Mustang project user documentation

Jochen Stärk

For Mustangproject 1.5.1, 2018-01-13

http://www.mustangproject.org

Inhaltsverzeichnis

About Mustangproject

Mustangproject is a Java-Library for extended („ZUGFeRD"-)metadata in PDF-invoices. It requires the Apache PDFBox library, uses PDF/A files as input and is, like Apache PDFBox subject to the APL-License and can therefore, within the terms of the Apache Public License, be used for free in commercial and noncommercial projects as long as e.g. a according „Notice"-file is placed.

Overview of ZUGFeRD-Solutions

	Platform	License	ZF Versions	Functionality				Viable for			Price
				Read PDF	create XML	write PDF	PDF/A-Conversion	Commercial software	Freeware	Open Source	
intarsys	Java	proprietary	1	Yes	Yes	Yes	Yes	Yes	Yes	No	On request
Konik	Java	AGPL	1	Yes	Yes	Yes	No	No	No	Yes	0 €
Mustang	Java	APL	1	Yes	Yes	Yes	No	Yes	Yes	Yes	0 €

https://github.com/akretion/factur-x	Python	BSD	1,2	No	No	Yes	No	Yes	Yes	Yes	0 €
https://github.com/stephanstapel/ZUGFeRD-csharp	C#	APL	1	Yes	Yes	No	No	Yes	Yes	Yes	0 €

Download/Project setup

Source code

Home of the Mustangprojekt source code is https://github.com/ZUGFeRD/mustangproject/

Project setup without Maven

With installed OpenOffice.org or LibreOffice and Eclipse for Java.

1. Start Eclipse, create a new Java-Eclipse-project, e.g. „MustangSample“.
2. Change to that folder.
3. Download
    1. Mustang
        1. the JAR file http://mustangproject.org/deploy/mustang-1.5.1.jar[1]
        2. the notice file http://mustangproject.org/deploy/NOTICE
    2. Download the sample
        1. Either download the sample PDF-A/1 sample invoice without ZUGFeRD metadata from http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20171118_506blanko.pdf
        2. Alternatively create the invoice PDF yourself
            1. Download http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20171118_506.odt
            2. Open this OpenOffice.org source file in Writer

1 If you anyway embed PDFBox (pdfbox, fontbox, preflight, xmpbox as well as their dependencies apache-commons-io and apache-commons-logging) you can also download the much lighter http://mustangproject.org/deploy/original-mustang-1.5.1.jar

   3. File|Export as PDF: Set the Checkbox PDF/A-1a in the export options
   4. Save the PDF file as „MustangGnuaccountingBeispielRE-20171118_506blanko.pdf“

4. Switch back to Eclipse. Add the downloaded JAR files to your project (right click on project name, Properties) add as „external Jar“ to the „Build Path“ in the „libraries“ tab.

With Maven

The following repository

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

serves the following dependency

```
<dependency>
  <groupId>org.mustangproject.ZUGFeRD</groupId>
  <artifactId>mustang</artifactId>
  <version>1.5.1</version>
</dependency>
```

It's a good idea to also import Apache Commons Logging and JAXB

```
<dependency>
      <groupId>commons-logging</groupId>
      <artifactId>commons-logging</artifactId>
      <version>1.1.1</version>
</dependency>
<dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.2.5</version>
</dependency>
```

Reading ZUGFeRD data

1. Download a proper sample with metadata like http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20171118_506.pdf .
2. Create a new class in the src folder, called Reader. Check the „Public static void main()“ checkbox.
3. Within the main method, enter „ZUGFeRDImporter zi=**new** ZUGFeRDImporter();“ and add the import by pressing STRG+SHIFT+O
4. use zi.extract(PDF-filename) and canParse() to find out if ZUGFeRD-Data is present.
5. After invoking zi.parse() you can access the getter-Methods like getAmount()
6. There are only getters for few properties but additional ones can be added easily. Which data is

available can be seen in the ZUGFeRD-invoice.xml file embedded any ZUGFeRD compliant PDF

Complete sample source code for reading ZUGFeRD data

```
package sample;

import org.mustangproject.ZUGFeRD.ZUGFeRDImporter;

public class Read {

      public static void main(String[] args) {
            ZUGFeRDImporter zi=new ZUGFeRDImporter();
            zi.extract("./MustangGnuaccountingBeispielRE-20171118_506.pdf");
            System.out.println("Reading ZUGFeRD");
            if (zi.canParse()) {
                  zi.parse();
                  System.out.println("Due amount:"+zi.getAmount());
                  System.out.println("BIC:"+zi.getBIC());
                  System.out.println("IBAN:"+zi.getIBAN());
                  System.out.println("Account holder name:"+zi.getHolder());
                  System.out.println("Document:"+zi.getForeignReference());
            }
      }
}
```

Writing a ZUGFeRD-PDF file

A sample for writing ZUGFeRD PDFs is more comprehensive, because

1) more data is being written than read in the read example and
2) the exporter interacts via interfaces with your software.

1. Create a new class in the src-folder, e.g. MustangWriter. Check the checkbox to generate „Public static void main()“ .
2. Change `public class MustangWriter` to `public class MustangWriter implements IZUGFeRDExportableTransaction`
3. Add the following classes in in the same file:
   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. Generate the imports by pressing CTRL+SHIFT+O
5. Click left on MustangWriter and press ALT+SHIFT+S, select Override/Implement Methods and press return.
6. Click on Contact and repeat the last step.
7. Click Item, mark the variables, press ALT+SHIFT+S and select „*Generate Getters and Setters*“ first. Mark all members and press return.
8. Click again on Item, press ALT+SHIFT+S and select „Generate Constructor using Fields“. Choose again all member variables and press return.
9. For Item, „add unimplemented methods“ will add two methods (getItemAllowances and getItemCharges) which will work even if they return null.
10. Repeat the last two steps for „Product“: Click Product, mark the variables, press ALT+SHIFT+S and select „Generate Getters and Setters“. Choose all members and press return.
11. Item also needs other methods besides the getter/setters, press ALT+SHIFT+S, and choose Override/Implement Methods
12. Click on Product again, press ALT+SHIFT+S and select „Generate Constructor using Fields“. Choose all members again and press return.
13. The following methods of Contact should return the following:

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

14. The following methods of the main class should return the following:

```
1. getDeliveryDate(): new
   GregorianCalendar(2017,Calendar.NOVEMBER,17).getTime()
2. Pressing CTRL+SHIFT+O twice will import the necessary GregorianCalendar
   and Calendar class
3. getDueDate(): new GregorianCalendar(2017,Calendar.DECEMBER,9).getTime()
4. getIssueDate(): new GregorianCalendar(2017,Calendar.NOVEMBER,18).getTime()
5. getNumber(): "RE-20171118/506"
6. getOwnBIC(): "COBADEFFXXX"
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() of the main class can now create products and return them as a arrays of items:

```
            Item[] allItems=new Item[3];
        Product designProduct=new Product("", "Künstlerische Gestaltung
(Stunde): Einer Beispielrechnung", "HUR", new BigDecimal("7.000000"));
        Product balloonProduct=new Product("", "Luftballon: Bunt, ca. 500ml",
"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. Now create a private void apply method in the main class
20. Please instantiate this main MustangWriter class in the main method and invoke the apply() function.
21. In the apply-method you can now
    1. create a new ZUGFeRDExporterFromA1Factory, run
    2. setProducer and setCreator on it (e.g. ZUGFeRDExporter ze=new ZUGFeRDExporterFromA1Factory().setProducer("string").setCreator("string")) and get the ZUGFeRDExporter from the factories load("./MustangGnuaccountingBeispielRE-20171118_506new.pdf") method. Feel free to use your own PDF/A-1 invoice file. In this chain (.setProducer.setCreator...) you can also insert setZUGFeRDVersion(2).
    3. use the PDFattachZugferdFile-method (with the IZUGFeRDExportableTransation, i.e. „this" as parameter) on the ZUGFeRDExporter and
    4. use export to save the PDF/A-3 file. The apply-method then looks – with according try/catch-blocks- as follows:

```
        try {
            System.out.println("Reading Blanko-PDF");
            ZUGFeRDExporter ze = new
ZUGFeRDExporterFromA1Factory().setProducer("My
Application").setCreator(System.getProperty("user.name")).load("./MustangGnuaccount
ingBeispielRE-20171118_506blanko.pdf");
            System.out.println("Generating and attaching ZUGFeRD-Data");
            ze.PDFattachZugferdFile(this);
            System.out.println("Writing ZUGFeRD-PDF");
            ze.export("./MustangGnuaccountingBeispielRE-
20171118_506new.pdf");
            System.out.println("Done.");
        } catch (IOException e) {
            e.printStackTrace();
        }
```

22. CTRL+SHIFT+O again helps with the imports
23. „My Application“ and `System.getProperty("user.name")` are stored in the meta data as „Producer“ (producing application) respectively „Creator“ (author). Please adjust accordingly.
24. Start it to write the ZUGFeRD invoice `MustangGnuaccountingBeispielRE-20171118_506new.pdf` to the file specified in export.
25. Adjust the NOTICE-File and add it to your application.
26. Make sure the XML data in ZUGFeRD-invoice.xml in the created file always matches the PDF content visible to humans.

The target file contains ZUGFeRD-invoice.xml instead of factur-x.xml from the official sample unless the ZUGFeRD-Version was set to 2 in the factory.

Complete source code example for writing ZUGFeRD PDFs

Please refer to the file MustangWriter.java in this directory.

Writing custom XML-Data

If you create your own ZUGFeRD-XML you can attach them using `setZUGFeRDXMLData`:

```
		ZUGFeRDExporter ze;
		try {

			System.out.println("Converting to PDF/A-3u");
			ze = new 
ZUGFeRDExporterFromA1Factory().setProducer("My 
Application").setCreator(System.getProperty("user.name")).load("./MustangGnuaccountingBeispielRE-
20171118_506blanko.pdf");


			System.out.println("Attaching ZUGFeRD-Data");
			String ownZUGFeRDXML = "<rsm:CrossIndustryDocument></rsm:CrossIndustryDocument>";
			ze.setZUGFeRDXMLData(ownZUGFeRDXML.getBytes());
			System.out.println("Writing ZUGFeRD-PDF");
			ze.export("./Target.pdf");
		} catch (IOException e) {
			e.printStackTrace();
		}
```

Mustangproject checks if the input PDF/A file looks halfway valid and the XML data contains „<rsm:CrossIndustry“ which is the case for both ZF1 (CrossIndustryDocument) and ZF2 files (CrossIndustryInvoice). Still in the factory you can use setZUGFeRDVersion and setZUGFeRDConformanceLevel to set the version respective profile of the XML you are inserting.

Supplementary functions

- `ZUGFeRDExporter.setTest()` sets a attribute in the XML structure used to identify test invoices.

- `ZUGFeRDExporter.ignoreA1Errors()` skips the check of the input file whether it's valid PDF/A-1
- A first attempt to migrate from ZF1 to ZF2 can be done with something like `String facturx=new ZUGFeRDMigrator().migrateFromV1ToV2(zugferdInvoice);`