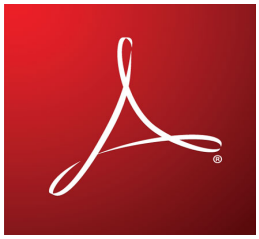
Working with PDF and Java

Adobe LiveCycle for Java Developers

A Tutorial from Technical Evangelism @ Adobe



Synopsis:

This course is designed to introduce Java ™ developers to the basic concepts involved in working with the International Standards Organization (ISO) Portable Document Format (PDF) as defined in the ISO specification. This course will take developers through simple exercises first by writing code in Java to manipulate PDF in various ways and then how to invoke services in Adobe LiveCycle ES using the Java Client libraries.

License:

Table of Contents

Working with PDF and Java

Course Leaders: Duane Nickull (JavaOne2009 – Moscone, San Francisco, CA)

Forward:

This course has been put together in hopes to provide developers a boot camp to learn all the basics concepts for working with ISO PDF documents from a Java Programming language environment. Although we tried hard to cover everything, it was simply not possible in such a short time. Our overall goal is to provide you with an introduction to and various pointers so you can make your own decision if you want to pursue this exciting new application development technology in the future. If you do, we will be providing additional references where you can continue learning and become part of the larger community after this course is over.

IMPORTANT: There is a high probability we will not get through the entire course during the time allotted at JavaOne 2009. This is by design. The course reflects a best case scenario whereby everyone covers the materials quickly. We felt it better to have extra rather than not enough content. If we do not get through the entire course, you can take the remaining labs by yourself as this instructional handout has sufficient notes to complete everything.

This course is written in such a manner that you should be able to take this alone as a self paced tutorial, although during the delivery of the course, we will be there to lead you through it and help in the event you encounter any problems.

We hope you enjoy this course as much as we enjoyed putting it together. Remember – we are here for you. Don't hesitate to ask any questions during the event and afterwards.

Audience Assumptions

- Attendees are familiar with PDF, but do not understand the specific binary

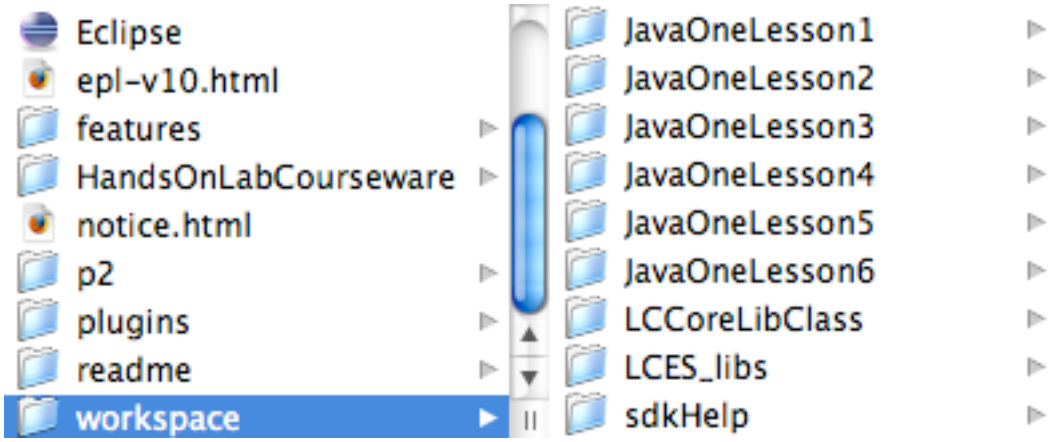
nature of it.
- Attendees are able to write some basic lines of Java and compile applications.
- Attendees have the courseware installed plus Java JDK 1.5
- Attendees are roughly familiar with XML syntax rules

Pre-Requisites

Before you can take this course, you must have the following software installed on your computer:

- Eclipse Ganymede 3.4
- Java JDK 1.5
- The courseware for this hands on lab
- Some PDF rendering software (Adobe Acrobat 9 recommended)

The course materials are designed to be used in a relative path and may be installed anywhere. For Mac OSX, the preferred location would be to place the files on your dektop. You should see the following structure:



Agenda

Getting inside PDF

LiveCycle ES Platform Overview

Java Architecture – Ways to manipulate PDF

Directly via Java API's

Remotely using Java Service Clients and the LC ES Enterprise Service Bus (Cloud Instance)

LC ES Service and Process Management

Where to find higher learning.

Abstract

The PDF ISO standard has experienced a large growth in adoption by government and enterprises. Many of these have requirements to round trip information between a J2EE environment and PDF forms or static documents. This hands on labs will be about 25% presentation and 75% coding and working with the PDF libraries.

The core Java PDF libraries will be explored included how to create PDF documents, how to read and write to and from file systems, how to get PDF attachments, how to access metadata libraries and more.

The lab environment will be set up with JDK, JBoss and Adobe LiveCycle ES. Developers wishing to continue with the development will be able to take the environment home.

Proposer Notes: (The syntax may change to reflect new re-factoring of the core libraries:

Before any code is written, approximately 20 minutes of background information will be given about the PDF file format, it's render process and how PDF is persisted on a file system. We will briefly look at advanced topics like signing PDF with digital signatures and document certification.

Teachers: Find the accompanying slide deck in the folder <eclipse_root>/HandsOnLabCourseware/*_TEACHERS_GUIDE.pdf

Open up the slide dec as well and walk through these slides:

Understanding PDF

- PDF files are made from "objects"
- Objects in a PDF Document have identifiers to reference them.
- Objects can occur in any order in a file
- Objects can refer to each other by number
- References can create a cross-linked set of objects (mathematical graph)
- Cross reference table maps object numbers to places within the file
- Let's look at "HelloWorld.pdf

Discuss COS based PDF vs XDP.

Discuss ZIP format

Discuss XMP – show XMP

Hello World PDF

Hello World

```
%PDF-1.2
1 0 obj
<<
 /Type /Page
 /Parent 5 0 R
 /Resources 3 0 R
 /Contents 2 0 R
>>
endobj

2 0 obj
<<
  /Length 51
>>
 stream
  BT
  /F1 24 Tf
  1 0 0 1 260 254 Tm
  (Hello World)Tj
  ET
 endstream
endobj
```

```
3 0 obj
<<
 /ProcSet[/PDF/Text]
 /Font <</F1 4 0 R >>
>>
endobj

4 0 obj
<<
 /Type /Font
 /Subtype /Type1
 /Name /F1
 /BaseFont/Helvetica
>>
endobj
```

```
5 0 obj
<<
 /Type /Pages
 /Kids [ 1 0 R ]
 /Count 1
 /MediaBox
   [ 0 0 612 446 ]
>>
endobj

6 0 obj
<<
 /Type /Catalog
 /Pages 5 0 R
>>
endobj

trailer
<<
 /Root 6 0 R
>>
```

First thing to do: Thank Jim King for the slides!!!!

We will see a lot of pages like this one so let me tell you carefully what you are looking at. I opened the example 01 PDF file in Microsoft Word as a "text" file. The characters from the PDF file have been formatted into three columns and I have also inserted some line breaks and indentations in order to make the text more readable. I also added headings and footings including page numbers.

Since the page displays "Hello World" you might expect the PDF file to have that character string somewhere within it. You would be right and I have highlighted this in red. We will work our way outward from this string and see what supporting material is required to turn that text string into a complete PDF document. Notice that "Hello World" is enclosed in parenthesis. This is how strings are represented in both PDF and PostScript.

```
%PDF-1.2
1 0 obj
<<
 /Type /Page
 /Parent 5 0 R
 /Resources 3 0 R
 /Contents 2 0 R
>>
endobj

2 0 obj
<<
  /Length 51
>>
 stream
  BT
  /F1 24 Tf
  1 0 0 1 260 254 Tm
  (Hello World)Tj
  ET
 endstream
endobj
```

```
3 0 obj
<<
 /ProcSet[/PDF/Text]
 /Font <</F1 4 0 R >>
>>
endobj

4 0 obj
<<
 /Type /Font
 /Subtype /Type1
 /Name /F1
 /BaseFont/Helvetica
>>
endobj
```

```
5 0 obj
<<
 /Type /Pages
 /Kids [ 1 0 R ]
 /Count 1
 /MediaBox
   [ 0 0 612 446 ]
>>
endobj

6 0 obj
<<
 /Type /Catalog
 /Pages 5 0 R
>>
endobj

trailer
<<
 /Root 6 0 R
>>
```

- Note that the material in the file is organized into 6 objects a "%PDF-1.2" header and a trailer.

- Each of the objects has a number followed by a zero, begins with "obj" and ends with "endobj".

- Jim King refers to PDF as "object oriented PostScript" because this object structuring is something that PDF has but PostScript does not.

- Note also that the file starts with "%PDF-1.2" which indicates that this is a PDF file following the 1.2 version of the PDF specification.
- Discuss backwards and forwards compatibility.

Exercise 1: Opening and Closing a PDF Document in Java

- Objective: Learn how to read and serialize a PDF to and from a file system.
- Expected duration: 14 minutes
- Brief description: Attendees will write a base class and add the following code:

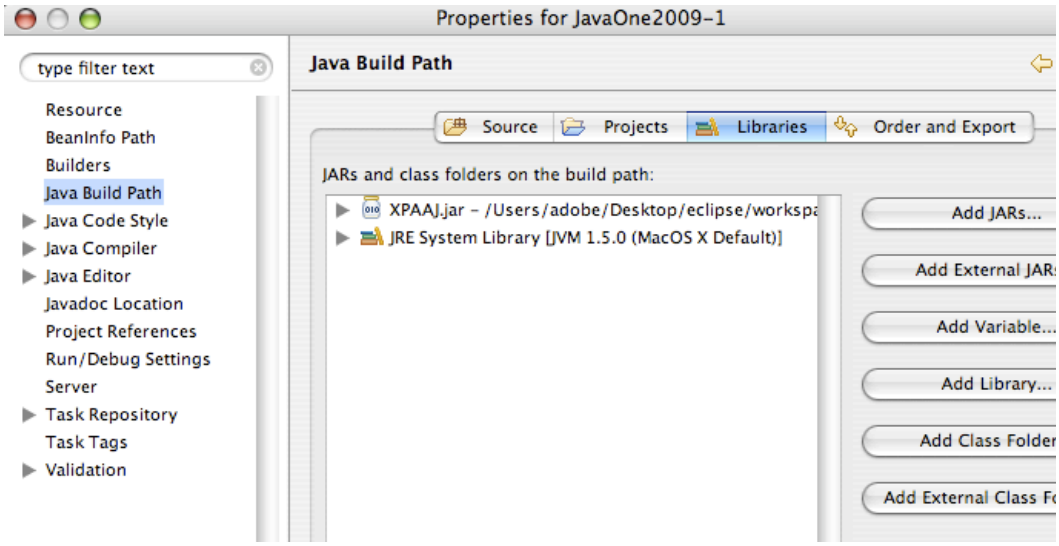
Step 1: Open the project under the folder <eclipse_root>/workspace/JavaOne2009-1/ and navigate to the source directory and open up JavaOne2009-1.java

Step 2: This project should be already linked to the XPAAJ.jar file. If it is not, take the following steps.

2a. – Select "Project" -> Build Path and you will see a window that looks similar
to the one below.

2b - Select the "Java Build Path" on the left hand side and you should see XPAAJ.jar in the path.

2c – if you do not see it, click "Add External Jars" on the right hand side and navigate to the <eclipse_root>/workspace/libs directory and you will see it.



Step 3: The source code will appear as follows. It requires you to add some code to

open up the PDF document.

```
import java.io.InputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.FileNotFoundException;
import java.io.IOException;
// this is from xpaaj.jar - check licenses before using.
// LiveCycle ES has newer JAVA libraries for manipulating PDF
import com.adobe.pdf.*;

public class JavaOne2009_1 {
    public static void main(String[] args)
            throws FileNotFoundException, IOException
       {
             // get pdf filename
             String inPdfName;
             if(args.length != 1 ) {
              System.out.println("\nCommand line format: java PDFExtractData
pdf-file");
              return;
             } else {
                   // message
                System.out.println("\nPDF data extraction using Adobe PDF
Libraries.");
                   inPdfName = new String(args[0]);
                   //PDFExtract(inPdfName);
             }
       }

       /**
        *     TODO: Add Method PDFExtract to open and extract data from PDF file
        */

}

/******* End of file *********************************/
```

Step 4 – uncomment the line of code that will call the new method we write. This is done by removing the two slashes before it to change it from this:

```
//PDFExtract(inPdfName);
```

to this:

```
PDFExtract(inPdfName);
```

Step 5 - Add the code to build a new method. We are going to use the method "OpenDocument" from the PDFFactory class. The API signature is as follows:

openDocument

Creates a new `PDFDocument` object based on a `java.io.InputStream` object that references an existing PDF or XDP file.

Syntax

```
public static PDFDocument openDocument(java.io.InputStream input)
                                throws java.io.IOException, PDFException
```

Parameters

`input`	A `java.io.InputStream` object that represents an existing PDF or XDP file.

Returns

A `PDFDocument` object.

Throws

`java.io.IOException` if the `java.io.InputStream` object is invalid.

`PDFException` if a PDF exception is raised. For more information, see "XML/PDF Access API for Java Exceptions" on page 29.

Step 6: start your method as follows:

```
public static void PDFExtract(String inPdfName)
	throws FileNotFoundException, IOException
	{
	// open PDF

	try {
		doc = PDFFactory.openDocument(inPdfFile);
	} catch (IOException e) {
			System.out.println("Error opening PDF file :" + inPdfName);
			System.out.println(e);
	}
```

Teachers notes:

- explain try..catch and exception handing – make sure attendees understand!

Step 7: add the following highlited code:

```
public static void PDFExtract(String inPdfName)
	throws FileNotFoundException, IOException
	{
	// open PDF
	System.out.println("\nOpen PDF Document ... ");
	PDFDocument doc = null;
	FileInputStream inPdfFile = new FileInputStream(inPdfName);
```

```
	try {
		doc = PDFFactory.openDocument(inPdfFile);
	} catch (IOException e) {
			System.out.println("Error opening PDF file :" + inPdfName);
			System.out.println(e);
	}
```

Teachers notes:

- Explain the fileinputstream and the new PDFDocument doc

Step 8: append this code to the end of your class:

```
	if(doc == null)
		System.out.println("Cannot open PDF file : " + inPdfName);
	else
		System.out.println( inPdfName + " was successfully opened.");
	}
```

Teachers notes:

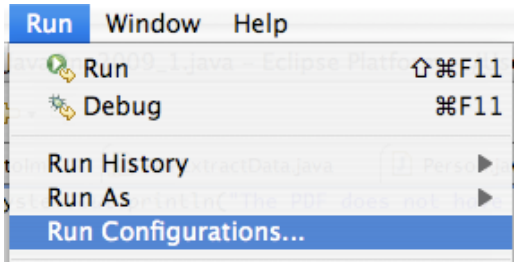
- explain why we are doing this (to test and make sure it runs)

Step 9: your code should now be ready to run. The completed file should look like this:

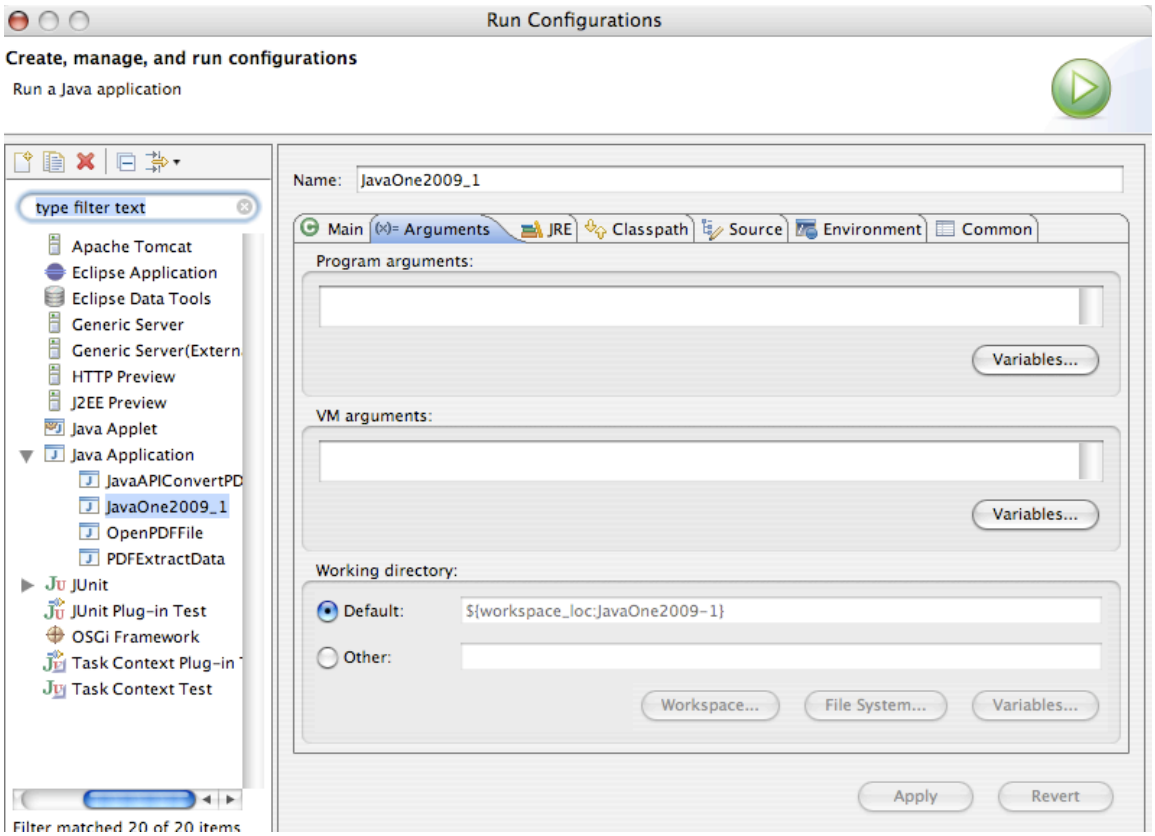
```
import java.io.InputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.FileNotFoundException;
import java.io.IOException;
// this is from xpaaj.jar - check licenses before using.
// LiveCycle ES has newer JAVA libraries for manipulating PDF
import com.adobe.pdf.*;

public class JavaOne2009_1 {
    public static void main(String[] args)
		throws FileNotFoundException, IOException
	{
		// get pdf filename
		String inPdfName;
		if(args.length != 1 ) {
		  System.out.println("\nCommand line format: java PDFExtractData
pdf-file");
		  return;
		} else {
			// message
```

```
                System.out.println("\nPDF data extraction using Adobe PDF
Libraries.");
                    inPdfName = new String(args[0]);
                    PDFExtract(inPdfName);
            }
      }

      /**
       *     TODO: Add Method PDFExtract to open and extract data from a PDF
file
       */
    public static void PDFExtract(String inPdfName)
      throws FileNotFoundException, IOException
      {
      // open PDF
      System.out.println("\nOpen PDF Document ... ");
      PDFDocument doc = null;
      FileInputStream inPdfFile = new FileInputStream(inPdfName);

      try {
            doc = PDFFactory.openDocument(inPdfFile);
      } catch (IOException e) {
                  System.out.println("Error opening PDF file :" + inPdfName);
                  System.out.println(e);
      }

      if(doc == null)
            System.out.println("Cannot open PDF file : " + inPdfName);
      else
            System.out.println( inPdfName + " was successfully opened.");
      }

}

/******* End of file *********************************/
```

Step 10 - Before you run the program, you have to specify arguments so Eclipse knows how to find the args. To do this, select "Run" -> "Run Configuration" from the top menu as shown below:



This will open up a dialog window that looks like this.



Click on the tab "Arguments" as shown above and you will need to enter the program arguments in the box. The simplest way to do this is to grab a command window (Win) or terminal (*.nix) and first navigate to the absolute Volume root.

Teachers notes:

Make sure instructions are understood for people on other operating systems!

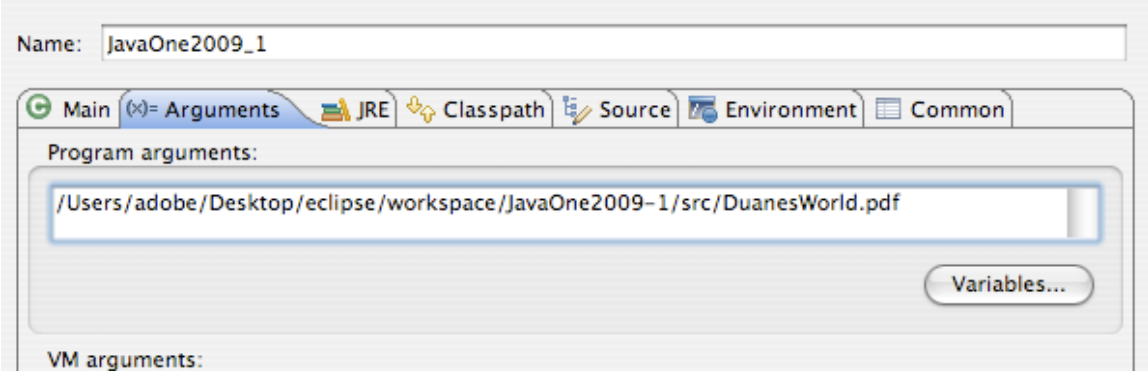
This can be done by entering "/" as show below:

```
Last login: Fri Mar 20 10:52:26 on console
Welcome to Darwin!
leamons-loaner:~ adobe$ cd /
leamons-loaner:/ adobe$ pwd
/
leamons-loaner:/ adobe$
```

navigate to the directory of <eclipse_root>/worspace/JavaOne2009-1/src/ then enter "pwd" to get the absolute path to that directory as shown below.

```
leamons-loaner:/ adobe$ cd Users/adobe/Desktop/eclipse/workspace/JavaOne2009-1/s
rc/
leamons-loaner:~/Desktop/eclipse/workspace/JavaOne2009-1/src adobe$ pwd
/Users/adobe/Desktop/eclipse/workspace/JavaOne2009-1/src
leamons-loaner:~/Desktop/eclipse/workspace/JavaOne2009-1/src adobe$ ls
DuanesWorld.pdf         JavaOne2009_1.java
leamons-loaner:~/Desktop/eclipse/workspace/JavaOne2009-1/src adobe$
```

Copy and paste the result from "pwd" into the arguments and concatenate the name of the *.pdf file you wish to use. In this case you should see one there called DuanesWorld.pdf.



The click "Apply". Eclipse should now know how to get the argument file and pass it to the java class when you run the program.

You are now ready to run your program. You should see the following in the eclipse console window:

```
PDF data extraction using Adobe PDF Libraries.

Open PDF Document ...
/Users/adobe/Desktop/eclipse/workspace/JavaOne2009-1/src/DuanesWorld.pdf was
successfully opened.
```

If you did, congtratulations!

Exercise 2: Let's find out some attributes about our PDF document

Excercise 1b: Getting the Version and size (number of Pages) of a PDF document. In this section we will return to the Java API docs to explore some of the core things that are possible with PDF and Java. The following lines of code will be added:

```
System.out.println("Document version is: " + doc.getVersion());
```

```
                System.out.println("Document is " + doc.getNumberOfPages() + "pages
long.");
```

- Exercise 2: Extracting XMP metadata. XMP is based of W3C RDF.
    - Objective: Learn how to grab the metadata out of a PDF package.
    - Expected duration: Approximately 15 minutes
    - Brief description: Attendees will add code similar to the following to the base classes

```
                // Export the XMP metadata
                System.out.println("\nExtracting document metadata ...");
                String DocMetadataFile = "DocMetadata.xml";
                boolean b = false;
                InputStream inputStream;
                inputStream = doc.exportXMP();
                if(inputStream == null)
                        System.out.println("No document level metadata was exported.");
                else {
                        System.out.println("Document metadata was exported.");

                        try {
                                b = saveFile(inputStream, DocMetadataFile);
                        } catch (Exception e) {
                                System.out.println("Error saving metadata file.");
                                System.out.println(e);
                        }
                        if(b == true)
                                System.out.println ("Document metadata was saved to file :
" + DocMetadataFile);
                        else
                                System.out.println("Document metadata was not saved.");
                }
```

- Exercise 3: Learning how to work with Image Metadata. Images are self contained within PDF documents and may contain their own XMP metadata including information about the RAW image capture.
    - Objective: lear how to iterate over the IMages and grab XMP metadata for each image.
    - Expected duration: 22 minutes
    - Brief description: Attendees will add the following code to their project.

```
                // Export image metadata
                System.out.println("\nExtracting image metadata ...");
                List images = doc.getImages();
                if(images.size() > 0)
                        System.out.println("Images were extracted. Number of images = " +
images.size());
                else
                        System.out.println("No images where found in the PDF file.");

                // Iterate over all of the images and export their metadata.
                int imageNumber = 0;
                Iterator iterator = images.iterator();
                while(iterator.hasNext()) {
                        imageNumber++;
                        PDFImage image = (PDFImage)iterator.next();

                        try {
```

```
                                // export image metadata
                                inputStream = image.exportXMP();
                                if(inputStream == null) {
                                        System.out.println("No metadata found for image #" +
imageNumber);
                                        continue;
                                }

                                System.out.println("Image #" + imageNumber + " - " +
"metadata was exported.");
                                String imageFilename = "ImageMetadata" + imageNumber +
".xml";
                                b = saveFile(inputStream, imageFilename);
                                System.out.println("Image metadata was saved to file : " +
imageFilename);
                        } catch (IOException e) {
                                System.out.println("Error exporting image metadata.");
                                System.out.println(e);
                        } catch (Exception e) {
                                System.out.println("Error saving image metadata.");
                                System.out.println(e);
                        }
                }
```

- Exercise 4: Extracting Annotations from a PDF document.
    - Objective: Learn how to grab the annotations out of a PDF package.
    - Expected duration: Approximately 20 minutes
    - Brief description: Attendees will add code similar to the following to the base classes

```
                // Export the Annotations
                System.out.println("\nExtracting annotaion data ...");
                inputStream = doc.exportAnnotations();
                if(inputStream == null)
                        System.out.println("No annotations were exported.");
                else {
                        System.out.println("Annotations were exported.");

                        String annotFile = "Annotations.xml";
                        b = false;
                        try {
                                b = saveFile(inputStream, annotFile);
                        } catch (Exception e) {
                                System.out.println("Error saving annotation file.");
                                System.out.println(e);
                        }
                        if(b == true)
                                System.out.println ("Annotations were saved to file : " +
annotFile);
                        else
                                System.out.println("Annotations were not saved to file.");

                }
```

- Exercise 5: Extracting text from a PDF document.
    - Objective: Learn how to grab the text out of a PDF package. Once text is derived, it may be used to build a search engine index or to use the text.
    - Expected duration: Approximately 20 minutes
    - Brief description: Attendees will add code similar to the following to the base

```
classes

       // save text string to a file.
    public static boolean saveStringToFile(String text, String filePath)
    {
                boolean b = false;
                try {
                        BufferedWriter outTxtFile = new BufferedWriter(new
FileWriter(filePath));
                        outTxtFile.write(text, 0, text.length());
                    outTxtFile.close();
                        b = true;
                } catch (IOException e) {
                        System.out.println("Error saving text file.");
                }
                return b;
       }

 - Exercise 6: Getting the form type.
    - Objective:  PDF documents may be forms. Form types will be discussed.
    - Expected duration:  Approximately 10 minutes
    - Brief description:  Attendees will add code similar to the following to the base
classes

                // Get the form type
                System.out.println("\nExtracting form data ...");
                FormType formType = doc.getFormType();

 - Exercise 7: Getting the form data and writing it to file.
    - Objective:  PDF documents may be forms. Form component values will be extracted and
persisted to disk.
    - Expected duration:  Approximately 20 minutes
    - Brief description:  Attendees will add code similar to the following to the base
classes

                // Export the form data
                String formDataFile = "FormData.xml";
                if(formType == FormType.XML_FORM)
                {
                        System.out.println("The PDF has an XML form.");
                        inputStream = doc.exportFormData(FormDataFormat.XFA);
                        if(inputStream == null)
                                System.out.println("No XML form data was exported.");
                        else {
                                System.out.println("XML form data was exported.");
                                try {
                                        b = saveFile(inputStream, formDataFile);
                                        System.out.println("XML form data was saved to file
: " + formDataFile);
                                } catch (Exception e) {
                                        System.out.println("Error saving XML Form Data.");
                                        System.out.println(e);
                                }
                        }
                }

- Exercise 8: LC ES: The big picture
    - Objective:  Attendees will be introduced to the LiveCycle Workbench tooling and
learn how to work with the java classes in an Eclipse based IDE and create a simple
process to add a signature field to a document.
```

- Expected duration: Approximately 20 minutes
- Brief description: Attendees will be introduced on how to accomplish many of the tasks they have just done within Eclipse in a non-programmatic environment.

About the Teachers

Duane Nickull is a Senior Technical Evangelist for Adobe Systems and host of the Adobe TV series *"Duane's World".* Duane has written or participated in most of the larger SOA standards work in the past decade. He currently chairs the OASIS Service Oriented Architecture Reference Model Technical Committee (SOA-RM TC) which has just delivered a Reference Model for Service Oriented Architecture as a full OASIS standard. He served as a Vice Chair of the United Nations Centre for Facilitation of Commerce and Trade (UN/CEFACT) between 2003 and 2006. Within the United Nations, he oversaw the UN's Electronic Business strategy and Service Oriented Architecture and modeling efforts. He has served as the project team lead of the United Nations (UN/CEFACT) Electronic Business Architecture Group (SOA) and a specially appointed liaison between the W3C, UN and OASIS standards consortiums. Additionally, Duane has served as the chair and lead system architect for the United Nation's Electronic Business Working Group, a direct sub-group of CEFACT TMG and on the CEFACT TMG Steering Committee. He also has served as the Co-chair of the ebXML Technical Architecture group as well as co-editor of that specification starting in 1999, largely recognized as the first post-internet and post XML SOA. Mr. Nickull has written and contributed many technical

articles and books on these subjects Mr. Nickull has been called Mr. SOA by his peers during introductions to speak on the subject due to his overwhelming experience writing and contributing to the major Service Oriented Architectures (SOA's). Between 1995 and 2006, he spoke at over 500 venues in various countries around the world Duane has recently renewed his work in the theoretical field of computational intelligence and has recently spoken several times to various audiences via the Ontolog Forum on event-causality aware inference engines coupled to a query-able ontology. In the field of semantic reconciliation, Duane was a co-inventor of the first Context-sensitive XML Search Engine (www.goxml.com) and the first web based XML E-Commerce ASP. He is named on pending patents pertaining to XML indexing and retrieval covering 51 unique points. He also served as Technical Director for XSLT.com during the 1990's until as recently as 2002. He lives in Vancouver, Canada with his wife and three children, plays in a rock band, actively snowboards, races Porsche 911's and mountain bikes. Duane made his living as a professional musician for several years.

Duanes Website: http://technoracle.blogspot.com