

```
1
2 import com.cete.dynamicpdf.*;
3 import com.cete.dynamicpdf.pageelements.*;
4 import com.cete.dynamicpdf.pageelements.barcoding.Ean13Sup5;
5 import java.io.BufferedReader;
6 import java.io.FileNotFoundException;
7 import java.io.FileReader;
8 import java.io.IOException;
9 import java.text.SimpleDateFormat;
10 import java.util.Date;
11 import java.util.GregorianCalendar;
12 import javax.servlet.ServletConfig;
13 import javax.servlet.ServletException;
14 import javax.servlet.ServletOutputStream;
15 import javax.servlet.http.HttpServlet;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.servlet.http.HttpServletResponse;
18
19 public class TimeMachineTaggedPdf extends HttpServlet {
20
21     ServletOutputStream sOut;
22     Template footerTemplate;
23     EvenOddTemplate headerTemplate;
24
25     public void init(ServletConfig servletConfig) throws ServletException {
26         super.init(servletConfig);
27
28     }
29     public void doGet(HttpServletRequest req, HttpServletResponse res)
30         throws IOException,ServletException {
31
32
33         sOut = res.getOutputStream();
34         footerTemplate = new Template();
35         headerTemplate = new EvenOddTemplate();
36
37         // Create a document and set it's properties
38         Document objDocument = new Document();
39         objDocument.setCreator("TimeMachineTaggedPdf.java");
40         objDocument.setAuthor("H. G. Wells");
41         objDocument.setTitle("The Time Machine");
42         objDocument.setTemplate(headerTemplate);
43
44         // Specify document as tagged PDF
45         objDocument.setTag(new TagOptions());
46
47         // Adds elements to the header and footer groups
48         setPageHeaderTemplate();
49         setPageFooterTemplate();
```

```
50
51 // Sets up outline hierarchy
52 Outline objTitlePageOutline = objDocument.getOutlines().add("The Time "+
53     "Machine", new XYDestination(1, 0, 0));
54 Outline objChaptersOutline = objTitlePageOutline.getChildOutlines()
55     .add("Chapters");
56
57 // Builds the report
58 buildDocument(objDocument, objTitlePageOutline, objChaptersOutline);
59
60
61 // Outputs the document to the current web page
62 objDocument.drawToWeb(req, res, sOut, "TimeMachineTaggedPdf.pdf");
63 sOut.close();
64 }
65
66 private void buildDocument(Document document, Outline titlePage,
67     Outline chapters) {
68 // Adds Title page to document
69 document.getSections().begin(NumberingStyle.NONE, "Cover");
70 addTitlePage(document);
71 document.getSections().begin(NumberingStyle.NUMERIC, "Page", footerTemplate);
72
73 // Adds Chapters to the document
74 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
75     "_Chapter1.txt"), "1", "Chapter 1", chapters);
76 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
77     "_Chapter2.txt"), "2", "Chapter 2", chapters);
78 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
79     "_Chapter3.txt"), "3", "Chapter 3", chapters);
80 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
81     "_Chapter4.txt"), "4", "Chapter 4", chapters);
82 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
83     "_Chapter5.txt"), "5", "Chapter 5", chapters);
84 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
85     "_Chapter6.txt"), "6", "Chapter 6", chapters);
86 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
87     "_Chapter7.txt"), "7", "Chapter 7", chapters);
88 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
89     "_Chapter8.txt"), "8", "Chapter 8", chapters);
90 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
91     "_Chapter9.txt"), "9", "Chapter 9", chapters);
92 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
93     "_Chapter10.txt"), "10", "Chapter 10", chapters);
94 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
95     "_Chapter11.txt"), "11", "Chapter 11", chapters);
96 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
97     "_Chapter12.txt"), "12", "Chapter 12", chapters);
98
```

```
99 // Add the Epilogue to the document
100 addChapter(document, getServletContext().getRealPath("/text/TimeMachine"+
101     "_Epilogue.txt"), "Epilogue", "Epilogue", titlePage);
102 }
103
104 private void setPageHeaderTemplate() {
105
106 // Adds elements to the header group
107 headerTemplate.getOddElements().add(new Label("The Time Machine",
108     -18, -18, 324, 11, Font.getTimesRoman(), 11, TextAlign.LEFT));
109 headerTemplate.getOddElements().add(new Label("H. G. Wells", 18, -18,
110     324, 11, Font.getTimesRoman(), 11, TextAlign.RIGHT));
111 headerTemplate.getEvenElements().add(new Label("H. G. Wells", -18, -18,
112     324, 11, Font.getTimesRoman(), 11, TextAlign.LEFT));
113 headerTemplate.getEvenElements().add(new Label("The Time Machine", 18,
114     -18, 324, 11, Font.getTimesRoman(), 11, TextAlign.RIGHT));
115 }
116
117 private void setPageFooterTemplate() {
118 // Adds elements to the footer group
119 PageNumberingLabel objPageNumberLabel = new PageNumberingLabel
120     ("-%SP% -", 0, 478, 324,
121     11, Font.getTimesRoman(),
122     11, TextAlign.CENTER);
123 objPageNumberLabel.setPageOffset(-1);
124 footerTemplate.getElements().add(objPageNumberLabel);
125 }
126
127 private void addTitlePage(Document document) {
128 //Adds a title page to the document
129 Page objPage = new Page(396, 540, 36);
130
131 String strDisclaimer = "This document is in the public domain. " +
132     "Permission to use, copy, modify, and " +
133     "distribute this document for any purpose and " +
134     "without fee is hereby granted, without any " +
135     "conditions or restrictions. The barcode below " +
136     "is for demonstration purposes only.";
137
138 GregorianCalendar gc = new GregorianCalendar();
139 Date date = gc.getTime();
140 SimpleDateFormat sd1 = new SimpleDateFormat("yyyy.MM.dd");
141 SimpleDateFormat sd2 = new SimpleDateFormat("HH.mm.ss");
142 String strGenerated = "Generated by\\nDynamicPDF Generator\\non "
143     + sd1.format(date) + ",\\nat " + sd2.format(date)
144     + " EST";
145 objPage.getElements().add(new Label("The Time Machine", 36, 36, 252, 30,
146     Font.getTimesBold(), 30, TextAlign.CENTER));
147 objPage.getElements().add(new Label("by H. G. Wells", 36, 96, 252, 22,
```

```
148         Font.getTimesBold(), 22, TextAlign.CENTER));
149     objPage.getElements().add(new Label("1895", 36, 148, 252, 22,
150         Font.getTimesBold(), 22, TextAlign.CENTER));
151     try {
152         objPage.getElements().add(new Image(getServletContext().getRealPath
153             ("images/DPDFLogo.png"), 62, 208, 0.21f));
154     } catch (FileNotFoundException ex) {
155         System.err.println("cannot load the image :"+ex);
156     }
157     objPage.getElements().add(new Label(strGenerated, 132, 208, 182, 54,
158         Font.getTimesRoman(), 11));
159     objPage.getElements().add(new Label(strDisclaimer, 36, 276, 252, 65,
160         Font.getTimesRoman(), 11, TextAlign.JUSTIFY));
161     objPage.getElements().add(new Ean13Sup5("201234567890", "90000", 82,
162         360));
163     objPage.setApplyDocumentTemplate(false);
164
165     document.getPages().add(objPage);
166 }
167
168 private void addChapter(Document document, String filePath, String title,
169     String bookmarkText, Outline parentOutline) {
170     // Retrieves the text from the sections file
171     String strSectionText = getTextFromFile(filePath);
172
173     // Adds the first page of the section
174     Page objPage = addSectionHeaderPage(document, title, bookmarkText,
175         parentOutline);
176     objPage.setApplyDocumentTemplate(false);
177
178     // Creates a TextArea for the sections text
179     TextArea objTextArea = new TextArea(strSectionText, 0, 146, 324, 322,
180         Font.getTimesRoman(), 11);
181     objTextArea.setLeading(14);
182     objTextArea.setParagraphSpacing(20);
183     objPage.getElements().add(objTextArea);
184     document.getPages().add(objPage);
185
186     // Creates a TextArea for the overflow text
187     objTextArea = objTextArea.getOverflowTextArea(0, 0, 324, 468);
188
189     // Loops until no overflow text is found.
190     while (objTextArea != null) {
191         // Adds a new page to the document
192         objPage = new Page(396, 540, 36);
193
194         objPage.getElements().add(objTextArea);
195         // Adds new page to the document
196         document.getPages().add(objPage);
```

```
197     // Creates a TextArea for the overflow text
198     objTextArea = objTextArea.getOverflowTextArea();
199 }
200 }
201
202 private Page addSectionHeaderPage(Document document, String title,
203     String bookmarkText,
204     Outline parentOutline) {
205     // Adds the first page of a section to the document
206     Page objPage = new Page(396, 540, 36);
207
208
209
210     objPage.getElements().add(new Bookmark(bookmarkText, 0, 0,
211         parentOutline));
212     objPage.getElements().add(new Label("The Time Machine", 0, 36, 324, 30,
213         Font.getTimesBold(), 30,
214         TextAlign.CENTER));
215     objPage.getElements().add(new Label(title, 0, 96, 324, 22,
216         Font.getTimesBold(), 22,
217         TextAlign.CENTER));
218     objPage.getElements().add(new Line(120, 128, 204, 128));
219
220     return objPage;
221 }
222
223 private String getTextFromFile(String filePath) {
224     // Opens a text file and returns the text from it.
225     StringBuffer contents = new StringBuffer();
226
227     BufferedReader reader = null;
228     try {
229         reader = new BufferedReader(new FileReader(filePath));
230         String line = null;
231         while ((line = reader.readLine()) != null){
232             contents.append(line);
233             contents.append(System.getProperty("line.separator"));
234         }
235     } catch (FileNotFoundException ex1) {
236         System.err.println("Invalid file path :"+ex1);
237     } catch (IOException ex2){
238         System.err.println("cannot read from the file :"+ex2);
239     } finally {
240         try {
241             reader.close();
242         } catch (IOException ex3) {
243             System.err.println("cannot close the file :"+ex3);
244         }
245     }
```

```
246     return contents.toString();  
247 }  
248  
249  
250  
251 }
```