Developing with Tables in Web Dynpro

SAP NetWeaver 04

Copyright

© Copyright 2007 SAP AG. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft, Windows, Outlook, and PowerPoint are registered trademarks of Microsoft Corporation.
IBM, DB2, DB2 Universal Database, OS/2, Parallel Sysplex, MVS/ESA, AIX, S/390, AS/400, OS/390, OS/400, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere, Netfinity, Tivoli, Informix, i5/OS, POWER, POWER5, OpenPower and PowerPC are trademarks or registered trademarks of IBM Corporation.

Adobe, the Adobe logo, Acrobat, PostScript, and Reader are either trademarks or registered trademarks of Adobe Systems Incorporated in the United States and/or other countries.
Oracle is a registered trademark of Oracle Corporation.

UNIX, X/Open, OSF/1, and Motif are registered trademarks of the Open Group.
Citrix, ICA, Program Neighborhood, MetaFrame, WinFrame, VideoFrame, and MultiWin are trademarks or registered trademarks of Citrix Systems, Inc.

HTML, XML, XHTML and W3C are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

Java is a registered trademark of Sun Microsystems, Inc.

JavaScript is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MaxDB is a trademark of MySQL AB, Sweden.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver, and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves informational purposes only. National product specifications may vary.

These materials are subject to change without notice. These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

Icons in Body Text

Icon	Meaning
⚠	Caution
❖	Example
💡	Note
⬆	Recommendation
<>	Syntax

Additional icons are used in SAP Library documentation to help you identify different types of information at a glance. For more information, see *Help on Help* → *General Information Classes and Information Classes for Business Information Warehouse* on the first page of any version of *SAP Library*.

Typographic Conventions

Type Style	Description
Example text	Words or characters quoted from the screen. These include field names, screen titles, pushbuttons labels, menu names, menu paths, and menu options. Cross-references to other documentation.
Example text	Emphasized words or phrases in body text, graphic titles, and table titles.
EXAMPLE TEXT	Technical names of system objects. These include report names, program names, transaction codes, table names, and key concepts of a programming language when they are surrounded by body text, for example, SELECT and INCLUDE.
`Example text`	Output on the screen. This includes file and directory names and their paths, messages, names of variables and parameters, source text, and names of installation, upgrade and database tools.
`Example text`	Exact user entry. These are words or characters that you enter in the system exactly as they appear in the documentation.
`<Example text>`	Variable user entry. Angle brackets indicate that you replace these words and characters with appropriate entries to make entries in the system.
`EXAMPLE TEXT`	Keys on the keyboard, for example, `F2` or `ENTER`.

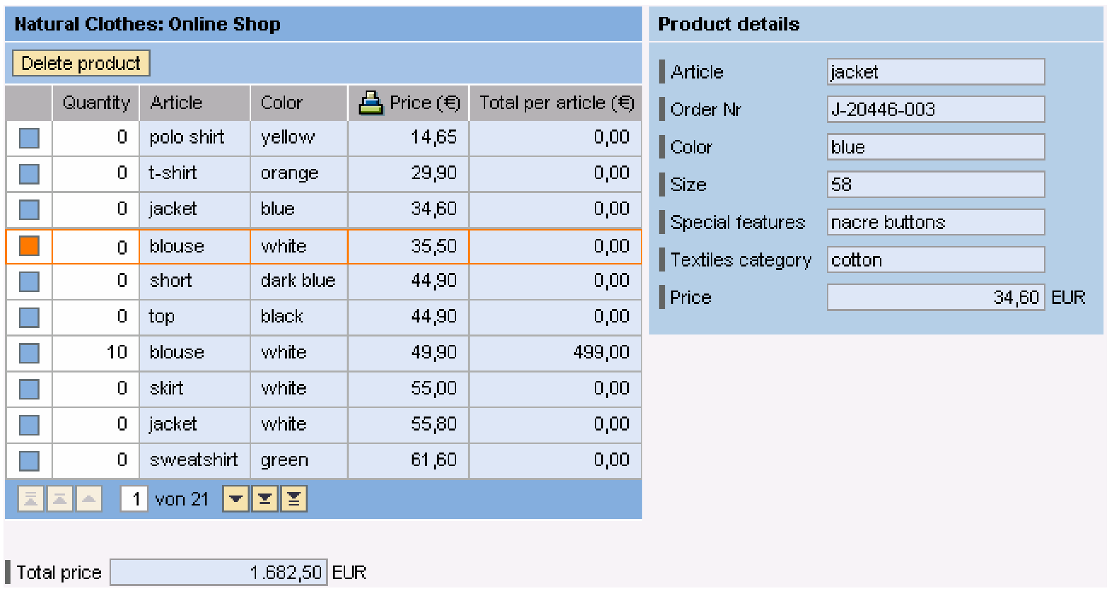
Developing with Tables in Web Dynpro

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 map it onto the view context
- Create a detailed form and map 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
- Format according to different currencies



Natural Clothes: Online Shop

Delete product

	Quantity	Article	Color	Price (€)	Total per article (€)
🟦	0	polo shirt	yellow	14,65	0,00
🟦	0	t-shirt	orange	29,90	0,00
🟦	0	jacket	blue	34,60	0,00
🟧	0	blouse	white	35,50	0,00
🟦	0	short	dark blue	44,90	0,00
🟦	0	top	black	44,90	0,00
🟦	10	blouse	white	49,90	499,00
🟦	0	skirt	white	55,00	0,00
🟦	0	jacket	white	55,80	0,00
🟦	0	sweatshirt	green	61,60	0,00

1 von 21

Total price 1.682,50 EUR

Product details

Article	jacket
Order Nr	J-20446-003
Color	blue
Size	58
Special features	nacre buttons
Textiles category	cotton
Price	34,60 EUR

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 and currency formatting
- A *window* in which two views are grouped for 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

Prerequisites

Systems, Installations, and Authorizations

- The SAP NetWeaver Developer Studio is installed on your PC.
- You have access to the J2EE Engine.

Knowledge

- Basic knowledge of the Java programming language
- Knowledge of programming Web Dynpro applications

Importing the Project Template

Use

So that you can continue with the tutorial, the SAP Developer Network (SDN) provides:

- The project template *TutWD_Table_Init* (the starting point for this tutorial)
- The Web Dynpro project *TutWD_Table*, which corresponds to the result at the end of this tutorial
- The text of this tutorial as PDF file

Prerequisites

- You have access to the SAP Developer Network.
- You have installed the SAP NetWeaver Developer Studio.

Procedure

1. Log on to the SAP Developer Network through URL http://sdn.sap.com. If you do not have a user ID, you must register before you can proceed.
2. Choose *Home* → *Developer Areas* → *Web Application Server* → *Web Dynpro* and then *Code Samples* → *Sample Applications and Tutorials.*
3. Download the zip file *TutWD_Table_Init.zip* containing the project template *TutWD_Table_Init* and save the zip file to any directory on your local hard disk or directly in the work area of the SAP NetWeaver Developer Studio.
4. Unpack the content of the ZIP file *TutWD_Table_Init.zip*.
5. Start the SAP NetWeaver Developer Studio.
6. Import the Web Dynpro project *TutWD_Table_Init*
    a. In the menu, choose *File* → *Import…*
    b. In the next window, choose *Existing Project into Workspace* and click *Next* to confirm.
    c. Choose *Browse*, open the folder in which you unpacked the project *TutWD_Table_Init*, and select the project.
    d. Confirm by choosing *Finish*

Result

Now you can open the following structure of the project *TutWD_Table_Init* in the Web Dynpro Explorer.

Web Dynpro Project Structure

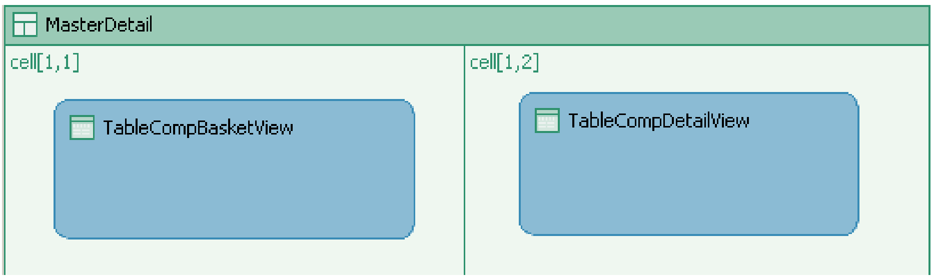
Init_Table_TutWD :Web Dynpro Project

TableApp :Web Dynpro Application

TableComp :Web Dynpro Component

MasterDetail :Viewset

.viewset *MasterDetail* The required views are embedded in the



TableCompBasketView :View

This view is provided for displaying the table with the products.

TableCompDetailView :View

This view is provided for displaying the detailed data of a selected product.

TableTutorialWindow :Window

In the window, both views are embedded as view set.

Local Dictionary Types

Quantity: Simple Data Type

This dictionary simple type (built-in type *integer*) is used to define *non-negative* quantity values by setting the value constraint *Minimum Inclusive* as 0.

Price: Structure Type

Structure type with two fields: AMOUNT of type *decimal* and CURRENCY of simple type *currency*.

Product: Structure Type

Structure type defining a product with 10 fields:
ARTICLE, COLOR, CURRENCY, ORDERNUMBER, PRICE, QUANTITY, SIZE, SPECIAL_FEATURES, TEXTILE_CATEGORY and TOTAL_PER_ARTICLE

Source

TableSorter.java: Java Class File

Utility class for sorting context node elements displayed in a Table UI Element.

The Web Dynpro project *TutWD_Table_Init* can already be run in the Web browser, but no content is displayed yet on the user interface.

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.

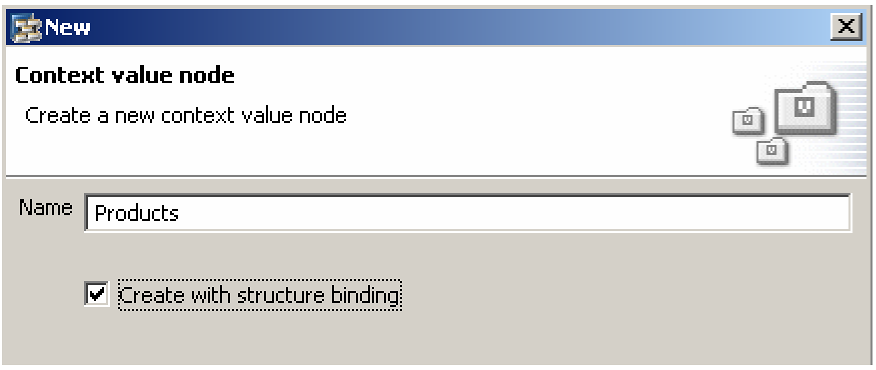
Finally, the UI elements – here the product table and the detailed form – must then be bound to the context of your view (*data binding*).

Procedure

Creating the Component Controller Context

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Component Controller*.
2. With the right mouse key, click the *Component Controller* and choose *Edit*.
3. Switch to the *Context* tab, click *Context* with the right mouse key, and choose *New* → *Value Node.*
4. Enter **`Products`** as name, activate *Create with structure binding*, and confirm with *Next >*.

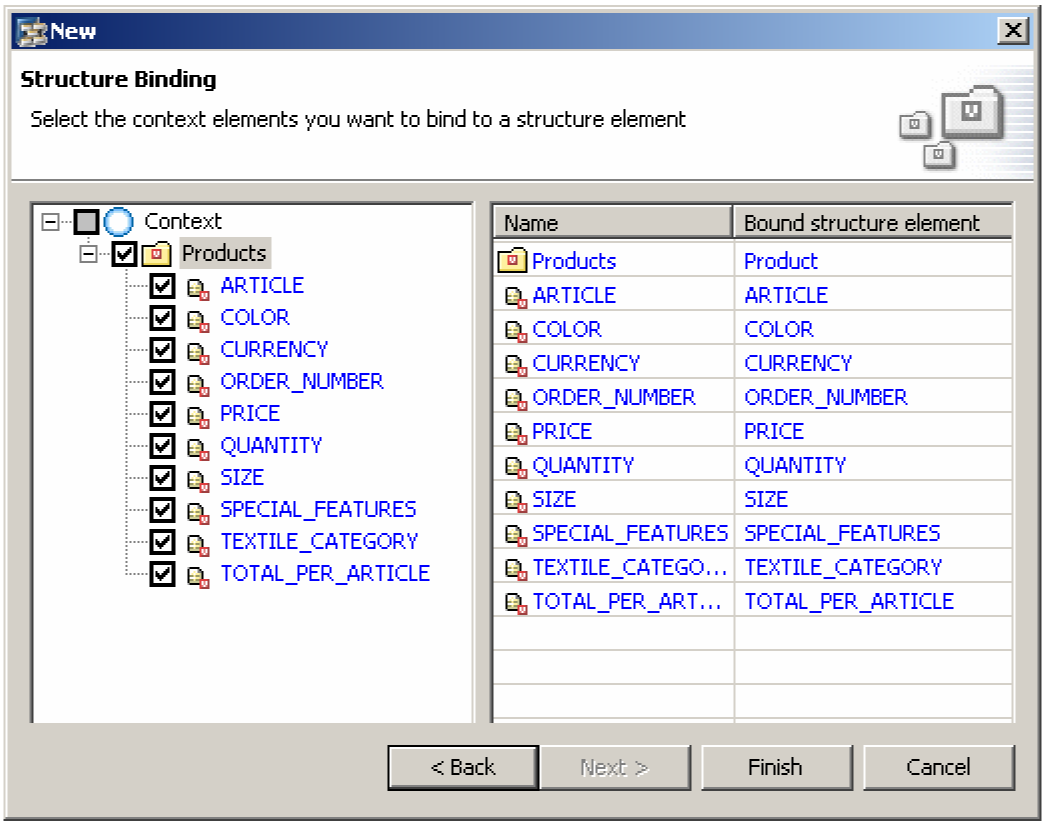
   In this manner, you can access the *Structures* provided in this project.



Name: Products

[✓] Create with structure binding

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



Structure Binding

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

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

Name	Bound structure element
Products	Product
ARTICLE	ARTICLE
COLOR	COLOR
CURRENCY	CURRENCY
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.

7. Choose the value *1..n* for the *selection* property of the value node *Products*. In this way it is possible to select several data records in the table.
8. For the *readOnly* property of the value attribute *TOTAL_PER_ARTICLE*, choose the value *true*. The setter method for this attribute is deleted.
9. For the *calculated* property of the value attribute *TOTAL_PER_ARTICLE*, choose the value *true*. This attribute can thus be calculated automatically at runtime.
10. 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 command.
3. Go to the `wdDoInit` method and remove the comment command for calling the`createBasket` method.
4. With the right mouse key, click the implementation and choose *Source* → *Organize Imports*.
5. Save the current status of your project by choosing *Save all Metadata.*

Mapping the View Context onto the Component Controller Context

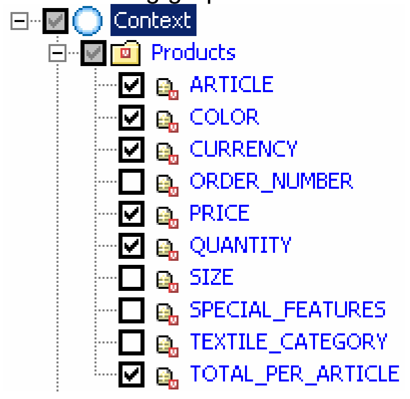
So that the data becomes available in both views, now each view must be mapped onto the component controller context.

Here you select attributes that are required in the respective view.

Procedure

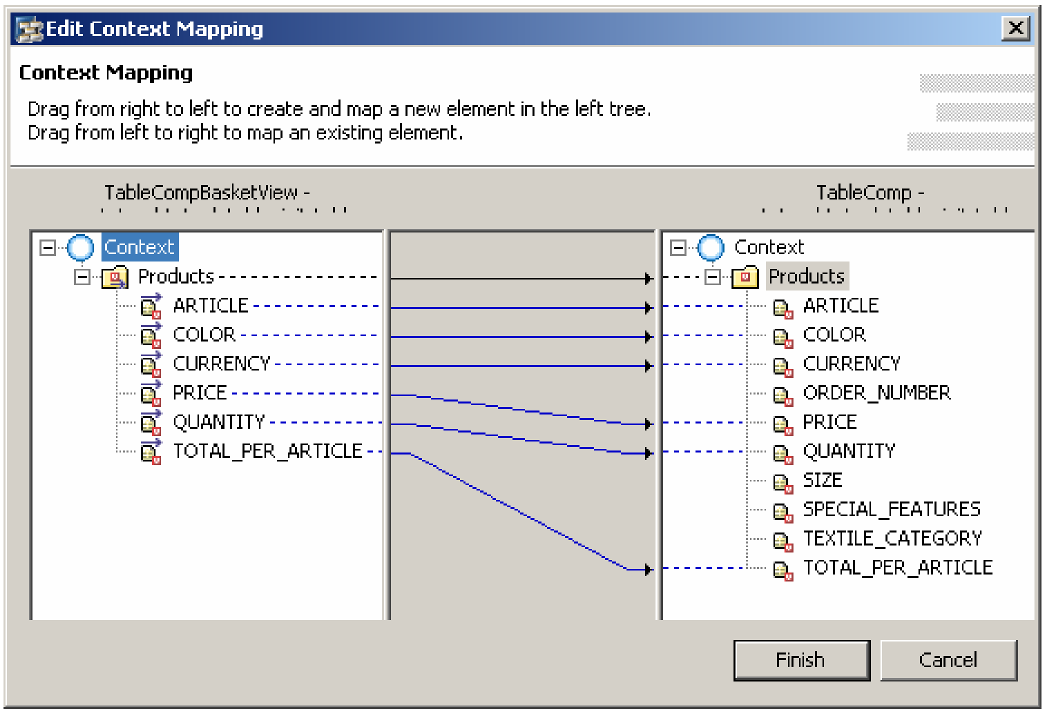
Mapping the TableCompBasketView onto the Component Controller Context

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.
2. Click on *Create a data link* and draw a line from the *TableCompBasketView* to the *Component Controller*. 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, CURRENCY, PRICE, QUANTITY, and TOTAL_PER_ARTICLE* and confirm by pressing *Ok*.
   The following graphic illustrates the mapping:



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

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



Context Mapping

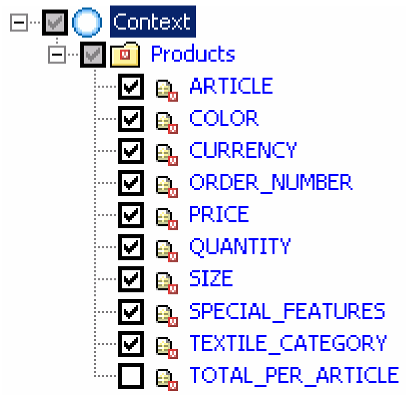
Drag from right to left to create and map a new element in the left tree.
Drag from left to right to map an existing element.

TableCompBasketView -	TableComp -
Context	Context
Products	Products
ARTICLE	ARTICLE
COLOR	COLOR
CURRENCY	CURRENCY
PRICE	ORDER_NUMBER
QUANTITY	PRICE
TOTAL_PER_ARTICLE	QUANTITY
	SIZE
	SPECIAL_FEATURES
	TEXTILE_CATEGORY
	TOTAL_PER_ARTICLE

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

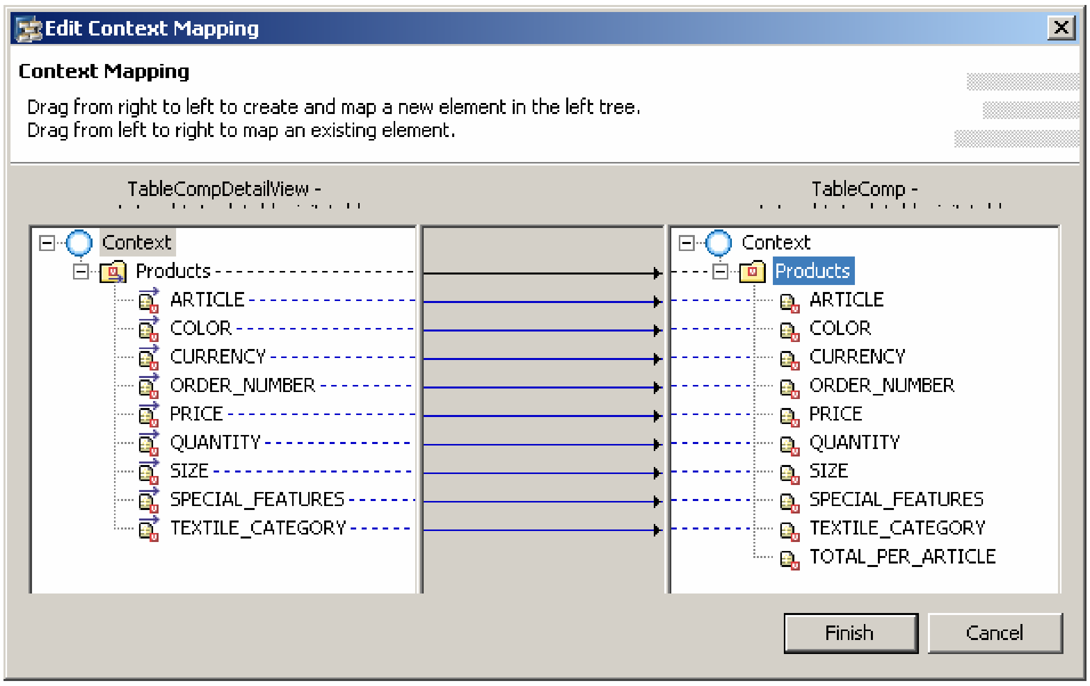
Mapping the TableCompDetailView to the Component Controller Context

1. Click on *Create a data link* and draw a line from the *TableCompDetailView* to the *Component Controller*. 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*.



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

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



Edit Context Mapping

Context Mapping

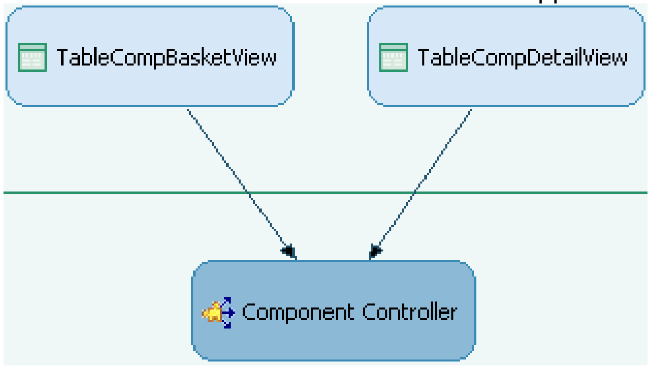
Drag from right to left to create and map a new element in the left tree.
Drag from left to right to map an existing element.

TableCompDetailView -	TableComp -
Context	Context
Products	Products
ARTICLE	ARTICLE
COLOR	COLOR
CURRENCY	CURRENCY
ORDER_NUMBER	ORDER_NUMBER
PRICE	PRICE
QUANTITY	QUANTITY
SIZE	SIZE
SPECIAL_FEATURES	SPECIAL_FEATURES
TEXTILE_CATEGORY	TEXTILE_CATEGORY
	TOTAL_PER_ARTICLE

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

Result

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

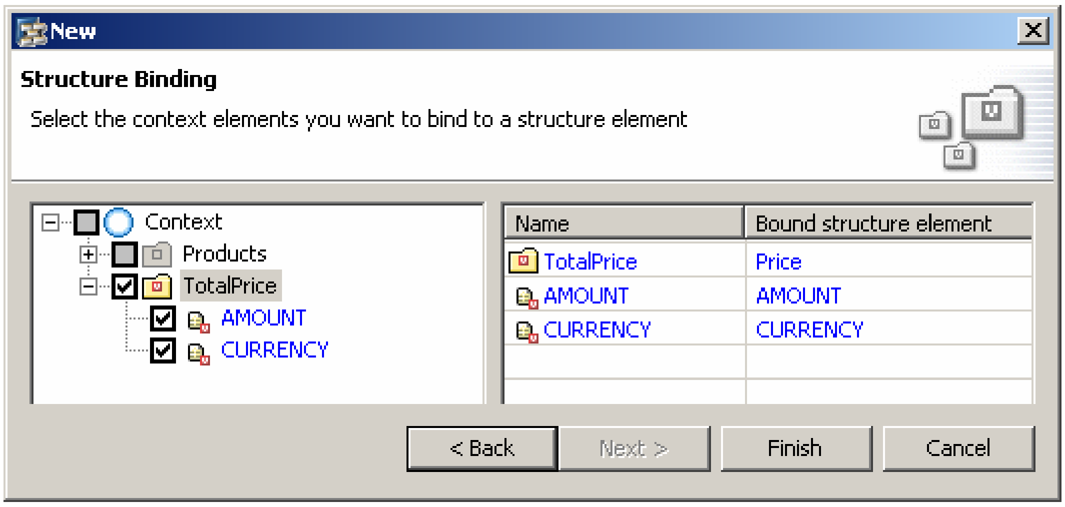


Enhancing the View Context

In the next step, you create context elements for the total price, the quantity, and the currency. 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.

Procedure

1. Choose TutWD_Table_Init → Web Dynpro → Web Dynpro Components → TableComp → Views → *TableCompBasketView.*
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 **`TotalPrice`** as name, activate *Create with structure binding*, and confirm with *Next* >.
5. Start *Dictionaries* → *Local Dictionary* → *com.sap.tut.wd.tutwd_table_init*, choose *Price* and click *Next* >.
6. Select all the available attributes by activating *TotalPrice* and choose *Finish*.
   The following graphic illustrates the structure binding:



New

Structure Binding

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

Name	Bound structure element
TotalPrice	Price
AMOUNT	AMOUNT
CURRENCY	CURRENCY

7. Select the *TotalPrice* context node.
   - Choose the value *1..1* for the *cardinality* property.
   - Choose the value *1..1* for the *selection* property.
8. Select the value attribute *AMOUNT*.
   - Choose the value *true* for the *calculated* property.
   - Choose the value *true* for the *readOnly* property. The set method is deleted.
9. Save the current status of your project by choosing *Save all Metadata.*

Result

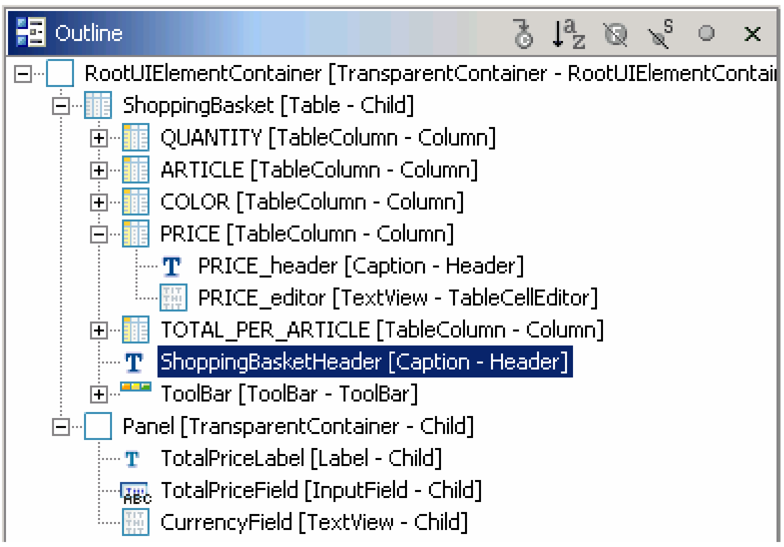
You have now created all the context nodes and attributes that are necessary in order to ensure availability and to save the data.

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.

Procedure

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView.*
2. Double-click *TableCompBasketView* to start editing. The *View Designer* appears and you are on the *Layout* tab.
3. In the *Outline*, click with the right mouse key on *RootUIElementContainer* and choose *Insert Child*.
4. As type, choose *Table,* assign as ID ShoppingBasket, and confirm with *Finish*.
5. In the *Outline*, click with the right mouse key on the table *ShoppingBasket* and choose *Insert Header*.



```
Outline
RootUIElementContainer [TransparentContainer - RootUIElementContai
  ShoppingBasket [Table - Child]
    QUANTITY [TableColumn - Column]
    ARTICLE [TableColumn - Column]
    COLOR [TableColumn - Column]
    PRICE [TableColumn - Column]
      PRICE_header [Caption - Header]
      PRICE_editor [TextView - TableCellEditor]
    TOTAL_PER_ARTICLE [TableColumn - Column]
    ShoppingBasketHeader [Caption - Header]
    ToolBar [ToolBar - ToolBar]
  Panel [TransparentContainer - Child]
    TotalPriceLabel [Label - Child]
    TotalPriceField [InputField - Child]
    CurrencyField [TextView - Child]
```

6. In the lower window, enter **`Shopping Basket`** as table title for the *text* property in the *Properties* tab.
7. Save the current status of your project by choosing *Save all Metadata.*

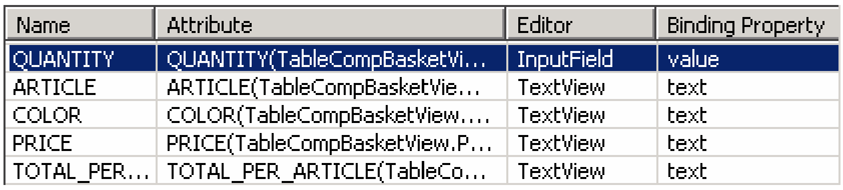
Binding the Table to the Context

At the current time, the contexts of the views and the component controller are bound with one another. To be able to display the data in the user interface, you will – in the next step – bind the table with the context of its view.

Procedure

1. If you have not done so already, double-click the *TableCompBasketView* and choose the tab *Layout*.
2. With the right mouse key, click the table *ShoppingBasket* and choose *Create Binding*.
3. Select the required value attributes *ARTICLE, COLOR, PRICE, QUANTITY*, and *TOTAL_PER_ARTICLE* and confirm by pressing *Next* >. In the next window, you can adapt the layout of the table.

4. Change the editor of the value attribute QUANTITY to *InputField* and confirm with *Next.*



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

5. In the lower window, go to the *Properties* tab and change the property *visibleRowCount* of the table to **`10`** in order to be able to always display 10 lines.

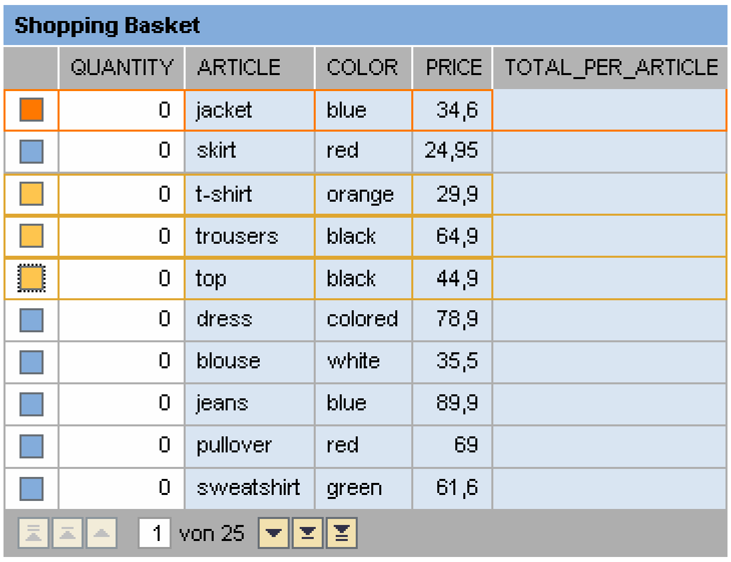
   You can process the titles of the columns by selecting the *Header* in the *Outline* under the required column, and then adapting its *text* property.

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

Result

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.

When the table was bound to the context, a column was generated for each value attribute and the data can now be displayed. The lead selection of the component controller enables you to select several data records.



Shopping Basket

	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	
	0	dress	colored	78,9	
	0	blouse	white	35,5	
	0	jeans	blue	89,9	
	0	pullover	red	69	
	0	sweatshirt	green	61,6	

1 von 25

Creating and Binding the Detailed Form

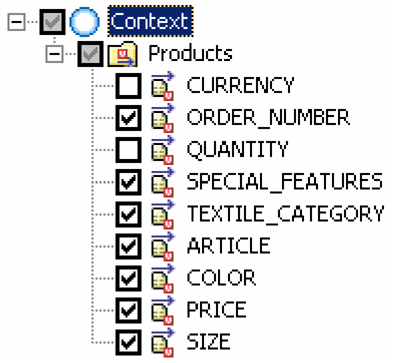
In the detailed form, additional data on the project that is selected in the product table is to be displayed. If the user selects another line, the data is to be automatically adapted to the data in the detailed form.

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.

Procedure

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompDetailView* and open this by double-clicking for editing.
2. In the *Outline*, click with the right mouse key on *RootUIElementContainer* and choose *Insert Child*.

3. As type, choose *Group,* assign as ID **`TotalPrice`**, and confirm with *Finish*.
4. With the right mouse key, click the *DetailGroup* and choose *Apply Template*. The *Template Wizard* is started.
5. Choose *Form*, enter **`DetailForm`** as name, and confirm with *Next >*.
6. 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*.



7. 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.
8. To set the heading of the detailed form, select the *DetailGroup_Header* in the *Outline* and enter **`Product Details`** for the *text* property.
9. Save the current status of your project by choosing *Save all Metadata.*

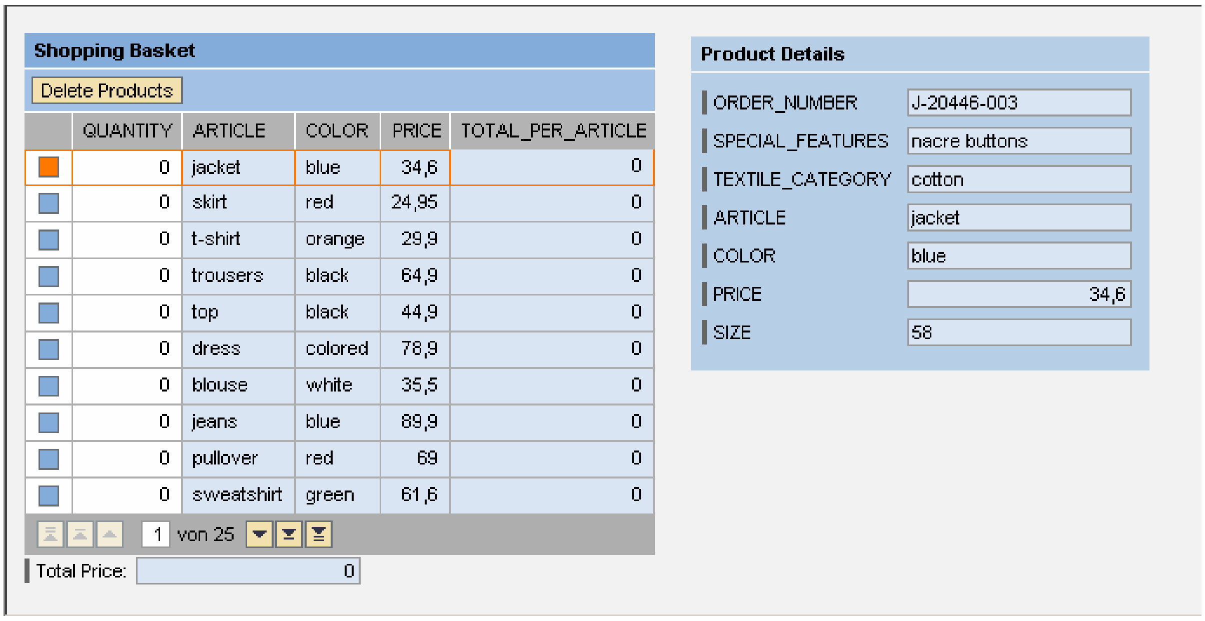
Result

At this point, you have already done the following:

- Created the context of the component controller and the view
- Mapped the contexts onto one another
- Created a table and a detailed form as UI elements
- Bound these to the context of the respective *view*

The basic tasks for creating a table and a form – the display of the required data – have been accomplished.

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
	0	skirt	red	24,95	0
	0	t-shirt	orange	29,9	0
	0	trousers	black	64,9	0
	0	top	black	44,9	0
	0	dress	colored	78,9	0
	0	blouse	white	35,5	0
	0	jeans	blue	89,9	0
	0	pullover	red	69	0
	0	sweatshirt	green	61,6	0

1 von 25

Total Price: 0

Product Details

ORDER_NUMBER	J-20446-003
SPECIAL_FEATURES	nacre buttons
TEXTILE_CATEGORY	cotton
ARTICLE	jacket
COLOR	blue
PRICE	34,6
SIZE	58

In the next steps, you will add other typical functions to the table.

- Delete
- Sort
- Calculate Totals
- Format According to Different Currencies

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.

Procedure

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*, click with the secondary mouse button on the table *ShoppingBasket* and choose *Insert ToolBar*.
4. In the *Outline*, click with the secondary mouse button on *ToolBar* and choose *Insert Child*.
5. Leave the *ToolBarButton* unchanged as type, assign as ID DeleteProductButton, and confirm with *Finish*.
6. 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.
7. Enter DeleteProducts as name and Delete Products as text. Leave the other settings unchanged and choose *Finish*.
8. Switch to the implementation of the event handler *onActionDeleteProduct* and insert the following code:

onActionDeleteProduct()

```
//@@begin onActionDeleteProduct(ServerEvent)
```

```
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));
  }
}
//@@end
```

To avoid problems with the index, the context is here executed in reverse. All elements selected in the table are deleted.

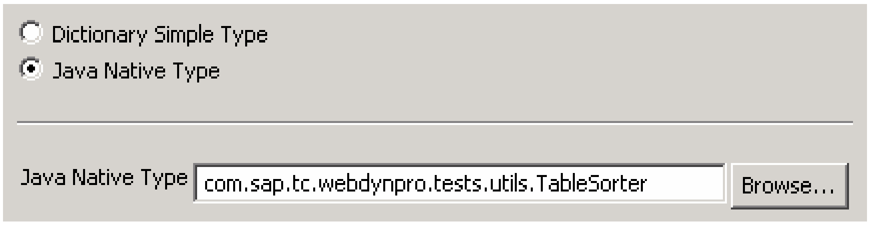
9. Save the current status of your project by choosing *Save all Metadata.*

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. However, this class is a preliminary version that is implemented directly in the runtime environment in a later release.

Procedure

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 [Browse...]

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()

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

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

onActionSort()

```
//@@begin onActionSort(ServerEvent)
wdContext.currentContextElement().getTableSorter().sort(wdEvent,
  wdContext.nodeProducts());
//@@end
```

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

You will find the program code for the sort class under *src* → *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.*

Result

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.

Procedure

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.* method:

getProductsTOTAL_PER_ARTICLE()

```
public java.math.BigDecimal
getProductsTOTAL_PER_ARTICLE(IPrivateTableComp.IProductsElement
element) {
  //@@begin
  return element.getQUANTITY() < 0
    ? new BigDecimal(0)
    : new BigDecimal(element.getQUANTITY())
        .multiply(element.getPRICE());
  //@@end
}
```

In this way, the total price for each product is automatically calculated at runtime from the quantity and the price of the product. 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*.
4. In the lower window in the *Properties* tab, go to *Event* and click the *onEnter* property.
5. Press the utmost right button … to define a new action, assign **`Roundtrip`** as name and text, leave the other settings unchanged and confirm with *Finish*.
6. Save the current status of your project by choosing *Save all Metadata.*

To make the application usable for Web Dynpro clients, the *onLeadSelect* event of the table *ShoppingBasket* should also be bound to this *Roundtrip* action. This is necessary because Web Dynpro clients do not create an automatic round trip when the lead selection is set. With the round trip the controller methods for price calculation (calculated context attribute getters) and the generic type validation for reporting negative quantity value entries (dictionary simple type QUANTIY) are processed on server side.

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*.

Procedure

1. Choose TutWD_Table_Init → Web Dynpro → Web Dynpro Components → TableComp → *Views* → *TableCompBasketView* and switch to the Layout tab.
2. In the *Outline*, click with the right mouse key on *RootUIElementContainer* and choose *Insert Child*.
3. As type, choose *TransparentContainer*, assign as ID **`TotalPrice`**, and confirm with *Finish*.
4. Proceed in the same manner as in steps 2 and 3, and add the following UI elements to the *TransparentContainer*.

Type	ID
Label	**`TotalPriceLabel`**
InputField	**`TotalPriceField`**

5. In *Outline*, select *TotalPriceLabel*, and in the lower window enter **`Total Price:`** in the *Properties* tab for the *text* property.

6. In the *Outline, select TotalPriceField*, in the lower window choose the *Properties* tab and click the pushbutton … in the *value* property, then choose the Context attribute *TotalPrice.AMOUNT,* and confirm with *Ok.*

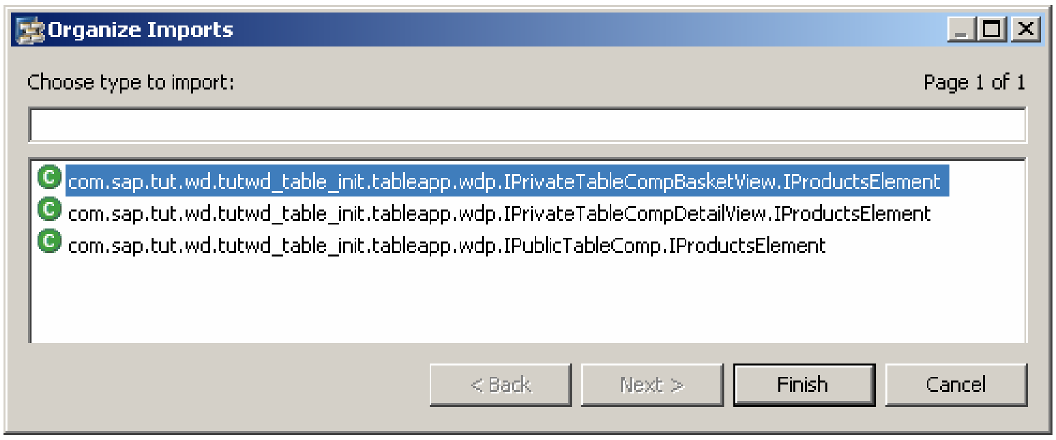
7. Switch to the *Implementation* tab page and add the following program code into the *getTotalPriceAMOUNT* method:

getTotalPriceAMOUNT()

```
public java.math.BigDecimal getTotalPriceAMOUNT(
  IPrivateTableCompBasketView.ITotalPriceElement element)
{
  //@@begin
  BigDecimal total = new BigDecimal(0), pricePerProduct;
  int n = wdContext.nodeProducts().size();
  for (int i = 0; i < n; ++i) {
    total = total.add(wdContext.nodeProducts()
      .getProductsElementAt(i).getTOTAL_PER_ARTICLE());
  }
  return total;
  //@@end
}
```

Step by step, the price for each product is determined here in a loop and these prices are then added to the total price. The total price per article is calculated by the first calculated context attribute getter method `getTOTAL_PER_ARTICLE()`.

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


Organize Imports

Choose type to import:

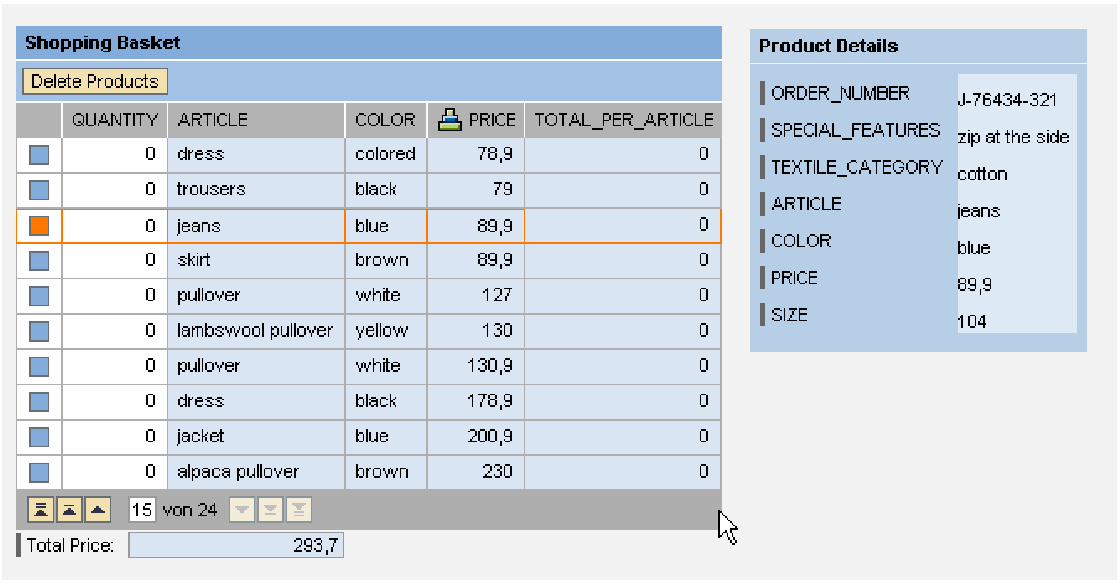
Page 1 of 1

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

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

Result

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.



Shopping Basket

Delete Products

	QUANTITY	ARTICLE	COLOR	PRICE	TOTAL_PER_ARTICLE
🟦	0	dress	colored	78,9	0
🟦	0	trousers	black	79	0
🟧	0	jeans	blue	89,9	0
🟦	0	skirt	brown	89,9	0
🟦	0	pullover	white	127	0
🟦	0	lambswool pullover	yellow	130	0
🟦	0	pullover	white	130,9	0
🟦	0	dress	black	178,9	0
🟦	0	jacket	blue	200,9	0
🟦	0	alpaca pullover	brown	230	0

15 von 24

Total Price: 293,7

Product Details

ORDER_NUMBER	J-76434-321
SPECIAL_FEATURES	zip at the side
TEXTILE_CATEGORY	cotton
ARTICLE	jeans
COLOR	blue
PRICE	89,9
SIZE	104

Setting and Changing the Currency

In this last step, you will set the currency and map this to the value attribute *CURRENCY*. All the fields concerned are then automatically changed when you change the currency.

Procedure

Setting the Currency

1. Choose *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompBasketView* and start this by double-clicking it.
2. In the *Outline*, click with the secondary mouse button on *RootUIElementContainer* and choose *Insert Child*.
3. As type, choose *TextView*, assign as ID **`CurrencyField`** , and confirm with *Finish*.
4. Select *CurrencyField*, go to the *Properties* window, click on the pushbutton … in the *text* property, choose the *Context* attribute *TotalPrice.CURRENCY*, and confirm with Ok.
5. Switch to the *Implementation* tab page.
6. Add the following program code into the *wdDoInit* method.

wdDoInit()

```
//@@begin wdDoInit()
if (wdContext.nodeProducts().size() > 0) {
  wdContext.currentTotalPriceElement().setCURRENCY(
    wdContext.nodeProducts().getProductsElementAt(0).getCURRENCY());
}
//@@end
```

Here, the currency of the products is set in the shopping basked for the context element *TotalPrice*. All the products in the shopping basket use the same currency.

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

Setting the Euro Label

One of the features of a new currency element is that this is displayed flush-right. This ensures that the suitable currency can be displayed afterwards with correct alignment.

1. Switch to the directory *TutWD_Table_Init* → *Web Dynpro* → *Web Dynpro Components* → *TableComp* → *Views* → *TableCompDetailView.*
2. Switch to the *Layout* tab.
3. Click with the right-mouse key on *TransparentContainer TotalPrice* in the *Outline* and choose *Insert Child*. As type, choose *Label*, assign as ID **`CurrencyLabel`**, and confirm with *Finish*.
4. Move the label by clicking it with the secondary mouse button and choose *Move Up* until it is under the *PRICE* input field.
5. Select the CurrencyLabel and click the pushbutton … in the *text* property in the lower window on the *properties* tab. Choose the *Products CURRENCY* and confirm with *Ok*.
6. Save your project by choosing *Save all Metadata.*

   You can change the currency by changing the passed parameter for `product.setCURRENCY("EUR")` in the implementation of the component controller in the `createBasket` method. All the currency labels then show the new value.

Building, Deploying, and Running the Table Tutorial

You have successfully completed the tutorial *Developing with Tables in Web Dynpro*.

Based on a predefined data structure, you have created the contexts of the component controller and the views and mapped them onto each other. You then bound the UI elements to the relevant view context and thus generated the required columns or fields.

For the table, you implemented the possibility to select several data records.

Based on this, you added the following functions:

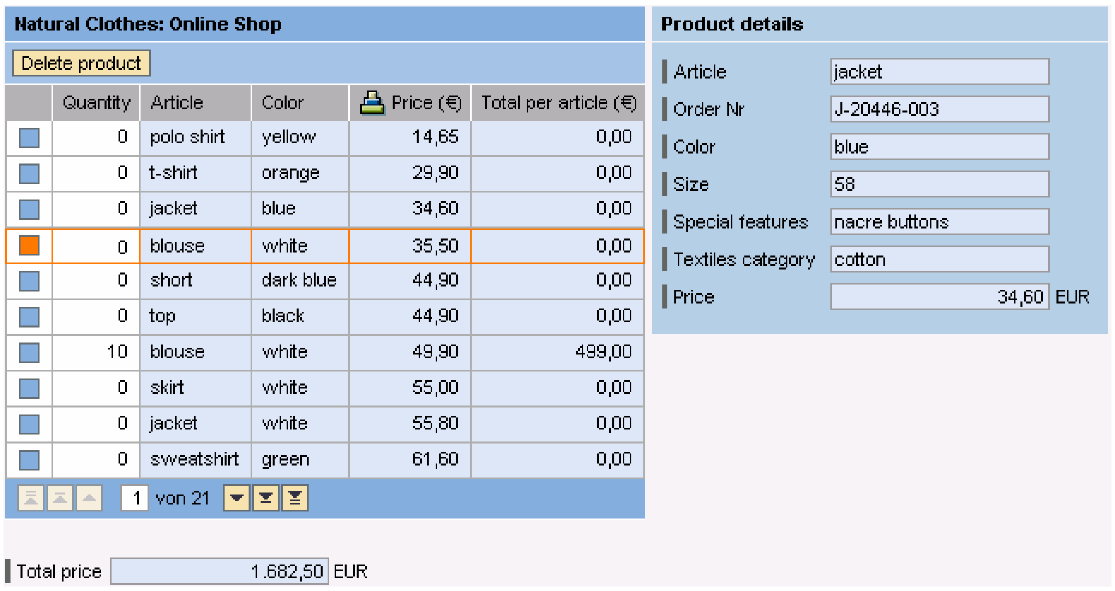
- Deletion of single or multiple rows
- Sorting of data records in ascending and descending order
- Calculation of subtotals and overall totals
- Setting and changing of currency format

Procedure

1. In the Web Dynpro Explorer, open the context menu for *TutWD_Table_Init* and choose *Rebuild Project*.
2. Open the context menu for the application *TableApp* and choose *Deploy New Archive and Run*.

Result

Once your application has started, you see the following:



Natural Clothes: Online Shop

Delete product

	Quantity	Article	Color	Price (€)	Total per article (€)
🟦	0	polo shirt	yellow	14,65	0,00
🟦	0	t-shirt	orange	29,90	0,00
🟦	0	jacket	blue	34,60	0,00
🟧	0	blouse	white	35,50	0,00
🟦	0	short	dark blue	44,90	0,00
🟦	0	top	black	44,90	0,00
🟦	10	blouse	white	49,90	499,00
🟦	0	skirt	white	55,00	0,00
🟦	0	jacket	white	55,80	0,00
🟦	0	sweatshirt	green	61,60	0,00

1 von 21

Product details

Article	jacket
Order Nr	J-20446-003
Color	blue
Size	58
Special features	nacre buttons
Textiles category	cotton
Price	34,60 EUR

Total price 1.682,50 EUR

The table displays the product data records. The selected data record is the lead selection and the corresponding detailed information is displayed in the details form.

You can select one or more data records and delete them by choosing *Delete product*.

Next to the column header *Price*, you see the symbol for *Sort ascending*; this function also becomes available for other columns when you click on the relevant column header. After sorting the data records in ascending order, the symbol changes to for *Sort descending*.

If you enter a value in the *Quantity* field and then press ENTER, the system calculates and displays the total for the article and overall total. Negative quantity values are not allowed; the generic validation service of the Java Dicitonary automatically reports an invalid context attribute message for every negative quantity value. This is based on the fact that the *Quantity* context attribute is of dictionary simple type *Quantity* with value constraint *Minimum Inclusive* = 0.