Developing Web Dynpro User Interfaces

SAP NetWeaver 04

Copyright

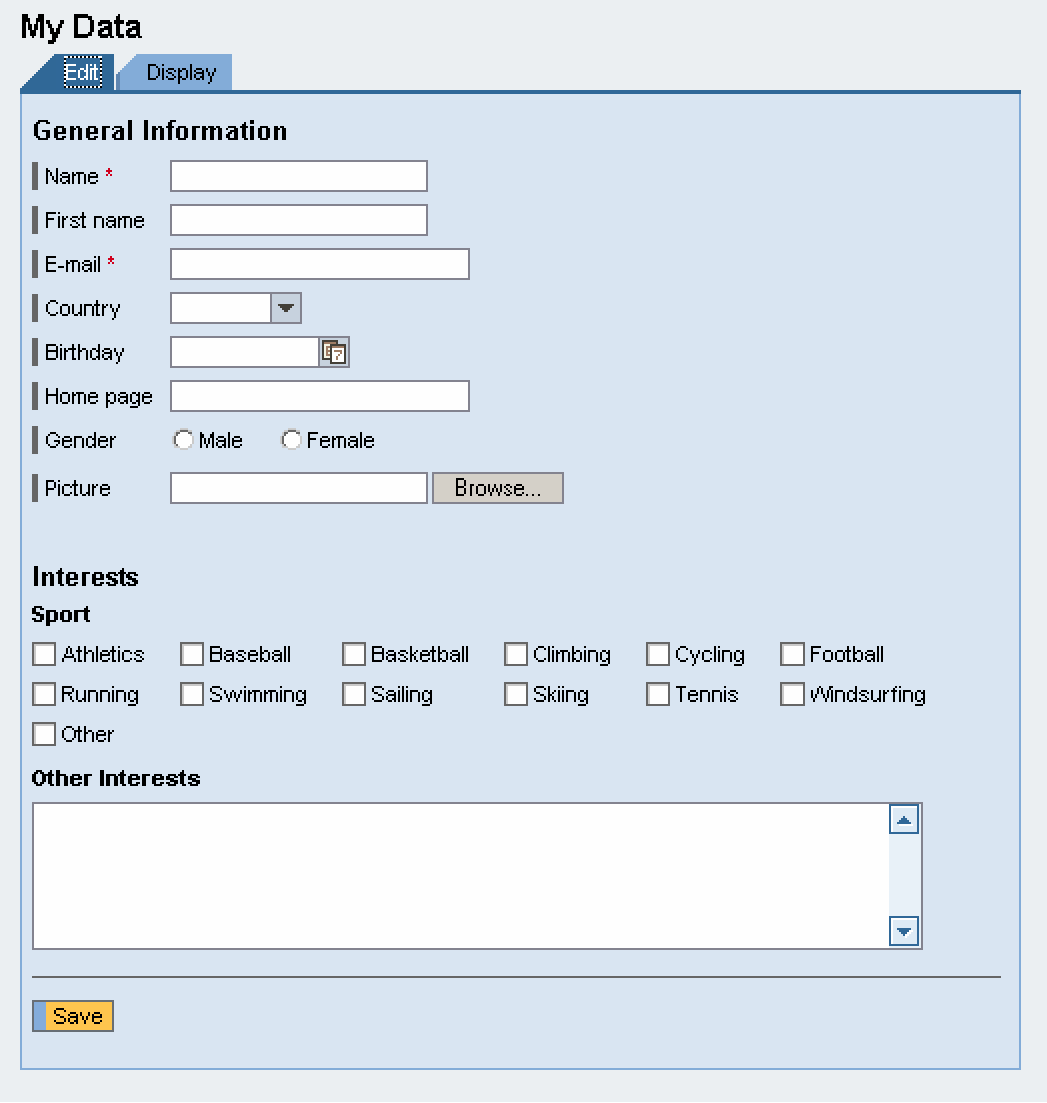
Developing Web Dynpro User Interfaces

Task

The aim of this tutorial is to explain to you the different Web Dynpro controls and to present them in relation to the use of the standard guidelines.

For this purpose, you will design a view that meets the requirements of the official Standards and Guidelines. On the other hand, you will complete a view that is already designed by adding the UI elements *Link*, *Image,* and *FileDownload.*

As an example scenario, an application is provided in which you can edit and display your own personal data.



The basic framework of the application is provided by the project template *TutWD_UI_Init.* Some methods and actions of the controller are already implemented in this so that the focus in this tutorial is on designing the user interface.

Objectives

After you have completed the tutorial, you will be able to use the following Web Dynpro controls:

Simple standard UI elements:

- ✓ Button
- ✓ CheckboxGroup
- ✓ DropDownByKey
- ✓ FileDownload
- ✓ FileUpload
- ✓ Image
- ✓ InputField
- ✓ Label
- ✓ LinkToURL
- ✓ RadioButtonGroupByKey
- ✓ TextEdit
- ✓ TextView
- ✓ ViewContainer

Simple standard UI elements:

- ✓ Tabstrip

Standard Container:

- ✓ TransparentContainer

Pattern:

- ✓ Message Area

Prerequisites

Systems, Installations, and Authorizations

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

Knowledge

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

Next Step

Importing a Project Template

Importing a Project Template

On the SAP Developer Network (SDN) at http://sdn.sap.com, the following Web Dynpro projects are available for this tutorial:

- The project template *TutWD_UI_Init* (the starting point for this tutorial)
- The completed Web Dynpro project *TutWD_UI* (corresponds to the project *TutWD_UI_Init* after completion of the tutorial)

The projects are available for download in corresponding zip files under the category *Home → Developer Areas → Web Application Server → Web Dynpro → Code Samples → Sample Applications and Tutorials*.

Prerequisites

□ You have access to the SAP Developer Network (http://sdn.sap.com).

□ You have installed the SAP NetWeaver Developer Studio.

Procedure

Importing the Project Template into the SAP NetWeaver Developer Studio

1. Call the SAP Developer Network via the URL http://sdn.sap.com and log on with the appropriate user ID and password. If you do not have a user ID, you must register before you can proceed.
2. Download the zip file TutWD_UI_Init.zip containing the project template *TutWD_UI_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.
3. Extract the content of the zip file “TutWD_Dynamic_Init.zip” in the work area of the SAP NetWeaver Developer Studio or in any directory on your local hard disk.
4. Start the SAP NetWeaver Developer Studio.
5. Import the Web Dynpro project *TutWD_UI_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 saved the project *TutWD_UI_Init*, and select the project.
   d. Confirm by choosing *Finish*

The Web Dynpro project *TutWD_UI_Init* appears in the Web Dynpro Explorer for further processing and completion of the tutorial.

Tabular Project Structure

Once you have imported the project template into the SAP NetWeaver Developer Studio, you will see the following structure in the Web Dynpro Explorer.

Web Dynpro Project Structure
Web Dynpro project: **TutWD_UI_Init**
Web Dynpro application: **UIApp**
Web Dynpro component: **UIComp**
View: **DisplayMyDataView** User interface elements are already contained in the *DisplayMyDataView*. In the last section, however, you will enhance this view with the *Link*-, *Image*- and *FileDownload* control.
View: **EditMyDataView** In the *EditMyDataView*, you will create the form for user inputs.
View: **UIView** The *UIView* is the view that will contain the tabstrip.
Window: **UIWindow**
Dictionaries→ Local Dictionary → Data Types → Simple Types
Country In the simple type *Country*, certain countries that will be displayed later in the dropdown list are stored.
Gender In the simple type *Gender*, the attributes “Male” and “Female” – which are to be displayed later in the selection list – are stored.

Next Step

Structure of the Tutorial

Structure of the Tutorial

General overview of interface design

- Practical field
  - Designing *UIView*
  - Defining *UIWindow*
- Basic theory
  - Setting up groups in Web Dynpro
  - Button versus link
- Practical field
  - Setting up *EditMyDataView*
  - Setting up *DisplayMyDataView*

Next Step

General Overview of Graphic User Interface Design

General Overview of Graphic User Interface Design

To ensure that the user has the best possible orientation within the application, the latter should be se up in as consistent and harmonious a way as possible. This can be achieved by reusing different design aspects, for example:

- Script features
- Special layouts
- Uniform treatment of certain elements

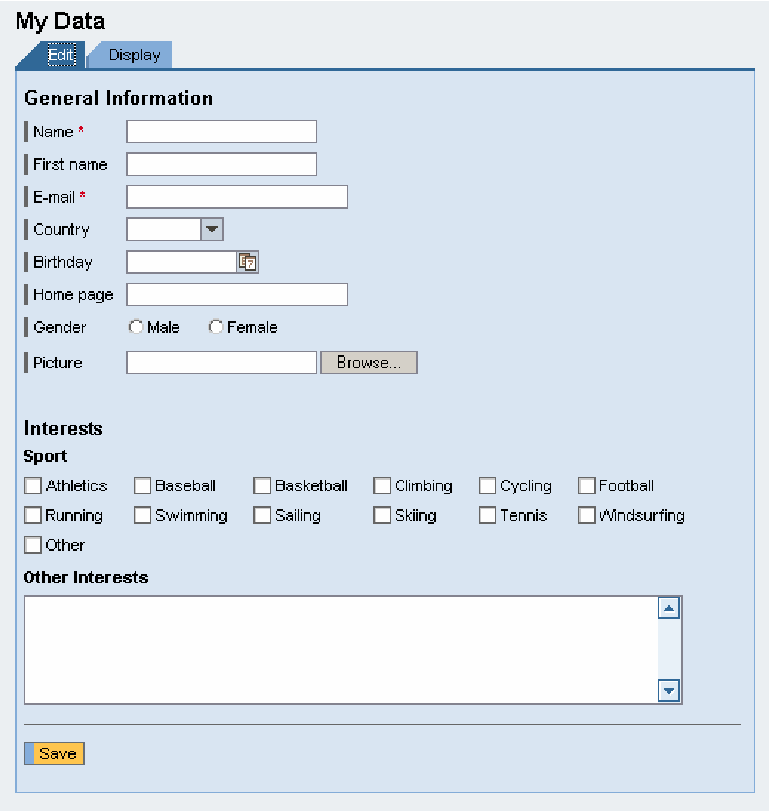
The interface design should not be set up in too complex a fashion, but should rather have a simple and consistent design.

Objects that belong together content-wise should be placed close together and separated from other objects. However, so that the structure does not distract from the actual information, superfluous lines and corners – that can result from frames, for example – should be avoided. If the entire user interface were structured with frames, it would not be possible for any area to attract the user's attention. Rather, this would have a confusing effect. For this reason, one should try to create visual groupings through free space – with or without lines.

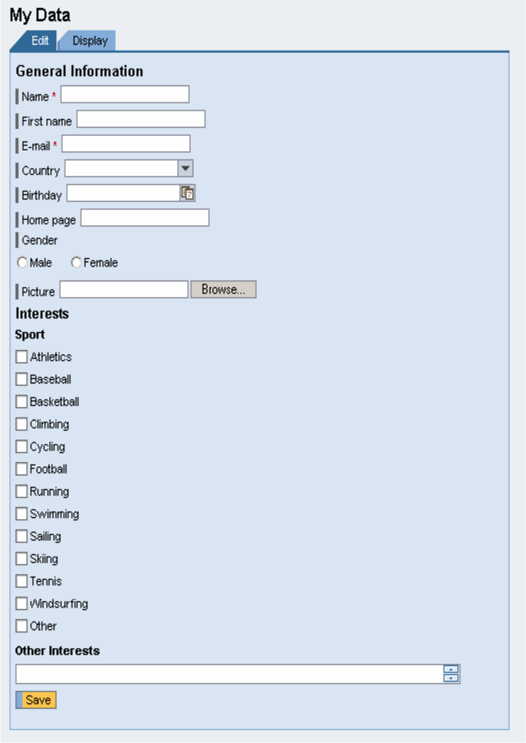
Web Dynpro therefore offers different controls and layouts that support consistent and harmonious user-interface design.

For more detailed information on UI design, refer to the Standards and Guidelines under (http://mysap.wdf.sap.corp:2080/resources/resources_internal/ur_guidelines/).

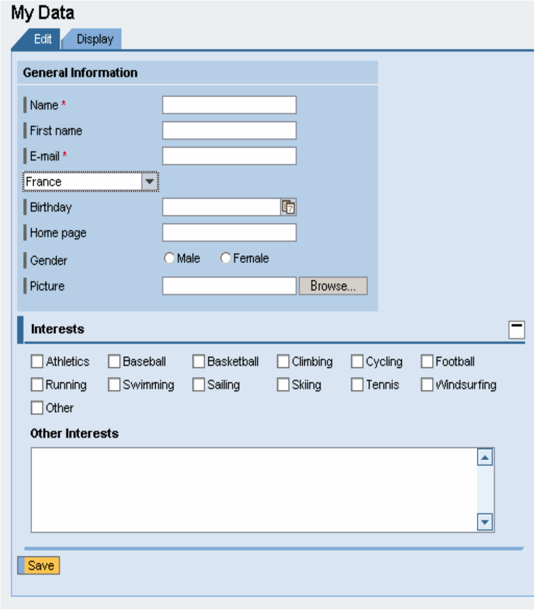
The interface for the tutorial is roughly designed as follows:



Examples of poor design:



- No RowLayout or FlowLayout should be used for input forms because then the input fields would not be aligned under one another. The alignment of the input fields, in this case, would be disquieting and disturbing.
- Subject areas should be separated from one another through blanks.
- Listing checkboxes under one another can waste space and force the user to scroll.
- If text needs to be entered, the field size should be considered beforehand.



Use the group UI elements in rare cases only – for example, if a space or lines are not sufficient. This avoids unnecessary lines and borders that could distract the user's attention from important content.

For each input element there is an appropriate label because, otherwise, the user would have an information gap and be confused.

Tray UI elements are to be used solely for displaying iViews. Since such are used mainly in portals, trays are used in this area only.

To avoid unclear, unstructured layout, use indents sparingly.

Next Step

Designing a UIView

Designing a *UIView*

In the *UIView*, a tabstrip UI element is mapped so that the user can switch between the edit and the display tabstrip. Both tabstrips embed the respective views *EditMyDataView* and *DisplayMyDataView.*

Tabstrip UI Element

The *tabstrip* UI element enables the user to switch easily between different views that share the same window area. The user can toggle between several tab pages by selecting a specific tab title. In this way, users are able to look at all alternative views.

The tabstrip has the disadvantage that it requires a large amount of memory in comparison with other navigation options. Another advantage is that there can be a limit with regard to the number of embedded views because of the memory space limitation.

Do's

- Tabstrips can contain dynamic information. In this way, the user can get an overview of important data, events, and requirements in the views quicker.
- Tabstrips are especially suited to views with different content – views that are quite different in how they look and would thus lead to inconsistent design.
- The tab pages can contain tables and *Group* UI elements.

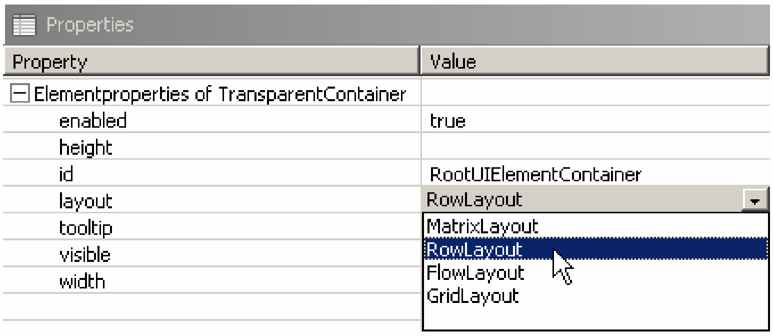
Don’ts

- Avoid tab headings that are too long and also too many tab pages
- Tab pages should not contain any icons.
- Tabstrips show the user that he or she can access the views in which ever sequence desired. If this is not the case, tabstrips should not be used.
- The tabstrip should not be scrolled. In this way, you avoid a flood of information on a very small area.
- Tabstrips must not be nested within each other because this would detract from the general clarity.

  For more information on nesting guidelines, refer to http://mysap.wdf.sap.corp:2080/community/design/layout2a.asp

Procedure

1. Open the *UIView* and switch to the *Layout* tab page.
2. Proceed to the *Outline* perspective, where you will find the standard generated *RootContainer (RootUIElementContainer[Transparent Container]).* Open its context menu by right-clicking the mouse and choose *Properties*.
3. In the *Properties* perspective of the *RootUIElementContainer*, change the *layout* to **`RowLayout`**.

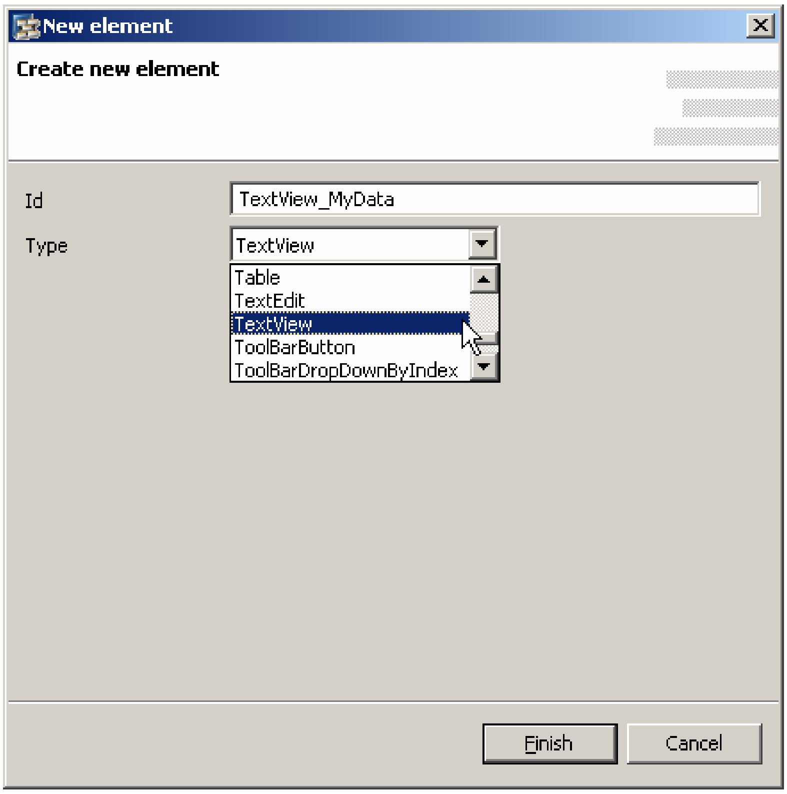


Properties

Property	Value
⊟ Elementproperties of TransparentContainer	
enabled	true
height	
id	RootUIElementContainer
layout	RowLayout
tooltip	
visible	
width	

4. In the *Outline* perspective, open the context menu for *RootUIElementContainer[Transparent Container]* and choose *Insert Child*.

5. Give the name **`TextView_MyData`** to the UI element and choose *TextView* as type for the UI element.



New element

Create new element

Id: TextView_MyData

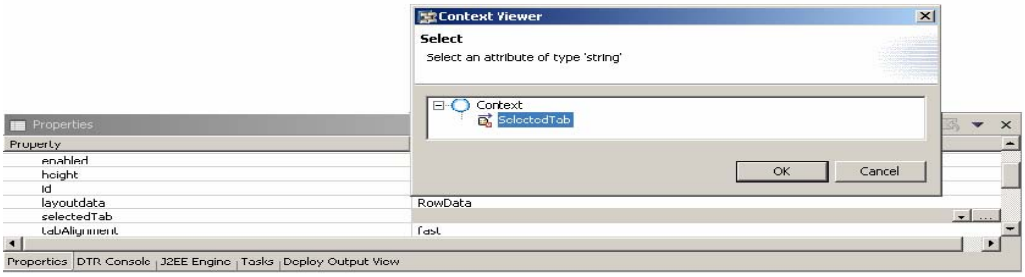
Type: TextView

6. Confirm by choosing *Finish*

7. Change the following properties for *TextView_MyData* in the *Properties* perspective.

8. To the *RootUIElementContainer* , add a further UI element of the type Tabstrip with the name **`Tabstrip_MyData`**. Here, in the context menu, choose *Insert Child*.

9. Bind the context attribute ***SelectedTab***, which already exists in the initial project template, to *Tabstrip_MyData* by assigning this context attribute to the property. In this way, you can define in the implementation which tab is to be displayed.



Context Viewer

Select

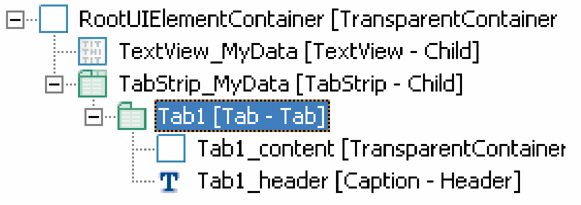
Select an attribute of type 'string'

- Context
  - SelectedTab

Property	
enabled	
height	
id	
layoutdata	RowData
selectedTab	
tabAlignment	fast

10. Change the property *width* from *Tabstrip_MyData* to **`560px`** and the property *layoutdata* to **`RowHeadData.`**

11. To add a tab page to the tabstrip, choose *Insert Tab* in the context menu for *Tabstrip_MyData*. A *Tab* Is automatically added to the tabstrip. This tab has a particular ID, a *TransparentContainer* (for the tab page content), and a tab heading of the type *Caption*.



- RootUIElementContainer [TransparentContainer
  - TextView_MyData [TextView - Child]
  - TabStrip_MyData [TabStrip - Child]
    - Tab1 [Tab - Tab]
      - Tab1_content [TransparentContainer
      - Tab1_header [Caption - Header]

Repeat this procedure one more time in order to finally have two automatically generated tabs with their respective IDs *Tab1* and *Tab2*.

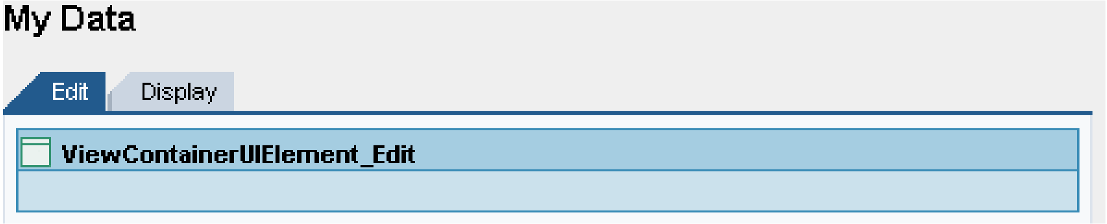
12. Afterwards, perform the following changes in **`TabStrip_MyData`**.

13. To *Tab1_content*, add the UI element *ViewContainerUIElement* with the name **`ViewContainerUIElement_Edit`** using *Insert Child* in the context menu.

14. Repeat the last step for *Tab2_content* and give the UI element the name **`ViewContainerUIElement_Display`**.

Result

You have created the layout for the UIView*.* In the *Layout* perspective, the following layout will be displayed if you press one of the title elements.



My Data

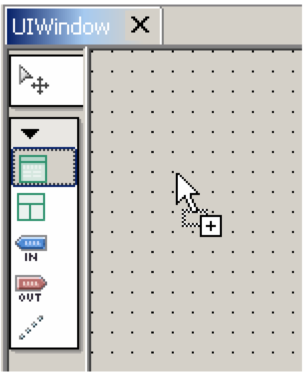
Next Step

Declaring a UIWindow

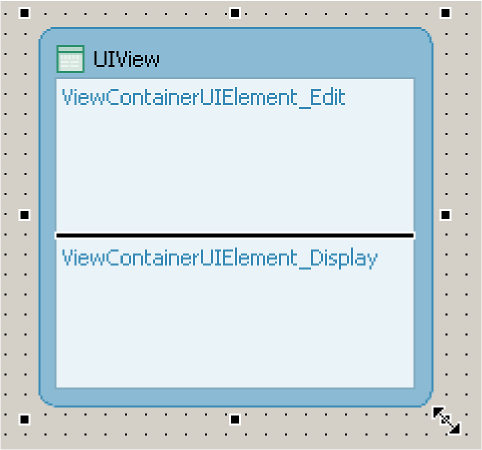
Declaring a UIWindow

Procedure

1. Double-click the *UIWindow* (TutWD_UI_Init → Web Dynpro → Web Dynpro Components → UIComp → Windows → UIWindow). The *Navigation Modeler* is started.
2. Choose the icon *Embed a view* and click the empty gray area.



3. In the dialog box, choose *Embed existing View* and confirm with *Next*.
4. From the View list, choose *UIView* and confirm with *Finish*.
5. Enlarge the *UIView*.



In the UIView, the *ViewContainerUIElements* that you created previously are displayed. Here you can now embed the views that are to be displayed in the respective tab page.

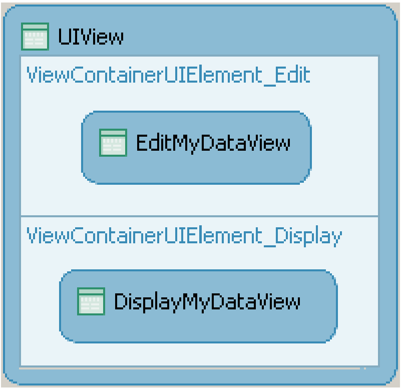
6. Choose the icon *Embed a view* and click in the *ViewContainerUIElement_Edit* container.

12. In the dialog box, choose *Embed existing view, Next,* and in the *View* list, choose the *EditMyDataView*. Confirm by choosing *Finish*

13. In the *ViewContainerUIElement_Display* container, embed the *DisplayMyDataView* in the same way – as described in step 2.

Result

You have now declared the *UIWindow* for the application.



Next Step

Setting Up Groups in Web Dynpro

Setting Up Groups in Web Dynpro

So that areas that belong together content-wise can be identified, groups should be created for them in the user interface. In particular, blanks should be used for this purpose. These can be implemented using the different layouts available. In addition, Web Dynpro provides horizontal and vertical separator elements.

However, avoid the use of **Group** UI elements and set these only when you have to highlight an extraordinary and particularly important area that might otherwise be overlooked.

Do not use the **Tray** UI element in Web Dynpro. It is provided for grouping iViews only.

Layouts

Different layouts can be assigned to the following container elements:

- TransparentContainer
- Group
- ScrollContainer
- (und Tray)

You have the following four layout variants at your disposal:

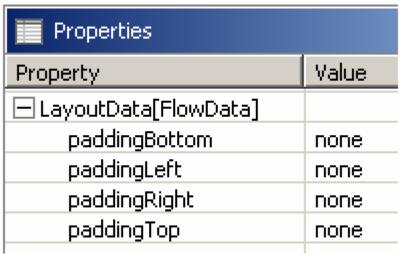
- FlowLayout
- MatrixLayout
- RowLayout
- GridLayout

The GridLayout will no longer be used in future because the MatrixLayout contains the most important properties of the GridLayout, but it also provides more flexibility.

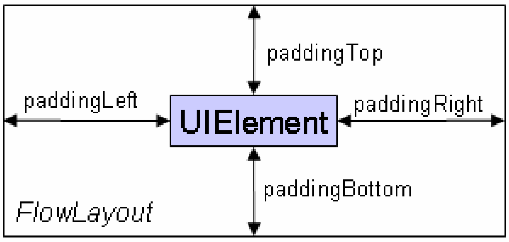
FlowLayout

The FlowLayout assigns the container children in a sequential manner. A FlowLayout depends on the client technology and the size of the browser window. To implement line breaks behind the container children, you can set the property `wrapping` to **`true`** for these.

To determine the distances between the container children, a LayoutData is assigned to each child. Here you can set the distance properties. The distances can be selected as `none`, `small`, `medium`, and `large.`



Property	Value
⊟ LayoutData[FlowData]	
paddingBottom	none
paddingLeft	none
paddingRight	none
paddingTop	none



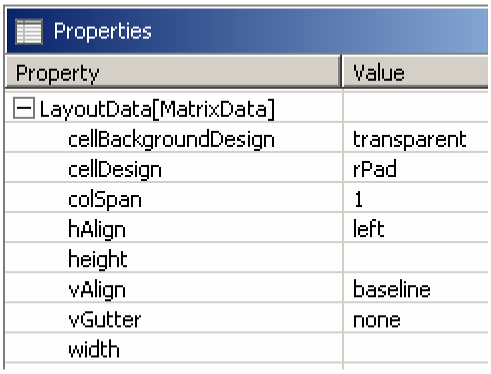
MatrixLayout

The matrix layout arranges the UI elements in a table-like manner in a grid structure.

You can use the properties `stretchedHorizontally` and `stretchedVertically` to specify whether or not the UI elements should match the container size.

In contrast to the GridLayout, for example, you cannot explicitly define the number of columns To create a line break for a Container Child UI element into the next line, assign **`MatrixHeadData`** to this in the property *`layoutdata`* .

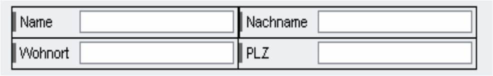
In the `LayoutData` of the container children, you can determine the properties of a MatrixLayout cell. You can align the UI element both horizontally – `hAlign` (left-justified, centered, right-justified), as well as vertically – `vAlign` (top-aligned-top, in the middle, bottom-aligned). The default settings for the alignment of the UI elements within the GridLayouts are left-justified and top-aligned.

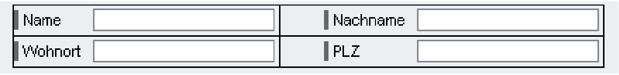


Properties

Property	Value
⊟ LayoutData[MatrixData]	
cellBackgroundDesign	transparent
cellDesign	rPad
colSpan	1
hAlign	left
height	
vAlign	baseline
vGutter	none
width	

Using the `vGutter` attribute, you can determine how large the additional distance between the cells should be. You can see the difference on the following two sdreens. Left without `vGutter`, and right with `vGutter` and the value `large.`

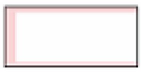
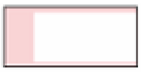
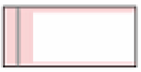
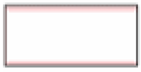
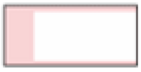
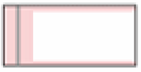
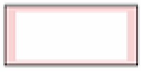
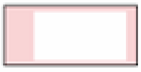
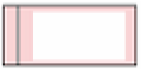
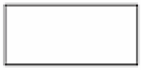
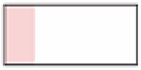
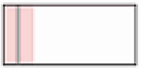
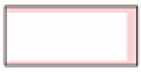
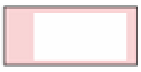
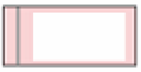




Using the `cellBackgroundDesign` property, you can determine the background color of the MatrixLayout cell.

border	This is the color of the cell borders. This value is used for nested matrix layouts to create grid net lines
fill1	This color corresponds to the value `primarycolor` of the `design` property of the Group UI element.
fill2	This color corresponds to the value `secondarycolor` of the `design` property of the Group UI element.
fill3	This color corresponds to the color value of the third level of a Tree UI element.
header	This color is identical to the color of the header area of a Tree UI element or a table.
plain	White background.
transparent	The background is transparent. The individual cells are displayed without grid net lines.

Using the `cellDesign` property, you can determine the space from the cell content to the outer border of the cell. It can have the following values:

Value	Spacing	W/o Gutter	With Gutter	With Gutter and Line
lPad	• Upper Margin: 2 Pixels • Lower Margin: 2 Pixels • Left Margin: 4 Pixels			
lrNoPad	• Upper Margin: 2 Pixels • Lower Margin: 2 Pixels			
lrPad	• Upper Margin: 2 Pixels • Lower Margin: 2 Pixels • Left Margin: 4 Pixels • Right Margin: 4 Pixels			
padless	• No spacing			
rPad	• Upper Margin: 2 Pixels • Lower Margin: 2 Pixels • Right Margin: 4 Pixels			

Vertical Separators and Spacing

The `vGutter` property lets you specify additional horizontal spacing easily. You can set these additional distances (known as gutters) with or without separators.

This type of layout is preferable to the *Grid Layout*, since it makes the layout structure in a container more consistent.

RowLayout

A RowLayout has a similar behavior to the MatrixLayout. However, it sequentially assigns the UI elements to exactly one column.

If you assign **`RowHeadData`** to a container-child UI element of the property `layoutdata`, it is exactly this UI element that is wrapped into the next line.

The row layout differs from the matrix layout in that the content is not organized in table cells – that is, the individual elements are not aligned vertically with each other. Whenever the row layout is implemented in an application, performance is better than if a matrix layout were used, but the layout flexibility is not compromised.

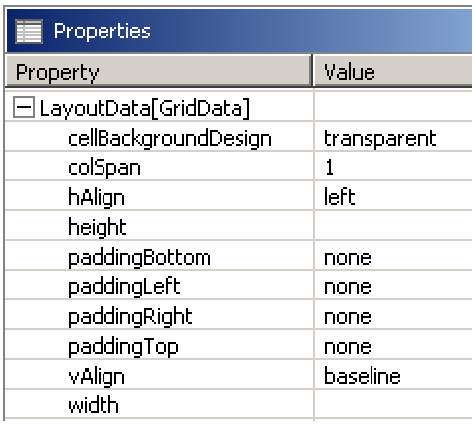
Vertical Separators and Spacing

The `vGutter` property of the corresponding *RowHeadData* object lets you specify additional horizontal distances and separators easily – in the same way as for MatrixLayout.

GridLayout

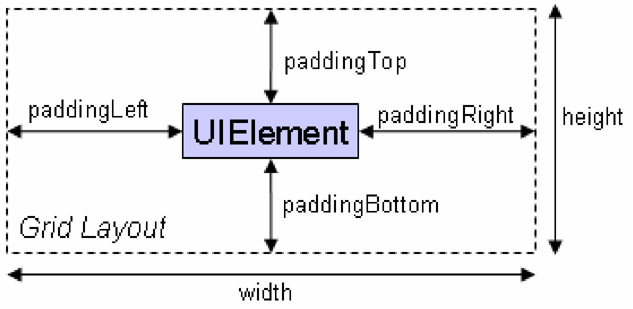
A GridLayout arranges the container children in a two-dimensional grid with a defined column number and any number of rows. The number of grid columns can be specified using the GridLayout property *colCount*.

The properties `cellBackgroundDesign`, `colSpan`, `hAlign`, `height`, `vAlign`, and `width` can be defined in the same way as MatrixLayout.



Properties

Property	Value
⊟ LayoutData[GridData]	
cellBackgroundDesign	transparent
colSpan	1
hAlign	left
height	
paddingBottom	none
paddingLeft	none
paddingRight	none
paddingTop	none
vAlign	baseline
width	



If a GridLayout cell is to include several cells of a row, you can determine this using the property `colspan`.

HorizontalGutter UI Element

The *HorizontalGutter* UI element helps you structure the layout and text parts, similar to the HTML tag <hr>. You can display the HorizontalGutter UI element either with or without a separator. You can determine this property at `hasRule`. You can set the width in the `width` property and specify the following sizes:

- em
- ex
- Pixels
- Percentages

The *HorizontalGutter* UI element provides three different heights: In the `height` property, the values mean the following:

- small → 7 Pixels
- medium → 17 Pixels
- large → 31 Pixels

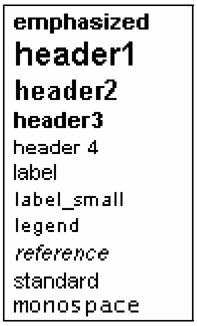
TextView

The T*extView* UI element provides an area for displaying a multiline text.

When using a *TextView* user interface element, you should always add a label to ensure accessibility of an application.

In order to ensure this accessibility, a TextView UI element must not be used as a label for the input fields. The *Label* UI element is provided for this purpose.

The `design` property describes the appearance of the *TextView* UI element. The Cascading Style Sheet (CSS) provided by SAP describes the different options for the `design` attribute display.



emphasized
header1
header2
header3
header 4
label
label_small
legend
reference
standard
monospace

emphasized	Highlights the text and applies the standard font size.
header 1	Highlights the text and applies the standard font size +4.
header 2	Highlights the text and applies the standard font size +2.
header 3	Highlights the text and applies the standard font size.
header 4	Highlights the text and applies the standard font size –1 (small) -> (like the *legend* value, but highlighted).
label	Displays the text using the standard font; a blank is always inserted after the text.

label_small	Displays the text using the standard font like for the *label* value, but with font size –1 like the font size for the *header4* value.
legend	Displays the text using the standard font size –1.
monospace	Displays the text using a non-proportional font size. Each letter takes up the same space.
reference	Displays the text in italics and applies the standard font size.
standard	Displays the text using the standard font size. No text attributes are defined for this value.

If you create a particular window area for a *TextView* UI element, you should make sure that this area provides sufficient space for translations of the texts. Texts in other languages (not English) could require approximately 30% more space.

Next Step

Button Versus Link

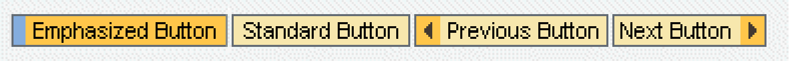
Button Versus Link

Button

The UI element *Button* represents the pushbutton on the screen. The user can execute statements and actions by clicking the pushbutton. The *Button* should only be used for unique and important tasks that reference a specific object – such as *Save* or *Print*, or navigation tasks. Button labels should always start with an uppercase letter; articles or short prepositions are exceptions in this case.

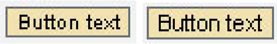
If too many buttons are placed on the user interface, the design will appear too complex. If you are unsure, use the *Link* UI element instead since it does not appear too dominant.

Buttons can have different `designs`.



The **`emphasized`** design should only be chosen if you want to complete a task by pressing the button. This button type should be placed to the left in a button group.

In addition, the button size can be determined using the `size` property and can have the values `default` (standard size) and `small`.



Deactivated Buttons and Hidden Buttons

Buttons with the `enable` property **`false`** are deactivated buttons and mean that this function is currently not available. For this reason, you should change this property for functions that are temporarily not available.

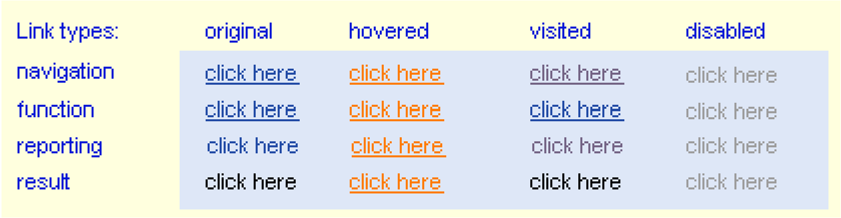
Hidden buttons that are not available continuously should not be displayed to the user. A typical application case for hidden buttons would be actions for which users do not have authorization. You can hide the buttons by changing the `visible` property.

Link

The *LinkToURL* UI element is a hypertext link. The navigation to this link leads to a user-defined Web resource (URL).

Using the `type` property, you can determine the graphic display of the UI element.

function	Link is displayed underlined in the standard design.
navigation	Link is displayed underlined and with a font color that is used for links already visited. This is the standard link for navigation purposes!
reporting	Link is displayed not underlined in the standard design.
result	Link is displayed not underlined. It is used in tables.



Link types:	original	hovered	visited	disabled
navigation	click here	click here	click here	click here
function	click here	click here	click here	click here
reporting	click here	click here	click here	click here
result	click here	click here	click here	click here

Next Step

Designing the Layout for EditMyDataView

Designing the Layout for EditMyDataView

In the *EditMyDataView*, the user should be able to specify his or her entries. For this purpose, you now design the input form.

Procedure

1. Open the **EditMyDataView** and switch to the *Layout* tab page.
2. Change the *Layout* for the **RootUIElementContainer** to RowLayout.

UI Elements for the RootUIElementContainer

3. Add the following UI elements and their respective properties to the **RootUIElementContainer**.

Properties	Value
TransparentContainer_General of the type *TransparentContainer*	
layout	MatrixLayout
layoutdata	RowHeadData
HorizontalGutter1 of the type *HorizontalGutter*	
layoutdata	RowHeadData
hasRule	False
TransparentContainer_Interests of the type *TransparentContainer*	
layout	MatrixLayout
layoutdata	RowHeadData
HorizontalGutter2 of the type *HorizontalGutter*	
layoutdata	RowHeadData
Button_Save of the type *Button*	
Design	Emphasized
Layoutdata	RowHeadData
onAction	Save
MessageArea1 of the type *MessageArea*	
Layout	RowHeadData

UI Elements of the TransparentContainer_General

4. Add the following UI elements to the **TransparentContainer_General**.

Properties	Value
TextView_Header1 of the type *TextView*	
design	header2
layoutdata	MatrixHeadData
text	General Information

TransparentContainer_Text of the type *TransparentContainer*	
Layout	MatrixLayout
Layoutdata	MatrixHeadData
width	445px
stretchedHorizontally	False

UI Elements of the TransparentContainer_Text

5. Add the following UI elements to the **TransparentContainer_Text**.

   You have two options on how to proceed, but in the end you have the same outcome.

First Option:

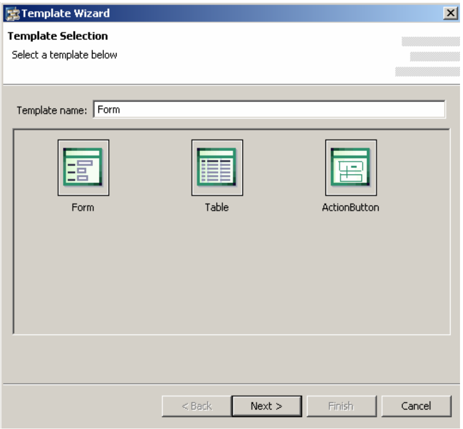
You create the individual elements, step by step, using *Insert Child* navigation. (In the case of large forms, this can be very time-consuming.)

Second Option:

Use the *Apply Template* function, which – like the *Insert Child* function – is in the context menu. You proceed as follows:

a. Right-click the *TransparentContainer_Text* and choose *Apply Template*.

b. Afterwards, the Template Wizard starts. Here you choose the function *Choose Form* and continue by pressing *Next*.



Template Wizard

Template Selection

Select a template below

Template name: Form

c. In the next step, choose all the context attributes that you find in the following table, confirm with *Next*, and then – in the following window on the right – change the sequence using the arrow keys. The corresponding labels, text fields, and even the data binding to the view context will be generated automatically.

Name
FirstName
Email
Country
Birthday
Homepage

d. The respective labels, text fields, and even the data binding to the view context are generated automatically.

e. In the last step, however, you have to enter the following elements using *Insert Child* navigation since there is no option for entering these elements in the form using *Apply Template*.

Label_Picture of the type *Label*	
layoutData	MatrixHeadData
text	Picture
vAlign	Top
FileUpload_Picture of the type *FileUpload*	
data	✓ MyData.Picture
fileName	MyData.PictureName
Label_Gender of the type *Label*	
layoutData	MatrixHeadData
text	Gender
vAlign	top
RadioButtonGroupByKey_Gender of the type *RadioButtonGroupByKey*	
selectedKey	✓ MyData.Gender
colCount	2

f. Add the following **Attributes** to the UI elements generated in step 5.

Name_label of the type *Label*	
vAlign	Top
Name of the type *InputField*	
State	Required
FirstName_label of the type *Label*	
text	First name
vAlign	top
Email_label of the type *Label*	
Text	E-mail
vAlign	top
Email of the type *InputField*	
State	required
Width	150px
Country_label of the type *Label*	
vAlign	top

Birthday_label of the type *Label*

Align	top
Birthday of the type *InputField*	
Width	75px
Homepage_label of the type *Label*	
Text	Home page
vAlign	top
Homepage of the type *InputField*	
Width	150px

UI Elements of the TransparentContainer_Interests

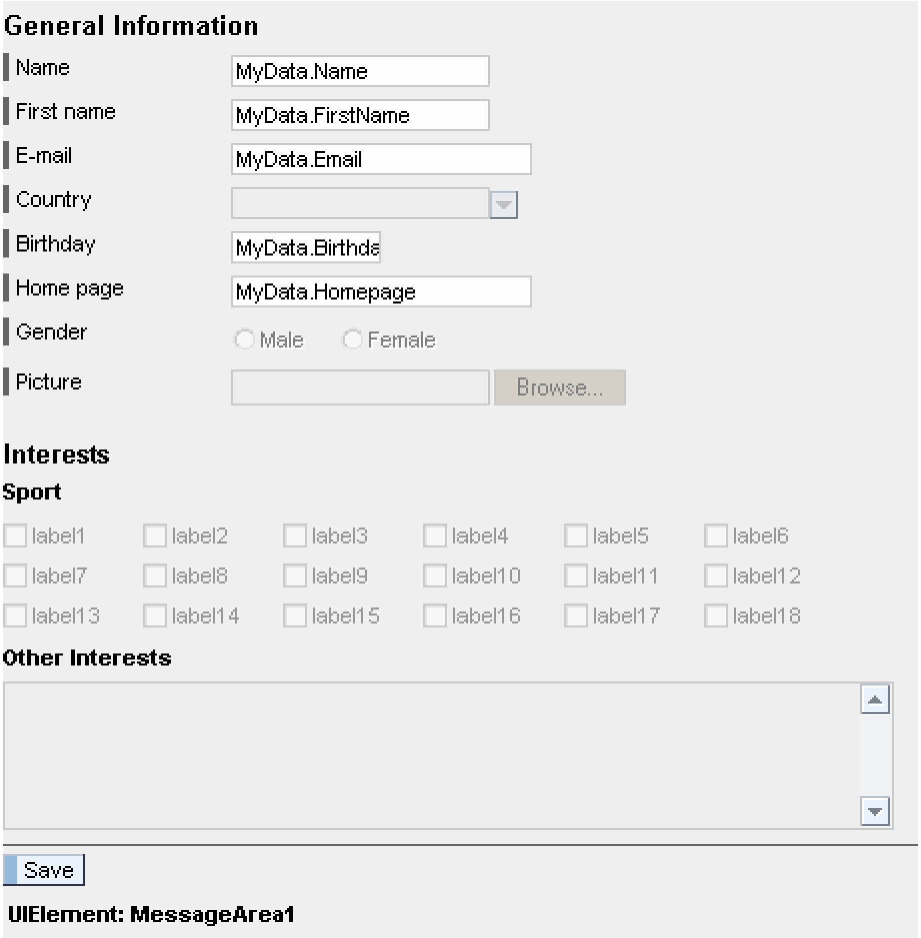
7. Add the following UI elements to the TransparentContainer_Interests.

Properties	**Value**
`TextView_Header2` of the type *TextView*	
Design	header2
Layout	MatrixHeadData
text	Interests
`TextView_Sport` of the type TextView	
Design	header3
Layoutdata	MatrixHeadData
text	Sport
`CheckBoxGroup_Sport` of the type CheckBoxGroup	
colCount	*6*
layoutdata	MatrixHeadData
texts	MyData.Sport.Sport The Context attribute – where the texts for the CheckBoxGroup are located – must be in a context node that has *Cardinaliy* `0..n` and the *Selection* `0..n` (to enable multiple selection). The initialization of the context takes place in the method `wdDoInit()`.
`TextView_Other` of the type TextView	
design	Header3
layoutdata	MatrixHeadData
text	Other Interests
TextEdit_Other of the type *TextEdit*	
Cols	85
Layoutdata	MatrixHeadData

Value	✓ MyData.OtherInterests

Result

You have enhanced the layout *EditMyDataView*.



General Information

Name	MyData.Name
First name	MyData.FirstName
E-mail	MyData.Email
Country	
Birthday	MyData.Birthda
Home page	MyData.Homepage
Gender	○ Male ○ Female
Picture	Browse...

Interests

Sport

□ label1	□ label2	□ label3	□ label4	□ label5	□ label6
□ label7	□ label8	□ label9	□ label10	□ label11	□ label12
□ label13	□ label14	□ label15	□ label16	□ label17	□ label18

Other Interests

UIElement: MessageArea1

Next Step

Designing the DisplayMyDataView

Designing the Layout of DisplayMyDataView

To display the user inputs, we recommend using the input layout. For this purpose, UI elements are already contained in the **DisplayMyDataView**. They are similar to those of *EditMyDataView*.

So that you can see that the data is not longer editable, you set the input fields to **readOnly**. Furthermore, you will enter the UI elements *Link, Image,* and *FileDownload*.

Procedure

1. Open the **DisplayMyDataView** and switch to the *Layout* tab page.
2. Set the *readOnly* property to **`true`**.
    - In the `TransparentContainer_Form`:
        - `Name`
        - `FirstName`
        - `Email`
        - `Country`
        - `Birthday`
        - `RadioButtonGroupByKey15`
    - In the `TransparentContainer_Interests`:
        - `CheckBoxGroup_Sport`
        - `TextEdit_Other`

To set a *Link* to the user's homepage, enter in the `TransparentContainer_Form` the following UI element with the corresponding properties:

Properties	Value
LinkToURL_Homepage of the type *LinkToURL*	
Reference	✓ MyData.Homepage
Text	MyData.Homepage

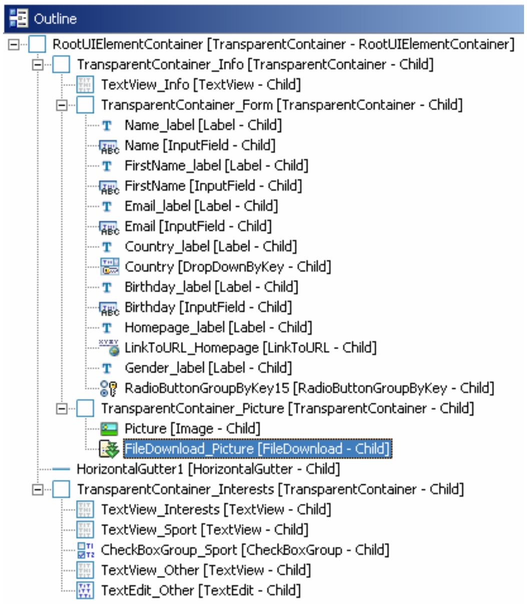
In the outline, choose `LinkToURL_Homepage` and open the context menu. Choose *Move Up* so often until the UI element is under `Homepage_Label`.

To display the screen with its corresponding download, enter the following UI elements with their respective properties in `TransparentContainer_Picture`:

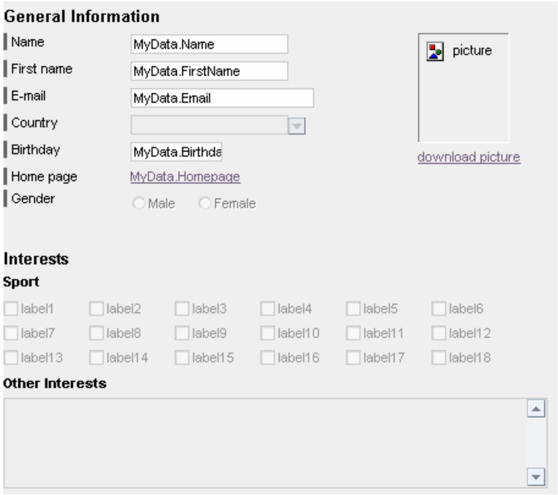
Properties	Value
Picture of the type *Image*	
alt	picture Determines an alternative text that is displayed whenever the graphic does not open or cannot be found.
Height	90px

Source	MyData.ImgSource Determines the Web address (URL) of the graphic through which the UI element receives the data.
Width	75px
FileDownload_Picture of the type *FileDownload*	
Data	MyData.File
layoutData	MatrixHeadData
Text	download picture

Result



```
Outline
⊟ RootUIElementContainer [TransparentContainer - RootUIElementContainer]
  ⊟ TransparentContainer_Info [TransparentContainer - Child]
      TextView_Info [TextView - Child]
    ⊟ TransparentContainer_Form [TransparentContainer - Child]
        Name_label [Label - Child]
        Name [InputField - Child]
        FirstName_label [Label - Child]
        FirstName [InputField - Child]
        Email_label [Label - Child]
        Email [InputField - Child]
        Country_label [Label - Child]
        Country [DropDownByKey - Child]
        Birthday_label [Label - Child]
        Birthday [InputField - Child]
        Homepage_label [Label - Child]
        LinkToURL_Homepage [LinkToURL - Child]
        Gender_label [Label - Child]
        RadioButtonGroupByKey15 [RadioButtonGroupByKey - Child]
    ⊟ TransparentContainer_Picture [TransparentContainer - Child]
        Picture [Image - Child]
        FileDownload_Picture [FileDownload - Child]
    HorizontalGutter1 [HorizontalGutter - Child]
  ⊟ TransparentContainer_Interests [TransparentContainer - Child]
      TextView_Interests [TextView - Child]
      TextView_Sport [TextView - Child]
      CheckBoxGroup_Sport [CheckBoxGroup - Child]
      TextView_Other [TextView - Child]
      TextEdit_Other [TextEdit - Child]
```



General Information

Name	MyData.Name
First name	MyData.FirstName
E-mail	MyData.Email
Country	
Birthday	MyData.Birthda
Home page	MyData.Homepage
Gender	○ Male ○ Female

Interests

Sport

□ label1	□ label2	□ label3	□ label4	□ label5	□ label6
□ label7	□ label8	□ label9	□ label10	□ label11	□ label12
□ label13	□ label14	□ label15	□ label16	□ label17	□ label18

Other Interests

Next Step

Executing the Application TutWD_UI_Init

Executing the Application TutWD_UI_Init

Now that you have reached this stage, you can start the fully developed example application in the Web Browser as described below.

Prerequisites

- You have started the SAP J2EE Engine.

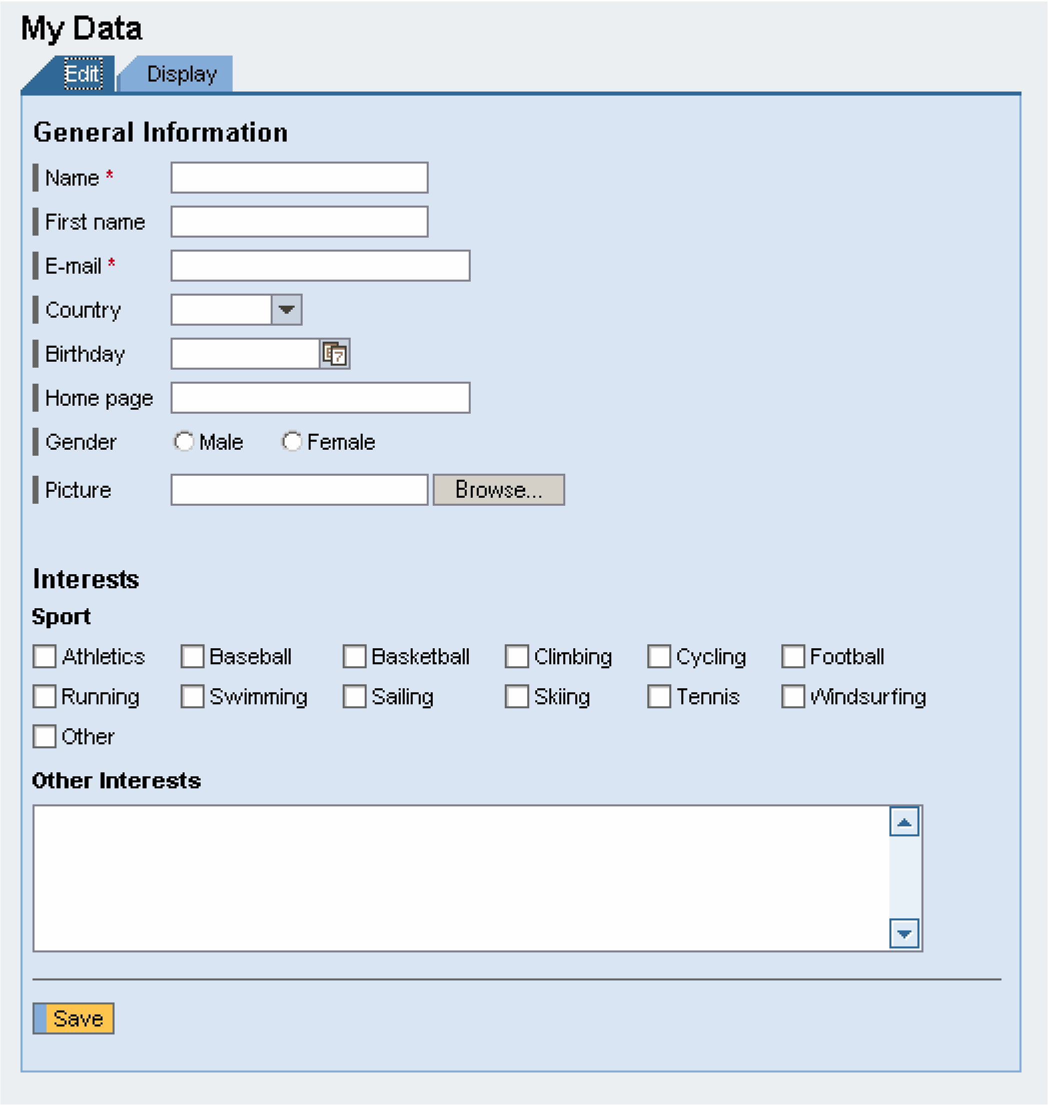
Procedure

Save the current state of the metadata for your project by choosing the icon *Save All Metadata*.

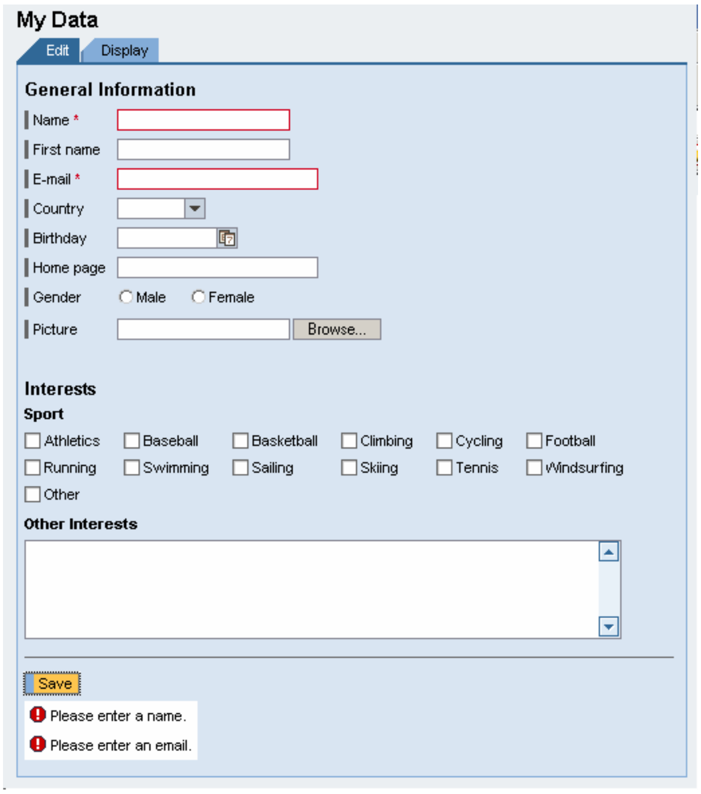
Open the context menu for the project node (**`TutWD_UI_Init`**) in the Web Dynpro Explorer and choose *Rebuild Project.*

In the Web Dynpro Explorer, open the context menu for the application object **`UIApp`**, and then choose *Deploy new archive and run*.

Result



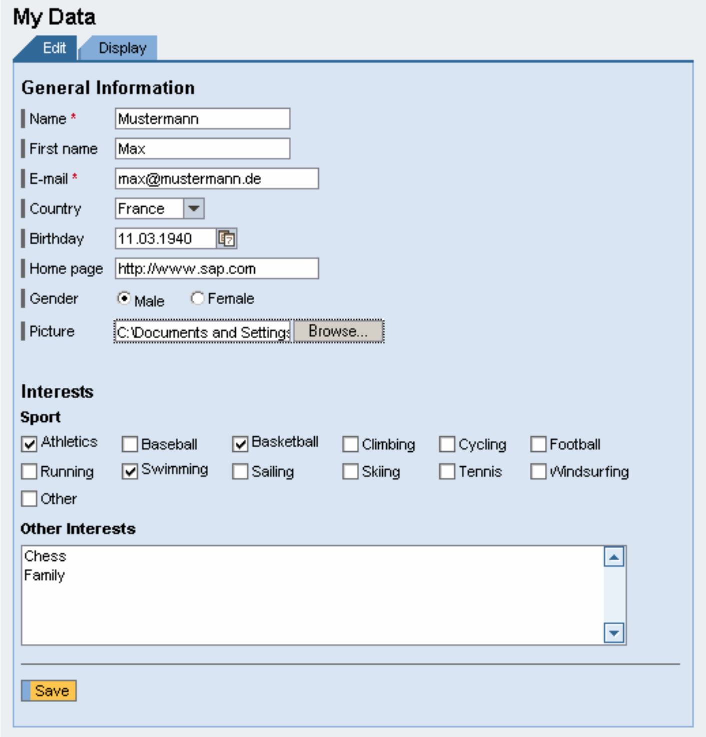
If you save the inputs without entering a name or an e-mail address beforehand, the error messages will appear at the lower left margin of the screen.



Please enter a name.

Please enter an email.

After you have made all the necessary inputs, you can save the application. The content of the display tab will then be displayed to you.



Name * Mustermann

First name Max

E-mail * max@mustermann.de

Country France

Birthday 11.03.1940

Home page http://www.sap.com

Gender ● Male ○ Female

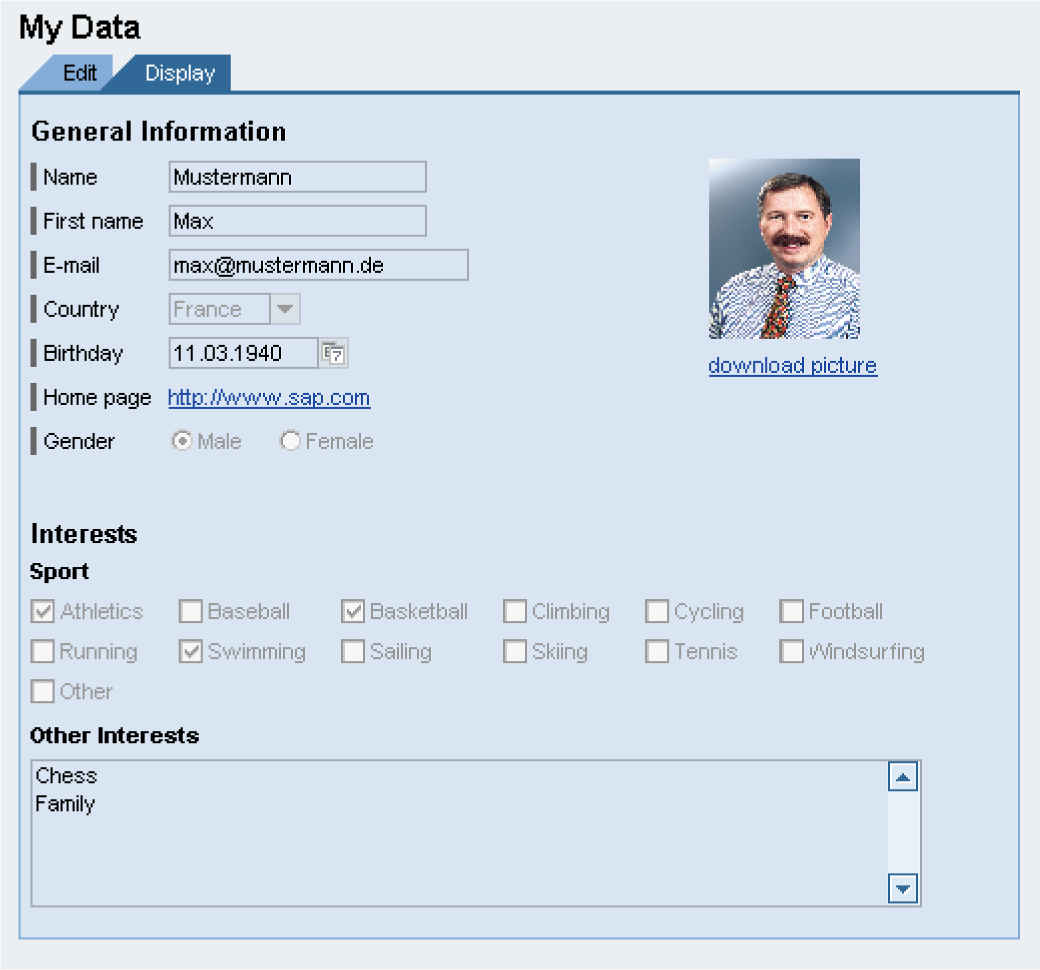
Picture C:\Documents and Settings

☑ Athletics □ Baseball ☑ Basketball □ Climbing □ Cycling □ Football

□ Running ☑ Swimming □ Sailing □ Skiing □ Tennis □ Windsurfing

□ Other

Chess
Family



My Data

General Information

Name: Mustermann

First name: Max

E-mail: max@mustermann.de

Country: France

Birthday: 11.03.1940

Home page: http://www.sap.com

Gender: ● Male ○ Female

Interests

Sport

☑ Athletics ☐ Baseball ☑ Basketball ☐ Climbing ☐ Cycling ☐ Football

☐ Running ☑ Swimming ☐ Sailing ☐ Skiing ☐ Tennis ☐ Windsurfing

☐ Other

Other Interests

Chess
Family

The selected screen can only be displayed if it is in the appropriate *mimes* folder of the component.