Disclaimer

Adobe® will support this configuration for development environments. Adobe has not fully tested this environment but will resolve issues that maybe found. Currently Adobe is using this configuration internally for development and does not know of compatibility issues. In future releases WSAD will be a fully tested environment for Form Server and other Adobe Server products.

Table of Contents

About Adobe LiveCycle Forms .........................................................................................................................................................2

Part I: Installing Adobe LiveCycle Forms on WSAD
1.0 Introduction ......................................................................................................................................................................................3
   Product support: Adobe LiveCycle Forms and IBM WSAD 5.1.1 .................................................................................................3
   Overview .............................................................................................................................................................................................3
   Pre-installation notes .........................................................................................................................................................................3
2.0 Install WSAD 5.1.1 with WAS 5.1 Base Edition ..............................................................................................................................4
   Update WSAD to the current fix pack ..............................................................................................................................................4
   Update the WAS 5.1 base edition in WSAD to 5.1.0.3 ..........................................................................................................................4
   Apply the IBM JDK interim patch to WAS 5.1 base edition ..............................................................................................................6
3.0 Install Adobe LiveCycle in WSAD ..................................................................................................................................................7
   Install Adobe LiveCycle Forms ......................................................................................................................................................7
   Deploy Adobe LiveCycle to WSAD test environment .....................................................................................................................7
   Create a new server and server configuration ................................................................................................................................8
   Deploy the Adobe LiveCycle web modules (.war files) .......................................................................................................................8
   Deploy Adobe LiveCycle enterprise applications (.ear files) ..........................................................................................................9
   Start the test application server ......................................................................................................................................................11
4.0 Test Adobe LiveCycle ..........................................................................................................................................................................12
   To deploy idp-servlet .......................................................................................................................................................................12
   To run idp-servlet .........................................................................................................................................................................12
5.0 Install WebSphere Studio/Adobe Designer plug-in ..........................................................................................................................12
   Congratulations you are now working with Adobe LiveCycle Forms on WSAD ............................................................................12

Part II: Installing and Deploying Tutorial Project
1.0 Setting up the Project .....................................................................................................................................................................13
   Adding external libraries .................................................................................................................................................................13
   Importing the .xdp files .................................................................................................................................................................14
   Creating the RenderToPDF servlet ...............................................................................................................................................14
   Inserting code ....................................................................................................................................................................................14
   Creating index.jsp ..............................................................................................................................................................................16
2.0 Running the application in WSAD ................................................................................................................................................16
3.0 Adding the RenderToHTML servlet ..........................................................................................................................................16
   Inserting code ....................................................................................................................................................................................17
   Modifying the RenderToPDF servlet .........................................................................................................................................19
   Modifying index.jsp .........................................................................................................................................................................20
4.0 Running the application ..................................................................................................................................................................20
About Adobe LiveCycle Forms

Overview

Whether users are internal or external, online or offline, you need a way to include them in forms-driven business processes. What’s more, you need to extend those business processes to users without requiring them to purchase additional software or download new plug-ins.

Using Adobe® Form Server, you can extend forms-driven processes beyond the walls of your organization by creating and delivering Adobe Portable Document Format (PDF) or HTML forms. Because these forms take advantage of a universal client—such as the free Adobe Reader® software and Web browsers—they can run on any platform, from PCs to handheld devices.

Adobe Form Server lets your partners, customers, and constituents easily access, fill, and submit online forms or save them locally for offline completion and submission at a later time. Users can submit the forms back into an organization’s core systems via Adobe Form Server to help increase the quality of data in core systems, improve customer service, and leverage existing IT investments.

Easily build dynamic forms based on user input

Using Adobe Form Server, organizations can create dynamic XML-based forms that capture information inside and outside the corporate firewall via universal clients, such as the free Adobe Reader or a Web browser. These forms can dynamically adapt and change in real time based on user input. This ability to provide more-customized forms improves the user experience—and helps ensure that organizations capture the right data.

- Empower business users to easily design forms with a point-and-click designer
- Use PDF to maintain the look and feel of existing forms
- Reduce training costs and increase user adoption by maintaining the familiarity of paper
- Extract XML data from submitted forms for integration with core systems
- Use XML tools to pre-populate fields and personalize forms

Enable secure offline forms completion

Adobe Form Server allows users to take dynamic PDF and HTML forms offline, save them locally, and fill them out at their convenience. Upon completion of the forms, users can go back online to submit them. Offline form completion enables enterprises to extend business processes to places where they could not be used previously, and users do not have to stay online to participate in business processes.

- Ensure authenticity and security of forms and the data they contain
- Allow users to digitally sign, validate, encrypt, and decrypt PDF files on the server
- Improve adoption rates of forms by offering them online and offline, to use inside and outside the firewall, and on any platform
Part I: Installing Adobe LiveCycle Forms on WSAD

1.0 Introduction

In this tutorial, you will learn how to install Adobe LiveCycle on IBM WebSphere Studio Application Developer 5.1.1 (WSAD) on a Windows XP system. Steps are the same for Windows 2003 server. You can create your own web applications and deploy them directly to the test server, which will give you not only hot deploy capabilities, but also the ability to step through code.

It is assumed that you have Adobe® Acrobat® or Adobe® Reader® software installed on your system. If not, please download the free Adobe Reader (6.0.2 or higher) at www.adobe.com.

Note: The installation steps are *all* required. The sequence is important. These steps will update WASD to [WSAD 5.1.1, fix level 3, WAS 5.1 Base (as part of WSAD) fix level 3. JDK patch level 1.4.1.]

Product support: Adobe LiveCycle Forms and IBM WSAD 5.1.1

Adobe will support this configuration for development environments. Adobe has not fully tested this environment but will resolve issues that may be found. Currently, Adobe is using this configuration internally for development and is not aware of compatibility issues.

In future releases, WSAD will be a fully tested environment for LiveCycle Forms and other Adobe Document Services.

Overview

- Install WSAD 5.1.1 with WAS 5.1 Base Edition.
- Update WSAD to the current fix pack.
- Update the WAS 5.1 Base Edition in WSAD to 5.1.0.3
- Apply the IBM JDK interim patch to WAS 5.1 Base Edition.
- Install the WebSphere Studio/Adobe Designer plug-in
- Install Adobe LiveCycle Forms, and test the product.

Pre-installation notes

Select: Control Panel > System > Advanced > Environment Variables

Set System Environment variables:
JAVA_HOME must be set to the IBM VM
[C:\IBM_WSAD\Application
Developer\v5.1.1\_jvm\jre]

The PATH variable should start with
[C:\IBM_WSAD\Application
Developer\v5.1.1\_jvm\jre\bin]
2.0 Install WSAD 5.1.1 with WAS 5.1 Base Edition.

We recommend setting the following base directory for the installation of WSAD 5.1.1:

[C:\IBM_WSAD\Application Developer\v5.1.1\]

A WSAD trial version is available from IBM here:


Please follow the IBM supplied instructions for installation.

Update WSAD to the current fix pack

Launch WSAD. When it has fully launched select Help > New Updates.

WSAD will search for updates.

Install the updates.

Allow WSAD to restart.

Shut down WSAD.

Update the WAS 5.1 base edition in WSAD to 5.1.0.3

IBM WebSphere Application Server, Version 5.1.0.3 Cumulative Fix for Windows

The Patch is located on the IBM site at:
Click here to get the Patch
(http://www-1.ibm.com/support/docview.wss?rs=0&q1=was+5.1.0.3&uid=swg24006675&loc=en_US&cs=utf-8&cc=us&lang=en)

Download the Windows Base version. You may also need to get the Update installer for 5.1 which is located here:
http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg24006036

Follow the instructions to install.

Example:
Extract [C:\Software\IBM\was510 Cf3_win.zip to c:\IBM_WSAD\patch]
Run [C:\IBM_WSAD\patch\updateWizard.bat]

Note: Make sure that the installer does not locate an existing install of WebSphere, select the “Specify product information” check box, and point the installer to the base_v51 folder in WSAD, or if the Update Wizard points to the wrong directory, check Specify product information… and click on Browse button. We pointed the installer to:
[C:\IBM_WSAD\Application Developer\v5.1.1\runtimes\base_v51]

Select to Install fix packs and browse to the [path/fixpacks] directory.
Install should look as follows:
Apply the IBM JDK interim patch to WAS 5.1 base edition

Install the JDK patch from IBM - PQ88973_win -- JDK iFix-- SDK 141 SR2b for WebSphere Application Server 5.1. Windows Patch, located on the IBM site:

Get the Patch and the Updater
(http://www-1.ibm.com/support/docview.wss?rs=180&context=SSEQTP&uid=swg24007337)

Extract the updater to a different directory, such as: C:\IBM_WSAD\PQ88973

Copy the PQ88973_win.jar archive to

    [C:\IBM_WSAD\patch]

Run the updater and install fixes

    [C:\IBM_WSAD\patch\updateWizard.bat]
3.0 Install Adobe LiveCycle in WSAD

Now you can begin the installation of Adobe LiveCycle Forms.

Install Adobe LiveCycle Forms

Select typical install and use the default directory, typically [C:\Program Files\adobe\idp]

Deploy the service infrastructure by copying the following files to the target directories

<table>
<thead>
<tr>
<th>Files to move</th>
<th>Original location</th>
<th>Target directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentServicesLibrary.jar</td>
<td>C:\Program Files\adobe\idp\lib</td>
<td>C:\IBM_WSAD\Application Developer\v5.1.1\runtimes\base_v51\lib</td>
</tr>
<tr>
<td>all DLL files</td>
<td>C:\Program Files\adobe\idp\deployment\binaries</td>
<td>C:\IBM_WSAD\Application Developer\v5.1.1\runtimes\base_v51\installedApps\svcNative</td>
</tr>
</tbody>
</table>

Deploy the Adobe web archives and applications by copying the following files to the target directories

<table>
<thead>
<tr>
<th>Files to move</th>
<th>Original location</th>
<th>Target directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>adobe-FontManager.war</td>
<td>C:\Program Files\adobe\idp\deployment</td>
<td>[C:\IBM_WSAD\Application Developer\v5.1.1\runtimes\base_v51\installable Apps]</td>
</tr>
<tr>
<td>adobe-FormServer.war</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adobe-PDFManipulation.war</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adobe-TrustManager.war</td>
<td></td>
<td></td>
</tr>
<tr>
<td>adobe-XMLForm.war</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DataManagerService.war</td>
<td></td>
<td></td>
</tr>
<tr>
<td>formServer.ear</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Deploy Adobe LiveCycle to WSAD test environment

Launch WSAD.
Create a new server and server configuration

In the J2EE perspective, right-click Servers, select New > Server and Server Configuration. Enter MyServer in Server name field.

Deploy the Adobe LiveCycle web modules (.war files)

1. Select Window > Open Perspective > J2EE.
2. In the navigation tree, right-click on the Enterprise Applications option and select New > Enterprise Application Project. You will need to create a wrapper EAR file for each WAR file that you wish to deploy.
3. In the New Enterprise Application Project Wizard, accept the default “Create J2EE 1.3 Enterprise Application project.”
4. In the project name field, provide a name for the Enterprise Application:

<table>
<thead>
<tr>
<th>FontMS</th>
<th>for adobe-FontManager.war</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSS</td>
<td>for adobe-FormServer.war</td>
</tr>
<tr>
<td>PDFMM</td>
<td>for adobe-PDFManipulation.war</td>
</tr>
<tr>
<td>TMS</td>
<td>for adobe-TrustManager.war</td>
</tr>
</tbody>
</table>
5. Click Next to accept the given name.
6. Do not select anything in the last wizard panel, simply click Finish.
7. Right-Click on the newly created Enterprise Application and select “Import>Import…”. In the Import dialog, select “WAR file” and click Next.
8. Click Browse and navigate to the \[C:\IBM_WSAD\Application Developer\v5.1.1\runtimes\base_v51\installableApps\] directory, and select the module you want to deploy.
9. Click New to create a new Web Module project. In the Project Name field, type the appropriate context root:

<table>
<thead>
<tr>
<th>FontManager</th>
<th>for adobe-FontManager.war</th>
</tr>
</thead>
<tbody>
<tr>
<td>FormServerService</td>
<td>for adobe-FormServer.war</td>
</tr>
<tr>
<td>PDFManipulation</td>
<td>for adobe-PDFManipulation.war</td>
</tr>
<tr>
<td>TrustManager</td>
<td>for adobe-TrustManager.war</td>
</tr>
<tr>
<td>XMLForm</td>
<td>for adobe-XMLForm.war</td>
</tr>
<tr>
<td>DataManager</td>
<td>for DataManagerService.war</td>
</tr>
</tbody>
</table>

10. Click Finish to create the new Web Module project
11. Repeat steps #2 thru #10 for the remaining five .war files.

**Deploy Adobe LiveCycle enterprise applications (.ear files)**

1. In the J2EE Hierarchy right-click on the Enterprise Applications node and select the Import > Import EAR... option.
2. Click Browse and locate the formServer.ear file \[C:\IBM_WSAD\Application Developer\v5.1.1\runtimes\base_v51\installableApps\]. In the Project name field, change the value to formServer_EAR and click Next.
3. In the next panel, make sure to check the AdobeCSAUtils.jar in the Utility JARs and web libraries section.
4. Click Finish.

5. While in the J2EE Hierarchy, expand the EJB Modules tree and right-click on the formServerEJB and select the Generate > Deployment and RMIC Code option.

6. Right-click on “MyServer” created earlier and select the “Add and remove projects…” option.

7. In the dialog, click on the “Add All” button and then click the Finish button.
Start the test application server

Right-click on the test server and "MyServer" click on the “Start” option.

You should now see Form Server running in the Console.
4.0 Test Adobe LiveCycle

To deploy idp-servlet

Follow the steps that describe how to deploy the formServer.ear file above. Instead of pointing to the formServer.EAR file, browse to the idp-servlet.ear sample application located in the product root/samples/simple/servlet/j2ee directory [C:\Program Files\adobe\idp\samples\simple\servlet\j2ee\idp-servlet.ear].

Select MyServer. Right-click and select 'Restart.'

To run idp-servlet

1. Open your web browser, and type the URL

   http://localhost:9080/idp/samples/simple/servlet/

2. Select 'Run Sample'.

3. Fill in the form and click Submit.

   If the server is configured correctly you will be presented with an XML representation of the data you entered in the form.

5.0 Install WebSphere Studio/Adobe Designer plug-in

Download the Adobe Designer 6.0 WebSphere Studio Plug-in (ZIP: 90 KB or EXE: 115 KB).

This plug-in extends WebSphere Studio with the ability to edit Adobe forms directly from any WebSphere Studio project using the Adobe Designer 6.0.

Note: Registration with Adobe Solutions Network (ASN) is required to access this software.

Congratulations you are now working with Adobe LiveCycle Forms on WSAD.

Please proceed to the next section.
Part II: Installing and Deploying Tutorial Project

Overview

This tutorial steps you through the process of using WSAD to create a dynamic Web project using Form Server. In the first part of the tutorial, the `renderForm()` method is used to render the XML Form template named `MortgageApplication.xdp` to PDF.

In the second part of the tutorial, the `renderForm()` method is used to render the template `CustomerInfo.xdp` to HTML. When the user clicks the Submit button, the `processFormSubmission()` method is used to extract the data from the form. The `renderForm()` method is used again to render `MortgageApplication.xdp` again as PDF, but now the application contains the data entered into `CustomerInfo.xdp`.

Whether users are online or offline, internal or external, Adobe LiveCycle Forms lets organizations deploy secure XML-based forms as Adobe PDF or HTML over any platform or device — from PCs to handhelds — without requiring any new software or plug-ins.

1.0 Setting up the Project

In this step you will create a dynamic Web project in WSAD.

1. Un-compress the tutorial files located in “WSAD_Tutorial_Files.zip” to your Desktop.
2. Open WSAD 5.1.1.
4. For Project Name, enter FS-sample.
5. Select Finish.
6. When prompted to switch to the Web Perspective, select Yes.

Adding external libraries

Add the library for the Form Server API calls that will be used by the Java code. This makes the library available for compiling the Java code.

1. Right click FS-sample project, select Properties. Select Java Build Path > Libraries tab.
2. Click Add External JARs, select `formserver-client.jar` (in C:/Program Files/Adobe/idp/lib).
3. Click OK.

Add the library to the WEB-INF/lib directory. The library is needed when the servlet runs.

1. Expand FS-sample project. WebContent > WEB-INF. Select ‘lib’ and Right-click.
2. Select Import > File system. Click Next.
3. In From directory, click Browse.
4. Select c:\Program Files\Adobe\idp\lib.
5. Add a checkmark beside `formserver-client.jar`.
6. Click Finish.
Importing the .xdp files

Two XDP files are provided. One is designed to be transformed to HTML, the other is designed to be transformed to PDF. These are the forms that are rendered by Form Server in the servlets you will create. In this step you import the XDP files into the project.

1. Click on the plus sign next to the FS-sample project to view the contents.
2. Right click Web Content. Select Import > File system.
3. Click Next.
4. From Directory, select Browse. Import un-zipped .XDP Files. Put a checkmark next to XDP Files to select both files (CustomerInfo.xdp & MortgageApplication.xdp).
5. Click Finish.

Creating the RenderToPDF servlet

The RenderToPDF servlet will convert an XDP file to PDF using the renderForm() method of Form Server. The XDP file is MortgageApplication.xdp was created in the Adobe Designer tutorial.1

Right click on FS-sample, and select New > Servlet.

For Class name, enter RenderToPDF.

Accept defaults in the following screens. Click Finish.

Inserting code

In this step you will add the code to render the MortgageApplication.xdp to PDF.

1. Double click RenderToPDF.java to open the file.
   At the top of the file are import statements for the object used in the code.
2. Add these import statements below the existing import statements:

   ```java
   import java.io.File;
   import java.io.FileInputStream;
   import javax.servlet.ServletOutputStream;
   import com.adobe.formServer.client.EJBClient;
   import com.adobe.formServer.interfaces.IFormServer;
   import com.adobe.formServer.interfaces.IOutputContext;
   ```

3. Redirect the doGet() method to the doPost() method by inserting the following code:

   ```java
doPost(req, resp);
   ```

   The servlet will respond in the same way to both a doGet() and a doPost() call.

---

1 Adobe Designer tutorial files are available at:
http://partners.adobe.com/asn/developer/ibm/DesignerTutorialFiles.zip
Adobe Designer tutorial instructions are available at:
4. In the `doPost()` method, insert the following code which connects to the Form Server EJB client, then calls the `renderForm()` method of Form Server to display an XDP file as PDF:

```java
try {
    EJBClient oEJBClient = new EJBClient();
    IFormServer oIFS = (IFormServer) oEJBClient;

    // placeholder for retrieving data from temporary file
    //
    // build up the URL context
    // for example, http://localhost:9080/FS-sample/RenderToPDF
    //
    String contextURL =
    req.getScheme() + "://" + req.getServerName() + ":" + req.getServerPort() + req.getContextPath();

    // call the IFormServer.renderForm() method
    // to render the XDP file as PDF.
    //
    IOutputContext oIOC =
    oIFS.renderForm(
        "MortgageApplication.xdp",
        "PDFForm", // render to PDF form
        null, null,
        "Mozilla/3",
        contextURL, "RenderToPDF",
        contextURL, null);

    // Set content length of the response object
    // and send the form rendered as PDF to the browser.
    //
    resp.setContentType(oIOC.getContentType());
    byte[] bContent = oIOC.getOutputContent();
    resp.setContentLength(bContent.length);
    ServletOutputStream oOutput = resp.getOutputStream();
    oOutput.write(bContent);
} catch (Exception e) {
    response.setContentType("text/html");
    PrintWriter out = resp.getWriter();
    out.println("<html>");
    out.println("<body>");
    out.println("Exception: " + e.getMessage());
    out.println(" </body>");
    out.println("</html>");
}
```

5. Select File > Save.
Creating index.jsp

Now create the web page that will call the servlet.

1. Right click FS-sample, select New > JSP File. Enter index.jsp in the File Name box. Accept the defaults.
2. Select Finish.
3. In index.jsp, type in:
   
   This page calls the servlet
4. Insert the cursor after the word servlet.
5. Select the Palette > HTML Tags.
6. Double click Link.
   The Insert Link dialog appears.
7. In the URL edit box, select Browse > Servlet.
8. Select RenderToPDF.
9. Click OK.
10. Save index.jsp

2.0 Running the application in WSAD

Now you will run the application in WSAD.

1. In the J2EE perspective, select MyServer under the Servers folder/directory. Right-click and select "Stop".
2. Return to Web perspective.
3. Right click on the project FS-sample, select Run on Server.
   The application is installed in WebSphere Studio test server. After the MyServer has started, the index.jsp page is displayed.
4. Click on the RenderToPDF link.
   MortgageApplication.xdp is displayed as PDF.

3.0 Adding the RenderToHTML servlet

In this step you will add the RenderToHTML servlet that renders an XDP file to HTML. The CustomerInfo.xdp has been designed for optimal display in a web browser. The XDP file has some fields with the same names as MortgageApplication.xdp. These fields will be extracted from CustomerInfo.xdp and imported into MortgageApplication.xdp. The RenderToPDF servlet will be modified slightly to retrieve the data and import the data into the form.

The RenderToHTML servlet converts CustomerInfo.xdp file to HTML using the renderForm() method of Form Server.

1. Right click on the FS-sample project, and select New > Servlet.
2. For Class Name, enter RenderToHTML.
3. Accept defaults in the following screens.
4. Select Finish.
Inserting code

Now you will add code to RenderToHTML.java.

1. Double click RenderToHTML.java.

At the top of the file are import statements for the object used in the code. Add these import statements below the existing import statements:

```java
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletOutputStream;
import com.adobe.formServer.client.EJBClient;
import com.adobe.formServer.interfaces.IFormServer;
import com.adobe.formServer.interfaces.IOOutputContext;
```

2. In the doGet() method, insert the following code which connects to the Form Server EJB client, then calls the renderForm() method of Form Server to display CustomerInfo.xdp as HTML:

```java
try {
    IFormServer oIFS;
    IOOutputContext oIOC = null;
    EJBClient oEJBClient = null;
    ServletOutputStream oOutput = resp.getOutputStream();
    oEJBClient = new EJBClient();
    oIFS = (IFormServer) oEJBClient;

    // build up the URL context
    // for example, http://localhost:9080/FS-sample/RenderToHTML

    String contextURL =
        req.getScheme()
            + "://" + req.getServerName()
            + ":" + req.getServerPort()
            + req.getContextPath();

    // call the IFormServer.renderForm() method
    // to display CustomerInfo.xdp as HTML.

    oIOC =
        oIFS.renderForm("CustomerInfo.xdp",
            "AHTML",
            new byte[0],
            null,
            "Mozilla/3",
            contextURL,
            "RenderToHTML",
            contextURL,
            null);
```
// Get the content type and length,  
// then send the PDF form to the browser.  
//
resp.setContentType(oIOC.getContentType());
byte[] cContent = oIOC.getOutputContent();
resp.setContentLength(cContent.length);
oOutput.write(oIOC.getOutputContent());
}
catch (Exception e){};

3. Select File > Save.

In the doPost() method add the code that is called when the Submit button is clicked. The Submit button calls the doPost() method of RenderToHTML servlet. In doPost(), the data is extracted from the form using the processFormSubmission() method of Form Server. At this point, the data would normally be saved to a database. For the sake of convenience, the data is saved to the servlet's tempdir. The path to the temporary file is forwarded to the RenderToPDF servlet.

4. Enter this code in the doPost() method.

try {
  IFormServer oIFS;
  IOutputContext oIOC = null;
  EJBClient oEJBClient = null;

  oEJBClient = new EJBClient();
oIFS = (IFormServer) oEJBClient;

  // Call the IFormServer.processFormSubmission() method.  
  //
oIOC = oIFS.processFormSubmission(req, "OutputType=0&XMLData=true");

  // This is where you might write the form data to a 
  // database or file.  
  // This servlet will write the data to the servlet's tempdir
  //
  File tempdir =
      (File) getServletContext().getAttribute("javax.servlet.context.tempdir");
  File f = File.createTempFile("formdata", ".xml", tempdir);
  FileWriter fwriter = new FileWriter(f);
fwriter.write(oIOC.getOutputString());
fwriter.close();
  req.setAttribute("FS-sample.tempfile", f.getPath());

  // forward the request to the RenderToPDF servlet
  //
  RequestDispatcher rd = req.getRequestDispatcher("RenderToPDF");
  rd.forward(req, resp);
}
catch (Exception e){};

5. Select File > Save.

Modifying the RenderToPDF servlet

The RenderToPDF servlet needs to be updated to receive the forwarded request from the doPost() method in RenderToHTML. In the doPost() method of RenderToPDF, the data will be retrieved from the temporary file and merged with MortgageApplication.xdp using the renderForm() method.

1. Double click on RenderToPDF.java to open the file.
2. In the doPost() method replace this line:
   // placeholder for retrieving data from temporary file
With this code:
   //
   // The RenderToHTML servlet set an attribute
   // containing the temp file name.
   // The temp file contains the XML
   // data extracted from the form.
   // The RenderToPDF servlet retrieves the attribute value,
   // and reads the XML data from the file.
   //
   String sPath = 
      (String) req.getAttribute("FS-sample.tempfile");
   File f = new File(sPath);
   FileInputStream fInput = new FileInputStream(f);
   Long length = new Long(f.length());
   byte[] bBuffer = new byte[length.intValue()];
   fInput.read(bBuffer);
   String sData = bBuffer.toString();

3. In the doPost() method, replace the third parameter (3rd line) in the renderForm() call with:
   oIOC =
      oIFS.renderForm(
         "MortgageApplication.xdp",
         "PDFForm",
         null,
         null,
   With this code:
   oIOC =
      oIFS.renderForm(
         "MortgageApplication.xdp",
         "PDFForm",
         bBuffer,
         null,

4. Select File > Save.
Modifying index.jsp

Now modify the web page to call the RenderToHTML servlet instead of the RenderToPDF servlet.

1. In WebContent, double click index.jsp to open the file.
2. Select and delete the link /FS-Sample/RenderToPDF.
3. Insert the cursor after the word servlet.
4. Select the Palette > HTML Tags.
5. Double click Link.
   The Insert Link dialog appears.
6. In the URL edit box, select Browse > Servlet.
7. Select RenderToHTML.
8. Click OK.
9. Save.

4.0 Running the application

Now you are ready to run the complete application.

1. Right click on the project FS-sample, select Run on Server.
2. The application is installed in WebSphere Studio test server (MyServer).
   Eventually the index.jsp page is displayed.
3. Click on RenderToHTML.
4. The CustomerInfo.xdp is displayed as HTML.
5. Enter values in the form fields.
6. Click Submit.
7. The values you entered for First Name, Email and SSN will display in the PDF version of the Mortgage application template.

MortgageApplication.xdp is displayed as PDF and contains the data from First Name, Email and SSN entered in the HTML rendition of CustomerInfo.xdp. (Only these three fields were finished in the MortgageApplication.xdp.)

The Submit button calls the doPost() method of RenderToHTML servlet. In doPost(), the data is extracted from the form using the processFormSubmission() method of Form Server. At this point, the data would normally be saved to a database. For the sake of convenience, the data is saved to the servlet’s tempdir. For example, a temporary file, formdata<###>.xml, is created in C:\WSAD\Application Developer\v5.1.1\runtimes\base_v51\temp\localhost\server1\FS-sample_war\FS-sample.war.

The path to the temporary file is forwarded to the RenderToPDF servlet. The RenderToPDF servlet receives the forwarded request (also in the doPost() method). The data is retrieved from the temporary file, and merged with MortgageApplication.xdp using the renderForm() method.

End of tutorial.