

Mustang project developer documentation

Jochen Stärk

Draft, 25.06.2014

www.mustangproject.org

Overview

	Platform	License	Functionality			Viable for			Price
			Read PDF	write XML	write PDF	Commercial software	Freeware	Open Source	
intarsys	Java	proprietary	✓	✓	✓	✓	✗	✗	On request
Konik	Java	AGPL	✓	✓	✓	✗	✗	✓	0 €
Mustang	Java	APL	✓	✓	✓	✓	✓	✓	0 €
https://github.com/stephanstapel/ZUGFeRD-csharp	C#	APL	✓	✓	✗	✓	✓	✓	0 €
https://github.com/opendatalab-de/zugferd	Java	APL	✓	✗	✗	✓	✓	✓	0 €

Mustang

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

1. Start Eclipse, create a new Java-Eclipse-project, e.g. „sample“. Change to that folder.
2. Download
 1. Apache PDFBox
 1. from <http://apache.openmirror.de/pdfbox/1.8.5/pdfbox-1.8.5.jar>
 2. from <http://apache.openmirror.de/pdfbox/1.8.5/preflight-app-1.8.5.jar>
 3. from <http://apache.openmirror.de/pdfbox/1.8.5/xmpbox-1.8.5.jar>
 2. Mustang
 1. from <https://github.com/Rayman2200/PDFA3/raw/master/mustang/target/mustang-1.0.jar>
 2. from <https://raw.githubusercontent.com/Rayman2200/PDFA3/master/mustang/src/main/java/org/mustangproject/ZUGFeRD/NOTICE>
3. Download the sample
 1. from http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140522_501.pdf
 2. the OpenOffice.org source from http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140522_501.odt
4. Open OpenOffice.org.
 1. Open the OpenOffice.org source file in writer
 2. File|Export as PDF: Set the Checkbox PDF/A-1a in the export options
 3. Save the PDF-Datei (without ZUGFeRD) as „blank.pdf“ in the sample-folder.
3. Switch back to Eclipse. Add all four downloaded JAR files to your project (Project properties) as „external Jar“ to the „Build Path“ .

Reading ZUGFeRD

4. Create a new class in the src folder, called Reader. Check the „Public static void main()“ checkbox.
5. Within the main method, enter „ZUGFeRDImporter zi=**new** ZUGFeRDImporter();“ and add the import by pressing STRG+SHIFT+O
6. use zi.extract(PDF-filename, in this case [MustangGnuaccountingBeispielRE-20140522_501.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140522_501.pdf)) and canParse() to find out if ZUGFeRD-Data is present.
7. After invoking zi.parse() you can access the getter-Methods like getAmount()
8. 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, including [MustangGnuaccountingBeispielRE-20140522_501.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140522_501.pdf)

Complete sample source code for reading

```
package sample;

import org.mustangproject.ZUGFeRD.ZUGFeRDImporter;

public class Read {

    public static void main(String[] args) {
        ZUGFeRDImporter zi=new ZUGFeRDImporter();
        zi.extract("./MustangGnuaccountingBeispielRE-20140522_501.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());
        }

    }

}
```

Writing ZUGFeRD

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 in a kind of „pull-method“. While this avoids redundant data a sample is more exhaustive because the sample has to store the data in the memory, which any productive software already does.

The alternative ZUGFeRD-Open-Source-project Konik follows a more conventional „push-method“ in which data is stored redundantly (if used alongside a ordinary sotware) by using setter-methods but which conveniently does not require you to cater for the availability of the getter methods.

1. Create a new class in the src-folder, e.g. MainClass. Check the checkbox to generate „Public static void main()“.
2. Change **public class** MainClass to **public class** MainClass **implements** IZUGFeRDExportableTransaction
3. in the same file, add **class** Contact **implements** IZUGFeRDExportableContact {}
4. in the same file, add **class** Item **implements** IZUGFeRDExportableItem {}
5. in the same file, add **class** Product **implements** IZUGFeRDExportableProduct {}

6. Generate the imports by pressing CTRL+SHIFT+O
7. Click left on MainClass and press ALT+SHIFT+S, select Override/Implement Methods and press return.
8. Click on Contact and repeat the last step.
9. Click on Item and repeat the last step.
10. Click on Products and repeat the last step.
11. 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(): ""`
 6. `getZIP(): "88802";`
12. The following methods of the main class should return the following:
 1. `getDueDate(): new GregorianCalendar(2014,Calendar.JUNE,12).getTime()`
 2. `getIssueDate(): new GregorianCalendar(2014,Calendar.MAY,22).getTime()`
 3. `getNumber(): "RE-20140522/501"`
 4. `getOwnBIC(): "COBADEFXXX"`
 5. `getOwnBankName(): ""`
 6. `getOwnIBAN(): "DE88 2008 0000 0970 3757 00"`
 7. `getOwnOrganisationName(): "Bei Spiel GmbH"`
 8. `getOwnTaxID(): "22/815/0815/4"`
 9. `getOwnVATID(): "DE136695976"`
 10. `getRecipient(): new Contact()`
 11. `getTotal(): new BigDecimal("496.00")`
 12. `getTotalGross(): new BigDecimal("571.04")`
 13. `getDeliveryDate() new Date();`
 14. `getOwnCountry() "DE"`
 15. `getOwnLocation() "Test city"`
 16. `getOwnStreet() "Test Street 22"`
 17. `getOwnZIP() "12345"`
 18. The Item- as well as the Product-class should return member variables in the overwritten methods, which should be set-able in the constructor.
 19. `getZFItems()` of the main class can now create products and return them as a array of items:


```

          Item[] allItems=new Item[3];
          Product design nProduct=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("171.20"),
new BigDecimal("1"), new BigDecimal("171.20"), designProduct);
          
```

```

        allItems[1]=new Item(new BigDecimal("0.79"), new BigDecimal("0.94"),
new BigDecimal("400"), new BigDecimal("376.04"), balloonProduct);
        allItems[2]=new Item(new BigDecimal("0.10"), new BigDecimal("0.12"),
new BigDecimal("200"), new BigDecimal("23.80"), airProduct);
        return allItems;

```

20. Now you instantiate this main class in the main method and invoke a apply() method you create.

21. In the apply-method you can now

1. load a PDDocument
2. instantiate a ZUGFeRDEExporter ,
3. invoke the ZUGFeRDEExporter's PDFmakeA3compliant (including the „Producer“, i.e. Application- and „Creator“ ,i.e. Author name parameters) and
4. finally use the PDFattachZugferdFile-method (with the IZUGFeRDEExportableTransation, i.e. „this“ as parameter) and
5. save the PDDocument again. The apply-method then looks - with according try/catch-blocks- as follows:

```

PDDocument doc;

try {
    doc = PDDocument.load("blank.pdf");
    // automatically add Zugferd to all outgoing invoices
    ZUGFeRDEExporter ze = new ZUGFeRDEExporter();
    ze.PDFmakeA3compliant(doc, "My Application",
        System.getProperty("user.name"), true);
    ze.PDFattachZugferdFile(doc, this);

    doc.save("unblank.pdf");
} catch (IOException e) {
    e.printStackTrace();
} catch (TransformerException e) {
    e.printStackTrace();
} catch (COSVisitorException e) {
    e.printStackTrace();
}

```

Complete source code example for writing ZUGFeRD PDFs

```

package sample;

import java.io.IOException;
import java.math.BigDecimal;
import java.util.Calendar;
import java.util.Date;
import java.util.GregorianCalendar;
import java.util.HashMap;

```

```

import javax.xml.transform.TransformerException;

import org.apache.pdfbox.exceptions.COSVisitorException;
import org.apache.pdfbox.pdmodel.PDDocument;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableContact;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableItem;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableProduct;
import org.mustangproject.ZUGFeRD.IZUGFeRDExportableTransaction;
import org.mustangproject.ZUGFeRD.ZUGFeRDExporter;

class Contact implements IZUGFeRDExportableContact {

    @Override
    public String getCountry() {
        return "DE";
    }

    @Override
    public String getLocation() {
        return "Spielkreis";
    }

    @Override
    public String getName() {
        return "Theodor Est";
    }

    @Override
    public String getStreet() {
        return "Bahnstr. 42";
    }

    @Override
    public String getVATID() {
        return "";
    }

    @Override
    public String getZIP() {
        return "88802";
    }
}

class Product implements IZUGFeRDExportableProduct {

    private String description, name, unit;
    private BigDecimal VatPercent;

    public Product (String description, String name, String unit, BigDecimal
VatPercent) {
        this.description=description;
        this.name=name;
        this.unit=unit;
        this.VatPercent=VatPercent;
    }
}

```

```

    }
    @Override
    public String getDescription() {
        return description;
    }

    @Override
    public String getName() {
        return name;
    }

    @Override
    public String getUnit() {
        return unit;
    }

    @Override
    public BigDecimal getVATPercent() {
        return VatPercent;
    }
}
class Item implements IZUGFeRDEExportableItem {

    private BigDecimal price, priceGross, quantity, totalGross;
    private Product product;

    public Item(BigDecimal price, BigDecimal priceGross, BigDecimal
quantity, BigDecimal totalGross, Product product) {
        this.price=price;
        this.priceGross=priceGross;
        this.quantity=quantity;
        this.totalGross=totalGross;
        this.product=product;
    }

    @Override
    public BigDecimal getPrice() {
        return price;
    }

    @Override
    public BigDecimal getPriceGross() {
        return priceGross;
    }

    @Override
    public IZUGFeRDEExportableProduct getProduct() {
        return product;
    }

    @Override
    public BigDecimal getQuantity() {
        return quantity;
    }
}

```

```

@Override
public BigDecimal getTotalGross() {
    return totalGross;
}

}

public class MainClass implements IZUGFeRExportableTransaction{

@Override
public Date getDueDate() {
    return new GregorianCalendar(2014,Calendar.JUNE,12).getTime();
}

@Override
public Date getIssueDate() {
    return new GregorianCalendar(2014,Calendar.MAY,22).getTime();
}

@Override
public String getNumber() {
    return "RE-20140522/501";
}

@Override
public String getOwnBIC() {
    return "COBADEFXXX";
}

@Override
public String getOwnBankName() {
    return "";
}

@Override
public String getOwnIBAN() {
    return "DE88 2008 0000 0970 3757 00";
}

@Override
public String getOwnOrganisationName() {
    return "Bei Spiel GmbH";
}

@Override
public String getOwnTaxID() {
    return "22/815/0815/4";
}

@Override
public String getOwnVATID() {
    return "DE136695976";
}
}

```



```

@Override
public IZUGFeRDEExportableContact getRecipient() {
    return new Contact();
}

@Override
public BigDecimal getTotal() {
    return new BigDecimal("496.00");
}

@Override
public BigDecimal getTotalGross() {
    return new BigDecimal("571.04");
}

@Override
public Date getDeliveryDate() {

    return new Date();
}

@Override
public String getOwnCountry() {
    return "DE";
}

@Override
public String getOwnLocation() {
    return "Test city";
}

@Override
public String getOwnStreet() {
    return "Test Street 22";
}

@Override
public String getOwnZIP() {
    return "12345";
}

@Override
public IZUGFeRDEExportableItem[] getZFItems() {
    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("171.20"),
new BigDecimal("1"), new BigDecimal("171.20"), designProduct);

```

```

        allItems[1]=new Item(new BigDecimal("0.79"), new BigDecimal("0.94"),
new BigDecimal("400"), new BigDecimal("376.04"), balloonProduct);
        allItems[2]=new Item(new BigDecimal("0.10"), new BigDecimal("0.12"),
new BigDecimal("200"), new BigDecimal("23.80"), airProduct);
        return allItems;
    }

    public void apply() {
        PDDocument doc;
        try {
            doc =
PDDocument.load("/home/jstaerk/workspace/sample/blank.pdf");
            // automatically add Zugferd to all outgoing invoices
            ZUGFeRDExporter ze = new ZUGFeRDExporter();
            ze.PDFmakeA3compliant(doc, "My Application",
                System.getProperty("user.name"), true);
            ze.PDFattachZugferdFile(doc, this);

            doc.save("unblank.pdf");
        } catch (IOException e) {
            e.printStackTrace();
        } catch (TransformerException e) {
            e.printStackTrace();
        } catch (COSVisitorException e) {
            e.printStackTrace();
        }
        System.out.println("Hello ZUGFeRD");
    }

    public static void main(String[] args) {
        MainClass write=new MainClass();
        write.apply();
    }
}

```