Tutorial: Implementing the UI Element RowRepeater

Applies to:

Web Dynpro for Java applications for SAP enhancement package 1 for SAP NetWeaver CE 7.1.

Summary

The tutorial describes how to implement a RowRepeater in a Web Dynpro application. The tutorial focuses on the special features of this UI element.

If you just want to check the running sample, download the complete project RowRepeater.

Author: SAP NetWeaver Product Management User Interaction, Stefanie Bacher with Marie-Fabienne Orcel, who created this tutorial during her internship in our team.

Company: SAP AG

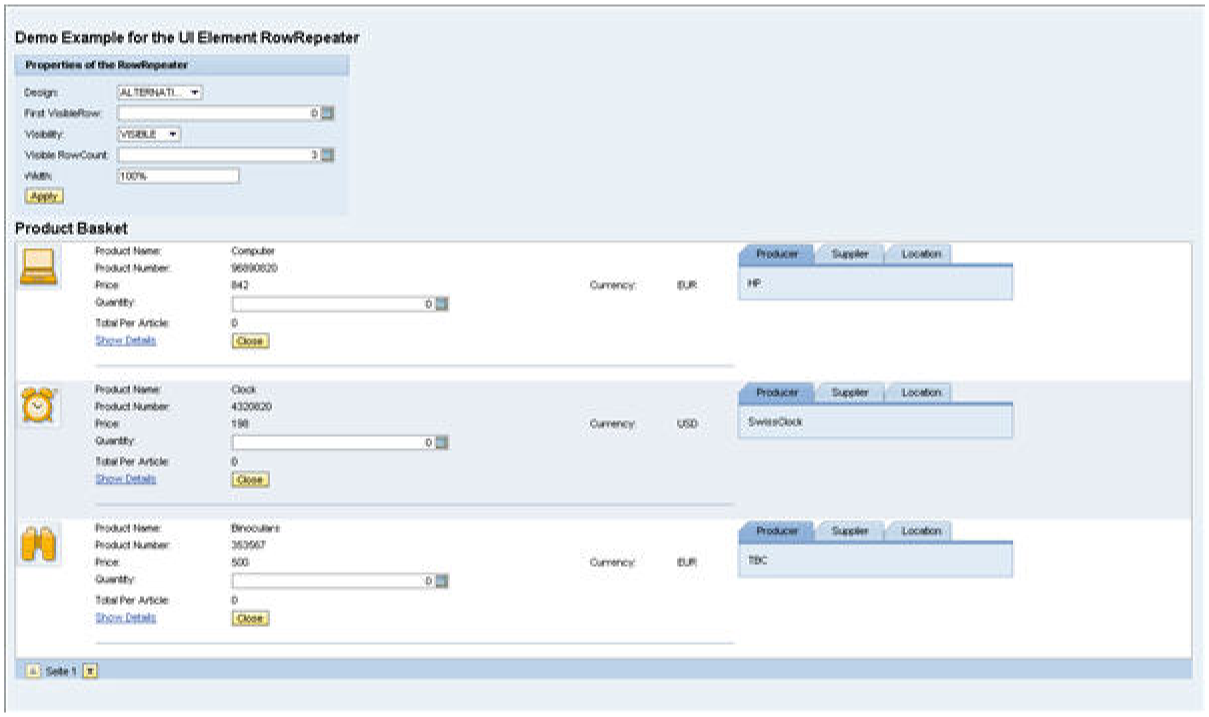
Created on: 08 January 2009

Table of Contents

Introduction

The RowRepeater is a UI element that creates a list of rows with the same layout. For each node element in the dataSource a row is painted.

This is an example of a RowRepeater:



Demo Example for the UI Element RowRepeater

Properties of the RowRepeater

Design:	ALTERNATI...
First VisibleRow:	0
Visibility:	VISIBLE
Visible RowCount:	3
Width:	100%

Apply

Product Basket

Product Name:	Computer			
Product Number:	96990620			
Price:	842	Currency:	EUR	Producer: HP
Quantity:	0			
Total Per Article:	0			
Show Details	Close			
Product Name:	Clock			
Product Number:	4320620			
Price:	198	Currency:	USD	Producer: SwissClock
Quantity:	0			
Total Per Article:	0			
Show Details	Close			
Product Name:	Binoculars			
Product Number:	353567			
Price:	500	Currency:	EUR	Producer: TBC
Quantity:	0			
Total Per Article:	0			
Show Details	Close			

In this tutorial, you will learn how to

- Design a view layout using the RowRepeater
- Implement a RowRepeater considering its specialties
- Implement an action that shows detailed information of a selected row

Creating the Context for the RowRepeater Data

To provide the RowRepeater with data, we must first store that data in the context of the *Component Controller*.

1. Add a new Node to the context root node and name it **Products**. The *Collection Cardinality* of *Products* node should be left at the default value of [0...n].
2. Now, manually add attributes to this node and set the data types according to the table below. Since *Total_Per_Article* should be a *calculated* attribute, select the *calculated* checkbox to true. This causes the NWDS to generate the appropriate Accessor/Mutator methods.

Attribute Name	Type	Calculated	Read-only
Currency	<currency>	no	false
Location	<string>	no	false
Picture	<string>	no	false
Price	<decimal>	no	false
Producer	<string>	no	false
ProductName	<string>	no	false
ProductNumber	<string>	no	false
Quantity	<integer>	no	false
Supplier	<string>	no	false
Total_Per_Article	<decimal>	yes	false

Creating the Context for the RowRepeater Properties

To enable the user to modify the layout of the RowRepeater, attributes, which will later be bound to the properties of the RowRepeater, are added to the context of the *Component Controller*.

1. Add the following attributes to the context root node

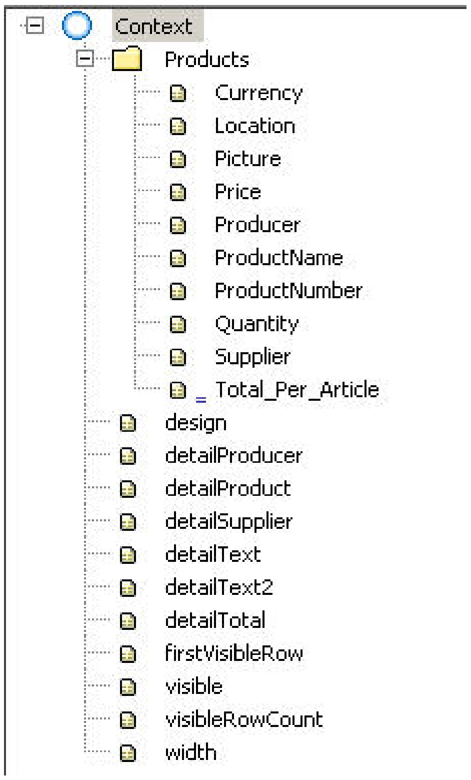
Attribute Name	Type	Calculated	Read-only
design	<RowRepeaterDesign>	no	false
firstVisibleRow	<integer>	no	false
visible	<Visibility>	no	false
visibleRowCount	<integer>	no	false
width	<string>	no	false

Note: The attributes *design* and *visible* have to be of type “Simple Type” of the package “UIElementDefinitions”.

Additionally some other attributes enhance the context and are put to the context root node. They are needed to display later detailed information of a selected row in another section.

Attribute Name	Type	Calculated	Read-only
detailProduct	<string>	no	false
detailProducer	<string>	no	false
detailSupplier	<string>	no	false
detailText	<string>	no	false
detailText2	<string>	no	false
detailTotal	<decimal>	no	false

The context of the Component Controller looks now like this:



Implementing additional functionality

As the *Total_Per_Article* attribute exists, it is used to calculate the total price per row.

1. Open the Java Editor of the *Component Controller* in the project structure, go to the section `//@@begin getProductsTotal_per_Article and //@@end` tags. Following code should be implemented:

```
// Method to get the total price per article by multiplying the price and the
quantity
 java.math.BigDecimal sum = null;
 if (
      element.getQuantity () < 0)
      {//ensure that no negative prices can be calculated
      sum = new BigDecimal (0);
      }
else
      {//calculate total
      sum = new BigDecimal (element.getQuantity ()).multiply (element.getPrice
      ());
      }
return (sum);
```

Providing some data

In order for the RowRepeater to display any data, we must first write some code to generate that data.

1. Open the Java Editor of the Component Controller and go to the section `//@@begin wdDoInit()` `and //@@end` tags. The following coding must be implemented.

```
IPrivateRowRepeater_ApplComp.IProductsElement elem;

        elem = wdContext.createAndAddProductsElement();
        elem.setProductName("Computer");
        elem.setProductNumber("96890820");
        elem.setPicture("Picture1.gif");
        elem.setPrice(new BigDecimal (842));
        elem.setProducer("HP");
        elem.setSupplier("Schenker");
        elem.setLocation("Mannheim, Germany");
        elem.setCurrency("EUR");
        elem = wdContext.createAndAddProductsElement();
        elem.setProductName("Clock");
        elem.setProductNumber("4320820");
        elem.setPicture("Picture2.gif");
        elem.setPrice(new BigDecimal (198));
        elem.setProducer("SwissClock");
        elem.setSupplier("DB");
        elem.setLocation("Palo Alto, US");
        elem.setCurrency("USD");
        elem = wdContext.createAndAddProductsElement();
        elem.setProductName("Binoculars");
        elem.setProductNumber("353567");
        elem.setPicture("Picture3.gif");
        elem.setPrice(new BigDecimal (500));
        elem.setProducer("TBC");
        elem.setSupplier("Meyers");
        elem.setLocation("Walldorf, Germany");
        elem.setCurrency("EUR");
        elem = wdContext.createAndAddProductsElement();
        elem.setProductName("Compass");
        elem.setProductNumber("095254");
        elem.setPicture("Picture4.gif");
        elem.setPrice(new BigDecimal (45));
        elem.setProducer("Nokia");
        elem.setSupplier("Schenker");
        elem.setLocation("Paris, France");
        elem.setCurrency("EUR");

        int count = 1;
        for (int i = 1; i < 25; i++){
                elem = wdContext.createAndAddProductsElement();
                elem.setProductName("ProductName" + i);
                elem.setProductNumber("Product Number #" + i);

                if(count > 4)  { count = 1;
                 }
                count++;
                elem.setPicture("Picture"+count+".gif");
                elem.setPrice(new BigDecimal (342 + i*5));
                elem.setProducer("Producer" +i);
                elem.setSupplier("Supplier" + i);
                elem.setLocation("Mannheim, Germany");
                if(i%2 == 0){ elem.setCurrency("USD");
                        }
                else{ elem.setCurrency("EUR");
                         }

                }
```

Note: The images used in this example are stored in the Mime Repository in directory <project name>/src/mimes/Components/<component name>; so that they can be called by their name (e.g. “Picture1.gif”).

Mapping the View Context onto the Component Controller Context

In order to make the data held in the context of the Component Controller available to the View Controller, we need to tell the View Controller *RowRepeater_View* that it should use the Component Controller as a data source. Once this has been done, the data in the Component Controller’s context is available to the View Controller’s context through a technique known as **Context Mapping.**

1. Double-click the component *RowRepeater_ApplComp* and create a data link between the *RowRepeater_View* icon and the Component Controller icon.
2. By using drag and drop, drag the *Products* node and the context attributes from the Component Controller context on the right (*RowRepeater_ApplComp)* to the left context root node (*RowRepeater_View*). Confirm with *Finish*.

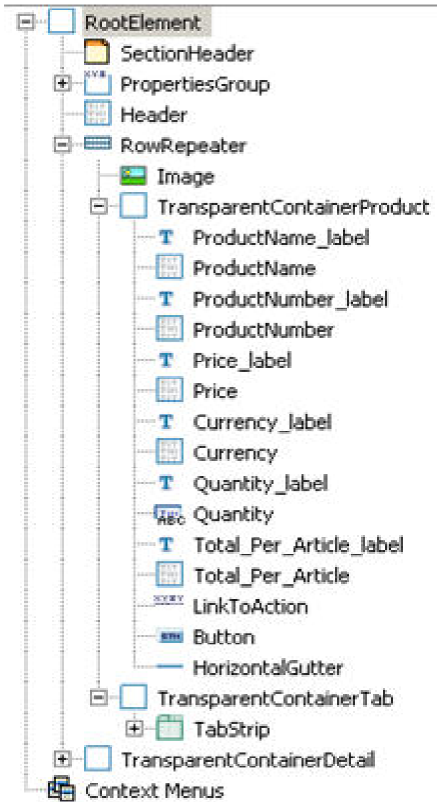
Designing the view layout with the RowRepeater

In this example a PropertiesGroup is designed which allows the user to modify the layout of the RowRepeater. Then the RowRepeater with its RowElements is designed, it displays the data hold in the *Products* node.

To provide more information about a product in a row, it is also necessary to design a section below the RowRepeater that shows detailed information.

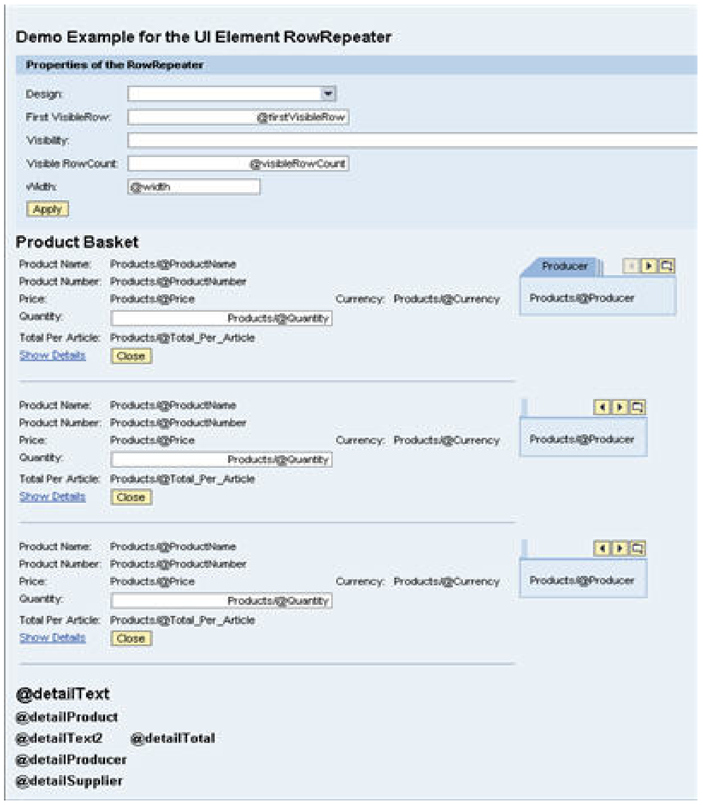
This is an example how a RowRepeater with its RowElements can be designed:

Outline View



```
RootElement
  SectionHeader
  PropertiesGroup
  Header
  RowRepeater
    Image
    TransparentContainerProduct
      ProductName_label
      ProductName
      ProductNumber_label
      ProductNumber
      Price_label
      Price
      Currency_label
      Currency
      Quantity_label
      Quantity
      Total_Per_Article_label
      Total_Per_Article
      LinkToAction
      Button
      HorizontalGutter
    TransparentContainerTab
      TabStrip
  TransparentContainerDetail
Context Menus
```

View Layout



Demo Example for the UI Element RowRepeater

Properties of the RowRepeater

Design:
First VisibleRow: @firstVisibleRow
Visibility:
Visible RowCount: @visibleRowCount
Width: @width
Apply

Product Basket

Product Name: Products.@ProductName
Product Number: Products.@ProductNumber
Price: Products.@Price Currency: Products.@Currency
Quantity: Products.@Quantity
Total Per Article: Products.@Total_Per_Article
Show Details Close
Producer
Products.@Producer

Product Name: Products.@ProductName
Product Number: Products.@ProductNumber
Price: Products.@Price Currency: Products.@Currency
Quantity: Products.@Quantity
Total Per Article: Products.@Total_Per_Article
Show Details Close
Products.@Producer

Product Name: Products.@ProductName
Product Number: Products.@ProductNumber
Price: Products.@Price Currency: Products.@Currency
Quantity: Products.@Quantity
Total Per Article: Products.@Total_Per_Article
Show Details Close
Products.@Producer

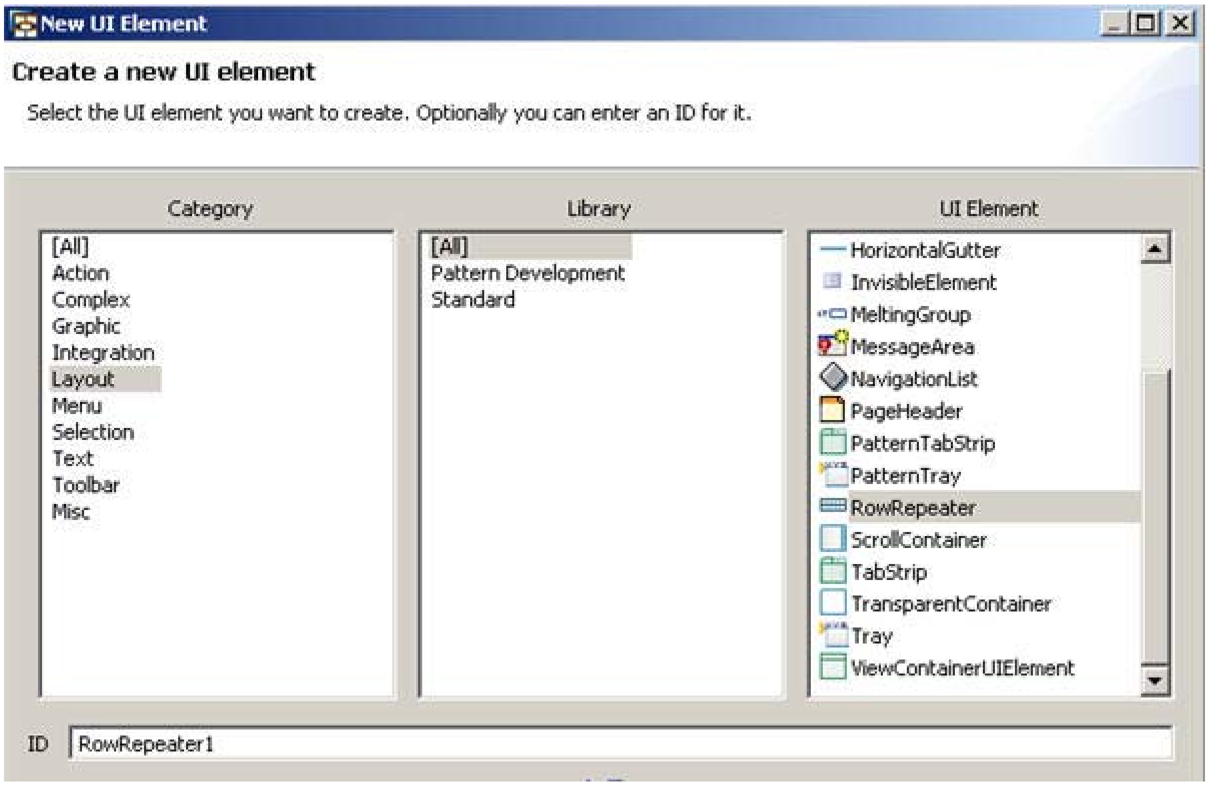
@detailText
@detailProduct
@detailText2 @detailTotal
@detailProducer
@detailSupplier

Designing the PropertiesGroup

1. Open the *Outline View* of the *RowRepeater_View*
2. Remove the *DefaultTextView* element
3. Insert a *Group* container in the *RootElement* container. Make a right click on the *Group* container and choose *"Apply Template", "Form"* and then select the context attributes: *design, firstVisibleRow, visible, visibleRowCount, width*. Confirm with *Finish*.
4. To provide a header for the whole view, insert a *SectionHeader* in the *RootElement* container.

Designing the RowRepeater

1. Insert the **RowRepeater** in the *RootElemen*t container


New UI Element

Create a new UI element

Select the UI element you want to create. Optionally you can enter an ID for it.

ID RowRepeater1

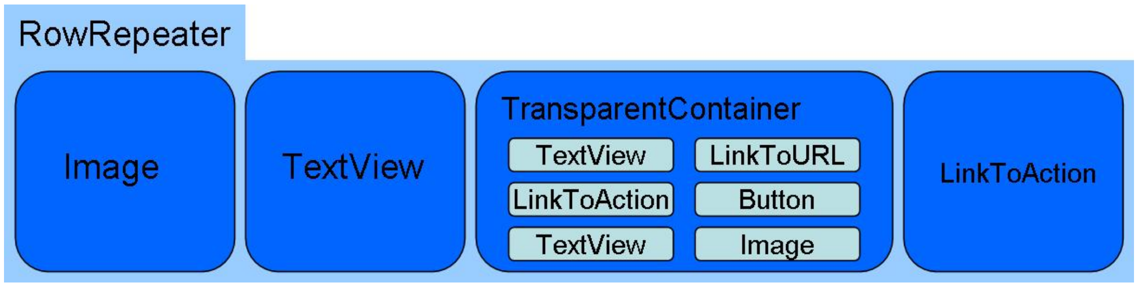
2. Now you can insert **RowElements** in the RowRepeater

Supported elements are the *TransparentContainer*, *Images*, *TextView*, *InputField*, Links (*LinkToAction*, *LinkToURL*), *Buttons* and *TabStrips*.

Each UI element in the RowRepeater is laid out in own column.

The RowRepeater is displayed in a MatrixLayout. For each RowElement a MatrixCell is used and a MatrixData with default values is assumed.

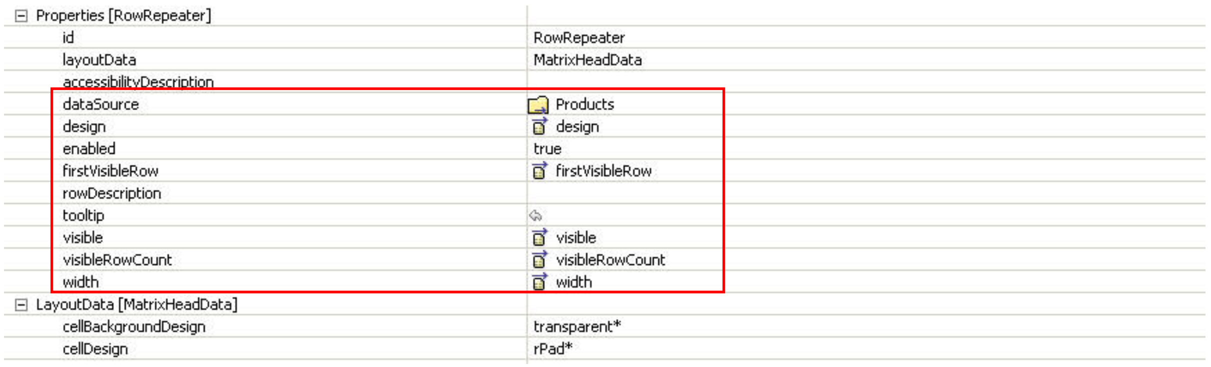
Note: If the RowElement is a TransparentContainer with MatrixLayout and scrollingMode = none, it will be treated different. The children of the TransparentContainer are now handled as cells of its MatrixLayout.



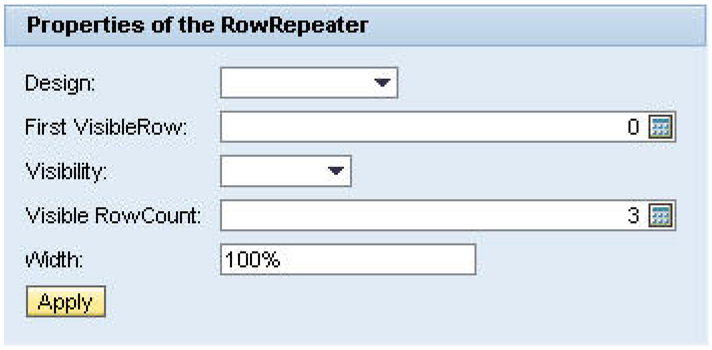
3. Insert in the RowRepeater an *Image*, bind the attribute *Product.Picture* to its source property
4. Insert a *TransparentContainer* and change the layout to *MatrixLayout* make a right click and choose *"Apply Template", "Form"* and then select following attributes of the *Products* node: *ProductName, ProductNumber, Price, Currency, Quantity and Total_Per_Article*. Click *Next* and change the Editor property of the attributes to *TextView* beside *Quantity*. Additionally insert a *LinkToAction*, a *Button* and a *HorizontalGutte*r in the *TransparentContainer*. All elements are now handled as cells of the *TranparentContainer*.
5. Insert another *TransparentContainer* in the RowRepeater, insert a *TabStrip*. Make a right click on the *TabStrip* and choose "*InsertTab*". Repeat this step two times.
6. Go to *Tab_Header* of the first Tab and write in the *text* property "Producer". Insert in the *TabContent* of this *Tab* a *TextView* and bind the *text* property to the Products attribute *Products.Producer*. Repeat this step for the other *Tab_Headers* with "Supplier" and "Location" and bind the *text* properties respectively to *Products.Supplier* and *Products.Location*.

Set the Properties of the RowRepeater

1. Go to the *Properties* tab of the RowRepeater. It is mandatory to bind the property *dataSource*. It is bound to the node that contains the run-time data, in this case to the *Products* node.
2. The other properties in this example are also bound to attributes of the context, so that the user can modify the layout of the RowRepeater as favored.



Properties [RowRepeater]	
id	RowRepeater
layoutData	MatrixHeadData
accessibilityDescription	
dataSource	Products
design	design
enabled	true
firstVisibleRow	firstVisibleRow
rowDescription	
tooltip	
visible	visible
visibleRowCount	visibleRowCount
width	width
LayoutData [MatrixHeadData]	
cellBackgroundDesign	transparent*
cellDesign	rPad*



Properties of the RowRepeater

Design:	
First VisibleRow:	0
Visibility:	
Visible RowCount:	3
Width:	100%

Implications of the properties

- **Design**: The layout of the RowRepeater; two possibilities: ***TRANSPARENT*** or ***ALTERNATING***.
- **FirstVisibleRow**: With this property the user can decide which row should be at the first position, i.e. which element is displayed first; it refers to the index of elements hold in the context and starts with 0.
- **Visibility:** The different types of visibility can be chosen: ***BLANK*** (UI element with visibility BLANK is invisible but takes view space, Using this visibility might reveal sensitive data), ***NONE*** (A UI element with visibility NONE is invisible and takes no view space. It cannot be made visible by end user or role personalization.), ***VISIBLE*** (A UI element with visibility VISIBLE is visible in its view), ***ALWAYS*** (A UI element with visibility ALWAYS is always visible in its view and cannot be hidden by end user or role personalization.), ***NOT_YET*** (A UI element with visibility NOT_YET is not yet visible in its view (i.e. treated like NONE) until it is made visible by end user or role personalization. This can be used to ship hidden parts of a screen which can then be made visible on demand.)
- **VisibleRowCount:** Defines how many rows are visible, i.e. how many elements.
- **Width:** Modifies the width of the RowRepeater; in can be defined in pixel (“100”) or in percentage (“100 %”).

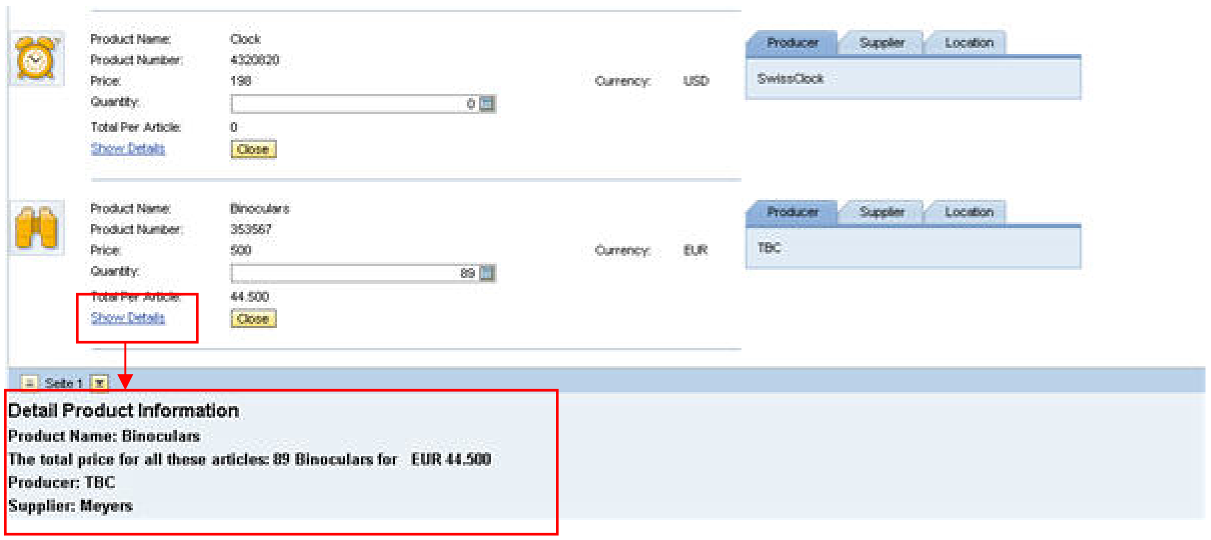
Note: It is recommended that the RowRepeater is not included in another layout (e.g. *Group*), in order that modifications of the “*width”* -property are not affected.

Designing the Detail Section

1. Insert a *TransparentContainer* in the *RootElement* container. Change the layout to *MatrixLayout*.
2. Insert six *TextViews* in the *TransparentContainer*. They will later display the detailed information of a selected element.
3. Bind the first *TextView* to the context attribute *detailText*. Repeat this step for the other *TextViews* and bind them respectively to the context attributes: *detailProduct, detailText2, detailTotal, detailProducer* and *detailSupplier*.

Action

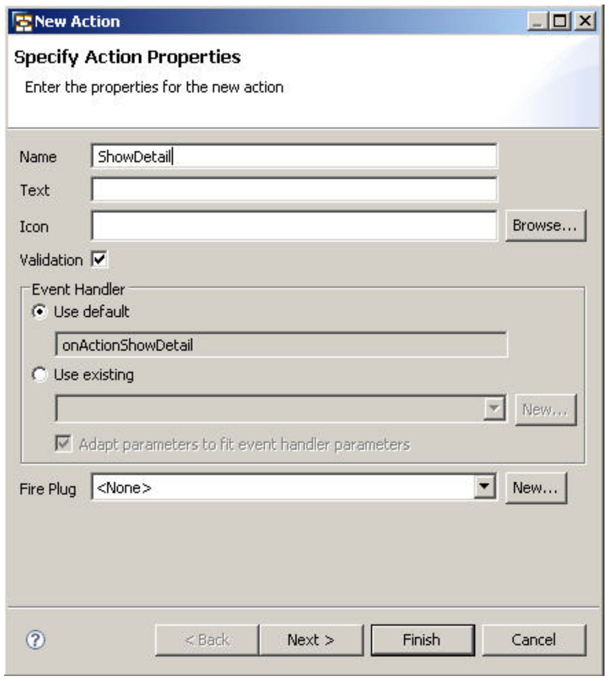
When the user clicks the *“Show Details”* link, a detail section below the RowRepeater will be shown. In order that only the data of the current selected row will be display, **parameter mapping** is necessary. When the user selects the link, the action event handler method in the view controller needs to know exactly which row has been selected to display the data of the current element.



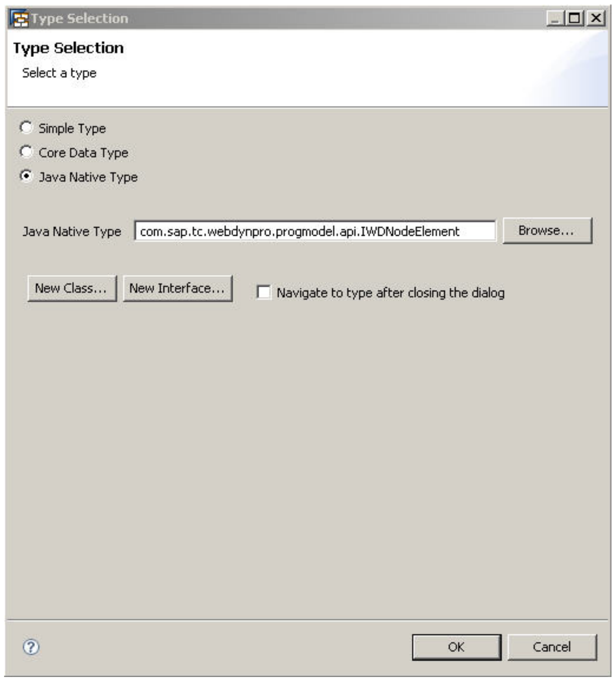
Detail Product Information
Product Name: Binoculars
The total price for all these articles: 89 Binoculars for EUR 44.500
Producer: TBC
Supplier: Meyers

Create Action ShowDetails

1. Open the *RowRepeater_View* with double-click
2. Create a new action *ShowDetails*
3. Click next and add a Parameter *element* of type "*IWDNodeElement*"; therefore when specifying the parameter properties chose *Browse,* then *Java Native Type.* Select *Browse* and enter *IWDNodeElement.* Confirm with *OK*


New Action
Specify Action Properties
Enter the properties for the new action
Browse...
Event Handler
New...
< Back
Next >
Finish
Cancel

Name: ShowDetail
Text:
Icon:
Validation ✓
Use default: onActionShowDetail
Use existing
Adapt parameters to fit event handler parameters ✓
Fire Plug: <None>

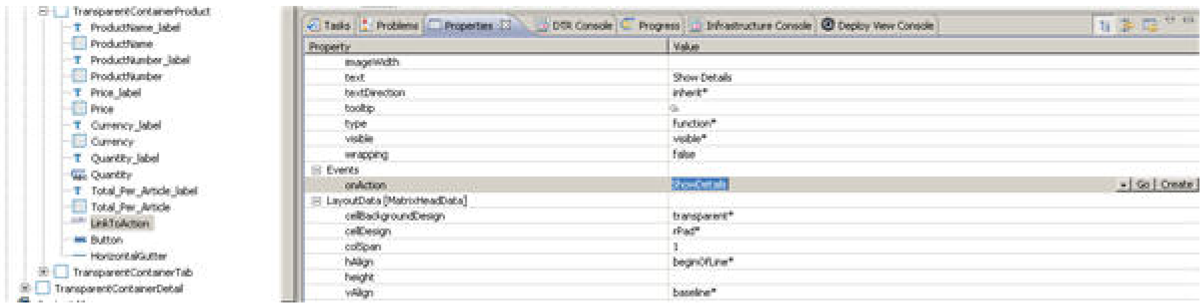


Simple Type
Core Data Type
[✓ Java Native Type]
Java Native Type: com.sap.tc.webdynpro.progmodel.api.IWDNodeElement
Navigate to type after closing the dialog

Binding the ShowDetails action

In order to make the *LinkToAction* UI element aware that it has to show detailed information, you must associate the *onAction* event with the *ShowDetail*s action.

1. Select in the *Outline View* the *LinkToAction* UI element. Switch to the Properties tab; in the drop down list for the *onAction* event, select the *ShowDetails* action.

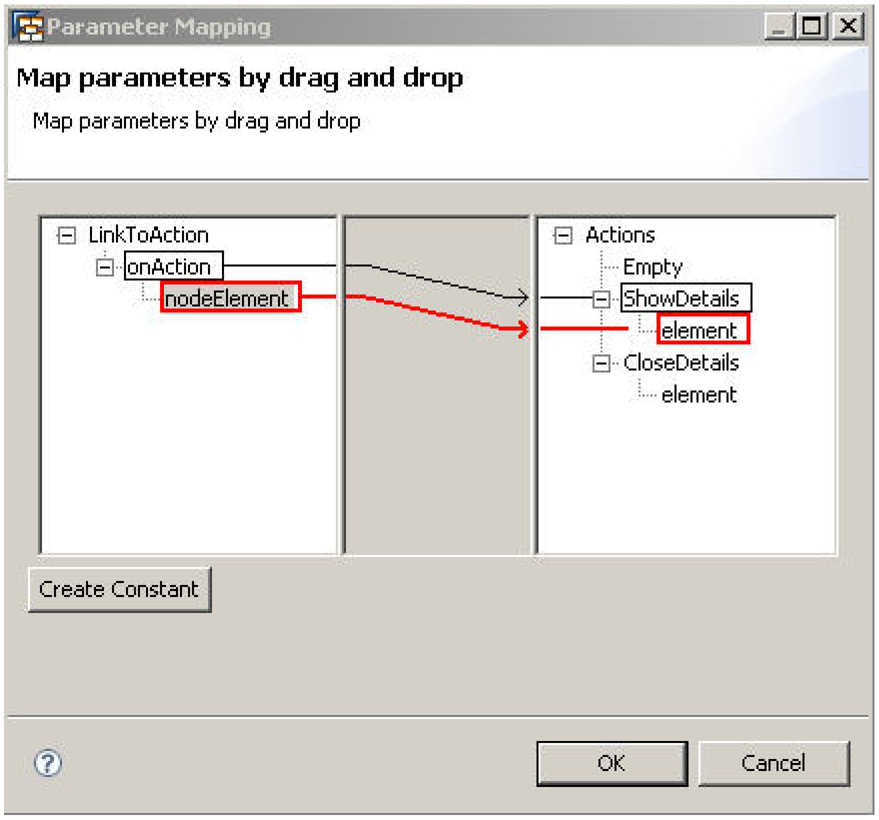


Property	Value
imageWidth	
text	Show Details
textDirection	inherit*
tooltip	
type	function*
visible	visible*
wrapping	false
Events	
onAction	ShowDetails
LayoutData [MatrixHeadData]	
cellBackgroundDesign	transparent*
cellDesign	rPad*
colSpan	1
hAlign	beginOfLine*
height	
vAlign	baseline*

Parameter Mapping

Parameter Mapping is needed to add a parameter to an event and transfer a value to the corresponding event handler. The *nodeElement* parameter references the corresponding node element in the context to which the UI element is assigned. When you click the *LinkToAction* in the *RowRepeater* this parameter can be used to determine in which row the link was pressed, so that only the detailed information of the current selected element is displayed.

1. Make a right click on the *LinkToAction* UI element in the *Outline View* and chose Parameter Mapping
2. Select the action's *nodeElement* parameter and drag and drop it to the event parameter *element* for the UI element.



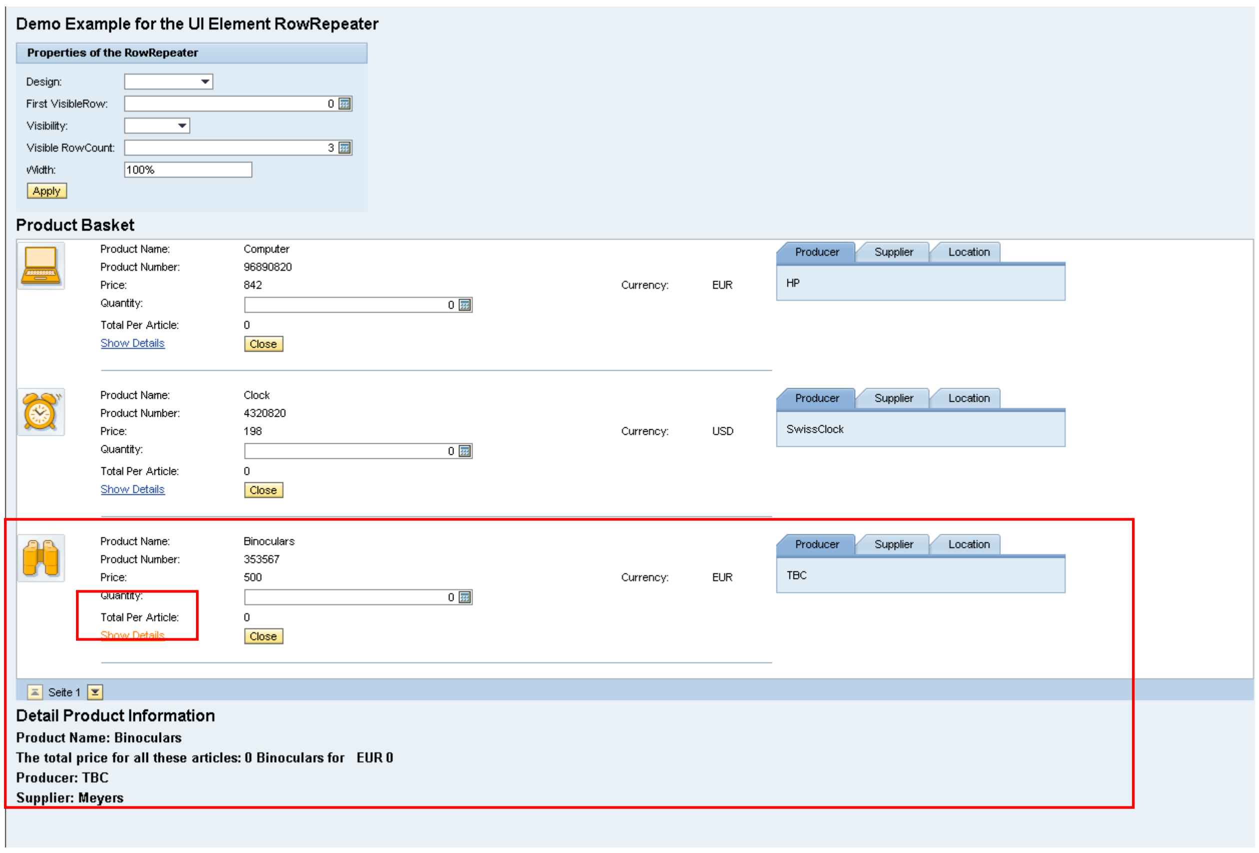
Showing detailed information

To display more information about a selected element, some functionality must be implemented in the *ShowDetails* action.

1. Open the Java Editor for the RowRepeater_View and go to the section `//@@begin onActionShowDetails(ServerEvent) and //@@end` tags
2. The following code must be inserted

```
// Get detailed information of a node element to display the data in a special
section
wdContext.currentContextElement ().setDetailText ("Detail Product
      Information");
wdContext.currentContextElement ().setDetailProduct ("Product Name:
      "+ (String) element.getAttributeValue ("ProductName"));
wdContext.currentContextElement ().setDetailText2 ("The total price for all
      these articles: "+ element.getAttributeValue ("Quantity") +
      "  "+  (String)  element.getAttributeValue  ("ProductName")  +"  for     "+
      (String) element.getAttributeValue ("Currency"));
wdContext.currentContextElement ().setDetailTotal ((BigDecimal) element.
      getAttributeValue ("Total_Per_Article"));
wdContext.currentContextElement ().setDetailProducer ("Producer:
      "+ (String) element.getAttributeValue ("Producer"));
wdContext.currentContextElement ().setDetailSupplier ("Supplier:
      "+ (String) element.getAttributeValue ("Supplier"));
```

The result is that only the detailed information of an element in the current row is displayed in the section below the RowRepeater.


Demo Example for the UI Element RowRepeater

Properties of the RowRepeater

Design:	
First VisibleRow:	0
Visibility:	
Visible RowCount:	3
Width:	100%

Product Basket

```
Product Name:       Computer                               Producer | Supplier | Location
Product Number:     96890820                               HP
Price:              842          Currency:    EUR
Quantity:           0
Total Per Article:  0
Show Details        Close

Product Name:       Clock                                  Producer | Supplier | Location
Product Number:     4320820                                SwissClock
Price:              198          Currency:    USD
Quantity:           0
Total Per Article:  0
Show Details        Close

Product Name:       Binoculars                             Producer | Supplier | Location
Product Number:     353567                                 TBC
Price:              500          Currency:    EUR
Quantity:           0
Total Per Article:  0
Show Details        Close
```

Detail Product Information

Product Name: Binoculars

The total price for all these articles: 0 Binoculars for   EUR 0

Producer: TBC

Supplier: Meyers

Create Action CloseDetails

This action is triggered when the user chooses the button *Close*, then the detailed information of a selected element in the section below will not more displayed.

This action is implemented as the action *ShowDetails*, the only difference is the implementation of the *CloseDetails* action.

1. Open the Java Editor for the RowRepeater_View and go to the section `//@@begin onActionCloseDetails(ServerEvent) and //@@end` tags
2. The following code must be inserted

```
// Close the detailed information
wdContext.currentContextElement ().setDetailText ("");
wdContext.currentContextElement ().setDetailText2 ("");
wdContext.currentContextElement ().setDetailProduct ("");
wdContext.currentContextElement ().setDetailTotal (null);
wdContext.currentContextElement ().setDetailProducer ("");
wdContext.currentContextElement ().setDetailSupplier ("");
```

The result is that detailed information of an element in the current row will not more be displayed in the section below the RowRepeater.

Related Content

UI Element Guide: RowRepeater

Parameter Mapping

For more information, visit the User Interface Technology homepage.

Copyright