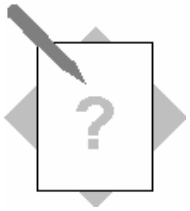
Tutorial

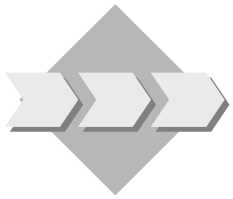


Interactive Forms Integration into Web Dynpro for Java

Topic: Offline interactive form based on download/upload functionality

At the conclusion of this exercise, you will be able to:

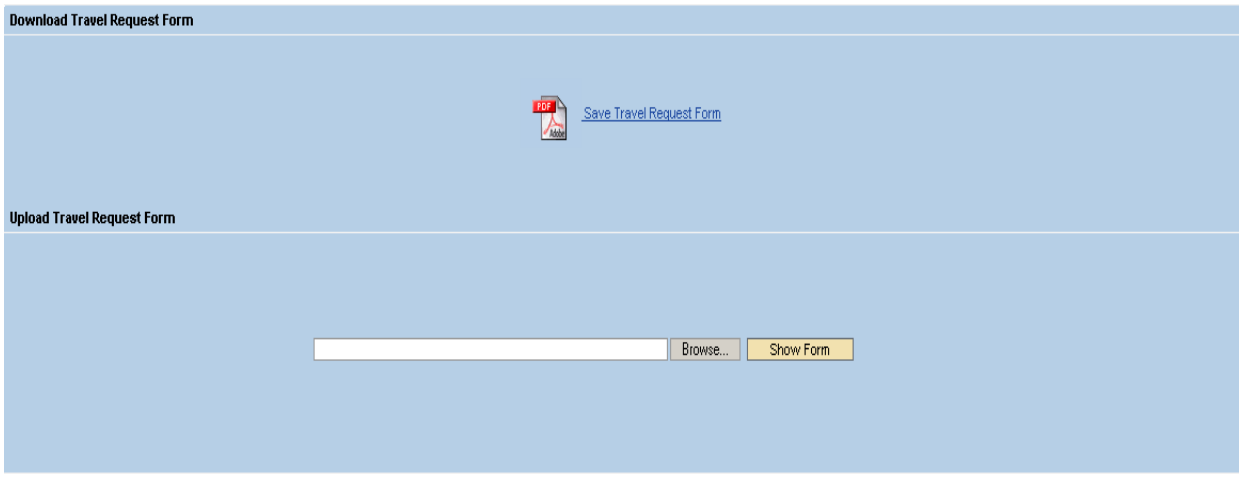
- Provide download functionality within a Web Dynpro application
- Provide upload functionality within a Web Dynpro application
- Integrate an interactive form within a Web Dynpro application



Think of a company where employees need to fill out forms offline. Offline in this case means that the employees make their form entries on their local host without a server connection (e.g. at home, at a customer site, while traveling, etc.).
This tutorial explains how to develop an offline interactive form scenario based on download/upload functionality. In this case, an employee downloads a form to his local computer, makes entries offline, and, after returning to his office, uploads the filled-out form. In a follow-up step, the data can be extracted and passed to a backend system. You can use the Adobe integration within Web Dynpro for this purpose. This exercise will show you which steps are necessary to develop such a scenario based on a travel request form.

1 Overview

The following screenshot shows the user interface of the Web Dynpro application you will develop in this exercise. In the upper view, you can download an interactive form to make your entries offline. In the lower view, you can upload the filled-out form.



2 Prerequisites

To be able to use PDF forms in Web Dynpro applications, the following prerequisites apply:

2-1 The SAP NetWeaver Developer Studio (Support Package Stack 11) including Adobe LiveCycle Designer (forms design tool) is installed on your computer. Use the SAP NetWeaver 04 installation CDs/DVDs if it is not yet installed.

2-2 You have access to the SAP J2EE Engine (Release 6.40).
(Note that you can download an evaluation version (Sneak Preview SAP Web Application Server 6.40 Java) from the SDN Download Area at https://www.sdn.sap.com/sdn/downloadarea.sdn.)

2-3 Adobe Reader 7.0.1 is installed on your computer. If this is not the case, download Adobe Reader 7.0.1 from the Adobe homepage (http://www.adobe.com) at http://www.adobe.com/products/acrobat/readstep2.html).

2-4 The Active Component Framework (ACF) of the Interactive Forms integration is installed on your computer.
With SP Stack 11, calling a Web Dynpro application that includes a PDF form for the first time should automatically install the ACF in the background. If you experience difficulties that may be related to the frontend installation, please see SAP Note 766191 on the SAP Service Marketplace for manual installation.

2-5 The Adobe document services are configured on the SAP J2EE Engine you are using. To access the Installation and Configuration Guides for Adobe document services, go to http://service.sap.com/nw04installation → SAP Web AS → SAP Web AS 6.40 SR1 and Related Documentation → Adobe Document Services.

2-6 Basic knowledge of developing Web Dynpro applications. For helpful information about Web Dynpro, go to https://www.sdn.sap.com/sdn/developerareas/webdynpro.sdn?node=linkDnode6-2.

3 Importing a Project Template

To restrict the development of this sample application to the actual content covered, there is a predefined Web Dynpro project template available in SDN under Web Dynpro Sample Applications and Tutorials. Follow the steps listed below to import the predefined Web Dynpro project:

3-1 Unzip the contents of the ZIP file *TutWD_UploadDownloadInteractiveForm_Init.zip* into the work area of the SAP NetWeaver Developer Studio or into a local directory.

3-2 Start the SAP NetWeaver Developer Studio.

3-3 Import (*File → Import → Existing Project into Workspace*) the Web Dynpro project *TutWD_UploadDownloadInteractiveForm_Init*.

3-4 The Web Dynpro project *TutWD_UploadDownloadInteractiveForm_Init* then appears in the Web Dynpro Explorer for further processing and editing in the context of this tutorial. The information display triggered by the Web Dynpro project *TutWD_UploadDownloadInteractiveForm_Init* can be ignored at this time, since we will extend the Web Dynpro project during the remainder of this exercise and will thus remove this information.

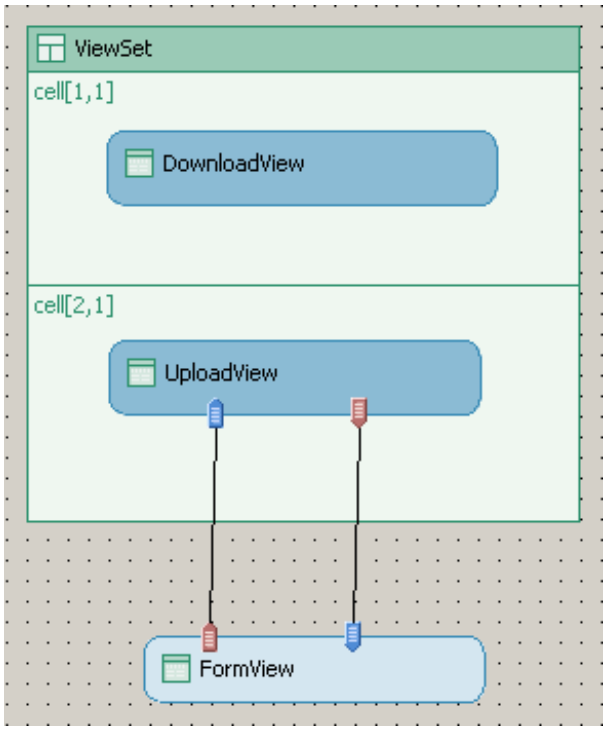
Note: Depending on which Java compiler preferences are set, you may see some warnings in the *Task* view after importing the project. If the severity level for problems of type *Unused imports* (set in *Window → Preferences → Java → Compiler*) has the value *Warning*, the compiler will issue a warning for unused import references. Ignore these warnings!

4 Initial Project Structure

After you have imported the Web Dynpro project template *TutWD_UploadDownload InteractiveForm_ Init*, the following project structure is displayed in the Web Dynpro Explorer:

Web Dynpro project structure
Web Dynpro project: **TutWD_UploadDownloadInteractiveForm_Init**
Web Dynpro application: **UploadDownloadInteractiveFormApp** The application *UploadDownloadInteractiveFormApp* displays the interface view of the Web Dynpro component *UploadDownloadInteractiveFormComp* in the browser window.
Web Dynpro component: **UploadDownloadInteractiveFormComp** This is the Web Dynpro component that contains our entire application.
View: **DownloadView** In this view, the download functionality is provided. View: **UploadView** In this view, the upload functionality is provided. View: **FormView** This view displays the content of the uploaded interactive form.
Window: **Window** Initially contains a view set with two view areas represented by two cells (ViewSet Definition: GridLayout with one column and two rows). The two views *DownloadView* and *UploadView* are already embedded. The *FormView* will become visible after the form was uploaded, and you can return from the *FormView* to the *UploadView*.

In the *Navigation Modeler*, which you can reach by double-clicking on the *Window* node, the initial Web Dynpro project looks as follows:



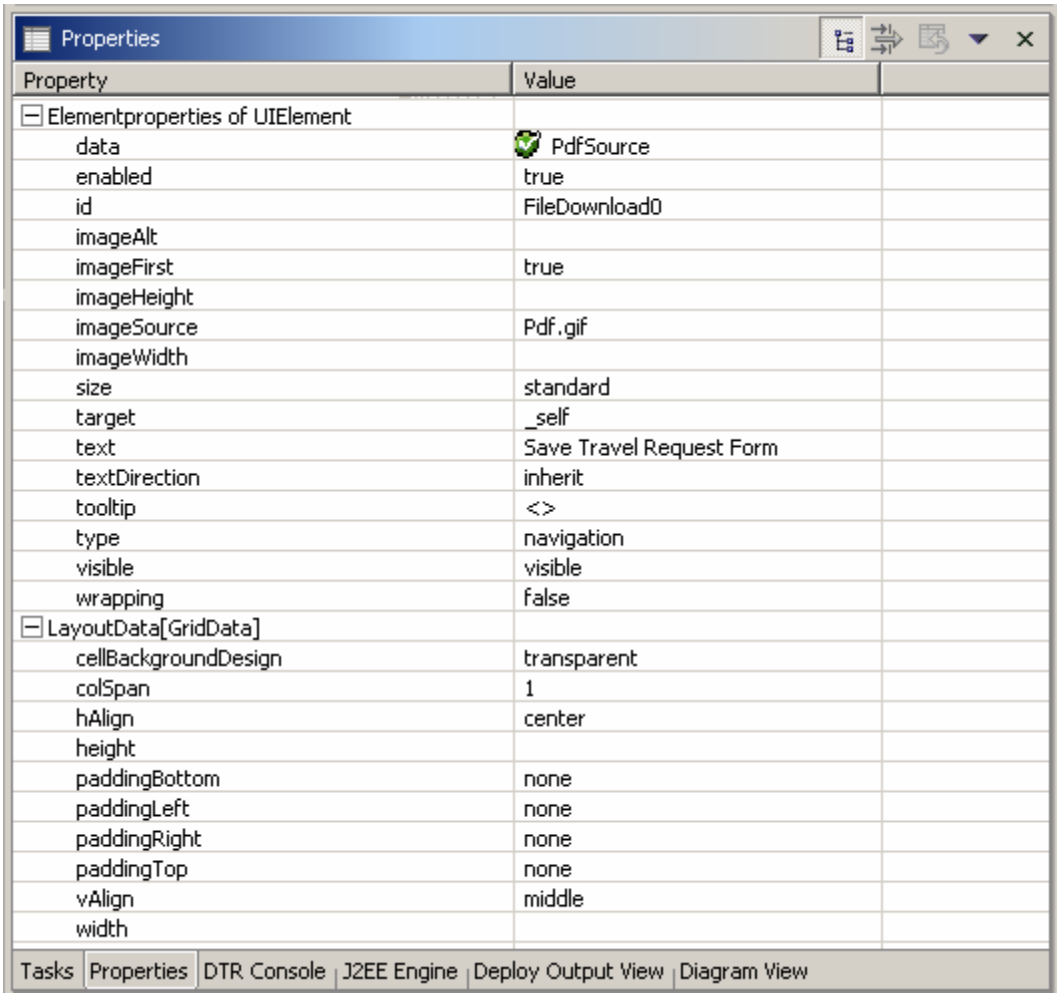
The *Window* contains a *ViewSet*, which consists of 2 view areas.
The upper view area contains the *DownloadView* and the lower one the *UploadView*.
The *FormView* is plugged via an inbound and an outbound plug to and from the *UploadView*.

5 Provide File Download Functionality

5-1 In the *Web Dynpro Explorer*, double-click the node for the *DownloadView* (*TutWD_Upload DownloadInteractiveForm_Init → Web Dynpro → Web Dynpro Components → UploadDownloadInteractiveFormComp → Views → DownloadView*), and choose the *Context* tab of the *DownloadView*.

5-2 Create a *Value Attribute* called *PdfSource* of type *binary*. The type can be changed in the *Properties* tab of this attribute.

5-3 Choose the *Layout* tab of the *DownloadView*. In the *Outline* view, create a new child in the *Group4* called *FileDownload0* of type *FileDownload*, and specify the properties for *data*, *imageSource*, *target*, *text*, *hAlign* and *vAlign* on the corresponding *Properties* tab according to the screenshot below:



Note that the *PdfSource* attribute will be used for the *FileDownload* element. The option *target* with attribute *_self* defines the browser window, which corresponds to the *FileDownload* element.

5-4 Using the code lines listed below, the *PdfSource* attribute is prefilled with the empty travel request form, which can be found within the *Navigation* view under the corresponding component folder in the mimes directory of the source folder *(→ src → mimes → Components → com.sap.tut. wd.uploaddownloadinteractiveform. UploadDownloadInteractiveFormComp*).
The prefilling is done during the initialization of the *DownloadView*, so that it is possible after the initialization process to download the empty PDF form with the help of the *FileDownload* element. To make this possible, choose the *Implementation* tab of the *DownloadView*, and then add the following source code at the end of the *wdDoInit* method:

```
IWDAttributeInfo attInfo =
      wdContext.currentContextElement().node().getNodeInfo().
      getAttribute("PdfSource");
ISimpleTypeModifiable type = attInfo.getModifiableSimpleType();
IWDModifiableBinaryType binaryType = (IWDModifiableBinaryType) type;
binaryType.setFileName("TravelRequest.pdf");  // Set file name
binaryType.setMimeType(WDWebResourceType.PDF);  // set the mime type

String fileName =
"temp\\webdynpro\\web\\local\\TutWD_UploadDownloadInteractiveForm_In
it\\Components\\com.sap.tut.wd.uploaddownloadinteractiveform.UploadD
ownloadInteractiveFormComp\\TravelRequest.pdf";
// path to the source file

/* converts the binary file to a byte array and moves it to the
context element */
try
{
  File file = new File(fileName);
  FileInputStream in = new FileInputStream(file);
  ByteArrayOutputStream out = new ByteArrayOutputStream();
  int length;
  byte[] part = new byte [10 * 1024];
  while ((length = in.read(part)) != -1)
  {
    out.write(part, 0, length);
  }
  in.close();
  wdContext.currentContextElement().
      setPdfSource(out.toByteArray());
}
catch(Exception e)
{
  throw new WDRuntimeException(e);
}
```

5-5 On the Implementation page of the *DownloadView*, choose *Source* → *Organize Imports* from the context menu. This will automatically add all necessary import statements to your source code.

5-6 Save the new metadata by choosing the  (*Save All Metadata)* icon from the toolbar.

6 Provide File Upload Functionality

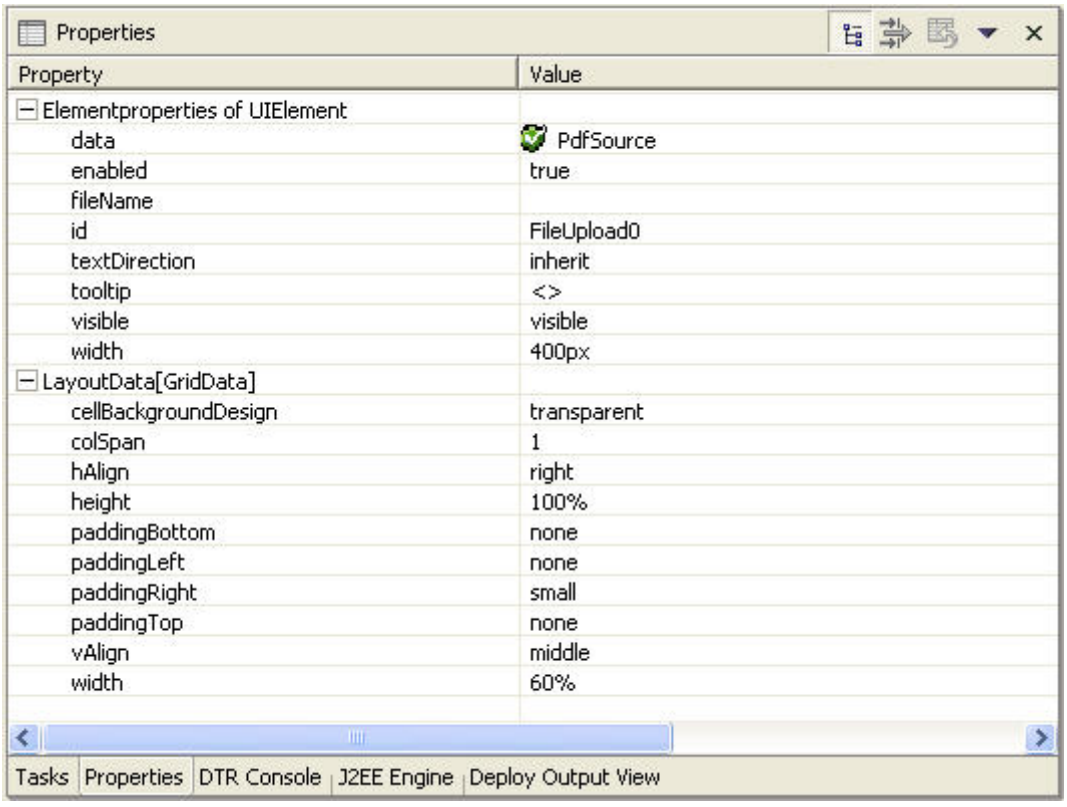
6-1 In the *Web Dynpro Explorer*, double-click the node for the *UploadView* (*TutWD_Upload DownloadInteractiveForm_Init → Web Dynpro → Web Dynpro Components → UploadDownloadInteractiveFormComp → Views → UploadView*), and choose the *Context* tab of the *UploadView*.

6-2 Create a *Value Attribute* called *PdfSource* of type *binary*. The type can be changed on the *Properties* tab of this attribute. This attribute is used as a data container for the upload process.

6-3 Choose the *Implementation* tab of the *UploadView*, and add the following code to the *wdDoInit* method:

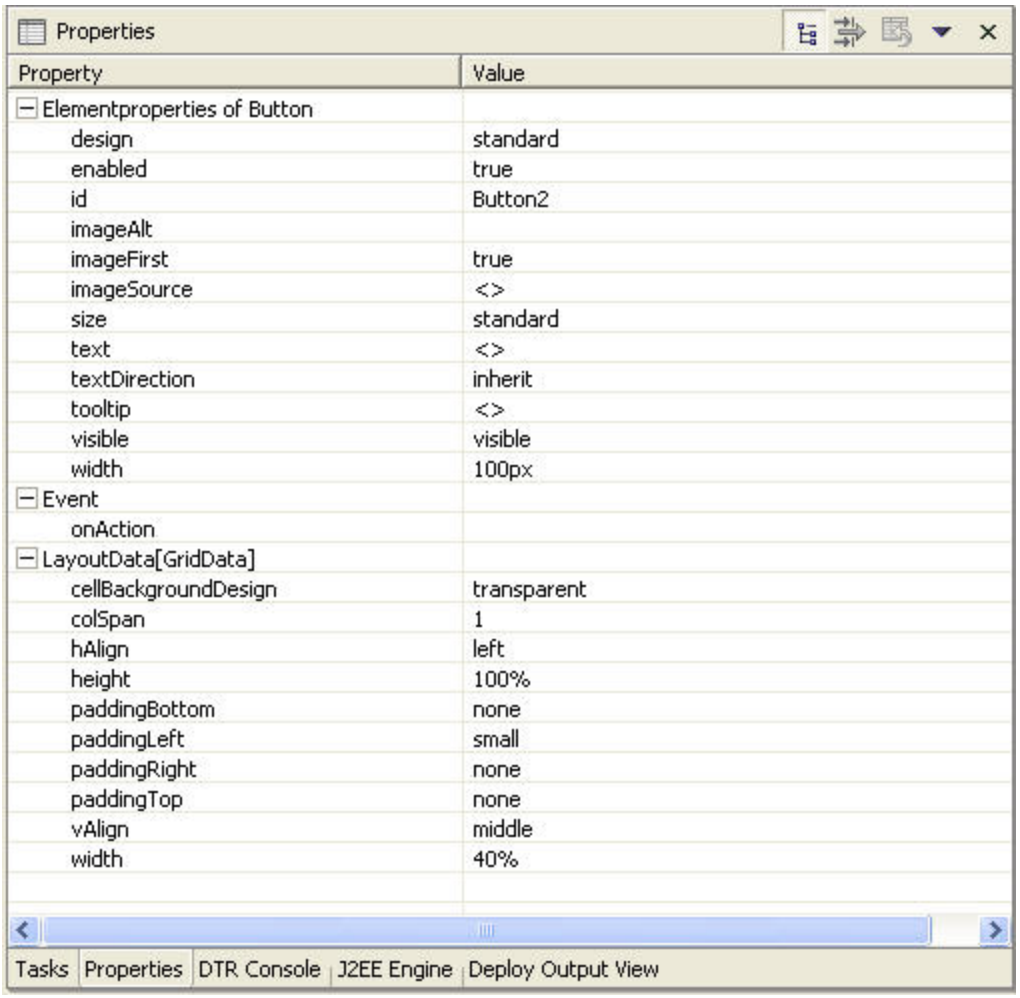
```
wdContext.getNodeInfo().getAttribute("PdfSource").
            getModifiableSimpleType();
```

6-4 Choose the *Layout* tab of the *UploadView*. In the *Outline* view, create a new child for *Group5* called *FileUpload0* of type *FileUpload*, and specify the properties for *data*, *width*, (and under *LayoutData*) *hAlign, height, paddingRight, vAlign*, and *width* in the corresponding *Properties* tab according to the screenshot below:



Note that the *PdfSource* attribute is used for the *FileUpload* element.

6-5 Create a new button called *Button2* as a child of *Group5* in the *Outline* view. By pressing this button, the selected document is uploaded to the *PdfSource* attribute. Set the properties *width*, (and under *LayoutData*) *hAlign*, *height*, *paddingLeft*, *vAlign*, and *width* in the corresponding *Properties* tab according to the screenshot below:



6-6 On the *Actions* tab of the *UploadView*, create a new action called *ShowFormPressed* with the text *Show Form*. Choose *ToFormView* as the fire plug. Click *Finish*.

6-7 On the *Properties* tab for *Button2*, assign this new action to the *onAction* property. To do so, select *Button2* in the *Outline* view, and then choose the *Properties* tab.

6-8 Choose the *Implementation* tab of the *UploadView*, and add the following code lines at the beginning of the *onActionShowFormPressed* method:

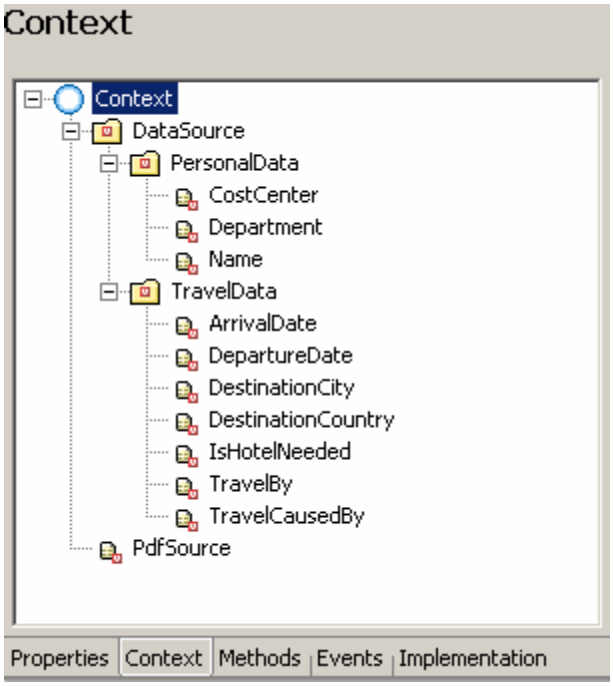
```
wdThis.wdGetUploadDownloadInteractiveFormCompController().
            wdGetContext().currentContextElement().
            setPdfSource(
            wdContext.currentContextElement().getPdfSource());
```

This code copies the uploaded PDF file to the corresponding component controller's context element, which is used in the *FormView* to display the uploaded PDF document.

7 Display Interactive Form

Create FormView Context

The data for the travel request form is already defined in the component controller *UploadDownloadInteractiveFormComp* (In the *Web Dynpro Explorer*, double-click the node for the *UploadDownloadInteractiveFormComp* (*TutWD_UploadDownloadInteractiveForm_Init → Web Dynpro → Web Dynpro Components → UploadDownloadInteractiveFormComp*), and choose the *Context* tab of the *UploadDownloadInteractiveFormComp*).



For displaying the data in the *FormView*, we need to set up the corresponding context structure.

7-1 Open the *Data Modeler* of the *UploadDownloadInteractiveFormComp* in the *Web Dynpro Explorer* view by double-clicking on the *UploadDownloadInteractiveFormComp* node. To edit the data link between *FormView* and the *Component Controller*, double-click the arrow between both elements.

7-2 Drag and drop the *DataSource* node on the right side onto the *Context* node on the left, check all boxes and confirm with *OK*. Do the same with the *PdfSource* attribute. Choose *Finish*. As a result, the context structure in the *FormView* is built and mapped to the component controller.

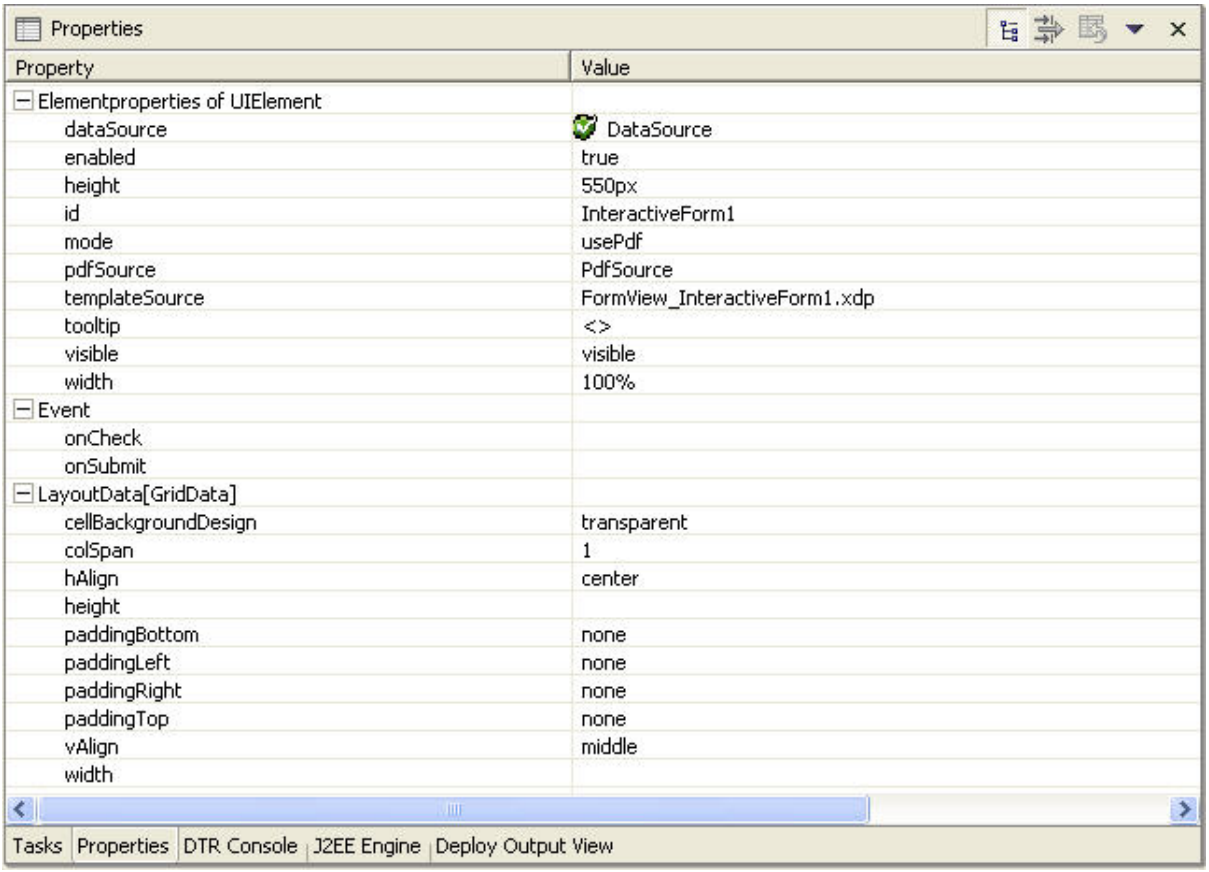
Design the FormView Layout

In this section, we will extend the *FormView* with an interactive form element and a toolbar button. The interactive form element is used for displaying the form and the button is used for returning to the *UploadView*.

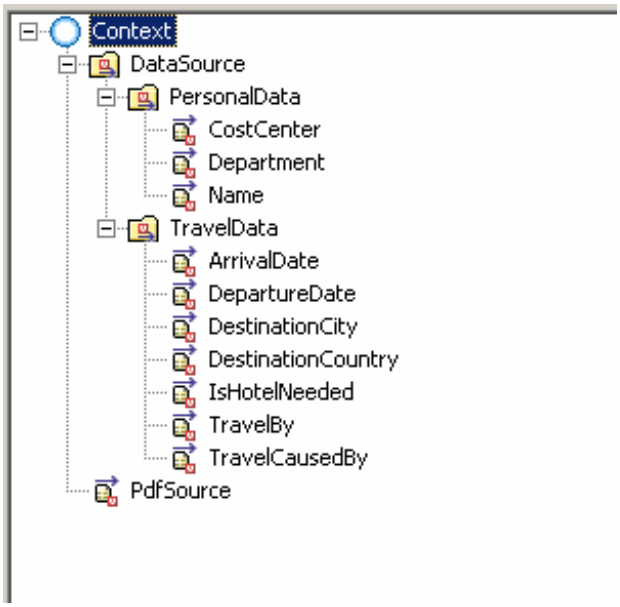
7-3 In the *Web Dynpro Explorer*, double-click the node for the *FormView* (*TutWD_UploadDownloadInteractiveForm_Init* → *Web Dynpro* → *Web Dynpro Components* → *UploadDownloadInteractiveFormComp* → *Views* → *FormView*), and choose the *Layout* tab of the *FormView*.

7-4 In the *Outline* view, insert a child of type *InteractiveForm* and ID *InteractiveForm1* in *Group2*

7-5 Select the *InteractiveForm1* element in the *Outline* view, switch to the corresponding *Properties* tab, and specify the properties as shown in the screenshot below. Reference the corresponding *dataSource* and *pdfSource* from the context, and set *height, mode, width*, *hAlign,* and *vAlign*:



The *dataSource* property is used to specify the data source. The data source encapsulates the data you can display in the form at runtime. For the *dataSource* property, you need to specify the path to the context node providing the data. In this step, we reference the *DataSource* context node, which is already defined in the context structure of the template we use (see *Context* tab of the *FormView*):



For more information related to context structures refer to the tutorial linked below, which you can find in the standard SAP Library documentation:

Application of Context Programming and Data Binding

The *usePdf* value of the mode property is used for displaying the original PDF document. The data source and the template for the creation of the PDF document are ignored.

The *pdfSource* property specifies the path of the context element that contains the PDF document. You must bind this property to a context attribute of type *binary*. In this tutorial, the context attribute has already been defined in the context structure (see graphic above). This property allows an application developer to access the binary file and download it to the local hard disk or read and send the data to a backend.

7-6 In the *Outline* pane, insert a *ToolBarItem* of type *ToolBarButto*n to the ToolBar, and call it *SubmitButton.*

7-7 Create a new action called *SubmitPressed* on the *Actions* tab of the *FormView*, and choose *ToUploadView* as the fire plug. Click *Finish*.

7-8 Choose the *Layout* tab, and select the *SubmitButton* in the *Outline* view. On the *Properties* tab for the *SubmitButton*, assign this new action to the *onAction* property, and set the *text* property to *Submit*.

7-9 Save the new metadata by choosing the (*Save All Metadata)* icon from the toolbar.

8 Building, Deploying and Running the Project

To complete the tutorial, *Rebuild Project* and *Deploy New Archive and Run*. A browser window displays the user interface that you know from the overview section. You can download the travel request form and make your entries offline with the stand-alone Adobe Reader. At a later stage, you can upload the filled-out form and display it.

Congratulations, you have completed the offline interactive form scenario based on download/upload functionality.