

Mustang Project Nutzerdokumentation

Jochen Stärk

zu Mustangproject 1.1.2, 15.06.2015

<http://www.mustangproject.org>

Inhaltsverzeichnis

Mustang Project Nutzerdokumentation.....	1
Über Mustangproject.....	1
Übersicht von ZUGFeRD-Lösungen.....	1
Download/Projekteinrichtung.....	2
Source code.....	2
Projekteinrichtung ohne Maven.....	2
Mit Maven.....	3
Lesen von ZUGFeRD-Daten.....	3
Komplettes Lesebeispiel	3
Schreiben einer ZUGFeRD-PDF-Datei.....	4
Komplettes Schreibbeispiel.....	7
Schreiben eigener XML-Daten.....	11
Zusatzfunktionen.....	11

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

https://github.com/stephanstapel/ZUGFeRD-csharp	C#	APL	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/1.8.8/pdfbox-1.8.8.jar>
 2. Downloaden Sie <http://apache.openmirror.de/pdfbox/1.8.8/preflight-app-1.8.8.jar>
 3. Downloaden Sie <http://apache.openmirror.de/pdfbox/1.8.8/xmpbox-1.8.8.jar>
 2. Mustang
 1. Downloaden Sie <http://mustangproject.org/deploy/mustang-1.1.2.jar>
 2. Downloaden Sie <http://mustangproject.org/deploy/NOTICE>
 3. Laden Sie
 1. das Lesebeispiel von http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20140703_502.pdf
 2. und entweder
 1. die Quelldatei im OpenOffice.org-Format http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613_503.odt und
 2. Öffnen Sie die Datei in OpenOffice.org
 3. Datei|Exportieren als PDF: Wichtig ist hier, dass Sie die Checkbox PDF/A-1a setzen
 4. Speichern Sie die PDF-Datei beispielsweise als „[MustangGnuaccountingBeispielRE-20150613_503blanko.pdf](http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613_503blanko.pdf)“ im MustangSample-Ordner

2. Alternativ laden Sie direkt die daraus erzeugte PDF-Datei noch ohne ZUGFeRD-Metadaten von http://www.mustangproject.org/MustangGnuaccountingBeispielRE-20150613_503blanko.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>1.1.2</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-20150613_503.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 der Exporter sich seine Daten direkt per Interface, solzusagen in einer Art „pull-Ansatz“ holt. So wird zwar eine redundante Datenhaltung vermieden, ein Beispielprogramm muss jedoch Sorge tragen die Daten in geeigneter Weise zumindest im Arbeitsspeicher vorzuhalten. Bei der Einbindung in eine produktive Warenwirtschaft entfällt dieser Schritt, da die Warenwirtschaft die Daten ja bereits zu beliebigen Zeitpunkten zur Verfügung stellt.

Das derzeit alternative ZUGFeRD-Open-Source-Projekt Konik (<http://konik.io>) verfolgt einen konventionellen „push-Ansatz“ in dem – für Beispielprogramme einfacher – Daten redundant durch setter-Methoden gesetzt werden.

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, priceGross, quantity, totalGross;
              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 „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. 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.
11. 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";
```

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

```
1. getDeliveryDate(): new GregorianCalendar(2014, Calendar.JULY, 3).getTime()
2. CTRL+SHIFT+O importiert die dazu nötige GregorianCalendar Klasse
3. getDueDate(): new GregorianCalendar(2014, Calendar.JULY, 24).getTime()
4. getIssueDate(): new GregorianCalendar(2014, Calendar.JULY, 3).getTime()
5. getNumber(): "RE-20140703/502"
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. getRecipient(): new Contact()
17. getTotal(): new BigDecimal("496.00")
18. getTotalGross(): new BigDecimal("571.04")
```

19. 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("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. Legen Sie eine neue Funktion, beispielsweise namens „apply“ an (private reicht).
21. In der Main-Methode der Hauptklasse instantiiert man jetzt die Klasse und rufen Sie dort apply() auf.
22. In der apply-Methode kann man jetzt ein PDDocument
 1. laden,
 2. einen ZUGFeRDEExporter instantiieren,
 3. dessen PDFmakeA3compliant (mit „Producer“, also Anwendungs- und „Creator“ also Autorennamen) und
 4. PDFattachZugferdFile-Methoden (mit this als IZUGFeRDEExportableTransaction) aufrufen sowie
 5. das PDDocument wieder speichern. Die apply-Methode sieht dann – mit entsprechenden try/catch-Blöcken- beispielsweise so aus:

```

PDDocument doc;

try {
    System.out.println("Lese Blanko-PDF");
    doc = PDDocument.load("./MustangGnuaccountingBeispielRE-
20150613_503blanko.pdf");
    // automatically add Zugferd to all outgoing invoices
    ZUGFeRDEExporter ze = new ZUGFeRDEExporter();
    System.out.println("Wandle in PDF/A-3 um");
    ze.PDFmakeA3compliant(doc, "My Application",
        System.getProperty("user.name"), true);
    System.out.println("ZUGFeRD-Daten generieren und anhängen");
    ze.PDFattachZugferdFile(doc, this);
    System.out.println("Schreibe ZUGFeRD-PDF");
    doc.save("./MustangGnuaccountingBeispielRE-20150613_503new.pdf");
    System.out.println("Fertig.");
} catch (IOException e) {
    e.printStackTrace();
} catch (TransformerException e) {
    e.printStackTrace();
} catch (COSVisitorException e) {
    e.printStackTrace();
}

```

23. Ein CTRL+SHIFT+O hilft wieder beim Hinzufügen der nötigen Imports
24. „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.
25. Passen Sie die NOTICE-Datei an und fügen Sie Ihrer Anwendung hinzu.

Komplettes Schreibbeispiel

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

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;

public class MustangWriter implements IZUGFeRDExportableTransaction {

    @Override
    public Date getDeliveryDate() {
        return new GregorianCalendar(2014, Calendar.JULY, 3).getTime();
    }

    @Override
    public Date getDueDate() {
        return new GregorianCalendar(2014, Calendar.JULY, 24).getTime();
    }

    @Override
    public Date getIssueDate() {
        return new GregorianCalendar(2014, Calendar.JULY, 3).getTime();
    }

    @Override
    public String getNumber() {
        return "RE-20140703/502";
    }

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

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

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

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

    @Override
    public String getOwnLocation() {
        return "Stadthausen";
    }
}
```

```

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

@Override
public String getOwnStreet() {
    return "Ecke 12";
}

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

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

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

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

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

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

@Override
public IZUGFeRExportableItem[] 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;
}

class Contact implements IZUGFeRExportableContact {

    @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 "DE999999999";
}

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

class Item implements IZUGFeRDEExportableItem {

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

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

    public BigDecimal getPrice() {
        return price;
    }

    public void setPrice(BigDecimal price) {
        this.price = price;
    }

    public BigDecimal getPriceGross() {
        return priceGross;
    }

    public void setPriceGross(BigDecimal priceGross) {
        this.priceGross = priceGross;
    }

    public BigDecimal getQuantity() {
        return quantity;
    }

    public void setQuantity(BigDecimal quantity) {
        this.quantity = quantity;
    }

    public BigDecimal getTotalGross() {
        return totalGross;
    }

    public void setTotalGross(BigDecimal totalGross) {
        this.totalGross = totalGross;
    }

    public Product getProduct() {
        return product;
    }
}

```

```

        public void setProduct(Product product) {
            this.product = product;
        }
    }

    class Product implements IZUGFeRExportableProduct {
        private String description, name, unit;
        private BigDecimal VATPercent;

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

        public String getDescription() {
            return description;
        }

        public void setDescription(String description) {
            this.description = description;
        }

        public String getName() {
            return name;
        }

        public void setName(String name) {
            this.name = name;
        }

        public String getUnit() {
            return unit;
        }

        public void setUnit(String unit) {
            this.unit = unit;
        }

        public BigDecimal getVATPercent() {
            return VATPercent;
        }

        public void setVATPercent(BigDecimal vATPercent) {
            VATPercent = vATPercent;
        }
    }

    private void apply() {
        PDDocument doc;
        try {
            System.out.println("Lese Blanko-PDF");
            doc = PDDocument.load("./MustangGnuaccountingBeispielRE-20150613_503blanko.pdf");
            // automatically add Zugferd to all outgoing invoices
            ZUGFeRExporter ze = new ZUGFeRExporter();
            System.out.println("Wandle in PDF/A-3u um");
            ze.PDFmakeA3compliant(doc, "My Application",
                System.getProperty("user.name"), true);
            System.out.println("ZUGFeRD-Daten generieren und anhängen");
            ze.PDFattachZugferdFile(doc, this);
            System.out.println("Schreibe ZUGFeRD-PDF");
            doc.save("./MustangGnuaccountingBeispielRE-20150613_503new.pdf");
            System.out.println("Fertig.");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

```

```

    } catch (TransformerException e) {
        e.printStackTrace();
    } catch (COSVisitorException e) {
        e.printStackTrace();
    }
}

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

```

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:

```

    doc = PDDocument.load("./Source.pdf");
    // automatically add Zugferd to all outgoing invoices
    ZUGFeRDExporter ze = new ZUGFeRDExporter();
    System.out.println("Converting to PDF/A-3u");
    ze.PDFmakeA3compliant(doc, "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(doc, null);
    System.out.println("Writing ZUGFeRD-PDF");
    doc.save("./Target.pdf");

```

Zusatzfunktionen

`ZUGFeRDExporter.setTest()` setzt ein Attribut im ZUGFeRD-XML, das benutzt wird um eine Testrechnung auszuzeichnen.