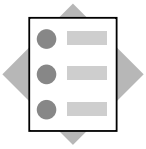
UI Elements

Contents:

- **Overview**
- **Table UI element**
- **Tree UI element (Optional)**

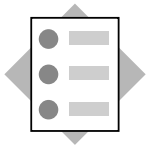
UI Elements: Objectives



After completing this lesson, you will be able to:

- **Modify and create UI elements.**
- **Create table UI elements.**
- **Create a tree UI element.**

UI Elements Overview: Topic Objectives



After completing this topic, you will be able to:

- **Explain which UI elements are available.**
- **Use the View Designer to create UI elements.**

UI Controls

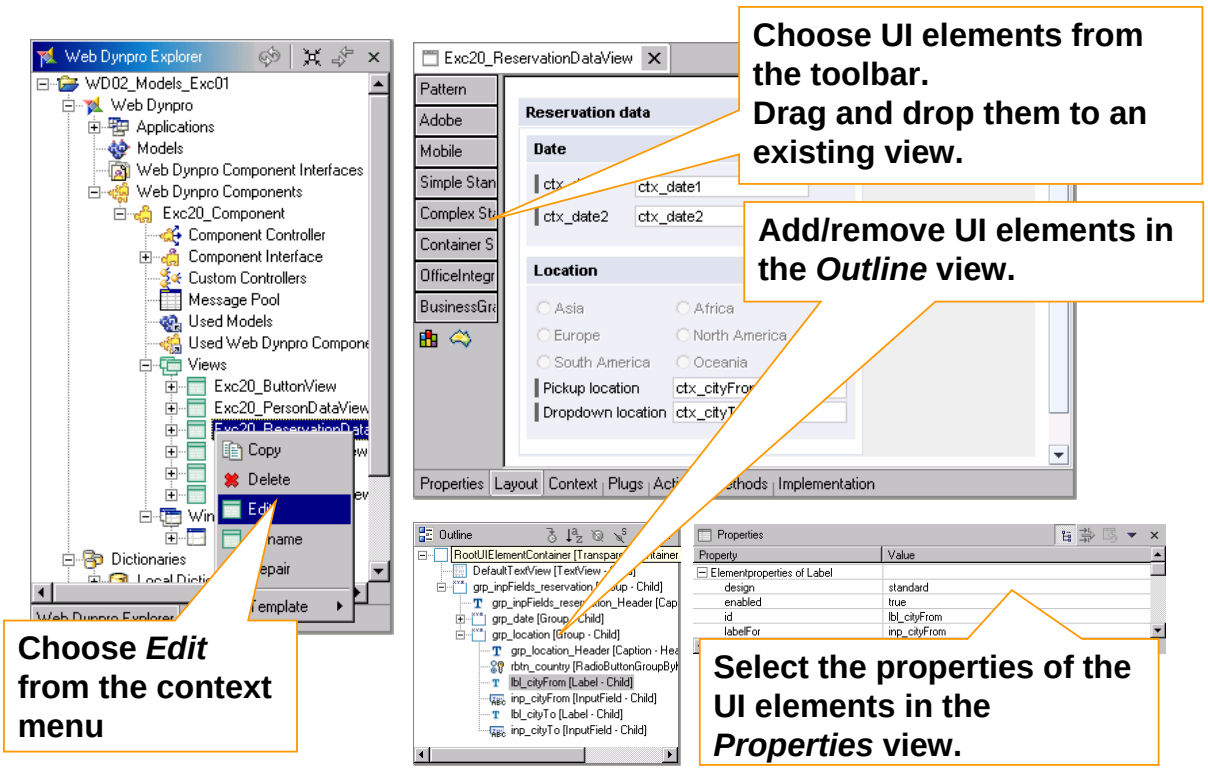
- Comprehensive set of standard UI controls
  - According to Unified Rendering and Unified UI Element standards

Button	Caption	Chart (onSelect events!)
Checkbox [group]	Dropdown list box	Group
HTML Frame	Image	Input field
Label	Link to action	Link to URL
Menu (only in tray)	Progress Indicator	Radio button [group]
Road Map	Scroll bar	Table
Tab strip	Text edit	Text view
Tool bar	Tray	Tree

- Table cells containing nested controls
  - Buttons, drop-down list, links, images, checkboxes, radio buttons
  - In-place editing (with automatic undo support!)

- There are numerous user interface elements available for designing the user interface of a Web Dynpro application. All available user interface elements are divided into categories and can be selected using buttons in the view designer.

View Designer



- **View Designer**

  The *View Designer* is a Web Dynpro tool that provides graphical support when implementing the user interface layout of a Web Dynpro application. The logical Web Dynpro element for the interface layout is the View.

  There are several standard interface elements available, all of which can be adapted to suit your requirements by adjusting the properties appropriately.

- **Opening the View Designer**

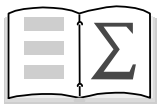
  The View Designer is used after you have created a view in the Navigation Modeler or Data Modeler.

  To open the View Designer, choose *Edit* from the context menu for the view name in the Web Dynpro Explorer; Navigation Modeler, or Data Modeler.

  The *Layout* tab leads you to the View Designer tool in the right screen area. The perspective that contains the work area of the View Designer is the *Diagram View*.

  If you want to enlarge the work area in which you position the individual interface elements, double-click the header bar in the editor. To revert to the original size, double-click the same bar again.

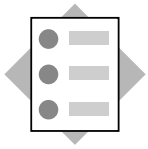
UI Elements Overview: Topic Summary



You should now be able to:

- **Explain which UI elements are available.**
- **Use the View Designer to create UI elements.**

Tables: Topic Objectives

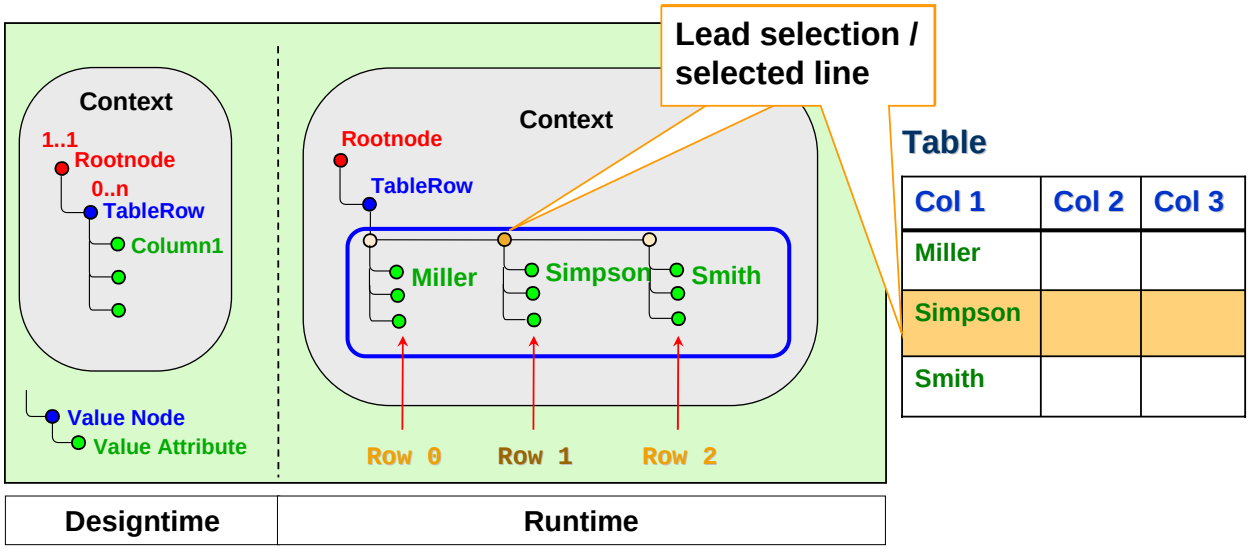


After completing this topic, you will be able to:

- **Create tables and fill them with data from the context.**

Filling Tables with structured context data

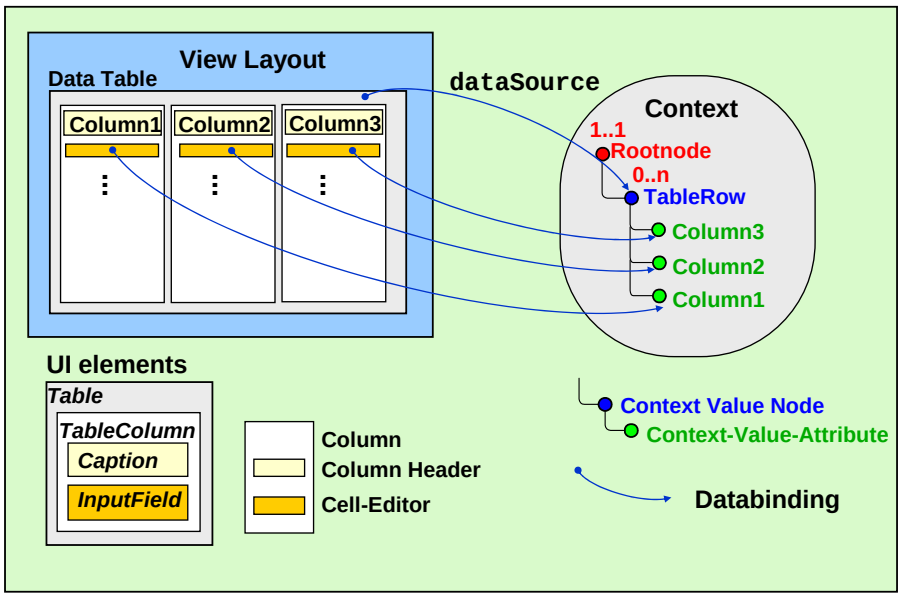
- Filling Tables on a Web User Interface can easily be done in two steps
  - Binding corresponding UI element properties to context elements
  - Programmatically adding value node elements to context value node



- **Table UI Element**
  The *Table* UI element allows the two-dimensional display of data in cells arranged into rows and columns. The UI element consists of a header area, context text area, and footer area. The lead selection of a row is highlighted in color when displayed on the screen. The Table UI element can contain any number of TableColumn elements.

Step 1: Populating Tables using Databinding

- Databinding is applied between
  - Table control and context value node (Property: *dataSource*).
  - Cell editor control in TableColumn control and context value attribute



- **Data Binding**

  A table receives its data from a context node – that is, the table property dataSource must be bound to a multiple context node.

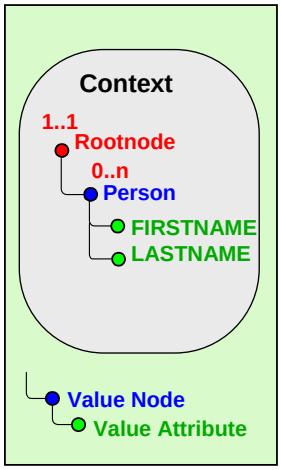
  At runtime, each node element of the node collection is a table row. The number of table rows is identical to the number of node elements. The selected table rows correspond to the node selection. If the selection of the context node changes, the selected table rows also change. The lead selection plays an important role for the table cell that can be edited, because it is predefined by the lead selection of the context node. A table cell can only be edited if the lead selection corresponds to this cell and its table cell editor allows the editing of the cell content.

  The table columns correspond to the context attributes and are described by the aggregation of TableColumn objects. They specify the number and order of columns as well as the headers and the width of the columns.

  The content of a table cell to be displayed is specified by the table cell editor of the column. The table cell editor can only display the content in a table cell, so it does not make any difference if the content cannot be edited

Step 2: Filling context value nodes

Creating typed Node Elements



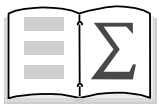

```
// Collection tableRows = new Vector();
1 IPrivate<MyView>.I<Person>Element personElement = null;
2 personElement = wdContext.currentPersonElement();
3 personElement.setFIRSTNAME("Homer");
4 personElement.setLASTNAME("Simpson");
5 tableRows.add(personElement);
6 // add collection of node elements to context value node
7 wdContext.nodePerson().bind(tableRows);
```

or

```
1 IPrivate<MyView>.I<Person>Element personElement = null;
2 personElement = wdThis.wdGet<MyComponentController>().
                  wdGetContext().createPersonElement();
3 personElement.setFIRSTNAME("Homer");
4 personElement.setLASTNAME("Simpson");
5 wdThis.wdGet<MyComponentController>().
  wdGetContext().nodePerson().addElement(person);
```


- **Filling Tables**
  The example shown in the graphs demonstrates two variants how you can fill the table UI with data from the associated context.

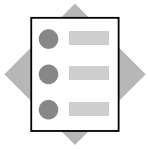
Table: Topic Summary



You should now be able to:

- **Create tables and fill them with data from the context**

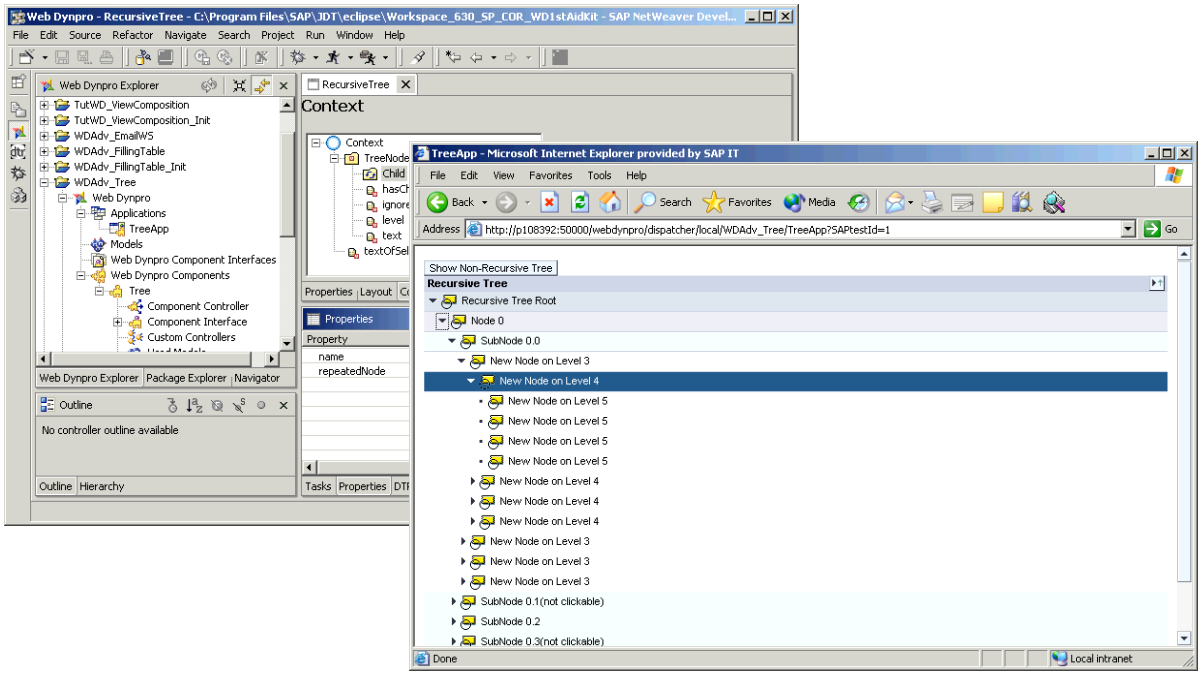
Trees: Topic Objectives



After completing this topic, you will be able to:

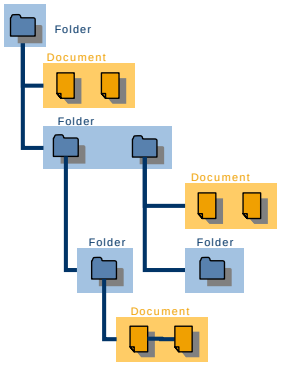
- **Create trees and fill them with data from the context.**

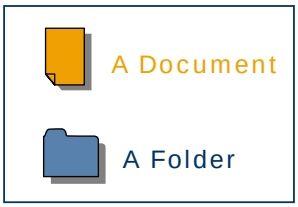
Example



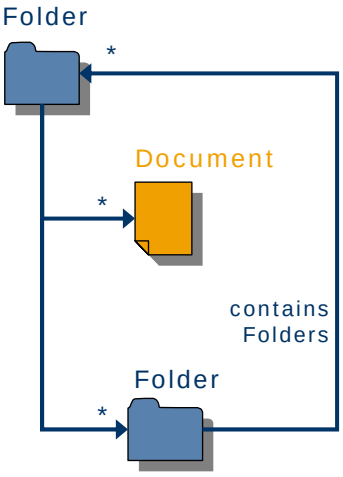
Trees and Recursive Context Nodes

Example

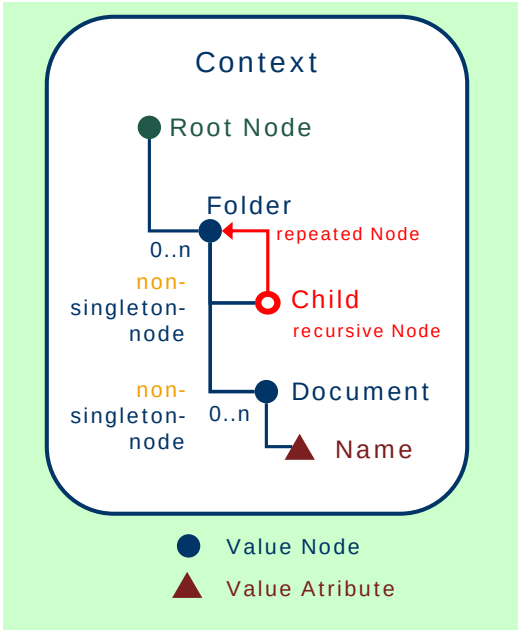




Schema



Context Structure

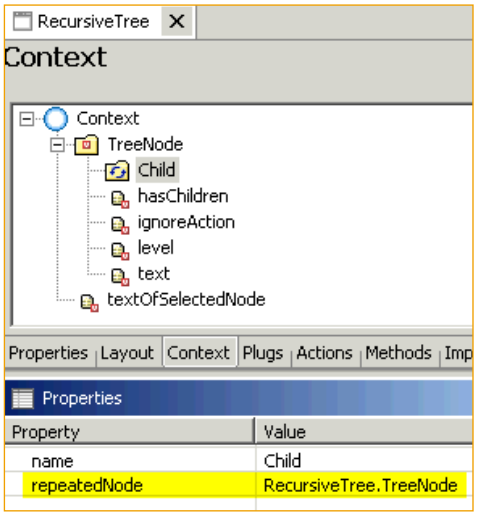


Designtime

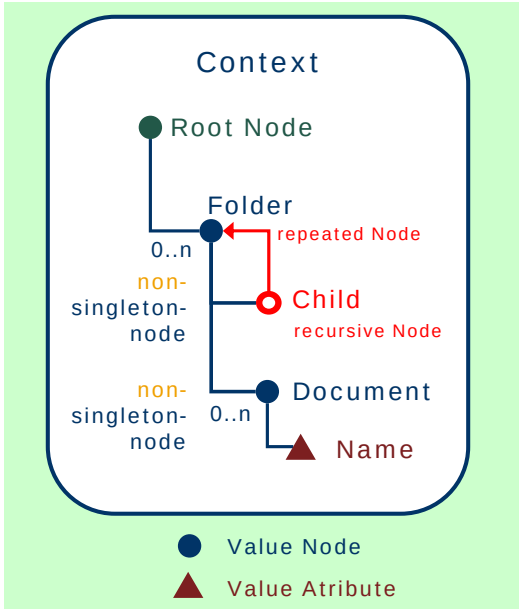
Trees and Recursive Context Nodes (2)

Recursive context nodes point back to some upper context node via property repeatedNode.

This recursive node is automatically non-singleton and exists once for each element of the parent node

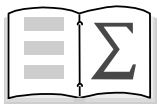


Context Structure



Designtime

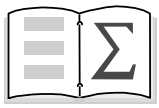
Trees: Topic Summary



You should now be able to:

- **Create trees and fill them with data from the context**

UI Elements: Summary



You should now be able to:

- **Modify and create UI elements.**
- **Create table UI elements.**
- **Create tree UI elements**