Gerando Documento PDF em Java

Anderson Rodrigues Araújo

Aprenda como gerar documentos pdf a partir de um JavaBean e um documento xsl-fo, utilizando as ferramentas JDOM e FOP.

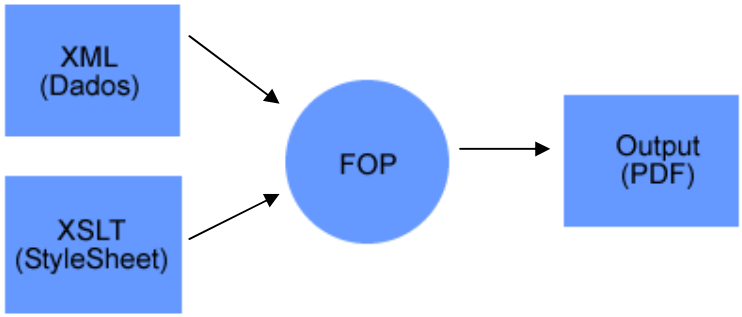
Introdução

Muitas vezes no desenvolvimento de um projeto, nos deparamos com a necessidade de mostrar as informações ao usuário, de uma forma mais trabalhada, até mesmo mais apresentável. Quando não se tem um gerador de relatórios à mão, uma boa opção para gerar o relatório em PDF, ou em outros formatos, é utilizar o FOP, da Apache.

Neste tutorial, iremos aprender como gerar um documento pdf a partir de um JavaBean com os dados e um documento xsl-fo (eu define o estilo da página), utilizando o JDOM e FOP.

Para gerar o documento pdf nós precisaremos:

- Um documento xsl-fo, que define o layout da página.
- Um documento xml, que será gerado a partir de um JavaBean, utilizando o JDOM.
- Um processador para ler os arquivos anteriores e gerar a saída em pdf (no nosso caso: o FOP);



O Arquivo xsl-fo

O arquivo xsl-fo (eXtensible Stylesheet Language – Formatting Objects) é o arquivo que definirá o formato da página, neste arquivo constam informações como margens da página, cabeçalho, rodapé, corpo, fonte, fluxo da informação, etc

O nosso arquivo xsl-fo (Cliente.xsl) ficaria assim:

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <xsl:stylesheet version="1.1" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:fo="http://www.w3.org/1999/XSL/Format" exclude-result-prefixes="fo">
3.   <xsl:output method="xml" version="1.0" omit-xml-declaration="no" indent="yes"/>
4.   <xsl:param name="versionParam" select="'1.0'"/>
5.   <xsl:template match="Cliente">
6.     <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
7.       <fo:layout-master-set>
8.         <fo:simple-page-master master-name="simpleA4" page-height="29.7cm" page-width="21cm"
           margin-top="2cm" margin-bottom="2cm" margin-left="2cm" margin-right="2cm">
9.           <fo:region-before extent="1.5in" padding="6pt 1in" border-bottom="0.5pt" display-
             align="after" />
10.          <fo:region-body margin-bottom="20pt" margin-top="80pt"/>
11.          <fo:region-after extent="1in" padding="3pt 1in" border-bottom="0.5pt" display-
             align="after" />
12.        </fo:simple-page-master>
13.      </fo:layout-master-set>
14.      <fo:page-sequence master-reference="simpleA4">
15.        <fo:static-content flow-name="xsl-region-before">
16.          <fo:block>
17.            <fo:external-graphic src="pdf.gif" content-height="1em" content-width="1em" />
```

```
18.           </fo:block>
19.         </fo:static-content>
20.         <fo:static-content flow-name="xsl-region-after" font-size="8pt" font-style="normal">
21.           <fo:block line-height="14pt" text-align="start" font-weight="bold">
22.             Locadora DVD Show
23.           </fo:block>
24.           <fo:block line-height="14pt" text-align="start">
25.             Rua Jose Bonifacio, 741 - Vila Independencia
26.           </fo:block>
27.           <fo:block line-height="14pt" text-align="start">
28.             Sao Paulo - SP - CEP: 01234-567
29.           </fo:block>
30.           <fo:block line-height="14pt" text-align="start">
31.             Tel: (11) 8888-8888 . Fax: (11) 8888-8889
32.           </fo:block>
33.         </fo:static-content>
34.         <fo:flow flow-name="xsl-region-body">
35.           <fo:block font-size="14pt" line-height="17pt" text-align="center" font-weight="bold">
36.             DECLARACAO
37.           </fo:block>
38.           <fo:block font-size="11pt" line-height="17pt" text-align="justify" space-before="50pt">
39.             Eu,
40.             <xsl:value-of select="nome"/>,
41.             <xsl:value-of select="profissao"/> domiciliado na
42.             <xsl:value-of select="logradouro"/>, no bairro
43.             <xsl:value-of select="bairro"/>, na cidade de
44.             <xsl:value-of select="cidade"/>, estado
45.             <xsl:value-of select="uf"/>, inscrito(a) no CPF / CNPJ
46.             <xsl:value-of select="cpf"/>, portador da cedula de identidade (RG) n.:
47.             <xsl:value-of select="rg"/>, declaro nesta data que:
48.           </fo:block>
49.           <fo:block font-size="11pt" line-height="17pt" text-align="justify" space-before="20pt">
50.             Recebi da Locadora DVD Show Ltda. todos os valores que me eram devidos,
51.             devidamente corrigidos.
52.           </fo:block>
53.           <fo:block font-size="11pt" line-height="17pt" text-align="justify" space-before="20pt">
54.             Declaro ainda que, a partir desta data, nada tenho a exigir da empresa
55.             Locadora DVD Show Ltda., civil e ou criminalmente, posto que esta empresa
56.             cumpriu com os termos da legislacao vigente.
57.           </fo:block>
58.           <fo:block font-size="11pt" line-height="17pt" space-before="20pt">
59.             Por ser verdade, firmo a presente em uma unica via.
60.           </fo:block>
61.           <fo:block font-size="11pt" line-height="17pt" space-before="60pt">
62.             _____________________________________________
63.           </fo:block>
64.           <fo:block font-size="11pt" line-height="17pt">
65.             <xsl:value-of select="nome"/>
66.           </fo:block>
67.       </fo:flow>
68.     </fo:page-sequence>
69.   </fo:root>
70.  </xsl:template>
71.</xsl:stylesheet>
```

Especificação XSL-FO

O W3C é o consórcio que define as especificações para utilização de arquivos XSL e XSLT. Este é o mesmo consórcio que define outras especificações como HTML e XML, entre outras.

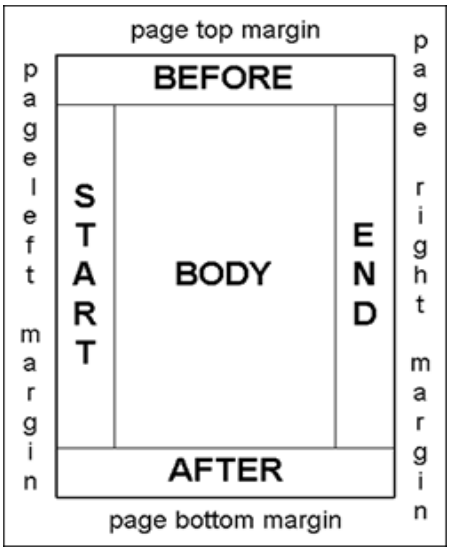
Entendento o arquivo Cliente.xsl

A primeira linha do arquivo, é uma tag obrigatório, que define que o conteúdo do arquivo é um xml, pois XSL-FO pertence à família XML.

A segunda linha definimos que o conteúdo representa um XSL, conforme a recomendação oficial de namespace da W3C.

Observe que na quinta linha definimos o template a ser aplicado à tag "Cliente" do arquivo xml.

Na tag "`<fo:layout-master-set>`" (sétima linha) definimos o elemento "simple-page-master", que define o layout das regiões da página. As regiões são as seguintes:



Note que no nosso arquivo definimos layout para três regiões: “before”, “body” e “after”.

Obs.: Pode haver mais de de um elemento “simple-page-master” dentro da tag “layout-master-set”.

Na linha 14 definimos a tag “page-sequence”. As páginas no documento são agrupadas em seqüências. Cada seqüência inicia uma nova página. Note que fazemos referência à “simple-page-master” de nome “A4”, que foi declarada anteriormente.

Dentro da tag “pág-sequence” temos 2 tags “static-content” que definem o conteúdo das regiões “before” e “after” e uma tag “flow” que define o fluxo da informação no corpo do documento. Cada tag “block” define um parágrafo no texto.

A tag “value-of” na linha 40, extrai o valor de um determinado atributo no arquivo xml, dentro do node selecionado, no nosso caso “Cliente”.

Gerando o xml com JDOM

O JDOm é uma API desenvolvida para facilitar a criação e manipulação de documentos xml, utilizando código Java.

Consideremos o seguinte objeto JavaBean:

```
package teste.pdf;

public class Cliente {

     private String nome;
     private String logradouro;
     private String bairro;
     private String cidade;
     private String uf;
     private Integer cpf;
     private String rg;
     private String profissao;

     //métodos getters e setters
}
```

Partindo de que o nosso objeto “Cliente” está populado, teremos a seguinte classe para gerar o xml:

```
package teste.pdf;

import java.io.FileOutputStream;
import org.jdom.Document;
import org.jdom.Element;
```

```
import org.jdom.output.Format;
import org.jdom.output.XMLOutputter;

public class GeraXml {

	private String fileName = "Cliente.xml";

	private void geraXml( Cliente cliente ){

		// elemento principal do xml
		Element Cliente = new Element("Cliente");
		//criando os elementos que irão fazer parte do xml
		Element logradouro = new Element("logradouro");
		Element bairro = new Element("bairro");
		Element cidade = new Element("cidade");
		Element uf = new Element("uf");
		Element cpf = new Element("cpf");
		Element rg = new Element("rg");
		Element profissao = new Element("profissao");

		//atribui os valores para os campos
		logradouro.setText( cliente.getLogradouro() );
		bairro.setText( cliente.getBairro() );
		cidade.setText( cliente.getCidade() );
		uf.setText( cliente.getUf() );
		cpf.setText( cliente.getCpf().toString() );
		rg.setText( cliente.getRg() );
		profissao.setText( cliente.getProfissao() );

		//atribui os campos para ao elemento principal
		Cliente.addContent( logradouro );
		Cliente.addContent( bairro );
		Cliente.addContent( cidade );
		Cliente.addContent( uf );
		Cliente.addContent( cpf );
		Cliente.addContent( rg );
		Cliente.addContent( profissao );

		//criando o documento
		Document doc = new Document();
		//atribui o elemento principal
		doc.setRootElement( Cliente );

		//cria o output
		XMLOutputter xout = new XMLOutputter( Format.getPrettyFormat() );
		try{
			xout.output( doc, new FileOutputStream( fileName ) );
		}catch (Exception e){
			e.printStackTrace();
		}
	}
}
```

Reflection

Se você está familiarizado com o uso de reflection, uma opção melhor seria criar uma classe mais genérica, que geraria o arquivo xml a partir de qualquer objeto, e não somente a partir da classe Cliente, como no nosso caso.

Gerando o pdf com FOP

FOP (Formating Object Processor) é um subprojeto do projeto XML da Apache. Ele é uma aplicação Java que o arquivo xsl-fo, e gera a saída para vários formatos (PDF, PCL, PS, SVG, TXT, entre outros).

```
package teste.pdf;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileOutputStream;

import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
```

```
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.sax.SAXResult;
import javax.xml.transform.stream.StreamSource;

import org.apache.avalon.framework.logger.ConsoleLogger;
import org.apache.avalon.framework.logger.Logger;
import org.apache.fop.apps.Driver;

public class GeraPdf {

	private TransformerFactory transformerFactory;
	private String pdfFile = "Cliente.pdf";


	public void GerarPdf(){
		this.transformerFactory = TransformerFactory.newInstance();
		//cria o driver
		Driver driver = new Driver();
		Logger logger = new ConsoleLogger(ConsoleLogger.LEVEL_INFO);
		driver.setLogger(logger);
		//seta o tipo de renderização
		driver.setRenderer( Driver.RENDER_PDF );
		try{
			//Seta o buffer para o output
			ByteArrayOutputStream out = new ByteArrayOutputStream();
			driver.setOutputStream(out);
			//Setup Transformer
			Source xsltSrc = new StreamSource( new File( xslFile ) );
			Transformer transformer = this.transformerFactory.newTransformer( xsltSrc ) ;
			Result res = new SAXResult( driver.getContentHandler() );
			//Setup o source
			Source src = new StreamSource( new File( xmlFile ) );
			//Inicia o processo de transformação
			transformer.transform(src, res);
			FileOutputStream fos = new FileOutputStream( this.pdfFile );
			fos.write( out.toByteArray() );
		}catch( Exception e) {
			e.printStackTrace();
		}
	}
}
```

Executando a aplicação

Temos agora a classe que será responsável por chamar todas as rotinas, para obtermos o resultado esperado:

```
package teste.pdf;

public class Application {

	public static void main(String[] args) {
		Cliente cliente = geraCliente();
		GeraXml geraXml = new GeraXml();
		//Chama a classe que gera o arquivo xml
		geraXml.geraXml( cliente );

		GeraPdf geraPdf = new GeraPdf();
		//Chama a classe que gera o arquivo pdf
		geraPdf.GerarPdf();

		System.out.println("Processamento efetuado com sucesso!");
	}

	/**
	 * Esta Classe popula um objeto do tipo Cliente para
	 * ser utilizado como teste.
	 */
	private static Cliente geraCliente(){
		Cliente cliente = new Cliente();
		cliente.setNome("Anderson Rodrigues Araujo");
		cliente.setProfissao("Analista de Sistema");
		cliente.setLogradouro("Rua Mauricio de Mendonca, 344");
		cliente.setBairro("Cohab II");
		cliente.setCidade("Mococa");
		cliente.setUf("SP");
		cliente.setCpf("286.694.458-59");
```

```
            cliente.setRg("29.890.025-7");
            return cliente;
        }
    }
```

Conclusão

Vimos aqui um pequeno exemplo de como gerar documentos pdf a partir de um JavaBean e um documento xsl-fo. Para isso utilizamos as API's do JDOM para gerar o arquivo xml e FOP para gerar a saída em pdf.

O exemplo que vimos aqui é relativamente simples, visto que existem vários outros recursos mais avançados, tanto para utilização dos arquivos xsl-fo, como para utilização do JDOM e FOP.

Para quem quiser se aprofundar no assunto:

FOP: http://xml.apache.org/fop
Tutorial sobre FOP: http://javaboutique.internet.com/tutorials/FOP
JDOM: www.jdom.org
Especificação XSL-FO: http://www.w3.org/TR/2001/REC-xsl-20011015/

Um abraço, e até a próxima.

Anderson Rodrigues Araújo (*andersonra@bol.com.br*)