

```
1
2 import com.cete.dynamicpdf.*;
3 import com.cete.dynamicpdf.pageelements.FormattedTextArea;
4 import com.cete.dynamicpdf.pageelements.FormattedTextAreaStyle;
5 import com.cete.dynamicpdf.pageelements.Image;
6 import com.cete.dynamicpdf.pageelements.PageNumberingLabel;
7 import java.io.BufferedReader;
8 import java.io.FileNotFoundException;
9 import java.io.FileReader;
10 import java.io.IOException;
11 import javax.servlet.ServletConfig;
12 import javax.servlet.ServletException;
13 import javax.servlet.ServletOutputStream;
14 import javax.servlet.http.HttpServlet;
15 import javax.servlet.http.HttpServletRequest;
16 import javax.servlet.http.HttpServletResponse;
17
18 public class FormattedTextContinuation extends HttpServlet {
19
20     ServletOutputStream sOut;
21     // Template for document elements
22     private Template template;
23     PageNumberingLabel objPageNumLabel;
24
25     public void init(ServletConfig config) throws ServletException {
26         super.init(config);
27
28     }
29
30
31
32     public void doGet(HttpServletRequest req, HttpServletResponse res)
33         throws IOException, ServletException {
34
35
36         sOut = res.getOutputStream();
37         template = new Template();
38         // Add the page numbering label to the page template
39         objPageNumLabel = new PageNumberingLabel("Page %%CP%% of %%TP%%", 0,
40             494, 684, 10, Font.getHelvetica(),
41             10, TextAlign.CENTER);
42         template.getElements().add(objPageNumLabel);
43         try {
44             Image objImage = new Image(getServletContext().getRealPath("/images"+
45                 "/DPDFLogo.png"), 342, 0, 0.18f);
46             objImage.setAlign(Align.CENTER);
47             template.getElements().add(objImage);
48         } catch (FileNotFoundException ex) {
49             System.err.println("cannot load the image :"+ex);

```

## FormattedTextContinuation.java

---

```
50     }
51
52     String strFormattedText = getTextFromFile(getServletContext()
53         .getRealPath("/text/FormattedText.txt"));
54
55     // Create a document and set it's properties
56     Document objDocument = new Document();
57     objDocument.setCreator("HTMLTextContinuation.java");
58     objDocument.setAuthor("ceTe Software");
59     objDocument.setTitle("Formatted Text Continuation");
60
61     // Add the template to the document
62     objDocument.setTemplate(template);
63
64     // Create an FormattedTextAreaStyle and set the initial properties for the
65     // FormattedTextArea
66     FormattedTextAreaStyle objStyle = new FormattedTextAreaStyle(FontFamily
67         .getHelvetica(), 10, false);
68
69     objStyle.paragraph.indent = 36;
70     objStyle.paragraph.spacingAfter = 6;
71     objStyle.paragraph.allowOrphanLines = false;
72     // Create an FormattedTextArea using the style
73     FormattedTextArea objFormattedTextArea = new FormattedTextArea(strFormattedText, 0, 60,
74         315, 424, objStyle);
75
76     // Loop untill all of the text is displayed
77     while (objFormattedTextArea != null) {
78         objFormattedTextArea = addTextToDocument(objDocument, objFormattedTextArea);
79     }
80
81     // Outputs the document to the current web page.
82     objDocument.drawToWeb(req, res, sOut, "FormattedTextContinuation.pdf");
83     sOut.close();
84 }
85
86 private FormattedTextArea addTextToDocument(Document document,
87     FormattedTextArea formattedTextArea) {
88     // Create a new page
89     Page objPage = new Page(PageSize.LETTER, PageOrientation.LANDSCAPE,
90         54.0f);
91     // Add left column FormattedTextArea to page
92     objPage.getElements().add(formattedTextArea);
93     // Create the right column FormattedTextArea
94     formattedTextArea = formattedTextArea.getOverflowFormattedTextArea(369, 60);
95     // Add it to the page if it had overflow text
96     if (formattedTextArea != null) {
97         objPage.getElements().add(formattedTextArea);
98     }
99 }
```

## FormattedTextContinuation.java

```
99 // Add page to document
100 document.getPages().add(objPage);
101
102 // Return the next FormattedTextArea if there is overflow text
103 if (formattedTextArea == null) {
104     return null;
105 } else {
106     return formattedTextArea.getOverflowFormattedTextArea(0, 60);
107 }
108 }
109
110 private String getTextFromFile(String filePath) {
111     // Opens a text file and returns the text from it.
112     StringBuffer contents = new StringBuffer();
113
114     BufferedReader reader = null;
115     try {
116         reader = new BufferedReader(new FileReader(filePath));
117         String line = null;
118         while ((line = reader.readLine()) != null){
119             contents.append(line);
120             contents.append(System.getProperty("line.separator"));
121         }
122     } catch (FileNotFoundException ex1) {
123         System.err.println("Invalid file path :"+ex1);
124     } catch (IOException ex2){
125         System.err.println("cannot read from the file :"+ex2);
126     } finally {
127         try {
128             reader.close();
129         } catch (IOException ex3) {
130             System.err.println("cannot close the file :"+ex3);
131         }
132     }
133     return contents.toString();
134 }
135 }
136 }
```