getRowCount()), row, 0);
            if (selectedItem != null)
                getTableDataModel().setValueAt(
                    new Double(selectedItem.getData()), row, 2);
            getTableDataModel().setValueAt(new Integer(1), row, 3);
            return DnDOperationException.REFRESH_TARGET;
        }
    }
    if (from.getComponent().getId().equals("basketTable")) {
        if (to.getComponent().getId().equals("garbageArea")) {
            getTableDataModel().removeRow(
                from.getRow() >= 0 ? from.getRow() : 0);
            return DnDOperationException.REFRESH_SOURCE;
        }
        return defaultStrategy.doDragOperation(from, to);
    }
    return DnDOperationException.NO_REFRESH;
}
}

```