Developing with Tables in Web Dynpro Java

Applies to:

Web Dynpro Java in SAP NetWeaver 7.0 (2004s)

Summary

This tutorial explains the basic functions of the Table UI element like creating the necessary UI elements, binding the data and mapping the context. Additionally it provides some additional features like sorting, displaying details in a form, calculating the totals and deleting rows.

Sample Project

If you want to follow the tutorial step-by-step, you may download the initial project template. If you just want to check the running sample, download the complete project:

- *TutWD_Table_Init.zip*: Initial Web Dynpro project template
- *TutWD__Table.zip*: Complete Web Dynpro project

Author(s): SAP NetWeaver Product Management

Company: SAP AG

Created on: 01 February 2008

Table of Contents

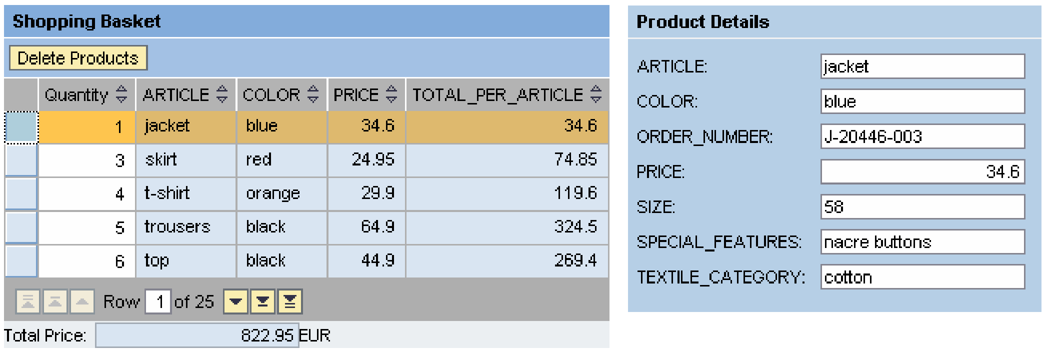
Developing with Tables in Web Dynpro

The display of data records in tables and forms and the editing of these – for example, selecting, deleting, or sorting – are central functions in Web applications.

The data structure that is the basis for this determines the layout of the table or a form. Therefore, you will first create the context in this tutorial and bind the data to this context. Based on the data structure that this available, you can bind the table and the form to the context during creation. The generation of columns and input fields as well as the assignment of fields to the attributes of the context then takes place automatically.

In this tutorial, you will learn how to:

- Create and map the context of the component controller and the views
- Create a table and bind its properties onto the view context
- Create a detailed form and bind it onto the view context
- Implement a function for sorting table rows
- Implement a function to delete single or several rows
- Implement a function for calculating totals for individual articles and implementing a total sum



Shopping Basket

Quantity	ARTICLE	COLOR	PRICE	TOTAL_PER_ARTICLE
1	jacket	blue	34.6	34.6
3	skirt	red	24.95	74.85
4	t-shirt	orange	29.9	119.6
5	trousers	black	64.9	324.5
6	top	black	44.9	269.4

Row 1 of 25

Total Price: 822.95 EUR

Product Details

ARTICLE:	jacket
COLOR:	blue
ORDER_NUMBER:	J-20446-003
PRICE:	34.6
SIZE:	58
SPECIAL_FEATURES:	nacre buttons
TEXTILE_CATEGORY:	cotton

The project template *TutWD_Table_Init* forms the basic structure of the application.

This includes:

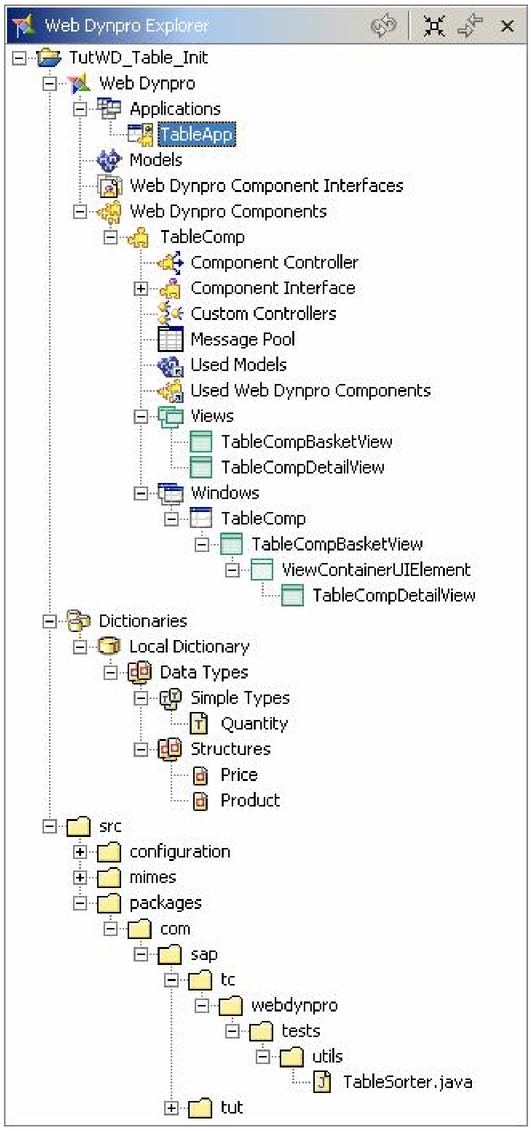
- A structure in the *Local Dictionary* that defines the fields of a product data record and thus the columns of the table to be created
- A second structure for price calculation
- A *window* in which two views are displaying the table and the detailed form
- Sample data for the shopping basket is stored in order to make creation of a data model unnecessary.
- A Java class *TableSorter* that provides the sort function

Importing a Project Template

If you are already familiar with Web Dynpro and the SAP NetWeaver Developer Studio you probably know how to import a project, otherwise you can find the necessary steps described at help.sap.com: [Importing a Project]

Initial Project Structure

Once the Web Dynpro project template *TutWD_Table_Init* has been imported, the following project structure should be displayed in the Web Dynpro Explorer:



```
TutWD_Table_Init
  Web Dynpro
    Applications
      TableApp
    Models
    Web Dynpro Component Interfaces
    Web Dynpro Components
      TableComp
        Component Controller
        Component Interface
        Custom Controllers
        Message Pool
        Used Models
        Used Web Dynpro Components
        Views
          TableCompBasketView
          TableCompDetailView
        Windows
          TableComp
            TableCompBasketView
              ViewContainerUIElement
                TableCompDetailView
  Dictionaries
    Local Dictionary
      Data Types
        Simple Types
          Quantity
        Structures
          Price
          Product
  src
    configuration
    mimes
    packages
      com
        sap
          tc
            webdynpro
              tests
                utils
                  TableSorter.java
          tut
```

Note: The dictionary simple type *Quantity* (built-in type integer) is used to prevent negative quantity values by setting the value constraint *Minimum Inclusive* as 0.

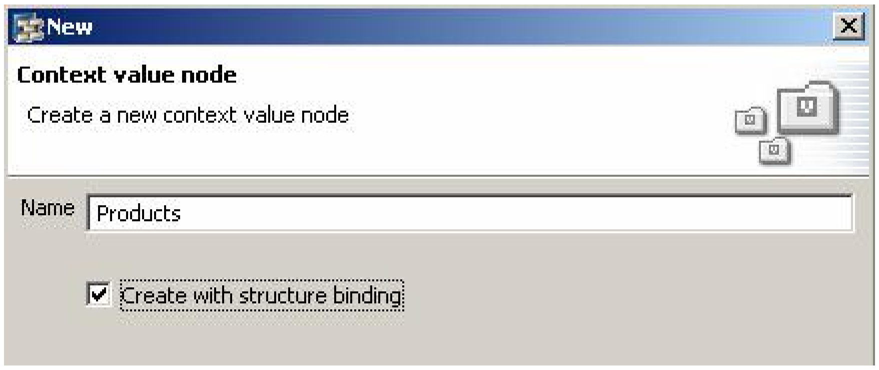
Creating the Component Controller Context

To provide the data to both views and keep them consistent, it is necessary to keep them at a higher level. The context of the component controller serves this purpose. The contexts of the views are then mapped to the context attributes of the component controller that they require.

Finally, the UI elements – here the product table and the detailed form – must then be mapped to the context of your view.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Component Controller*.
2. With the secondary mouse button, click the *Component Controller* and choose *Edit*.
3. Switch to the *Context* tab, click *Context* with the secondary mouse button, and choose *New* → *Value Node.*

4. Enter **Products** as name, activate C*reate with structure binding*, and confirm with *Next*. In this manner, you can access the *Structures* provided in this project.



New

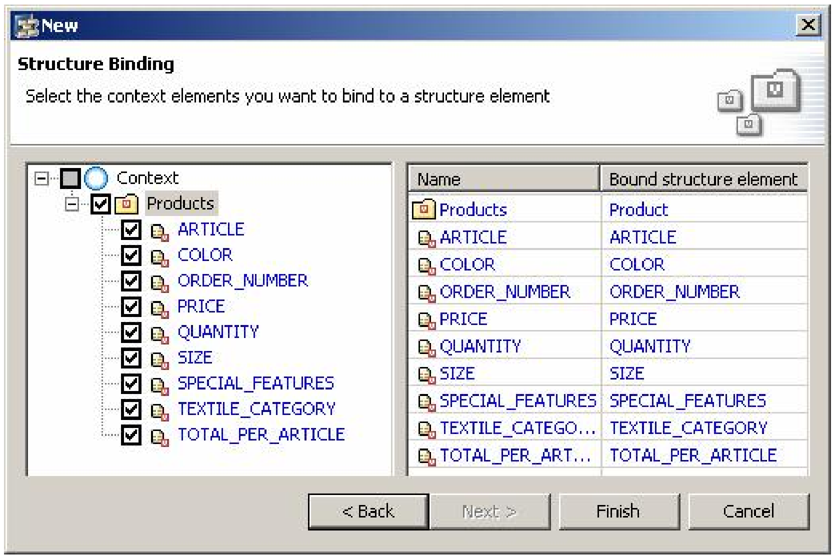
Context value node

Create a new context value node

Name: Products

[✓] Create with structure binding

5. Start *Dictionaries* → *Local Dictionary* → *com.sap.tut.wd.tutwd_table_init*, choose *Product*, and click *Next*. In the next window you can choose from the node attributes of the *context*.
6. Select all the available attributes by selecting the *Products* checkbox and confirm with *Finish*.



New

Structure Binding

Select the context elements you want to bind to a structure element

Context
- [✓] Products
  - [✓] ARTICLE
  - [✓] COLOR
  - [✓] ORDER_NUMBER
  - [✓] PRICE
  - [✓] QUANTITY
  - [✓] SIZE
  - [✓] SPECIAL_FEATURES
  - [✓] TEXTILE_CATEGORY
  - [✓] TOTAL_PER_ARTICLE

Name	Bound structure element
Products	Product
ARTICLE	ARTICLE
COLOR	COLOR
ORDER_NUMBER	ORDER_NUMBER
PRICE	PRICE
QUANTITY	QUANTITY
SIZE	SIZE
SPECIAL_FEATURES	SPECIAL_FEATURES
TEXTILE_CATEGO...	TEXTILE_CATEGORY
TOTAL_PER_ART...	TOTAL_PER_ARTICLE

You can now display and edit the properties of the individual node attributes by selecting the respective node attribute. The properties are then displayed in the lower window in the *Properties* tab. Change them accordingly to the table shown below.

Context Node/Attribute	Property	Value
Products	selection	1..n
TOTAL_PER_ARTICLE	readonly	true
	calculated	true

7. Save the current status of your project by choosing *Save all Metadata.*

Providing the Data

To be able to display some data as an example, data has been stored in the source code for this tutorial. The methods for displaying this data and for calling it are available in the implementation of the component controller and can now be activated.

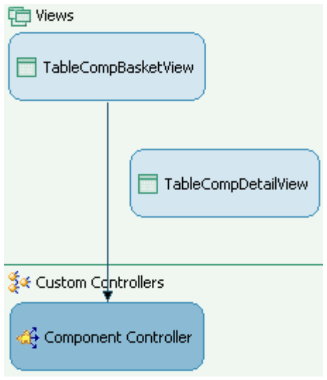
1. Switch to the Implementation tab page.
2. Select the comment lines of the method *createBasket()* and remove the comment commands.
3. Go to the `wdDoInit` method and remove the comment command for calling the `createBasket()` method.
4. Press the secondary mouse button and choose *Source* > *Organize Imports* from the context menu and *Save* all Metadata.

Mapping the View Context onto the Component Controller Context

So that the data becomes available in both views, now each view must be mapped onto the component controller context. Here you select attributes that are required in the respective view.

1. Navigate to *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* and start the *Data Modeler* in the *Diagram View* by double-clicking it or choose *Open Data Modeler* with the secondary mouse button.

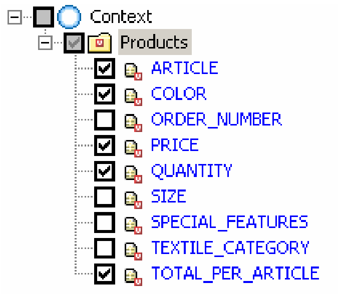
2. Click on *Create a data link* and draw a line from the *TableCompBasketView* icon to the *Component Controller icon*. You have started the assistant for *Context Mapping*.



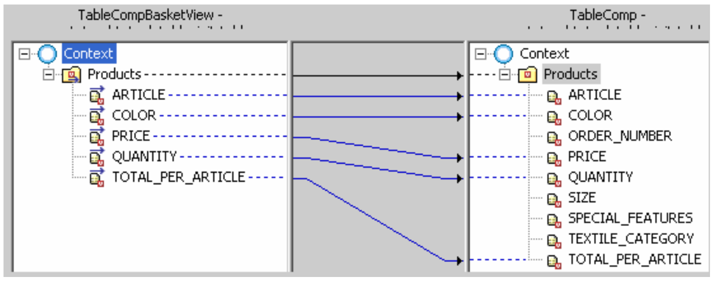
3. In the following screen, drag the *Product* value node from the component controller (right) to the root node *Context* in the *TableCompBasketView* (left).

4. Select the fields *ARTICLE, COLOR, PRICE, QUANTITY, and TOTAL_PER_ARTICLE* and confirm by pressing *Ok*.

   The following graphic illustrates the mapping:



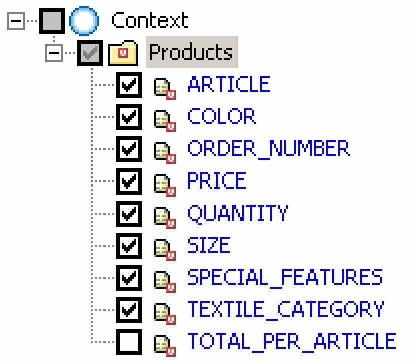
5. The following window shows the mapped elements. Confirm by choosing *Finish*



6. Save the current status of your project by choosing *Save all Metadata.*

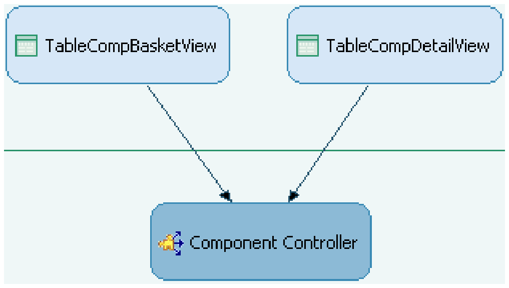
Mapping the TableCompDetailView onto the Component Controller Context

1. Click on *Create a data link* and draw a line from the *TableCompDetailView* icon to the *Component Controller* icon. You have started the assistant for *Context Mapping*.
2. In the following screen, drag the *Product* value node from the *component controller* (right) to the root node *Context* in the *TableCompDetailView* (left).
3. Select all the attribute by activating *Products* and then deactivate *TOTAL_PER_ARTICLE*. Confirm choosing *OK*.



4. Save the current status of your project by choosing *Save all Metadata.*

The contexts of the views have now been mapped to the context of the component controller.



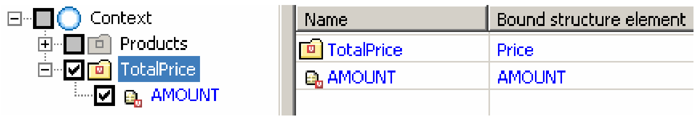
Enhancing the View Context

In the next step, you create context elements for the total price and the quantity. These need to be kept in the context so that different UI elements can access them. Since the data is required only in this view, it does not need to be mapped to the component controller context.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView.*

2. Switch to the *Context* tab, click *Context* with the secondary mouse button, and choose *New* → *Value Node.*

3. Enter **TotalPrice** as name, activate *Create with structure binding*, and confirm with *Next*.
4. Start *Dictionaries* → *Local Dictionary* → *com.sap.tut.wd.tutwd_table_init*, choose *Price* and click *Next*.
5. Select all the available attributes by activating *TotalPrice* and choose *Finish*.
   The following graphic illustrates the structure binding:



Name	Bound structure element
TotalPrice	Price
AMOUNT	AMOUNT

6. Set the *Context* Node/Atibute properties in the *Properties* tab as shown in the table below:

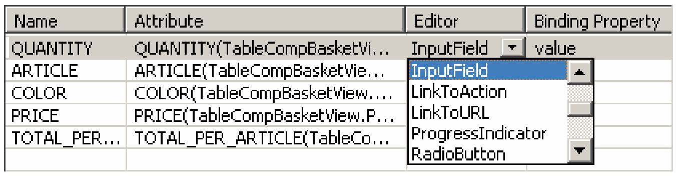
Context Node/Attribute	Property	Value
TotalPrice	cardinality	1..1
	selection	1..1
Amount	calculated	true
	readonly	true

7. Save the current status of your project by choosing *Save all Metadata.*

Creating the Table

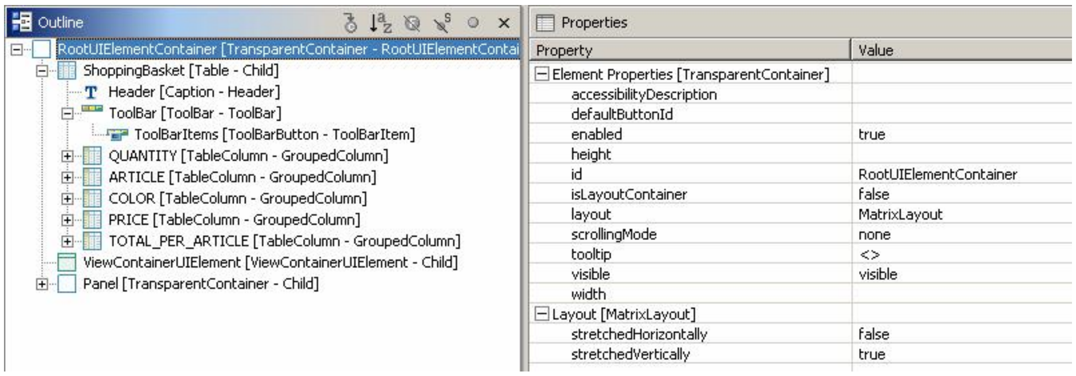
Here you create the *ShoppingBasket* table in the *TableCompBasketView*, in which products can be displayed after they have been bound to the context. After this step, the table consists only of a type of frame. The columns and column headings are generated in the next step when the table is bound to the context.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView.*
2. Double-click *TableCompBasketView* to start editing and switch to *Layout* tab.
3. In the *Outline*, click with the secondary mouse button on *RootUIElementContainer* and choose *Apply Template* from the context menu.
4. In the *Template Wizard Window* select *Table* and press *Next*.
5. Select the *Products* checkbox in order to get all its attributes selected automatically and press *Next*.
6. Change the editor of the value attribute *QUANTITY* to *InputField* and choose *Finish*.



Name	Attribute	Editor	Binding Property
QUANTITY	QUANTITY(TableCompBasketVi...	InputField	value
ARTICLE	ARTICLE(TableCompBasketVie...		
COLOR	COLOR(TableCompBasketView....		
PRICE	PRICE(TableCompBasketView.P...		
TOTAL_PER...	TOTAL_PER_ARTICLE(TableCo...		

7. In the *Outline*, select the Table UI Element and set its ID to **ShoppingBasket** in the *Properties* tab.
8. With secondary mouse button insert a table *Header* as child of the *ShoppingBasket* and set its text property to **Shopping Basket**.
9. In the *Outline*, move the Table UI element called *ShoppingBasket* to the first position using drag and drop.
10. Add a *Toolbar* UI element to the table, add a *ToolBarButton* as *ToolBarItem* to it and set its text property to **Delete Products**.



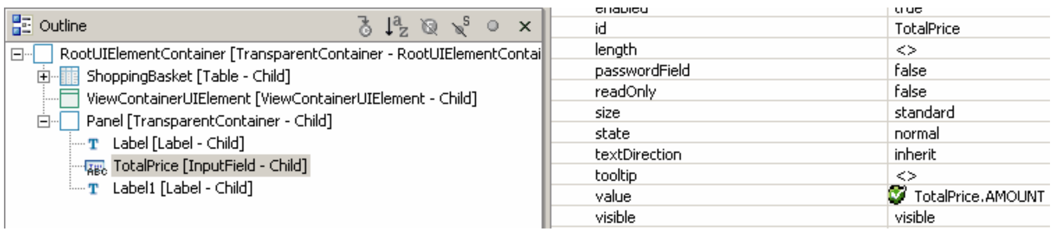
Outline

- RootUIElementContainer [TransparentContainer - RootUIElementContai
  - ShoppingBasket [Table - Child]
    - Header [Caption - Header]
    - ToolBar [ToolBar - ToolBar]
      - ToolBarItems [ToolBarButton - ToolBarItem]
    - QUANTITY [TableColumn - GroupedColumn]
    - ARTICLE [TableColumn - GroupedColumn]
    - COLOR [TableColumn - GroupedColumn]
    - PRICE [TableColumn - GroupedColumn]
    - TOTAL_PER_ARTICLE [TableColumn - GroupedColumn]
  - ViewContainerUIElement [ViewContainerUIElement - Child]
  - Panel [TransparentContainer - Child]

Properties

Property	Value
Element Properties [TransparentContainer]	
accessibilityDescription	
defaultButtonId	
enabled	true
height	
id	RootUIElementContainer
isLayoutContainer	false
layout	MatrixLayout
scrollingMode	none
tooltip	<>
visible	visible
width	
Layout [MatrixLayout]	
stretchedHorizontally	false
stretchedVertically	true

11. Select the *TotalPrice* InputField UI element and bind its value to context attribute **AMOUNT** of the *TotalPrice* node.



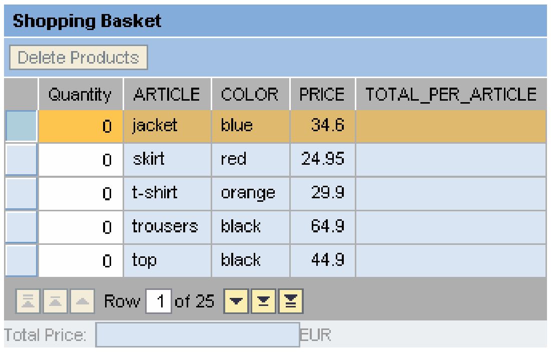
Outline

- RootUIElementContainer [TransparentContainer - RootUIElementContai
  - ShoppingBasket [Table - Child]
  - ViewContainerUIElement [ViewContainerUIElement - Child]
  - Panel [TransparentContainer - Child]
    - Label [Label - Child]
    - TotalPrice [InputField - Child]
    - Label1 [Label - Child]

id	TotalPrice
length	<>
passwordField	false
readOnly	false
size	standard
state	normal
textDirection	inherit
tooltip	<>
value	TotalPrice.AMOUNT
visible	visible

12. Save the current status of your project by choosing *Save all Metadata.*

After you have started the application with *Deploy New Archive and Run* in the context menu of the application *TableApp*, you will see the table displayed below.



Shopping Basket

Delete Products

Quantity	ARTICLE	COLOR	PRICE	TOTAL_PER_ARTICLE
0	jacket	blue	34.6	
0	skirt	red	24.95	
0	t-shirt	orange	29.9	
0	trousers	black	64.9	
0	top	black	44.9	

Row 1 of 25

Total Price: EUR

Creating and Binding the Detailed Form

In the detailed form, additional data on the project that is selected in the product table is to be displayed. If the user selects another line, the data is to be automatically adapted to the data in the detailed form.

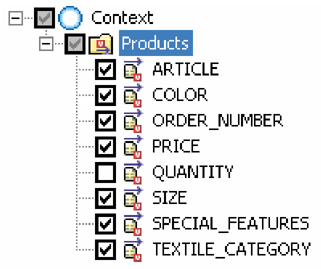
In the following step, you will create the detailed form and will bind it to the context using the function *Apply template*. Here the required fields are generated and the functions mentioned above are automatically at your disposal.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompDetailView* and open this by double-clicking for editing.

2. With secondary mouse button select the *Group* element and choose *Apply Template*. The *Template Wizard* is started.

3. Choose *Form*, enter **DetailForm** as name, and confirm with *Next*.

4. In the *Products* context node, activate the value attributes *ARTICLE*, *COLOR*, *ORDER_NUMBER*, *PRICE*, *SIZE*, *SPECIAL_FEATURES*, *TEXTILE_CATEGORY*, and confirm with *Next* >. In the next window, you can change the fields before they are generated. Then confirm by pressing *Finish*.

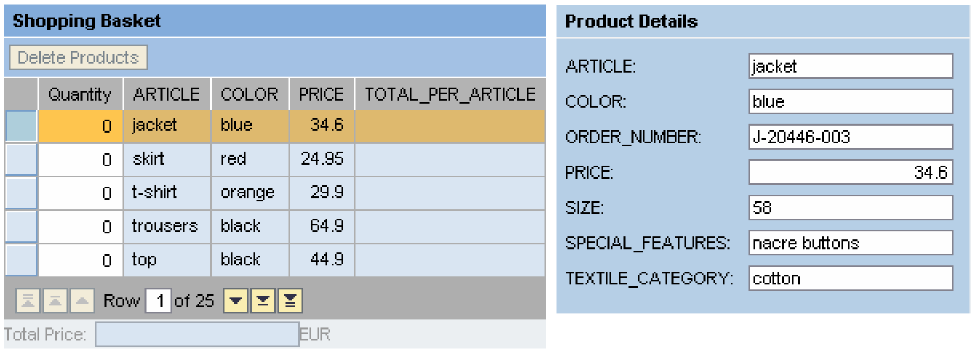


- Context
  - Products
    - [✓] ARTICLE
    - [✓] COLOR
    - [✓] ORDER_NUMBER
    - [✓] PRICE
    - [ ] QUANTITY
    - [✓] SIZE
    - [✓] SPECIAL_FEATURES
    - [✓] TEXTILE_CATEGORY

5. Since the values of the input fields are not to be changeable, set the *readOnly* property to *true* for each input field in the lower window in the *Properties* tab.

6. Save the current status of your project by choosing *Save all Metadata.*

After you have started the application with *Deploy New Archive and Run* in the context menu of the application *TableApp*, you will see the table displayed below. The data is bound and is therefore displayed in both UI elements. You can select one or several data records, whereby the data record whose detailed data is displayed in the form is highlighted in color during multiple selection.



Shopping Basket

Delete Products

Quantity	ARTICLE	COLOR	PRICE	TOTAL_PER_ARTICLE
0	jacket	blue	34.6	
0	skirt	red	24.95	
0	t-shirt	orange	29.9	
0	trousers	black	64.9	
0	top	black	44.9	

Row 1 of 25

Total Price: EUR

Product Details

ARTICLE:	jacket
COLOR:	blue
ORDER_NUMBER:	J-20446-003
PRICE:	34.6
SIZE:	58
SPECIAL_FEATURES:	nacre buttons
TEXTILE_CATEGORY:	cotton

Deleting Single or Several Rows

Since you have already defined the lead selection when you created and bound the table to the context in addition to implementing the option of multiple selection, you now simply need to read this in the next step in order to be able to delete the selected data records.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView* and start this by double-clicking for editing.

2. Switch to the *Layout* tab.

3. In the *Outline*, select the *ToolBarButton* and click the pushbutton ... in the *value* field of the *onAction* event property in the *Properties* tab. The *New Action* window opens.

4. Enter **DeleteProducts** as name and **Delete Products** as text. Leave the other settings unchanged and choose *Finish*.

5. Switch to the implementation of the event handler *onActionDeleteProduct* and insert the following code:

```
onActionDeleteProduct()
int n = wdContext.nodeProducts().size();
int leadSelected = wdContext.nodeProducts().getLeadSelection();
```

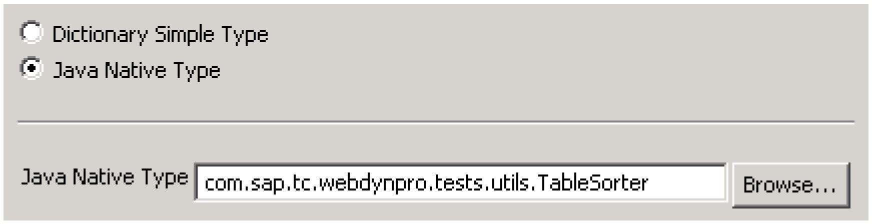
```
// loop backwards to avoid index troubles
for (int i = n - 1; i >= 0; --i)
{
   if (wdContext.nodeProducts().isMultiSelected(i) || leadSelected == i )
   {
     wdContext.nodeProducts().removeElement(wdContext.nodeProducts().
             getElementAt(i));
   }
}
```

6. Save the current status of your project by choosing *Save all Metadata.*

Sorting

To be able to sort a column, you must use a separate Java class for sorting the context elements. In this scenario, there is already a special class called *TableSorter*, which provides this function.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView* and start this by double-clicking for editing.
2. Switch to the *Context* tab page.
3. Using the secondary mouse button, click *Context*, choose *New* → *Value Attribute*, enter **TableSorter** as name, and confirm by pressing *Finish*.
4. Select the value attribute *TableSorter* and choose the *type* property in the lower window in the *Properties* tab.
5. Click the pushbutton ..., choose *Java Native Type* and enter **com.sap.tc.webdynpro.tests.utils.TableSorter**. Confirm choosing *OK*.


( ) Dictionary Simple Type
(●) Java Native Type

Java Native Type: com.sap.tc.webdynpro.tests.utils.TableSorter

6. Switch to the *Actions* tab, and create an action using *New*. Assign **Sort** as name and text, and confirm by pressing *Finish*.
7. Switch to the *Implementation* tab page and add the following program code into the *wdDoModifyView* method:

```
wdDoModifyView()
if (firstTime)
{
   IWDTable table = (IWDTable) view.getElement("ShoppingBasket");
   wdContext.currentContextElement().setTableSorter( new
               TableSorter(table, wdThis.wdGetSortAction(), null));
}
```

8. Switch to the method *onActionSort* and enter the following source code:

```
onActionSort()
wdContext.currentContextElement().getTableSorter().sort(wdEvent,
     wdContext.nodeProducts());
```

At runtime, a context element provided in the context is addressed in this way. It, in turn, addresses the current table sorter.

Note: You will find the program code for the sort class under sap → packages in the package com.sap.tc.webdynpro.tests.utils.TableSorter.

9. Start the context menu of the *implementation* and choose *Source* → *Organize Imports.*
10. Save the current status of your project by choosing *Save all Metadata.*

After you have started the application using *Deploy New Archive and Run* in the context menu of the application *TableApp*, you can click the title row of the column according to which you wish to sort the data records. Beside the title, you will see the symbol for *Sort ascending*. Click on this symbol and the data records are sorted accordingly. The symbol then changes to *Sort descending*.

Calculating the Total per Article

The *calculated* property of the value attribute *TOTAL_PER_ARTICLE* was defined in the component controller context as *true*. This makes it possible to have this attributed calculated at runtime.

Inserting the Calculation

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Component Controller* and start this by double-clicking for editing.
2. Switch to the *Implementation* tab.
3. Insert the following source text into the method *getProductsTOTAL_PER_ARTICLE.* In this way, the total price for each product is automatically calculated at runtime from the quantity and the price of the product.

```
getProductsTOTAL_PER_ARTICLE()
if (element.getQUANTITY() < 0)
     return new BigDecimal(0);
else
     return new
     BigDecimal(element.getQUANTITY()).multiply(element.getPRICE());
```

Note:.
Since the constraint of Simple Type QUANTITY, that prevents negative values, is checked after the value is calculated, it is possible to calculate negative values before the error message is forwarded to the user. Even it is not possible for the user to proceed before he has entered a valid value, we recommend to include this if-clause to prevent the data to be stored in the context. So you can ensure that negative product quantity values are not added to the total price per article.

4. Start the context menu of the *implementation* and choose *Source* → *Organize Imports.*
5. Save the current status of your project by choosing *Save all Metadata.*

Updating the Calculation with the ENTER Key

The prices are updated whenever the lead selection is changed - that is, whenever the user selects another line. In this way, a roundtrip is triggered, which updates the view and thus executes the required calculations of the *calculated attributes*.

Intuitively, the user will however expect that his input will be updated after pressing the `ENTER` key. In the following step, you will add this function by adding an empty action to the *Quantity* column. In this way, a round trip can be triggered by pressing the `ENTER` key.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView* and start this by double-clicking for editing.
2. Switch to the *Layout* tab.
3. In the *Outline*, select the *Quantity*_editor of the Sopping Basket table and click the pushbutton ... in the value field of the onAction event property in the Properties tab. The New Action window opens.
4. Enter **Roundtrip** as name and **Roundtrip** as text. Leave the other settings unchanged and choose Finish.
5. Save the current status of your project by choosing *Save all Metadata.*

Note: To make the application usable for Web Dynpro clients, the onLeadSelect event of the table should also be mapped to this Roundtrip action. This is necessary because Web Dynpro clients do not create an automatic round trip when the lead selection is set.

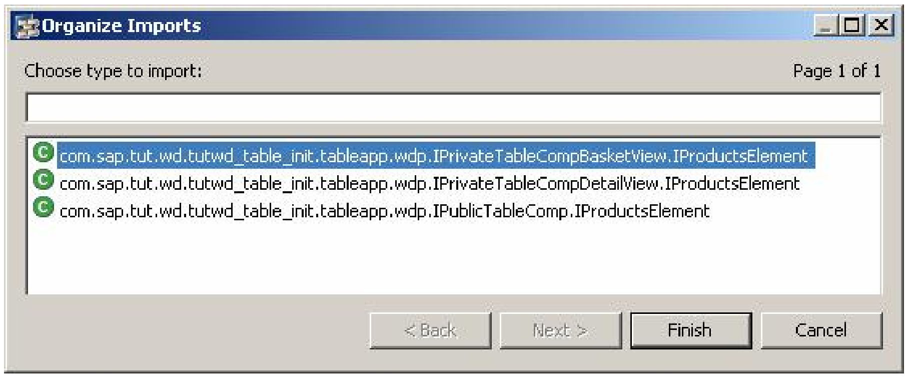
Calculating the Total

To display the total sum of products in the shopping basket, you need to create a separate field for the total price. Like *TOTAL_PER_ARTICLE*, this is to be a *calculated attribute*.

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView* and switch to the *Implementation* tab.
2. Add the following program code into the *getTotalPriceAMOUNT* method:

```
getTotalPriceAMOUNT()
BigDecimal total = new BigDecimal(BigInteger.ZERO);
int n = wdContext.nodeProducts().size();
for (int i = 0; i < n; ++i)
{
       IProductsElement item =
wdContext.nodeProducts().getProductsElementAt(i);
       int quantity = item.getQUANTITY();
       BigDecimal price = item.getPRICE();
       total = total.add(new
       BigDecimal(BigInteger.valueOf(quantity)).multiply(price));
}
return total;
```

3. Start the context menu of the *implementation* and choose *Organize Imports.* Choose `com.sap.tut.wd.tutwd_table_init.tableapp.wdp.IPrivateTableCompBasketView.IProductsElement` and confirm with *Finish*.



com.sap.tut.wd.tutwd_table_init.tableapp.wdp.IPrivateTableCompBasketView.IProductsElement
com.sap.tut.wd.tutwd_table_init.tableapp.wdp.IPrivateTableCompDetailView.IProductsElement
com.sap.tut.wd.tutwd_table_init.tableapp.wdp.IPublicTableComp.IProductsElement

4. Save the current status of your project by choosing *Save all Metadata.*

After you have started the application with *Deploy New Archive and Run* in the context menu of the application *TableApp*, you can have the total per article and the total sum calculated by entering a quantity and pressing the `ENTER` key.

Related Content

Table (UI Element Guide on SAP Help Portal)

Copyright