Developing State-of-the-Art Table User Interfaces in Web Dynpro Java

SAP NetWeaver 7.0

Bertram Ganz, SAP NetWeaver UI Foundation OPS
Learning Objectives

- Provide an overview of the new Web Dynpro table functions in SAP NetWeaver 7.0
- Layout Web Dynpro tables
- Use cell variants
- Size tables using absolute and relative widths
- Explain grouped table headers
- Talks about some of the new table functions in SAP NetWeaver CE 7.1
Web Dynpro Table UI Elements
Designing Tables – Sizing, Layouting, Grouping
Further Notes and Topics
Outlook – Table in SAP NetWeaver CE 7.1
Summary
Web Dynpro Table UI Elements

New Feature Overview

Web Dynpro Table Metamodel

Using Table Cell Variants
Table Functions in SAP NetWeaver 7.0

Grouping Values
- Table cells can be merged on the basis of their grouping value.

Hierarchical Column Headers (See Exercise 2)
- Headers can be arranged in a hierarchy.

Horizontal Scrolling
- A table can display fewer columns than nested in the table.
- Scroll to invisible columns.
Table Functions in SAP NetWeaver 7.0

Fix Table Columns Right or Left Aligned
- You can also fix columns aligned to the right or left.
- All scrollable columns should have the same widths to avoid resizing of the table after every scrolling action.

Table Cell Variants (See Exercise 1)
- Use different table cell editors in the same table column for different table rows.
Table Functions in SAP NetWeaver 7.0

**Fixed Top Table Cell**
- Position cell editor(s) at fixed position on top of the table rows
- Replace the generic table filter row with custom filter editors like checkboxes, dropdowns

**Fixed Bottom Table Cell**
- Position cell editor(s) at fixed position at bottom of displayed table rows
Table Functions in SAP NetWeaver 7.0

Fixed Table Layout

- Fix table layout to predefined sizes
- Avoid *jumping* table columns based on differing text lengths in different table rows
- Hidden content gets displayed via tooltip

Table Grid Modes

- Explicitly show or hide borders of table columns and table rows
Table Functions in SAP NetWeaver 7.0

**Table Row Popin**
- Insert popin container displayed between table rows
- Popins refers to the whole table row

**Table Column Popin (also named Cell Popin)**
- Insert popin container displayed between table rows
- Popin refers to a specific cell in a table column
- Associated cell is assigned the background color of the popin
Further Table Enhancements in SAP NetWeaver 7.0

Enhanced Table Selection Behavior

- Support multiple selection without changing the lead selection with property `selectionMode=multiNoLead`
- Define selectable and non-selectable table rows with `Table UI Element` property `rowSelectable`

Enhanced Table Eventing

- `onFilter` Table event: IWDTable.IWDOnSort
- `onSort` Table event: IWDTable.IWDOnFilter
- Parameter enrichment for table’s `onLeadSelectEvent`: `newRow`, `oldRow`

Further Table Enhancements

- Display Empty Table Text
- Custom extension columns for SAP NetWeaver Portal based Web Dynpro applications using an Adaptive RFC Model.
Table UI Element Associations

Association Name: IWDTTable

Association Cardinality: 0..1

Association Source UI Element: IWDTTable

Association Target UI Element: IWDCaption
Adding Table UI Element Associations

UI element association “Column” is deprecated. Use “GroupedColumn” instead.
Abstract APIs

IWDAbstractMasterTableColumn
IWDAbstractTableCellVariant
IWDAbstractTableColumn
IWDAbstractTreeTableColumn

View Element APIs

IWDTable
IWDTableColumn
IWDTableCellEditor
IWDTableColumnGroup
IWDTableMarkableCellEditor
IWDTablePopin
IWDTablePopinToggleCell
IWDTableSingleMarkableCell
IWDTableStandardCell
IWDTreeByNestingTableColumn

WDTableMethods

Helper Class

WDTableMethods.IWDTableMethods

Enumerations

WDTableCellDesign
WDTableColumnFixedPosition
WDTableColumnHAlign
WDTableColumnSortDirection
WDTableCompatibilityMode
WDTableDesign
WDTableGridMode
WDTablePopinTitleDesign
WDTableSelectionChangeBehaviour
WDTableSelectionMode

Events

IWDAbstractTreeTableColumn
 .IWDOnLoadChildren
IWDAbstractTableColumn
 .IWDOnAction
IWDTable.IWDOnSort
IWDTable.IWDOnLeadSelect
IWDTablePopinToggleCell
 .IWDOnToggle
Using Table Cell Variants

With the 0:n association **CellVariant** of the **TableColumn** UI element, you can display *different* cell editors in the *same* table column.

---

**Table Cell Variant**
- = InputField

**Table Cell Variant**  
- = TextView

**Table Cell Design**
In a typical scenario you define an InputField UI element as the standard cell editor and a TextView UI element for the associated cell variant of type TableStandardCell.
1 Define Required Context Elements

- Every table node element must be enriched with a context attribute storing the selected cell variant ID per node element.
- Add non-singleton node CellInfo with cardinality 1..1 to (mapped) table data node.
- Add calculated context attribute of type String to get variantKey per node element (row) at runtime.

2 Add UI Elements to View Layout

- Add CellVariant of type Table-StandardCell to Table Column.
- Add cell editor of type TextView to TableStandardCell cell variant.
Adding a Cell Variant to a Table Column (2)

3. Assign Variant Key to Added Cell Variant
   - Every cell variant added to a table column must be specified with a corresponding variant key.
   - Set property `variantKey` for the added cell variant of type `TableStandardCell`.
   - Example: `variantKey = SOLD`

4. Define Binding Relation from TableColumn UI Element to View Context
   - Property `selected-CellVariant` of the `TableColumn` UI element defines the cell variant to be displayed per table row.
   - Bind this property to the calculated context attribute `SelCellVariant`. 
Table Cell Variants and Context Binding

**Table in View Layout**

- ShoppingBasket [Table - Child]
  - ShoppingBasketHeader [Caption - Header]
  - Toolbar [Toolbar - Toolbar]
- QUANTITY [TableColumn - GroupedColumn]
  - QUANTITY_header [Caption - Header]
- QUANTITY_editor [InputField - TableCellEditor]
- TableStandardCell [TableStandardCell - CellVariant]
- TextView [TextView - Editor]

**Context**

- **TableDataNode**
  - Article
  - IsSold
- **CellInfo**
- **SelCellVariantCalc** (calculated attribute)

**Design time**

**Runtime**

- **TableDataNode**
  - **CellInfo**
    - jacket
    - true
    - SOLD
  - **CellInfo**
    - skirt
    - false

**variantKey = SOLD**

**selectedCellVariant**

**Table in View UI**

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Article</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sold</td>
<td>jacket</td>
<td>blue</td>
</tr>
<tr>
<td></td>
<td>skirt</td>
<td>red</td>
</tr>
</tbody>
</table>

© SAP AG 2007, SAP NetWeaver F UI OPS / Bertram Ganz / 19
5. Implement Calculated Context Attribute Getter Method in View Controller

- Open view controller “Implementation” tab.
- Implement calculated context attribute getter to retrieve variant keys per node element (table row).
- For a calculated context attribute `CellInfo.SeCellVariant` this getter method is named `getCellInfo-SeCellVariant()`.
- Returned variant keys must match the defined cell variant keys (see step 3).
- Use empty String “” as key for standard cell editor.
public java.lang.String getCellInfoQUANTITY_SelCellVariant(IPrivateTableCellVariantsView.ICellInfoElement element) {

    //@@begin getCellInfoQUANTITY_SelCellVariant(
    //IPrivateTableCellVariantsView.ICellInfoElement)

    IPrivateKeyTableCellVariantsView.IProductsElement prodEl =
    (IPrivateTableCellVariantsView.IProductsElement) element
    .node().getParentElement();

    return prodEl.getISSOLD()?
    MyCellVariantKey.SOLD.toString():
    "" // if VariantKey does not exist
    ;

    //@@end

}
Table Cell Designs When Using Cell Variants

With the property `cellDesign` defined on `TableColumn` UI Element and `table cell variant` (TableStandardCell UI Element) level you can change the design of the displayed table cells and consequently table rows.

How can table rows have the same cell design when using table cell variants?

Same table cell design for all cells in same table row
Define Required Calculated Context Attribute

- Every table node element must be enriched with a context attribute storing the selected cell design per node element.
- Add non-singleton node CellInfo with cardinality 1..1 to (mapped) data node.
- Add calculated context attribute of dictionary simple type `com.sap.ide.webdynpro.uielementdefinitions.TableCellDesign` to get cell design per node element (table row) at runtime.

**Data Type**

- calculated context attribute *(readOnly = true)*
2 Changing Cell Designs for Different Table Columns (2)

Define Required Data Binding Relations

- Bind property `cellDesign` of all `TableColumn` UI elements to the same calculated context attribute `SelectedCellDesignCalc`.
- Bind property `cellDesign` of associated table cell variants (`TableStandardCell UI Elements`) to the same calculated context attribute `SelectedCellDesignCalc`.
Changing Cell Designs for Different Table Columns (3)

3 Implement Calculated Attribute Getter in View Controller

- Calculate cell design value of type `WDTableCellDesign` for every table cell.
- All different `cellDesign` properties (see step 1) are bound to the same calculated attribute → calculated attribute getter method is invoked for every displayed table cell, or for every context attribute of all displayed table node elements.
- Make sure, that the calculated attribute getter returns the same cell design value for the same node element.

```java
public com.sap.tc.webdynpro.clientserver.uielib.standard.api.WDTableCellDesign getCellInfoSelectedCellDesignCalc(IPrivateTableCellVariantsView.ICellInfoElement element) {
    //@@begin getCellInfoSelectedCellDesign(
    //  IPrivateTableCellVariantsView.ICellInfoElement)

    IPrivateTableCellVariantsView.IProductsElement prodEl =
        (IPrivateTableCellVariantsView.IProductsElement) element.
        node().getParentElement();

    return prodEl.getISSOLD() ? WDTableCellDesign.BADVALUE_LIGHT :
        WDTableCellDesign.STANDARD;

    //@@end
}
```
EXERCISE 1

Using Table Cell Variants

20 Minutes
Designing Tables – Sizing, Layouting, Grouping

Sizing and Layouting Table UIs

Grouping Table Columns
# Sizing and Layouting Tables – Use Case Scenarios

<table>
<thead>
<tr>
<th>Use Case Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1. How to size tables with absolute widths</strong></td>
</tr>
<tr>
<td>- My table width is driven by the widths of the embedded table columns</td>
</tr>
<tr>
<td>- I want to avoid varying table widths when scrolling</td>
</tr>
<tr>
<td><strong>2. How to size tables with relative widths</strong></td>
</tr>
<tr>
<td>- My table must span 100% width of its embedding container UI element</td>
</tr>
<tr>
<td>- I want to set minimum widths for all columns</td>
</tr>
<tr>
<td><strong>3. How to size tables with fixed table layout</strong></td>
</tr>
<tr>
<td>- My table must be sized with absolute not relative or minimum widths</td>
</tr>
<tr>
<td>- I want to avoid varying column width when scrolling</td>
</tr>
<tr>
<td><strong>4. How to wrap texts in TextView table cell editors</strong></td>
</tr>
<tr>
<td>- I want to reduce the column width by wrapping the texts of the displayed TextView cell editors.</td>
</tr>
<tr>
<td><strong>5. How to apply horizontal table scrolling</strong></td>
</tr>
<tr>
<td>- My table must display several columns and gets a large width even when applying text wrapping</td>
</tr>
<tr>
<td>- I want to display a subset of fixed columns and scroll the other column horizontally.</td>
</tr>
</tbody>
</table>
Relevant Properties for Sizing Web Dynpro Tables Widths

To change the widths of Web Dynpro Tables and Table Columns three specific UI element properties must be defined:

- **Table UI Element Property** `fixedTableLayout`
- **Table UI Element Property** `width`
- **TableColumn UI Element Property** `width`

Depending on the defined combination of property values different sizing results can be achieved.

1. **Table width**
   - `Table width`
   - `500px`

2. **TableColumn width**
   - `TableColumn width`
   - `150px`
   - `50%`
   - `50%`

3. **Table fixedTableLayout**
   - `Table fixedTableLayout`
   - `true`
When to Apply *Absolute* Table Widths

Set **absolute** table column widths when the table width is driven by or can increase with the displayed table columns!

- Keeps the table width independent of the embedding container width
- Table width adapts to the width of its table column content (cell editor texts, column header texts)

**NODE: Slide content only valid for Table UI property `fixedTableLayout = false`**
Absolute Table Widths are *Minimum Widths*

With *Table* UI element property `fixedTableLayout = false` all absolute table widths values are *minimum widths*.

Without defining any table width property, your table gets implicitly sized by its content

- Widths of column header texts
- Widths of cell editor content in visible rows

By defining suitable minimum widths you can reduce the probability of an implicit increase of table widths based on larger header or cell editor content widths!

**NOTE:** Slide content only valid for *Table* UI property `fixedTableLayout = false`
The type of the rendered table cell editor can affect the column width (with Table UI element property `fixedTableLayout=false`):

- In a readOnly table an `InputField` cell editor is rendered as a `TextView`.
- In an `InputField` you can scroll the “hidden” text content → the absolute table column size does not get enlarged and remains 150px.
- In a `TextView` you cannot scroll → the absolute (minimum) table column size gets enlarged → the table width gets enlarged.
- **SOLUTION:** Set Table UI property `fixedTableLayout` to true.

**Note:** Slide content only valid for Table UI property `fixedTableLayout = false`.
Implicit Sizing of Table Widths

Absolute Widths are Minimum Widths
When to Apply *Relative* Table Widths

Set a relative table width when the table width is driven by the width of its parent container but not of the table content!

- Keeps the table width relative to the width of the embedding container
- Table width only adapts to the width of its table columns when “large” column header texts or cell content gets displayed.
Sizing Tables with Relative Widths

A relative table width is controlled by the defined width of its embedding (scroll) container.

When defining relative widths a table gets still implicitly sized by its content.

With Table UI element property fixedTableLayout = false all relative table width values are minimum widths related to the absolute width of the embedding container.

Set the Table UI element width to 100% to span the whole width of the embedding container by default!

NOTE: Slide content only valid for Table UI property fixedTableLayout = false.
Combining *Absolute* with *Relative* Table Widths (1)

You can combine *relative* and *absolute* widths in the UI elements *Table* and *TableColumn*.

- Set *relative* table width
- Set *absolute* table width
- Set some columns with *absolute* widths
- Set other columns with *relative* widths summed up to 100%

**NOTE:** Slide content only valid for *Table UI property fixedTableLayout = false*
Relative Table Widths

Mixing Relative with Absolute Widths

DEMO
Apply text wrapping of TextView cell editor content to …

- keep the column width close to the defined minimum widths, as long as cell text does not enlarge it,
- avoid or delay horizontal scrolling based on long texts.

NOTE: Text wrapping only works for text containing whitespace

NOTE: Slide content only valid for Table UI property fixedTableLayout = false
Using a Fixed Table Layout

By setting the Table UI element property `fixedTableLayout` to `true` you can achieve fixed widths instead of minimum table (column) widths.

- **PRO:** Avoid / Delay horizontal scrolling based on long texts in table
- **CON:** Hidden cell editor or column header texts are displayed in tooltips
- **CON:** Cannot be combined with cell editor text wrapping

```
NOTE: Slide content only valid for Table UI property fixedTableLayout = true
```
Fixed Table Columns and Horizontal Scrolling of Columns

Apply scrollable and fixed table columns to reduce the number of columns being displayed

- **PROS**: Display large tables with large number of columns, without ...
  - scrolling the embedding container,
  - reading cell content in tooltips based on heavily shrunk column widths

- **CONS**:
  - User must explicitly scroll columns to see “hidden” column content
  - Potential variation of displayed table columns based on varying column content widths (see note)

### Table - scrollableColCount

<table>
<thead>
<tr>
<th>Col 1</th>
<th>Col 2</th>
<th>Col 3</th>
<th>Col 4</th>
<th>Col 5</th>
<th>Col 6</th>
<th>Col 7</th>
<th>Col 8</th>
<th>Col 9</th>
<th>Col 10</th>
</tr>
</thead>
<tbody>
<tr>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
</tr>
<tr>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
<td>cell</td>
</tr>
</tbody>
</table>

**TableColumn**

- fixedPosition
  - left
  - notFixed
  - right
Table Text Wrapping
Using Fixed Table Layout
Scrolling Table Columns

DEMO
EXERCISE 2
Sizing and Layouting Table UIs
Tables columns headers can be grouped hierarchically by “inserting” `TableColumnGroup` UI elements between `Table` and `TableColumn` UI elements.
Grouped Table Columns – Metamodel

IWDCaption

IWDAbstractTableColumn

IWDTTable

IWDTTableColumn

IWDTTableCellEditor

IWDTTable-ColumnGroup

IWDLinkToURL, IWDRadioButton, IWDIImage, IWDToggleButton, IWDCheckBox, IWDButton, IWDTriStateCheckBox, IWDLink, IWDTextView, IWDFileDownload, IWDDropDownByIndex, IWDDropDownByKey, IWDFileUpload, IWDToolBarLinkToAction, IWDTTextEdit, IWDCaption, IWDToolBarLinkToURL, IWDValueComparison, IWDProgressIndicator, IWDLinkToAction, IWDInputField,
Migrating Deprecated Table UI Element Association

The UI element association Column from a Table UI element to a TableColumn UI element is deprecated:

- Do not use the Table context menu item Insert Column any more!
- Use the new association GroupedColumn instead

<table>
<thead>
<tr>
<th>Deprecated Association</th>
<th>New Association</th>
<th>Association Target UI Element</th>
</tr>
</thead>
<tbody>
<tr>
<td>Column</td>
<td>GroupedColumn</td>
<td>□ IWDTableColumn</td>
</tr>
<tr>
<td></td>
<td></td>
<td>□ IWDTableColumnGroup</td>
</tr>
</tbody>
</table>

Migrate deprecated UI element associations of the Table UI element with context function “Migrate Columns”
EXERCISE 3

Adding Grouped Table Columns

15 Minutes
Web Dynpro Table UI Elements
Designing Tables – Sizing, Layouting, Grouping
Further Notes and Topics
Outlook – Table in SAP NetWeaver CE 7.1
Summary
Further Notes and Topics (1)

Additional technical information on the Web Dynpro Table subject matter can be found in SAP Online Help and on SDN:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Tutorial</td>
<td>SDN Tutorial “Creating a Tree Structure in a Table”</td>
</tr>
<tr>
<td>Trees in Table</td>
<td>SDN Tutorial “Creating a Tree Structure in a Table”</td>
</tr>
<tr>
<td></td>
<td>Link URL: See Table Tutorial</td>
</tr>
<tr>
<td>Master Detail Table</td>
<td>SDN Tutorial “Context Programming and Data Binding”</td>
</tr>
<tr>
<td>Tutorial</td>
<td>Link URL: See Table Tutorial</td>
</tr>
<tr>
<td>Table Selection Behavior</td>
<td>SDN Article “Enhancing Table Performance”</td>
</tr>
</tbody>
</table>
### Further Notes and Topics (2)

Additional technical information on the Web Dynpro Table subject matter can be found in SAP Online Help and on SDN:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Link</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Downloading Files in Tables on-demand</strong></td>
<td>SDN Article “Uploading and Downloading Files in Web Dynpro Tables - SAP NetWeaver 7.0”</td>
</tr>
<tr>
<td><strong>Excel Export in Tables</strong></td>
<td>SDN Wiki Code Tutorial “Exporting Table Data Using On-Demand Streams - SAP NetWeaver 7.0”</td>
</tr>
<tr>
<td></td>
<td><a href="https://wiki.sdn.sap.com/wiki/x/0mQ">https://wiki.sdn.sap.com/wiki/x/0mQ</a></td>
</tr>
<tr>
<td><strong>Sorting in Tables</strong></td>
<td>SDN WebLog „Enhanced Web Dynpro Java TableSorter for SAP NetWeaver 04s“</td>
</tr>
<tr>
<td><strong>Vetoable Lead Selection Change</strong></td>
<td>Web Dynpro Runtime API JavaDoc: WDTableSelectionMode</td>
</tr>
</tbody>
</table>
Web Dynpro Table UI Elements
Designing Tables – Sizing, Layouting, Grouping
Further Notes and Topics
Outlook – Table in SAP NetWeaver CE 7.1
Summary
New Features in SAP NetWeaver CE 7.1

The UI element **Table** has been enhanced in SAP NetWeaver CE 7.1

- Multi sorting ability and visualize filtered column
- Horizontal and vertical table scrolling
- Provide scroll tip for scroll status (index scroll tips)
- Wrapping of table column headers
- Generic selection menu (Select/deselect all)
- Selectable columns

![Image of enhanced Table UI element]

**Scroll tip**

**Selection Menu**
Web Dynpro Table UI Elements
Designing Tables – Sizing, Layouting, Grouping
Interacting with Table UIs
Further Notes and Topics
Outlook – Table in SAP NetWeaver CE 7.1
Summary
Summary

- Have an overview of the new Web Dynpro table functions in SAP NetWeaver 7.0
- Know about table layouting functions
- Understand how to use cell variants
- Know how to size tables using absolute and relative widths
- Have a grasp of table grouping
- Know about some of the functions in SAP NetWeaver CE 7.1
Further Information

SAP Public Web:

Related SAP Education and Certification Opportunities:

Related SAP Content:
CD254 Developing State-of-the-Art Table UIs in WD Java - Exercises