

FormFieldReader.java

```
1 import com.cete.dynamicpdf.*;
2 import com.cete.dynamicpdf.merger.PdfDocument;
3 import com.cete.dynamicpdf.merger.forms.*;
4 import com.cete.dynamicpdf.pageelements.Row2;
5 import com.cete.dynamicpdf.pageelements.Table2;
6 import java.io.IOException;
7 import javax.servlet.ServletConfig;
8 import javax.servlet.ServletException;
9 import javax.servlet.ServletOutputStream;
10 import javax.servlet.http.HttpServlet;
11 import javax.servlet.http.HttpServletRequest;
12 import javax.servlet.http.HttpServletResponse;
13
14 public class FormFieldReader extends HttpServlet {
15
16     ServletOutputStream sOut;
17
18     public void init(ServletConfig servletConfig) throws ServletException {
19         super.init(servletConfig);
20     }
21
22     public void doGet(HttpServletRequest req, HttpServletResponse res)
23         throws IOException, ServletException {
24         sOut = res.getOutputStream();
25         PdfDocument pdfDocument = new PdfDocument(getServletContext().getRealPath("/pdfs/fw9AcroForm_14_filled.pdf"));
26
27         Table2 table = new Table2(0, 0, 512, 676, Font.getHelvetica(), 12);
28         table.getBorder().setWidth(1f);
29         table.getBorder().setColor(Grayscale.getBlack());
30         table.setRepeatColumnHeaderCount(1);
31         table.setRepeatRowHeaderCount(1);
32
33         buildTable(table);
34
35         createList(table, pdfDocument.getForm().getFields());
36
37         Document document = new Document();
38         addTableToPage(document, table);
39
40         Table2 overflowRowTable = table.getOverflowRows();
41
42         while (overflowRowTable != null) {
43             addTableToPage(document, overflowRowTable);
44             overflowRowTable = overflowRowTable.getOverflowRows();
45         }
46
47         document.drawToWeb(req, res, sOut, "FormFieldReader.pdf");
48         sOut.close();
49     }
}
```

```
50
51 private void buildTable(Table2 table) {
52     createColumns(table);
53     createRowHeadings(table);
54 }
55
56 private void createList(Table2 table, PdfFormFieldList fieldList) {
57     for (int i = 0; i < fieldList.size(); i++) {
58         Row2 row = table.getRows().add(20);
59         row.getCells().add(fieldList.getPdfFormField(i).getFullName());
60         row.getCells().add(getFieldType(fieldList.getPdfFormField(i)));
61         row.getCells().add(fieldList.getPdfFormField(i).getValue());
62
63         if (fieldList.getPdfFormField(i).hasChildFields() == true) {
64             createList(table, fieldList.getPdfFormField(i).getChildFields());
65         }
66     }
67 }
68
69 private String getFieldType(PdfFormField pdfFormField) {
70     if (pdfFormField instanceof PdfTextField) {
71         return "Text Field";
72     }
73
74     if (pdfFormField instanceof PdfButtonField) {
75         return "Button Field";
76     }
77
78     if (pdfFormField instanceof PdfChoiceField) {
79         return "Choice Field";
80     }
81
82     if (pdfFormField instanceof PdfSignatureField) {
83         return "Signature Field";
84     }
85
86     return "Container Field";
87 }
88
89
90
91 private void addTableToPage(Document document, Table2 table) {
92     Page page = new Page(PageSize.LETTER);
93     if (table != null) {
94         page.getElements().add(table);
95     }
96
97     document.getPages().add(page);
98 }
```

```
99
100 private void createRowHeadings(Table2 table) {
101     Row2 row = table.getRows().add(40, Font.getTimesBold(), 12, Grayscale.getBlack(),
102                                     Grayscale.getLightGrey());
103     row.getCellDefault().setAlign(TextAlign.CENTER);
104     row.getCellDefault().setVAlign(VAlign.TOP);
105     row.getCells().add("FormField Name");
106     row.getCells().add("FormField Type");
107     row.getCells().add("FormField Value");
108 }
109
110 private void createColumns(Table2 table) {
111     table.getColumns().add(150);
112     table.getColumns().add(150);
113     table.getColumns().add(212);
114 }
115
116
117 }
```