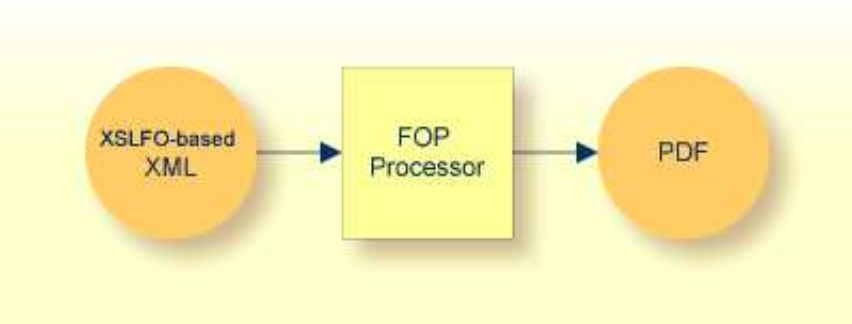
Introduction

This chapter covers the following topics:

- FOP - Formatting Objects Project
- Output Formats
- Printing
- Typical Example
- Browser Output of PDF Documents

FOP - Formatting Objects Project

There is a series of accepted standard modules coming from the Apache Group that form the base of Application Designer's PDF services. The so-called FOP standard (Formatting Objects Processor) is - though still in evolution - a healthy base for generating PDF documents through a certain XML-based specification: XSLFO. XSLFO consists out of a number of XML tag definitions by which a document is described. The XML definition is processed by a transform processor that produces PDF as output.

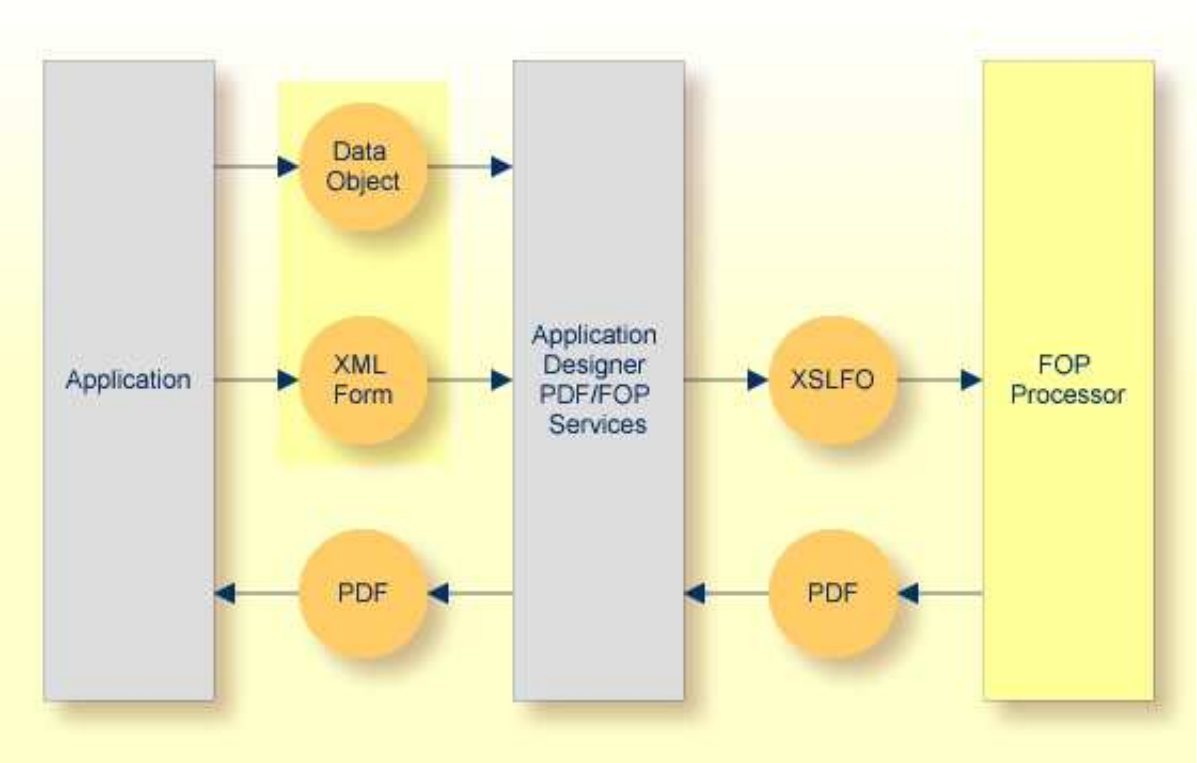


Though clear in concept and flexible in usage, XSLFO has a certain complexity: the number of tags and the number of properties inside the tags is quite high. Therefore, the effort for writing XSLFO-based XML documents is high and requires a profound understanding of FOP.

In addition, XSLFO is a pure output protocol - it describes a document before being transferred to PDF. The task of building the document and filling the right data in is not covered by FOP. This is where Application Designer's services come in.

Application Designer Services in Front of FOP

The following image summarizes the architecture of Application Designer's FOP services:



An application passes an XML form and a data object to the PDF/FOP services.

The XML form contains:

- Special abbreviation tags provided by Application Designer that simplify the creation of standard documents.
- Special tags provided by Application Designer that represent certain output controls (such as a grid) and contain binding information to a data object.
- Any XSLFO tag that you require. This means, you can directly use XSLFO statements and embed them into the form definition.

The data object (adapter) contains:

- All the "net data" that is mixed into the form. Net data is provided in the same way as you may be used to from working with Application Designer HTML pages: by corresponding properties that are referenced from the XML form definition.

The Application Designer PDF/FOP services build proper XSLFO out of the form and the data object. The XSLFO is passed to the normal FOP processing. The result is a PDF stream that is passed back to the calling application.

Output Formats

FOP is an abstract document definition. It is not bound to PDF. Application Designer offers both interfaces to create FOP-XSLO documents by the use of forms and it offers interfaces for printing FOP to a printer.

You can use other FOP output formats as well. Application Designer offers to you to directly generate the XSLFO result and pass it back to you as string.

Printing

Printing of application forms can be done in two ways:

- Server side PDF creation.
- Server side printing.

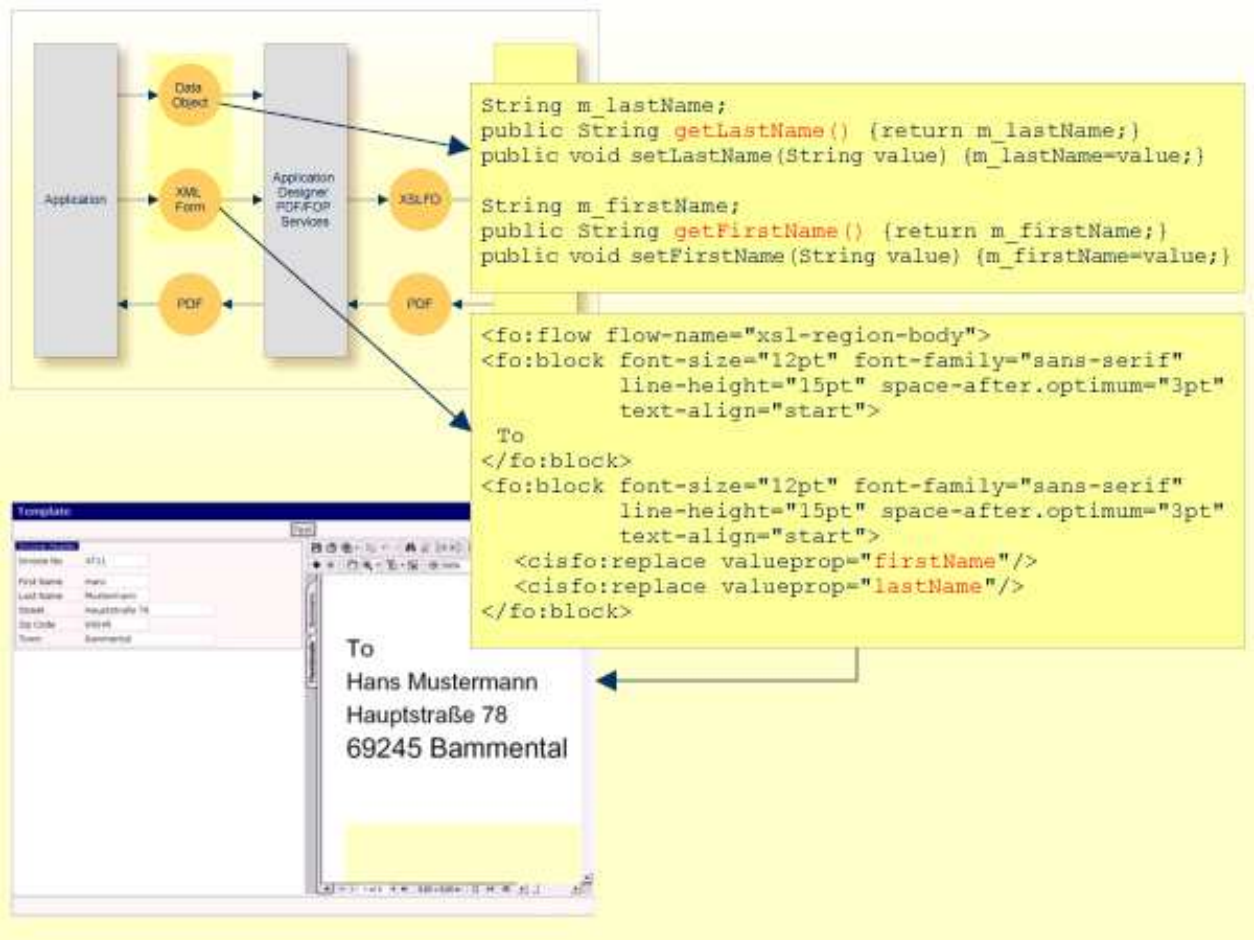
The server side PDF creation is what this document is about. Your server program uses the Application Designer APIs for creating PDF via FOP. The PDF files can be made available for the client (i.e. reachable by URL) and the client can integrate the files into its pages. Typically, the Adobe Reader is started on the client and the user can decide whether to print the document.

The server side printing is available by using a corresponding API. You can create an FOP document and call the print API. You have to define the printer on which you want to print.

Note:
Server side printing requires JDK 1.4 to be used inside your application server. (JDK 1.3. only allows to print on the standard printer of your application server.)

Typical Example

The following image shows an example of how the framework is applied:



You see that the binding between the form's XML and the data object is the same as you know from the binding between Application Designer HTML pages and adapter objects. In fact, it is the same technology now transferred to PDF output scenarios.

The following topics are covered below:

- Data Object (Adapter)
- Ways of Usage

Data Object (Adapter)

At runtime, the data object provides the data that is filled into the form. The same rules that you already know from the Application Designer adapter objects also apply to the data object:

- Properties can be nested: `address.street` means that first `address` is accessed, then `street` within the address object.
- Properties can be implemented - at any level - by corresponding `set`/`get` methods and/or by implementing the interface `IDynamicAccess`.

For detailed information, see *Binding between Page and Adapter* in the *Special Development Topics*.

Ways of Usage

There are several ways of using the Application Designer PDF/FOP services.

- You may store the XML forms somewhere in the database or in the file system. By this you can, for example, allow your customer to set up own forms that bind to the same data objects but contain different rendering information (e.g. different logos, different order of columns).
- You may generate the XML form dynamically. If having a flexible grid of information, then the number of columns depends, for example, on certain parameters.

It is up to you to decide.

Browser Output of PDF Documents

After having created a PDF document, you typically want to output it somewhere in the browser. There are two questions that you have to consider:

- How do you store the generated PDF document in your server so that it is reachable by URL?
- How do you embed the document into Application Designer HTML pages?

The following topics are covered below:

- Storing the Generated PDF
- Embedding the PDF into your Application Designer Page

Storing the Generated PDF

The Application Designer PDF APIs will generate a PDF byte stream. You have to store this byte stream in a certain way so that it can be reached by a URL from the browser. There are several ways to do so:

- You can store the document in the file system of the application server, for example, somewhere inside the file system belonging to your web application. However, this is a "quick and dirty" way and only works properly for some application servers (e.g. Tomcat, Jetty, etc.): it assumes that there is a file system in which your application is stored; this is not required by Java EE - in fact, some application servers do not store their web applications in the file system at all, but hold them within databases.
- You can store the data in a way that it is not reachable by a "fixed URL to a file" but by a "soft, servlet-based" URL.

If using the second option and embedding PDF into Application Designer page processing, then Application Designer already provides a so-called "session buffer". Inside the session buffer, you can store any object under a certain name. It will return a URL that you can pass to the browser as the URL to the stored content.

An example is provided in the section *The First PDF Document*.

Embedding the PDF into your Application Designer Page

- You can define a SUBPAGE control. A SUBPAGE control is a frame that can be dynamically loaded with a URL. This means you can store the PDF result somewhere in the file system so that it is accessible via a URL. The URL is passed to the SUBPAGE control - and the PDF is shown inside the frame accordingly.
- You can define an own frame in your frameset arrangement and send the URL of the PDF document to this frame.

An example is provided in the section *The First PDF Document*.