Server side PDF generation based on LaTeX templates

ISTVÁN BENCZE, BALÁZS FARK, LÁSZLÓ HATALA, PÉTER JESZENSZKY
University of Debrecen
Faculty of Informatics
Egyetem t.
H-4032, Debrecen, Hungary
jeszy (at) inf dot unideb dot hu

Abstract

We present a web-based addressbook application that can generate customized PDF documents using LaTeX template documents. The application is hosted on a server and users can access its functions using a web browser. A working LaTeX system must be installed only on the server side. Each registered user can manage his or her own addressbook. They can upload LaTeX templates and can generate multiple PDF documents from a template. Templates are customized to each selected recipient, substituting the appropriate addressbook data element into them. An example application might be an invitation card or a letter that must be sent to different recipients. Moreover, users can create simple documents (e.g. letters) using builtin templates and a simple web-based document editor.

Introduction

The Portable Document Format (PDF) has become one of the most widely used electronic document formats for publishing documents on the Web. It has many advantages that made it very popular. Some of them are the following:

- It is an open standard.
- It is device and platform independent.
- It is suitable for both viewing and printing.
- It is a file format, not a programming language like PostScript. A PostScript file contains code that must be interpreted, whereas a PDF file is rather a description, that results in faster and computationally less expensive processing.
- PDF files are searchable.

The goal of this paper is to give an overview of the tools and techniques that can be used to generate PDF documents in Java applications.

The first section presents a brief overview of the family of PDF tools that are available in Java.

In the following section we present our solution that is based on LaTeX template documents and on access to an external LaTeX system.

The last section is devoted to our sample LaTeX template-driven web application that generates PDF documents.

Overview of creating PDF documents in Java applications using conventional tools

This section gives an overview of the widely used solutions for the dynamic creation of PDF documents in Java applications. These tools can be classified as:

- XSL-FO formatters,
- PDF class libraries,
- reporting tools.

In the following we restrict our attention to open source solutions.

XSL-FO formatters

XSL-FO is an XML vocabulary for document formatting, a TeX-like typesetting language that uses XML syntax. It is a part of XSL, a family of W3C standards for the transformation and formatting of XML documents.

Because of the verbosity of the syntax, XML documents using the XSL-FO vocabulary are not edited manually. In order to use XSL-FO one needs an XML document and an appropriate XSLT stylesheet to transform it into another XML document that uses the XSL-FO markup vocabulary. (The transformation is executed by an XSLT processor, which is commonly available in Java environments.)

Then the XSL-FO document is converted into a readable or printable format by a so-called formatter. The most widely used output format is PDF.

Although the current version of the XSL specification became a W3C recommendation in 2001, none of the existing XSL-FO software products (including commercial products) implements the full standard.

Apache FOP [3] is an open source formatter, implemented in Java, that is a partial implementation of the XSL specification. FOP provides a Java API to access all of its functionality, thus it can be embedded into Java applications without difficulty.

XSL-FO might be a good solution if your data is in XML. There are ready-to-use XSLT stylesheets for standard XML document formats, such as DocBook XML, to transform them into XSL-FO. Writing your own stylesheet is not an easy job. Although there are graphical authoring tools, a sound knowledge of XSLT and XSL-FO is required.

PDF class libraries

Several Java class libraries are available to create and work with PDF documents. Unfortunately most of them are commercial products.

For example, fourteen PDF class libraries are listed in the appropriate category of the Google Directory [1] at the present time, and only three of them are available as open source software. Another reference [2] provides a list of open source PDF libraries in Java and contains six entries at present.

PDF class libraries can be classified as low-level or high-level.

Low-level PDF libraries provide low-level access to the contents of PDF documents and allow the creation of PDF documents in Java applications. To work with these APIs the programmer must be quite familiar with the PDF document format. It might be very difficult and cumbersome to use them.

In contrast, high-level PDF libraries use object models to model the logical structure of PDF documents. These logical models consist of Java objects that represent building blocks such as pages, chapters and paragraphs. Manipulating the object model programmers can access and modify the content of the underlying PDF documents.

PJX

A typical example of a low-level PDF library is PJX [4]. In order to use it one must know all about the PDF document format.

PDFBox

PDFBox [5] is a high-level class library. According to the project's web site it is used in several open source and commercial software products.

It allows the programmer to access and manipulate individual pages within a document. The content of pages can be accessed as a stream of objects, and it is easy to add text and images.

Unfortunately the API does not provide access to higher level building blocks such as chapters or paragraphs. For example, in order to add some text one must position to the right location within a page.

Although the API documentation is quite good and there are also some example programs and a developer's guide, unfortunately the latter is not very extensive.

iText

iText [6] is another high level class library that is more user friendly than PDFBox. It uses a higher level abstraction of documents. The building blocks of documents are chapters, sections, paragraphs, list, tables etc. This model looks like a document object model of an XML document. It is well documented; a very good tutorial is also available. According to the project web site, a book on iText will be published by Manning Publications this year.

Reporting tools

These are software tools that can generate business reports based on templates and data in databases and other data sources. Visual report designers may assist in the preparation of the reports. Templates are typically stored as XML documents that can also be edited by hand.

For a comprehensive list of open source reporting tools see [7]. Reporting tools offer varying features and capabilities; for example, they support different data sources and output formats. Some of them can produce PDF output and some can not.

JasperReports

JasperReports [8] is an excellent and powerful open source reporting tool that is written entirely in Java. It has a Java API that provies full programmatic control over the entire reporting process form report definition to report generation.

Report templates are defined by XML documents or defined programmatically, but open source and commercial visual report designer tools are available too. Compiled templates can be populated with data that is passed as parameters by the application or that comes from various data sources. A wide range of data sources is supported, such as relational databases (via JDBC and also via Hibernate), EJBs, XML documents and CSV files. When a template is filled the resulting report can be viewed, printed or exported to PDF, XML, HTML, CSV, XLS or RTF.

JasperReport is a professional-quality tool with many other features, such as i18n and integrated charting support.

DataVision

DataVision is an open source reporting tool that is very similar to JasperReports. It is also open source and written in Java, and it can be incorporated into a Java application easily. Reports can be created using a visual report designer tool and stored as XML files that can also be edited manually. The generated reports can be viewed, printed and exported to tab or comma-separated text files, DocBook, HTML, PDF and XML.

Compared to JasperReports, it has fewer features, for example it supports only relational databases (via JDBC) and plain text files as data sources.

It is mentioned here because to the best of our knowledge it is the only reporting tool than can export to LaTeX. Note that it uses LaTeX only as an output format; the user may use the resulting LaTeX documents to produce PDF or PostScript files. DataVision itself does not interpret LaTeX files to produce PDF, it uses the iText PDF library instead to generate PDF files directly.

Problems with the above solutions when using LaTeX templates

Some problems with the solutions presented in the preceding section are summarized below.

Problems with XSL-FO

Lack of stylesheets for non-standard XML formats If data is stored in a non-standard XML format and a stylesheet is not available to transform it into XSL-FO, it may be a difficult task to create an appropriate stylesheet.

Problems with PDF class libraries

Lack of flexibility As documents are created programmatically, any change in the output PDF file requires modification of the source code and the application must be recompiled.

Difficulty of use Low-level PDF class libraries require in-depth knowledge of the PDF format, making it extremely difficult to generate a PDF file. Even in the case of high-level libraries it may be difficult to achieve the right text layout.

Problems with report generators

Non-general purpose They are useful for generating business reports that contain tables and charts, based on data sources. They may not be the best solution to generate conventional documents such as letters. Typesetting large chunks of text and achieving the right layout may be difficult.

Common problems

Quality The aesthetic quality of the generated PDF documents is often poor compared with PDF files that are produced by LaTeX.

Java-TeX integration

Accessing TeX from Java

TeX and LaTeX offer the highest typographic quality. They can produce publication-quality PDF files with a professional appearance. It would be very useful if Java applications could benefit from it.

Unfortunately we have no knowledge of any existing standard tool to integrate Java and TeX.

Such an integration may work as follows. To produce high quality PDF output a Java application generates a TeX file. This is a trivial task since TeX source is plain text. Then the resulting TeX file is passed to a TeX system, that will turn it into DVI, PostScript or PDF.

The TeX system is accessed by using the `java.lang.Runtime` class, which allows the Java application to interface with the operating system. The application can have complete control over the TeX compilation process, it can interrupt the process if necessary and it also has access to the files that are produced by the TeX system.

Extending the above scenario with the use of TeX templates offers greater flexibility. In this case the Java application does not generate a TeX file from scratch,

but it reads a template and populates it with data. (Report generators operate in this way.)

Template engines

As described in Wikipedia, a template engine is a piece of software that processes an input text (the template) to produce one or more output texts. Template engines are widely used and very popular in web application development to create dynamic web content. Their most important advantage is that they separate application logic from web page layout.

Many Java-based template engines are available; see [10] for a list of open source Java template engines. They are used not only on the server side to generate HTML, but also they may be used in other applications to produce arbitrary textual output, even source code.

Generating LaTeX sources using FreeMarker

FreeMarker [11] is one well-known general-purpose open source template engine implemented in Java. Although it is typically used to generate HTML web pages in servlet-based MVC applications, we use it to produce LaTeX sources based on templates, that are turned into PDF.

FreeMarker has a powerful template language. Directives such as `if`, `switch` and `list` provide programming capabilities, and other common programming language constructs as variables, expressions and user defined functions are also available in templates.

Just as in the case of web applications the template engine is frequently used to incorporate database content into templates. In the example below we use Hibernate to access a relational database.

Hibernate [12] is the most popular solution for object-relational mapping (ORM) in the Java world. Hibernate provides transparent persistence for Java objects, that allows applications to store, update and delete objects in a relational database. It also provides query and object retrieval facilities. It is simple to use FreeMarker and Hibernate together.

Here is an example template fragment for producing a LaTeX table with FreeMarker:

```
\begin{tabular}{ll}
\toprule
Title &   ISBN\\
\midrule
<#list HibernateUtil.query("from Book b
  where b.year = 2006 order by b.title")
    as book>
    ${book.title}   &   ${book.isbn}\\
  </list>
  \bottomrule
\end{tabular}
```

In the example, `HibernateUtil` is a helper class whose static `query(String query)` method executes a HQL query,[1] and returns query results as a list of objects. Here we retrieve all books in the database that are published this year sorted by title. The table contains titles and ISBN numbers of the books and the output would look like this:

Title	ISBN
Aglaja. Apokrif	9630779668
Kazár szótár	9637448306
Utazás a tizenhatos mélyére	9631425169

Related projects

Although they have not influenced our work, the NTS and ε_{χ}-TeX projects must be mentioned here.

NTS stands for New Typesetting System. The goal of the project was to re-implement TeX in Java, but it was discontinued.

NTS has been replaced by ε_{χ}-TeX [13], that is a TeX-compatible typesetting system written in Java. Originally it was started as an attempt to enhance NTS, but later the entire system was rewritten from scratch.

The system is under development. Although a downloadable installer is available at the website of the project, the development is currently in pre-alpha stage.[2]

The project is a very promising initiative, but there is much to do. If it becomes available it will provide a more flexible Java-TeX integration.

The TeX Catalogue [14] contains two packages that support database access, namely SQLTeX and LaTeXDB.

SQLTeX is a Perl script that reads an input TeX file containing SQL commands and produces an output in which the commands are replaced with the results. LaTeXDB is a similar preprocessor but it is

[1] HQL stands for Hibernate Query Language, the fully object-oriented query language of Hibernate.

[2] Namely, it is only a development release that is not "feature complete". The next so-called alpha release will be delivered for more general testing.

implemented in Python. Both packages support only MySQL databases.

In either case, TeX input files may contain constructs that look like commands (for example `\sqldb`, `\sqlrow` or `\texdbconnection`) but are not in fact TeX commands. (This means that TeX files containing them will not compile.) They will be interpreted and replaced by the preprocessor to produce TeX files that should compile without any errors. In that sense, SQLTeX and LaTeXDB operate in a similar way to FreeMarker, but using TeX syntax.

A sample web application generating PDF

We have developed a web application to demonstrate the above approach in practice. Using the web application requires registration. Each registered user can manage his or her own addressbook and can generate PDF files based on LaTeX templates. The user selects entries of the addressbook and these are used to populate the template with data. A separate PDF file is generated for each selected entry that will be offered for download in a single ZIP file.

For example, this can be used to generate letters in PDF that are customized for each recipient. The PDF files are produced by LaTeX, thus guaranteeing a certain quality. Many users do not have LaTeX installed on their computer, but via the web application they have access to a LaTeX system. (It is also possible not to use the addressbook at all, and simply produce single PDF files.)

After logging in users have the following options:

- manage addressbook (add, delete and modify entries),
- upload an existing template and generate PDF(s),
- create a new template with a simple web-based editor and generate PDFfile(s).

If the third option is selected the user is presented with a list of predefined templates. These templates are LaTeX document skeletons that are stored on the computer hosting the web application. The following templates are installed by default: article, book, report, letter, empty.[3] The document editor is initialized with the selected template.

The templates that are uploaded or edited by the user should be valid LaTeX documents that should compile without any errors, although they may contain constructs that have special meaning. Text surrounded by `'@'` characters is a variable reference, and a replacement text will be substituted for it.

Some variable references have a predefined meaning, for example

- `@current.name@` means the full name of a person in an addressbook entry;
- `@current.name.firstname@` is the first name of a person in an addressbook entry;
- `@current.addresses.country@` is the country of the default postal address of a person in an addressbook entry;
- `@current.addresses.home.zipcode@` means the zip code of the home address of a person in an addressbook entry;
- `@current.phonenumbers.office@` is the office telephone number of a person in an addressbook entry.

These variable references can be used to generate multiple PDF files from a single template based on addressbook entries. Any other variable references such as `@signature@` are called static variable references, which will be replaced by static replacement text.

The user is presented with a list that contains all static variable references that occur in the template. For each of them a replacement text may be specified.

The next step is to select the output format, the possible choices being DVI, PostScript and PDF.

If the template does not contain any variable references, or contains only static variable references, then a single result file will be generated. Otherwise as the last step the user must select at least one addressbook entry, and a DVI, PostScript or PDF file will be generated for each of them.

The results are offered for download in a ZIP file that contains the generated DVI, PostScript or PDF file(s) together with the log file(s) and LaTeX source(s).

The `\write18{`*command*`}` construct allows the execution of operating system commands and is a potential security risk. Thus, `\write18` should be disabled, especially in web applications such as this (normally this is the default in TeX systems).

The following technologies and software products were used in the development: JDK 5.0, Apache Tom-

[3] Additional templates can be added easily.

cat, JavaServer Pages (JSP), PostgreSQL, and Hibernate. Note that we did not use FreeMarker, as there was no need for such a complex template engine in this application.

References

[1] A list of PDF class libraries for Java. http://www.google.com/Top/Computers/Programming/Languages/Java/Class_Libraries/Data_Formats/PDF/

[2] Open source PDF libraries in Java. http://java-source.net/open-source/pdf-libraries

[3] Apache FOP. http://xmlgraphics.apache.org/fop/

[4] PJX. http://www.etymon.com/epub.html

[5] PDFBox — Java PDF library. http://www.pdfbox.org/

[6] iText, a free Java-PDF library. http://www.lowagie.com/iText/

[7] Open Source Charting & Reporting Tools in Java. http://java-source.net/open-source/charting-and-reporting

[8] JasperReports. http://jasperreports.sourceforge.net/

[9] DataVision. http://datavision.sourceforge.net/

[10] Open Source Template Engines in. Java http://java-source.net/open-source/template-engines

[11] FreeMarker. http://freemarker.sourceforge.net/

[12] Hibernate. http://www.hibernate.org/

[13] ε_{χ}-TEX. http://www.extex.org/

[14] The TEX Catalogue Online. http://texcatalogue.sarovar.org/