How to Extract Data from an Offline Interactive Form using XML Parsing in Web Dynpro Java

Applies to:

This article applies to WebDynpro Java/Adobe Interactive Forms development in SAP NetWeaver 7.0 and Adobe LiveCycle Designer 8.0 or higher release. Demo application created for this article has been developed in SAP NetWeaver 7.0 EHP1. For more information, visit the Web Dynpro Java homepage.

Summary

This article tells about how to extract data from an Offline Interactive Form using XML parsing based on upload functionality in WebDynpro Java.

Author: Suresh Thiyagarajan

Company: Tata Consultancy Services

Created on: 24 November 2010

Author Bio

Suresh Thiyagarajan is a SAP NetWeaver Consultant working for Tata Consultancy Services, India. His areas of expertise in SAP technologies include WebDynpro Java, WebDynpro ABAP, Enterprise Portal and Adobe Interactive Form development.

Table of Contents

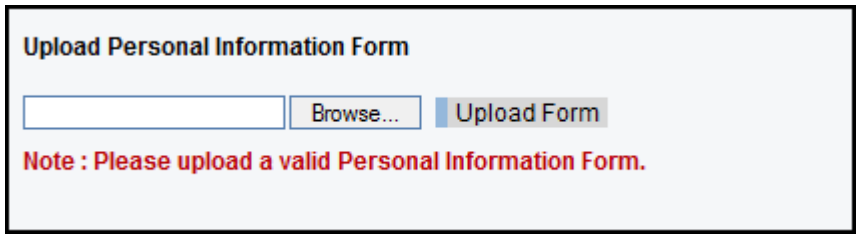
Introduction

This is the final part of a three part series on Adobe Interactive Forms development. In Part 1, we have seen how to add and remove subform instances at runtime. In Part 2, we have seen how to extract data from subform instances using XML parsing in an Online Interactive Form through WebDynpro Java. In part 3, we will discuss how to extract data from an Offline Interactive Form in WebDynpro Java. Extracted data from the Adobe Form will be stored in a value node and finally updated to R/3 table using a RFC.

Scenario

In this article, the below shown screen will be used to illustrate the offline data extraction. Clicking on *'Browse'* button will open a file browser that lets you select the PDF file from your computer. Upon clicking the *'Upload Form'* button, the data will be extracted from the Adobe Form and stored in a WebDynpro value node.

The file upload feature of WebDynpro will be used to retrieve the contents of the offline PDF. XML parsing methods will be then used to extract the data. We will continue to use the *Personal Information Form* from part 2 in this article as well.



Note: The PDF file being uploaded has to be of the same type as the Personal Information Form in part 2.

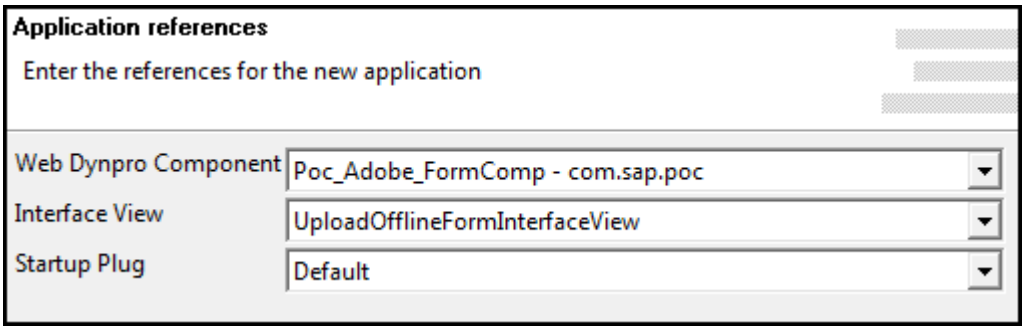
WebDynpro Development Component

We will continue to use the DC *poc_adobe_form* from part 2, but let us create a new application, window and view for this part.

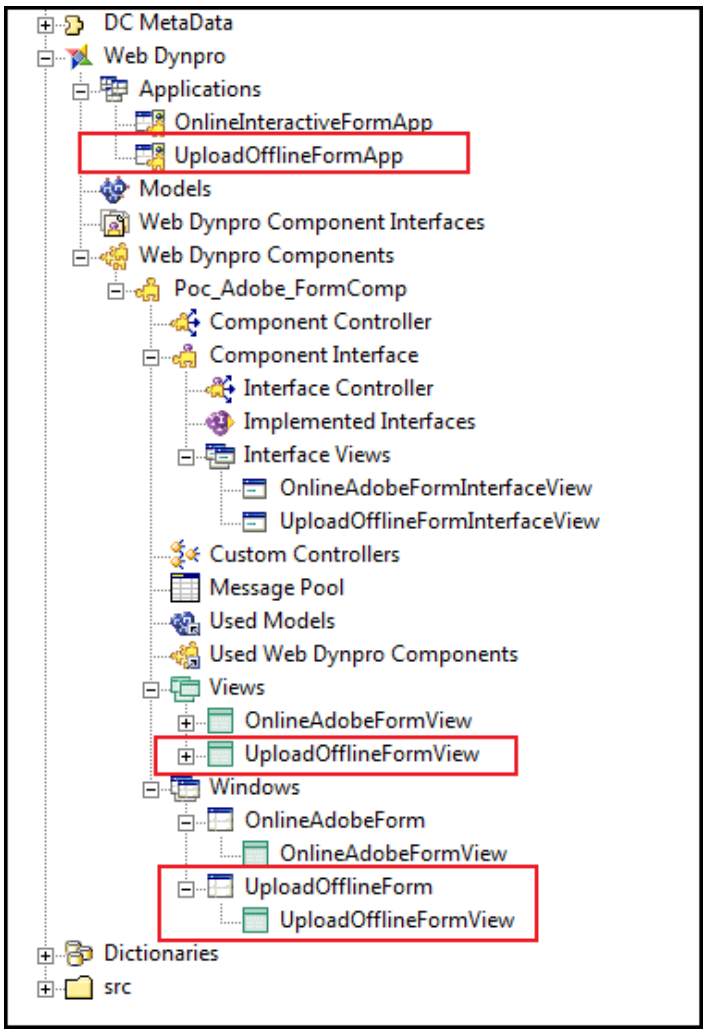
View – A new view *UploadOfflineFormView* will be used to design the above sample screen.

Window – A new window *UploadOfflineForm* will be used to embed the view '*UploadOfflineFormView'.*

Application – *UploadOfflineFormApp* will be created with the reference of WebDynpro Component and Interface View as shown below.

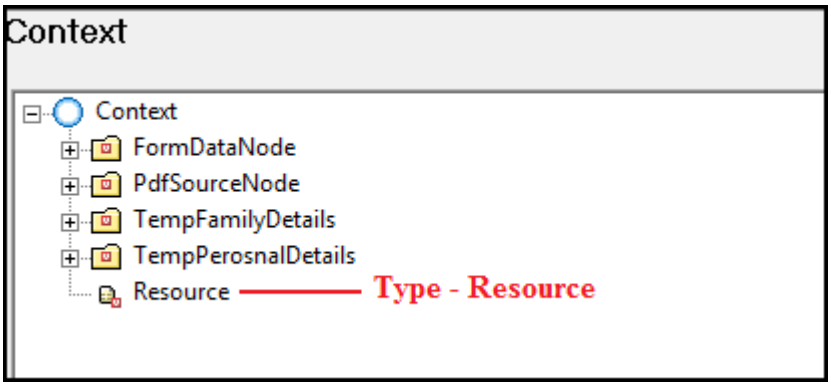


A diagram of the Development Component is given below for reference.



Creating context structure

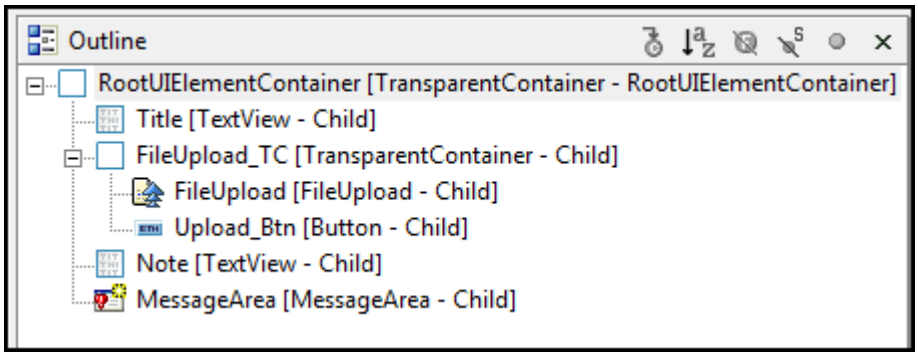
For this part, an additional attribute has to be created to store the uploaded PDF file. Details of the attribute are given below.



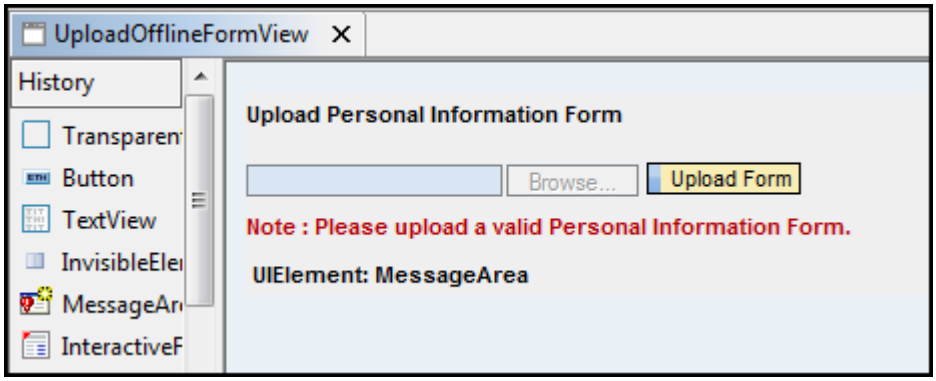
Resource – The *resource* property of the FileUpload UI element is bound to this attribute; it holds the file data for the PDF document at runtime.

Creating UI elements in the view

Open the *Layout* tab of the view '*UploadOfflineFormView'* and design the view with *FileUpload* and *Button* UI elements in *Outline* tab as shown below.



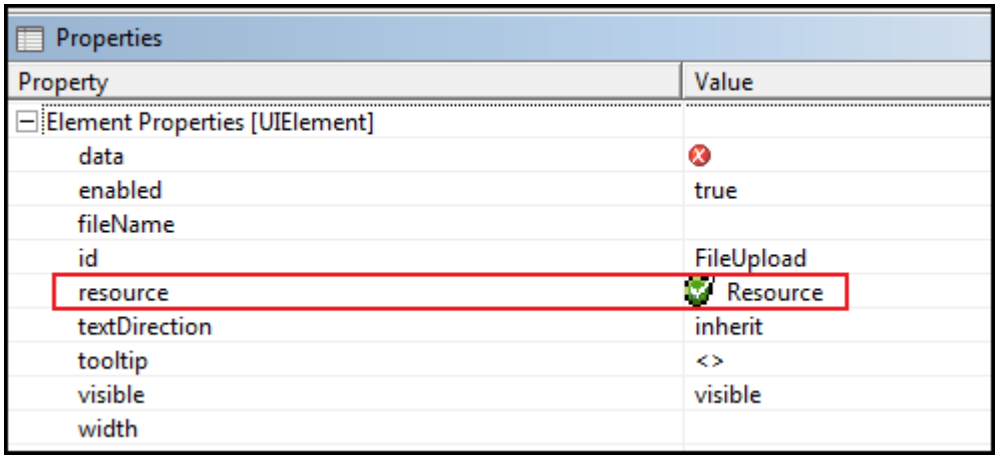
Below screen shows design time Layout of the view *'UploadOfflineFormView',*



Data binding between UI element and context

Select *FileUpload* UI from *Outline* tab and go to *Properties* perspective.

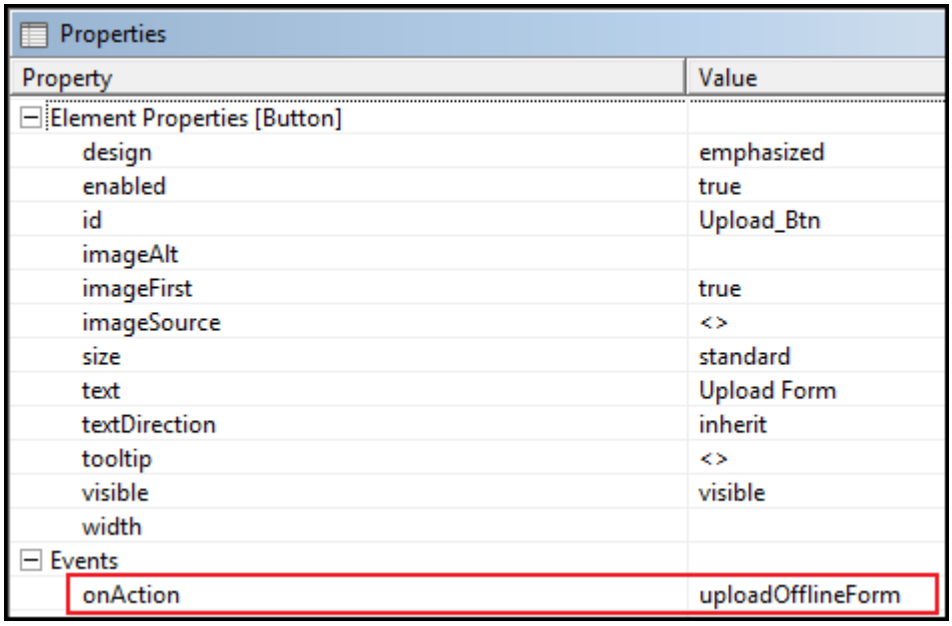
Bind the *resource* property of the *FileUpload* UI to the new attribute *Resource* as shown below.



Creating action handler for 'onAction' event in the Button UI

Now an action handler for *'onAction'* event of the button '*Upload Form'* needs to be created in the WebDynpro view.

For this go to *Upload_Btn* UI → *Properties* tab → select *'onAction'* event → Create an action handler with name *'uploadOfflineForm'*.



Switch to the '*UploadOfflineFormView'* view and select the *Implementation* page.

IWDMessageManager class is used to display Success or Error message. To access this message class globally either in the view or component controller, create an object in the component controller (//@@begin others) and initialize the method in *wdDoInit ()* as shown below.

```
//@@begin others
IWDMessageManager msgMgr = null;
//@@end
```

```
public void wdDoInit ()
{
  //@@begin wdDoInit ()
  msgMgr = wdComponentAPI.getMessageManager();
  //@@end
 `
```

Add the following source code to the '*uploadOfflineForm'* action handler.

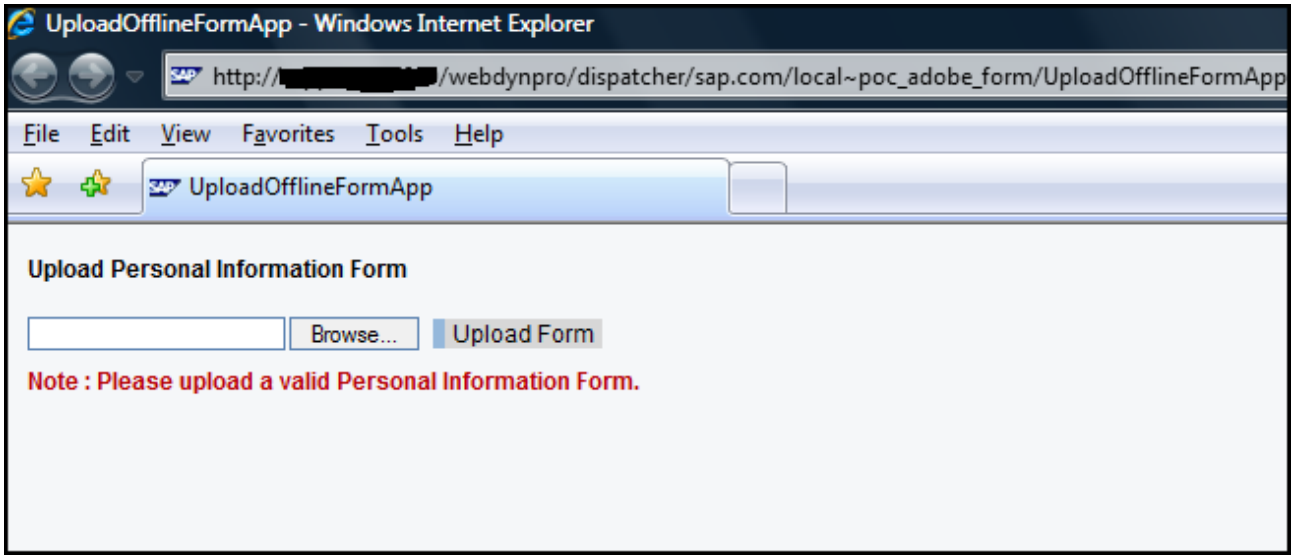
Code Snippet of 'onActionuploadOfflineForm()' action handler

```
public void onActionuploadOfflineForm(com.sap.tc.webdynpro.progmodel.api.IWDCustomEvent wdEvent
)
{
//@@begin onActionuploadOfflineForm(ServerEvent)
try{
   wdContext.currentPdfSourceNodeElement().setPdfSource(null);
   wdContext.nodeFormDataNode().invalidate();
   //    if resource context attirbute is null, then raise a exception
   if(wdContext.currentContextElement().getAttributeValue("Resource")!=null)
   {
        //   assign resource type to IWDResource interface to get file extension
        IWDResource fileResource = wdContext.currentContextElement().getResource();
        //   if uplaoded file is not a PDF file, then raise a exception
       if(fileResource.getResourceType().getFileExtension().equalsIgnoreCase("PDF"))
       {
          //   reading no. of bytes from input stream of the resource attribute
          byte[] b = new byte[wdContext.currentContextElement().getResource().read(false).available()];
          //   reads some number of bytes from the input stream and stores them into the buffer array b
          wdContext.currentContextElement().getResource().read(false).read(b);
          //  displaying the bytes length of the uploaded adobe form
          wdComponentAPI.getMessageManager().reportSuccess("Adobe Form Bytes Length :"+b.length);
          //  store the byte array in the pdfSoruce context attribute
          wdContext.currentPdfSourceNodeElement().setPdfSource(b);
          //  extracts data from a PDF document and copies the values into the WebDynpro Context
          WDInteractiveFormHelper.transferPDFDataIntoContext(wdContext.currentPdfSourceNodeElement
().getPdfSource(), wdContext.nodeFormDataNode());
          // store parsed data from PersonalDetails node to TempPersonalDetails node
          wdContext.currentTempPerosnalDetailsElement().setFirstName(wdContext.currentPersonalDetails
Element().getFirstName());
          wdContext.currentTempPerosnalDetailsElement().setLastName(wdContext.currentPersonalDetails
Element().getLastName());
          //  call the component controller methods
          wdThis.wdGetPoc_Adobe_FormCompController().getFormData();
          wdThis.wdGetPoc_Adobe_FormCompController().displayParsedData();
       }
       else
       {
          msgMgr.reportException("Please select a valid Personal Information Form",false);
       }
    }
    else
    {
        msgMgr.reportException("Please browse a Personal Information Form",false);
    }
 }
 catch(Exception e)
 {
     msgMgr.reportException("Error in uploading the Adobe Form :"+e.getLocalizedMessage(),false);
 }
  //@@end
```

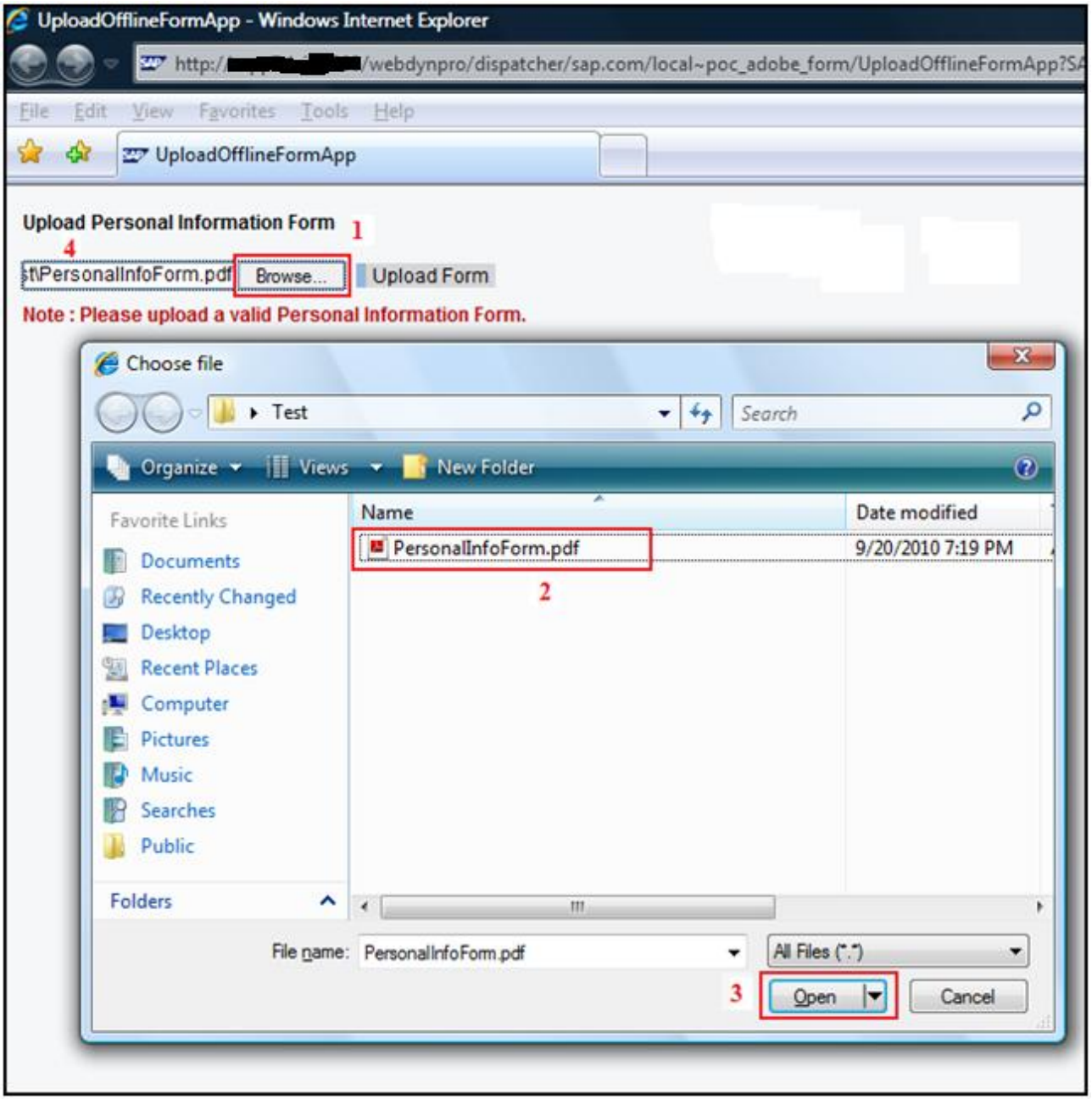
Note: You may check the Part 2 document to get the code snippet for getFormData() and displayParsedData() methods.

Build & Deploy

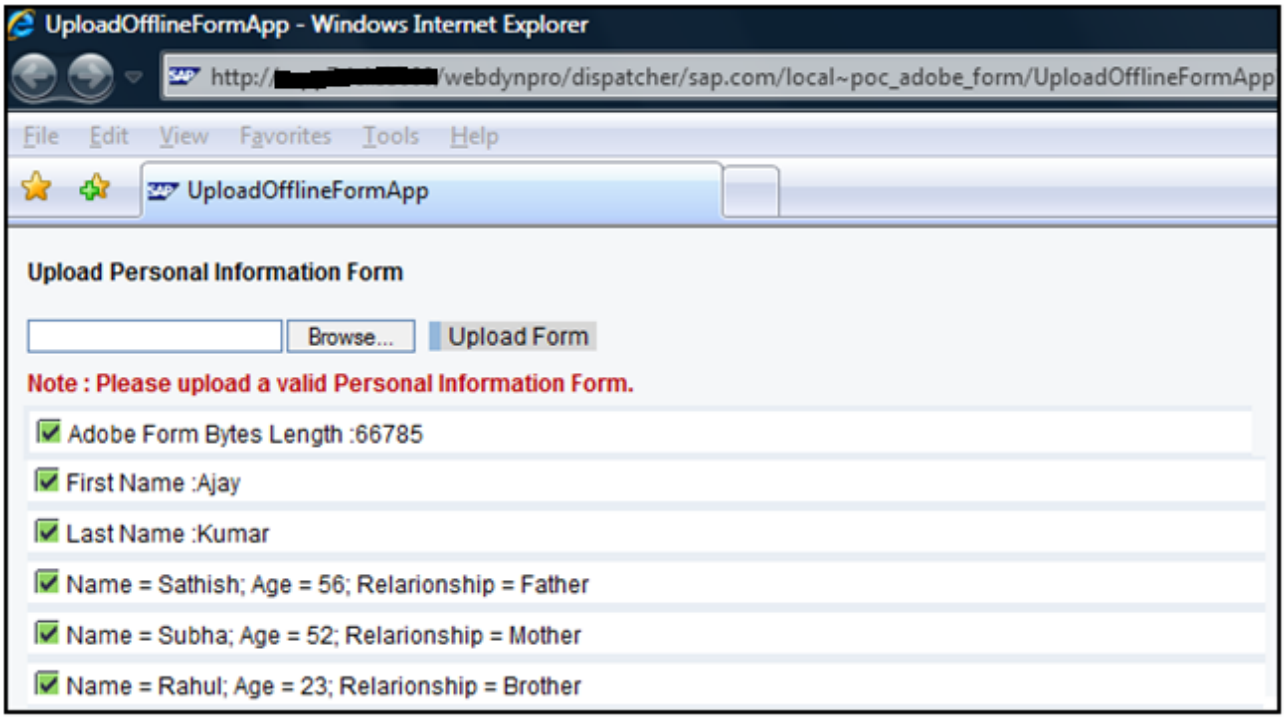
Build and deploy the project from NetWeaver Developer Studio. Upon executing the *UploadOfflineFormApp* application, the Internet Explorer browser should look similar to the screen below.



Clicking on *'Browse'* button opens the file browser.



Upon clicking the *'Upload Form'* button, data is extracted from the Adobe Form and stored in the value nodes in WebDynpro context. The value nodes are then iterated through and the parsed data is displayed using message class as shown below.



Related Content

How to Add or Remove Subform Instance at Runtime in Adobe Interactive Forms

How to Extract Data from an Online Interactive Form using XML Parsing in WebDynpro Java

Parsing XML file in Java

About DOM - Link 1 and Link 2

SAP Interactive Forms by Adobe

User Interface Technology Homepage

For more information, visit the WebDynpro Java Homepage

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.