Mustang Project Nutzerdokumentation

Jochen Stärk

zu Mustangproject 1.3.0, 27.03.2015

http://www.mustangproject.org

Inhaltsverzeichnis

Über Mustangproject

Mustangproject ist eine Java-Bibliothek zur Unterstützung von erweiterten („ZUGFeRD“-)Metadaten in PDF-Rechnungsdateien. Sie verwendet als Eingabe PDF/A-Dateien, benötigt die Apache PDFBox-Bibliothek und unterliegt wie diese der APL-Lizenz. Sie ist daher, entsprechend den Richtlinien der Apache Public License, unter Einbettung einer entsprechenden „Notice“-Datei kostenlos einsetzbar in kommerziellen und nichtkommerziellen Projekten.

Übersicht von ZUGFeRD-Lösungen

	Plattform	Funktionsumfang				Geeignet für			Preis
		Lesen	XML erzeugen	PDF Schreiben	PDF/A-Umwandlung	Kommerz. Software	Freeware	Open Source	
intarsys	Java	Ja	Ja	Ja	Ja	Ja	Ja	Nein	a.A.
Konik	Java	Ja	Ja	Ja	Nein	Nein	Nein	Ja	0 €
Mustang	Java	Ja	Ja	Ja	Nein	Ja	Ja	Ja	0 €

https://github.com/stephanstapel/ZUGFeRD-csharp	C#	Ja	Ja	Nein	Nein	Ja	Ja	Ja	0 €

Download/Projekteinrichtung

Source code

Heimat der Mustangprojekt-Quelltexte ist https://github.com/Rayman2200/PDFA3

Projekteinrichtung ohne Maven

Mit installiertem OpenOffice.org oder LibreOffice und Eclipse for Java.

1. Starten Sie Eclipse, Neues Java-Eclipse-Projekt erstellen, beispielsweise „MustangSample“.
2. Wechseln Sie in der Shell in den erstellten Ordner.
3. Download von
    1. Apache PDFBox
        1. Downloaden Sie http://apache.openmirror.de/pdfbox/1.8.11/pdfbox-1.8.11.jar
        2. Downloaden Sie http://apache.openmirror.de/pdfbox/1.8.11/preflight-app-1.8.11.jar
        3. Downloaden Sie http://apache.openmirror.de/pdfbox/1.8.11/xmpbox-1.8.11.jar
    2. Mustang
        1. Downloaden Sie http://mustangproject.org/deploy/mustang-1.3.0.jar
        2. Downloaden Sie http://mustangproject.org/deploy/NOTICE
    3. Laden Sie
        1. das Lesebeispiel von http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20151008_504.pdf
        2. und entweder
            1. die Quelldatei im OpenOffice.org-Format http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20151018_504.odt und
                1. Öffnen Sie die Datei in OpenOffice.org
                2. Datei|Exportieren als PDF: Wichtig ist hier, dass Sie die Checkbox PDF/A-1a setzen
                3. Speichern Sie die PDF-Datei beispielsweise als „MustangGnuaccountingBeispielRE-20151008_504blanko.pdf“ im MustangSample-Ordner

2. Alternativ laden Sie direkt die daraus erzeugte PDF-Datei noch ohne ZUGFeRD-Metadaten von http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20151008_504blanko.pdf herunter.

4. Wechseln Sie zurück zu Eclipse. Fügen Sie durch Rechtsklick auf das Projekt („Eigenschaften“) alle vier heruntergeladenen JAR-Dateien Projekteigenschaften als „externe Jars“ zum „Java Build Path“ Reiter „Bibliotheken“ hinzu.

Mit Maven

Mit folgendem Repository

```
<repositories>
    <repository>
        <id>mustang-mvn-repo</id>
        <url>https://raw.github.com/Rayman2200/PDFA3/mvn-repo/</url>
    </repository>
</repositories>
```

und folgender Dependency

```
<dependency>
  <groupId>org.mustangproject.ZUGFeRD</groupId>
  <artifactId>mustang</artifactId>
  <version>1.3.0-SNAPSHOT</version>
</dependency>
```

Bei der Gelegenheit kann man auch gleich PDFBox importieren:

```
<dependency>
    <groupId>org.apache.pdfbox</groupId>
    <artifactId>pdfbox</artifactId>
    <version>1.8.8</version>
</dependency>
```

Lesen von ZUGFeRD-Daten

5. Erstellen Sie eine neue Java-Klasse unterhalb von src, beispielsweise MustangReader. inklusive „Public static void main()“
6. Geben Sie innerhalb von Main „ZUGFeRDImporter zi=**new** ZUGFeRDImporter();“ ein und lassen Sie den Import durch STRG+SHIFT+O ergänzen
7. verwenden Sie zi.extract(PDF-Dateiname) und ggf. canParse() um festzustellen ob es sich um ZUGFeRD-Daten handelt.
8. Nach zi.parse() haben Sie Zugriff auf die getter wie getAmount()
9. Welche Daten enthalten sind, können Sie der XML-Datei entnehmen die im ZUGFeRD-Beispiel-PDF eingebettet ist

Komplettes Lesebeispiel

```
package sample;

import org.mustangproject.ZUGFeRD.ZUGFeRDImporter;
```

```
public class Read {

	public static void main(String[] args) {
		ZUGFeRDImporter zi=new ZUGFeRDImporter();
		zi.extract("./MustangGnuaccountingBeispielRE-20151008_504.pdf");
		System.out.println("Lese ZUGFeRD");
		if (zi.canParse()) {
			zi.parse();
			System.out.println("Fälliger Betrag:"+zi.getAmount());
			System.out.println("BIC:"+zi.getBIC());
			System.out.println("IBAN:"+zi.getIBAN());
			System.out.println("Kontoinhaber:"+zi.getHolder());
			System.out.println("Rechnungsnr:"+zi.getForeignReference());

		}


	}

}
```

Schreiben einer ZUGFeRD-PDF-Datei

Ein Beispielprogramm zum Schreiben ist deshalb umfangreicher, weil erstens mehr Daten in einer differenzierteren Struktur geschrieben werden als derzeit beim Lesen benötigt und zweitens dem Exporter die Daten per Interface zur Verfügung gestellt werden müssen.

1. Erstellen Sie eine neue Klasse unterhalb von src, beispielsweise MustangWriter inklusive des obligatorischen „Public static void main()“ .
2. Ändern Sie `public class MustangWriter` in `public class MustangWriter implements IZUGFeRDExportableTransaction`
3. Fügen Sie innerhalb der Klasse MustangWriter folgende Klassen hinzu

```
1. class Contact implements IZUGFeRDExportableContact {}
2. class Item implements IZUGFeRDExportableItem {
      private BigDecimal price, quantity;
      private Product product;
   }
3. class Product implements IZUGFeRDExportableProduct {
      private String description, name, unit;
      private BigDecimal VATPercent;
}
```

4. Generieren Sie die Imports durch Drücken von STRG+SHIFT+O
5. Markieren Sie den Klassennamen MustangWriter und drücken Sie ALT+SHIFT+S, wählen Sie Override/Implement Methods und drücken Return.
6. Klicken Sie auf Contact und Wiederholen Sie den letzten Schritt.
7. Klicken Sie auf Item, markieren Sie die Variablen und wählen Sie zuerst „*Generate Getters and Setters*“ nach drücken von ALT+SHIFT+S. Wählen Sie alle Member aus und drücken Sie Return.
8. Klicken Sie erneut auf Item, drücken von ALT+SHIFT+S und wählen „Generate Constructor using Fields“. Wählen Sie erneut alle Member aus und drücken Sie Return.

9. Wenden Sie die beiden letzten Schritte auch auf Product an: Klicken Sie auf Product, markieren Sie die Variablen und wählen Sie „Generate Getters and Setters“ nach drücken von ALT+SHIFT+S. Wählen Sie alle Member aus und drücken Sie Return.
10. Item benötigt neben den getter/setter auch noch andere Methode, wählen Sie Item aus, drücken Sie ALT+SHIFT+S, wählen Sie Override/Implement Methods
11. Klicken Sie erneut auf Product, drücken von ALT+SHIFT+S und wählen „Generate Constructor using Fields“. Wählen Sie erneut alle Member aus und drücken Sie Return.
12. Folgende Methoden von Contact sollten Folgendes zurückgeben:

```
1. getCountry(): "DE"
2. getLocation(): "Spielkreis"
3. getName(): "Theodor Est"
4. getStreet(): "Bahnstr. 42"
5. getVATID(): "DE999999999"
6. getZIP(): "88802";
```

13. Folgende Methoden der Hauptklasse -MustangWriter- sollten folgendes zurückgeben:

```
1. getDeliveryDate(): new
   GregorianCalendar(2015,Calendar.OCTOBER,7).getTime()
```

2. Zweimaliges CTRL+SHIFT+O importiert die dazu nötige GregorianCalendar und Calendar Klasse

```
3. getDueDate(): new GregorianCalendar(2015,Calendar.OCTOBER,29).getTime()
4. getIssueDate(): new GregorianCalendar(2015,Calendar.OCTOBER,8).getTime()
5. getNumber(): "RE-20151008/504"
6. getOwnBIC(): "COBADEFXXX"
7. getOwnBankName(): "Commerzbank"
8. getOwnCountry() "DE"
9. getOwnIBAN(): "DE88 2008 0000 0970 3757 00"
10.   getOwnLocation() "Stadthausen"
11.   getOwnOrganisationName(): "Bei Spiel GmbH"
12.   getOwnStreet() "Ecke 12"
13.   getOwnTaxID(): "22/815/0815/4"
14.   getOwnVATID(): "DE136695976"
15.   getOwnZIP(): "12345"
16.   getOwnOrganisationFullPlaintextInfo(): "Bei Spiel GmbH\n"+
   "Ecke 12\n"+
   "12345 Stadthausen\n"+
   "Geschäftsführer: Max Mustermann"
17.   getRecipient(): new Contact()
```

18. getZFItems() der Hauptklasse kann jetzt Produkte anlegen und diese als Array von Posten (Items) zurückliefern:

```
            Item[] allItems=new Item[3];
         Product designProduct=new Product("", "Künstlerische Gestaltung
(Stunde)", "HUR", new BigDecimal("7.000000"));
         Product balloonProduct=new Product("", "Luftballon", "C62", new
BigDecimal("19.000000"));
         Product airProduct=new Product("", "Heiße Luft pro Liter", "LTR", new
BigDecimal("19.000000"));
```

```
            allItems[0]=new Item(new BigDecimal("160"), new BigDecimal("1"),
designProduct);
            allItems[1]=new Item(new BigDecimal("0.79"), new BigDecimal("400"),
balloonProduct);
            allItems[2]=new Item(new BigDecimal("0.10"), new BigDecimal("200"),
airProduct);
            return allItems;
```

19. Legen Sie eine neue Funktion, beispielsweise „apply“ an (private reicht).
20. In der Main-Methode der Hauptklasse instantiiert man jetzt die Klasse und rufen Sie dort apply() auf.
21. In der apply-Methode kann man jetzt
    1. einen ZUGFeRDExporter instantiieren,
    2. dessen PDFmakeA3compliant (mit Dateinamen, „Producer“, also Anwendungs- und „Creator“ also Autorenname) aufrufen,
    3. PDFattachZugferdFile-Methoden (mit this als IZUGFeRDExportableTransaction) aufrufen sowie
    4. die export-Funktion aufrufen. Die apply-Methode sieht dann – mit entsprechenden try/catch-Blöcken- beispielsweise so aus:

```
            try {
                  System.out.println("Lese Blanko-PDF");
                  // automatically add Zugferd to all outgoing invoices
                  ZUGFeRDExporter ze = new ZUGFeRDExporter();
                  System.out.println("Wandle in PDF/A-3u um");
                  ze.PDFmakeA3compliant("./MustangGnuaccountingBeispielRE-
20151008_504blanko.pdf", "My Application",
                              System.getProperty("user.name"), true);
                  System.out.println("ZUGFeRD-Daten generieren und anhängen");
                  ze.PDFattachZugferdFile(this);
                  System.out.println("Schreibe ZUGFeRD-PDF");
                  ze.export("./MustangGnuaccountingBeispielRE-
20151008_504new.pdf");
                  System.out.println("Fertig.");
            } catch (IOException e) {
                  e.printStackTrace();
            } catch (TransformerException e) {
                  e.printStackTrace();
            } catch (JAXBException e) {
                  e.printStackTrace();
            }
```

22. Ein CTRL+SHIFT+O hilft wieder beim Hinzufügen der nötigen Imports
23. „My Application“ und `System.getProperty("user.name")` werden in den Metadaten als „Producer“ (in etwa: erstellende Anwendung) beziehungsweise „Creator“ (in etwa: Autor) gespeichert. Bitte passen Sie die Werte entsprechend Ihrer Anwendung an.
24. Starten Sie, es sollte eine valide ZUFeRD-Rechnung in `./MustangGnuaccountingBeispielRE-20151008_504new.pdf`

erstellt werden.

25. Passen Sie ggf. die NOTICE-Datei an und fügen Sie Ihrer Anwendung hinzu.

Komplettes Schreibbeispiel

Siehe MustangWriter.java in diesem Verzeichnis.

Schreiben eigener XML-Daten

Sollten Sie eine eigene Implementierung verwenden um ZUGFeRD-XML-Daten zu erzeugen können Sie diese mit `setZUGFeRDXMLData` schreiben, `PDFattachZugferdFile` enthält dann einen null-Parameter wie folgt:

```
// automatically add Zugferd to all outgoing invoices
ZUGFeRDExporter ze = new ZUGFeRDExporter();
System.out.println("Converting to PDF/A-3u");
ze.PDFmakeA3compliant("./Source.pdf", "My Application",
        System.getProperty("user.name"), true);
System.out.println("Attaching ZUGFeRD-Data");
String ownZUGFeRDXML = "<some><xml attrib='value'/></some>";
ze.setZUGFeRDXMLData(ownZUGFeRDXML.getBytes());
ze.PDFattachZugferdFile(null);
System.out.println("Writing ZUGFeRD-PDF");
ze.Export("./Target.pdf");
```

Zusatzfunktionen

- `ZUGFeRDExporter.setTest()` setzt ein Attribut im ZUGFeRD-XML, das benutzt wird um eine Testrechnung auszuzeichnen.
- `ZUGFeRDExporter.ignoreA1Errors()` überspringt die Überprüfung der Eingangsdatei auf PDF/A-1-Fehler