

Name:	Plex
Beschreibung:	Bibliothek für den einfachen Zugriff auf Microsoft Excel Dateien
Version:	0.1
Release-Datum:	14.09.2010
Downloads:	plex.zip (vollständige Distribution)
Lizenz:	LGPL
Historie:	0.1 - Erste offizielle Version.
Subversion:	https://www.kasisoft.com/svn/plex/trunk
JIRA:	http://www.kasisoft.com/jira/browse/PLEX

1 Motivation

Aufgrund meiner beruflichen Erfahrung weiß ich, daß zahlreiche technische Informationen mit Hilfe von [Microsoft Excel](#) Dateien ausgetauscht werden. Mit Hilfe der [Apache POI](#) Bibliothek können diese Dateien für die weitere Verarbeitung ausgewertet werden. Bei der Verarbeitung mußten jedoch immer die gleichen Probleme gelöst werden:

- Erfassung der Tabellen-Struktur.
- Umwandlung der einzelnen Zell-Inhalte in entsprechende Datentypen.
- Fehlerbehandlung, falls Zell-Inhalte unpassend sind.
- Repräsentation der Daten in einer handlicheren Datenstruktur.

Aus diesem Grund habe ich *Plex* geschrieben mit dessen Hilfe die Verarbeitung von MS Excel Dateien vereinfacht werden soll. Im Wesentlichen handelt es sich hierbei um einen Importer, der einen Datenbestand aus MS Excel Dateien in ein oder mehrere Java *TableModel* Instanzen importiert. Der Import-Prozess wird dabei gesteuert durch eine Deklaration des Datenbestandes.

2 Einführung

Die Verarbeitung von MS Excel Dateien besteht eigentlich nur aus zwei Zutaten: Der PLEX-Deklaration und dem Importer, welcher Teil von Plex ist. Bei der PLEX-Deklaration handelt es sich um eine XML-Beschreibung der Tabellen-Strukturen. Diese dient als Eingabe für den Importer um die Tabellen-Daten laden zu können. Folgender Code-Ausschnitt zeigt den Import eines Daten-Bestandes:

```
File    declaration = new File( "mydecl.plex" );
Importer importer   = null;
try {
    importer = new Importer( declaration.toURI().toURL() );
} catch( MalformedURLException ex ) {
    // we've made sure that the resource and thus the URL is correct, so ignore this
} catch( PLEXException ex ) {
    // the declaration file was obviously invalid
}

try {
    File    excel    = new File( "systemmatrix.xls" );
    PlainExcel plex    = importer.runImport( excel );
    System.out.println( plex );
} catch( PLEXException ex ) {
    // the import failed for some reason
}
```

Dieses einfache Beispiel zeigt wie mit Hilfe des Importers eine *PlainExcel* Instanz erzeugt wird, welche dem importierten Datenbestand entspricht. Dabei handelt es sich im Wesentlichen um eine Kollektion von *PlainSheet* Instanzen welche jeweils die Schnittstelle *TableModel* realisieren. Dadurch lassen sich schnell einfache Anzeigen für die Visualisierung der Datenbestände verwirklichen.

Der eigentliche Aufwand für den Import besteht in der Erstellung der PLEX-Deklaration und ggf. der Realisierung einiger Schnittstellen zur Auswertung der MS Excel Datei.

3 Die PLEX-Deklaration

Bei der PLEX-Deklaration handelt es sich um eine XML-Beschreibung der MS Excel Datei, die zur Steuerung des Importers dient. Eine passende Schema-Beschreibung ist Teil der PLEX-Distribution. Die Beschreibung sieht dabei immer wie folgt aus:

```
<plex>

  <general>
    <!-- [1] Beschreibung der APIs -->
    <interface
      api="transform"
      id="cleanup"
      classname="com.kasisoft.lgpl.libs.plex.impl.CleanupTransform"
    />
  </general>

  <!-- Es muß mindestens immer ein Tabellenblatt deklariert werden. -->
  <sheet name="namelist" firstrow="0">
    <!-- [2] Beschreibung der Spalten -->
  </sheet>

</plex>
```

Im ersten Teil [1] werden die API-Implementationen deklariert. Hier werden verschiedene Implementationen mit eindeutigen IDs versehen um für die Auswertung der MS Excel Datei genutzt werden zu können. Die Klasse *CleanupTransform* stellt beispielsweise sicher, daß Zell-Inhalte niemals leere Zeichenketten sind und daher entweder *null* oder einen nicht-leeren Wert beinhalten.

Der zweite Teil [2] beschreibt die Spalten einer Tabelle je Tabellenblatt und damit die Inhalte, welche importiert werden soll. Alle Daten die sich "außerhalb" der Deklaration befinden, werden ignoriert.

3.1 Die Tabellenblätter

Jedes *sheet* Element beschreibt die zu importierenden Daten zu einem Tabellenblatt innerhalb der MS Excel Datei. Hierzu kann über die Attribute *name* oder *namepattern* die Identifikation der Excel Tabellenblätter erfolgen. Während *name* ein bestimmtes Tabellenblatt identifiziert, können mit einem regulären Ausdruck im Attribut *namepattern* mehrere gleichartige Tabellenblätter beschrieben werden.

Mit folgender Deklaration wird nur ein Tabellenblatt mit dem Namen "personen" importiert, sofern dieses existiert:

```
<sheet name="personen">
  <!-- Beschreibung der Spalten -->
</sheet>
```

Der Import gleichartiger Tabellenblätter kann wie folgt deklariert werden:

```
<sheet namepattern="statistic-[0-9]{1,3}">
  <!-- Beschreibung der Spalten -->
</sheet>
```

Hier werden alle Tabellenblätter importiert, deren Namen mit "statistics-" beginnen und mit einer 1-3 stelligen Zahl enden (bspw. "statistics-76", "statistics-9" etc.)

3.2 Datenbereich: Die Zeilen

Für die Festlegung der zu importierenden Zeilen muß die erste Zeile angegeben. Eine Obergrenze wird gegenwärtig nicht angegeben, da diese durch die letzte Zeile mit Inhalten vorgegeben ist.

Der einfachste Weg die erste Zeile anzugeben ist die Verwendung des Attributes *firstrow*:

```
<sheet name="fluffy" firstrow="20">
  <!-- Beschreibung der Spalten -->
```

```
</sheet>
```

Mit dieser Deklaration werden ab Zeile 20 alle Daten importiert. Da die Zeilenangabe mit 0 entspricht dies in Excel der Zeilennummer 21. Wenn man es mit gleichartigen MS Excel Dateien zu tun hat bei denen die erste Zeilennummer jedoch variiert kann man alternativ die erste Zeile dynamisch ermitteln lassen. Dazu muß eine entsprechende API Funktion wie folgt deklariert und verwendet werden:

```
<plex>

  <general>
    <interface
      api="row"
      id="rowlookup"
      classname="com.kasisoft.lgpl.libs.plex.impl.SimpleRowResolver"
    />
  </general>

  <sheet name="namelist">
    <firstrowdetect refid="rowlookup">
      <arg>1</arg>
    </firstrowdetect>
    <!-- Beschreibung der Spalten -->
  </sheet>

</plex>
```

Im *general* Block wurde der Lookup-Mechanismus für die erste Zeile deklariert. Hier wurde eine Implementation *SimpleRowResolver* angegeben, die Teil von PLEX ist. Selbstverständlich können auch eigene Implementationen der Schnittstelle *RowResolver* genutzt werden. Diese Funktion liefert die erste Zeile mit Dateninhalten. Im Element *firstrowdetect* wird diese Funktion benutzt wobei das Argument '1' einen zu addierenden Offset angibt. Welche Argumente möglich sind hängt natürlich von der konkreten Implementation ab.

Wenn wir annehmen, daß die Daten auf dem Tabellenblatt *namelist* ab Excel-Zeile 34 beginnen, dann liefert die Funktion selbst den Wert 33. Inclusive des Offsets 1 entspräche dies einer ersten Zeile 34 (Excel-Zeile 35).

3.3 Datenbereich: Die Spalten

Das wichtigste Beschreibungselement für die zu importierenden Daten ist die Festlegung der Spalten. Hierzu gehört immer die Angabe des *title* Attributes welche auch den Title (=Name) jeder Spalte einer *PlainSheet* Instanz angibt.

Die einfachste Möglichkeit zur Festlegung der Spalte ist die Festlegung der Position innerhalb der MS Excel Datei:

```
<sheet name="namelist" firstrow="1">
  <column title="callname" column="2">
  </column>
</sheet>
```

Analog zur Angabe der Zeilen sind auch die Spaltennummern 0-basiert, d.h. hier wird die Spalte 'C' für den Import festgelegt. Zur besseren Lesbarkeit kann man jedoch auch den Spaltennamen direkt angeben (keine Unterscheidung zwischen Groß- und Kleinschreibung):

```
<sheet name="namelist" firstrow="1">
  <column title="callname" column="Aa">
  </column>
</sheet>
```

Das hier gezeigte Beispiel entspricht der Spaltennummer 26. Wie bei den Zeilen kann auch für die Spalten ein Lookup-Mechanismus genutzt werden:

```

<plex>

  <general>
    <interface
      api="column"
      id="columnlookup"
      classname="com.kasisoft.lgpl.libs.plex.impl.SimpleColumnResolver"
    />
  </general>

  <sheet name="namelist" firstrow="1">
    <column title="callname">
      <columndetect refid="columnlookup">
      </columndetect/>
    </column>
  </sheet>

</plex>

```

Die hier verwendete Implementation der Schnittstelle *ColumnResolver* ermittelt einfach nur die erste Spalte mit einem Zelleninhalt.

3.4 Behandlung von Zellinhalte

Jede Zelle in einer MS Excel Datei hat einen bestimmten Typen, der wie folgt durch PLEX abgebildet wird:

MS Excel	Java
Blank (Leere Zelle)	<i>null</i>
Formula	<i>null</i>
Error	<i>null</i>
Boolean	java.lang.Boolean
Numeric	java.lang.Double
String	java.lang.String

Abgesehen von der Tatsache, daß die Excel Typen innerhalb einer Spalte nicht zwingenderweise konsistent sein müssen, kann es noch weitere Gründe geben die Zell-Inhalte nachzuverarbeiten. Für jede Spalten-Deklaration können beliebig viele Verarbeitungsfunktionen angegeben werden, die eine Umwandlung des Zell-Inhaltes erlauben:

```

<plex>

  <general>
    <interface
      api="transform"
      id="cleanup"
      classname="com.kasisoft.lgpl.libs.plex.impl.CleanupTransform"
    />
  </general>

  <sheet name="namelist" firstrow="1">
    <column title="callname" column="A">
      <transformer refid="cleanup"/>
    </column>
  </sheet>

</plex>

```

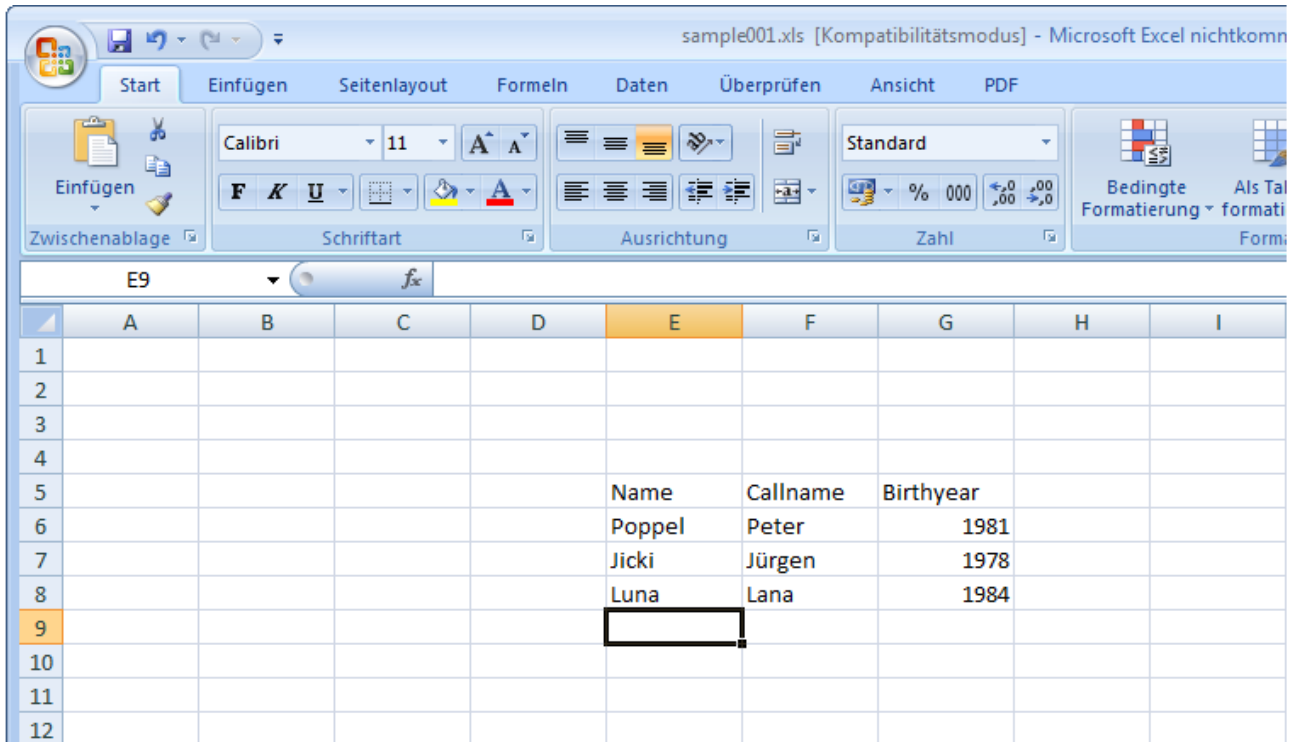
Die *transformer* Elemente werden in der angegebenen Reihenfolge ausgeführt und können nach Belieben eingesetzt werden.

4 Beispiel-Deklarationen

Um den Einstieg in den Import von MS Excel Dateien zu erleichtern werden hier einige Beispiele besprochen, die einige typische Anwendungsfälle beschreiben. Die hier aufgeführten Beispiele sind auch Teil der PLEX-Distribution.

4.1 Einfache festgelegte Tabellen [sample-001]

Gegeben sei folgende Tabelle, deren Daten frei in einem Tabellenblatt liegen:



	A	B	C	D	E	F	G	H	I
1									
2									
3									
4									
5					Name	Callname	Birthyear		
6					Poppel	Peter	1981		
7					Jicki	Jürgen	1978		
8					Luna	Lana	1984		
9									
10									
11									
12									

Mit der folgenden Deklaration können die gewünschten Daten einfach importiert werden:

```
<plex>

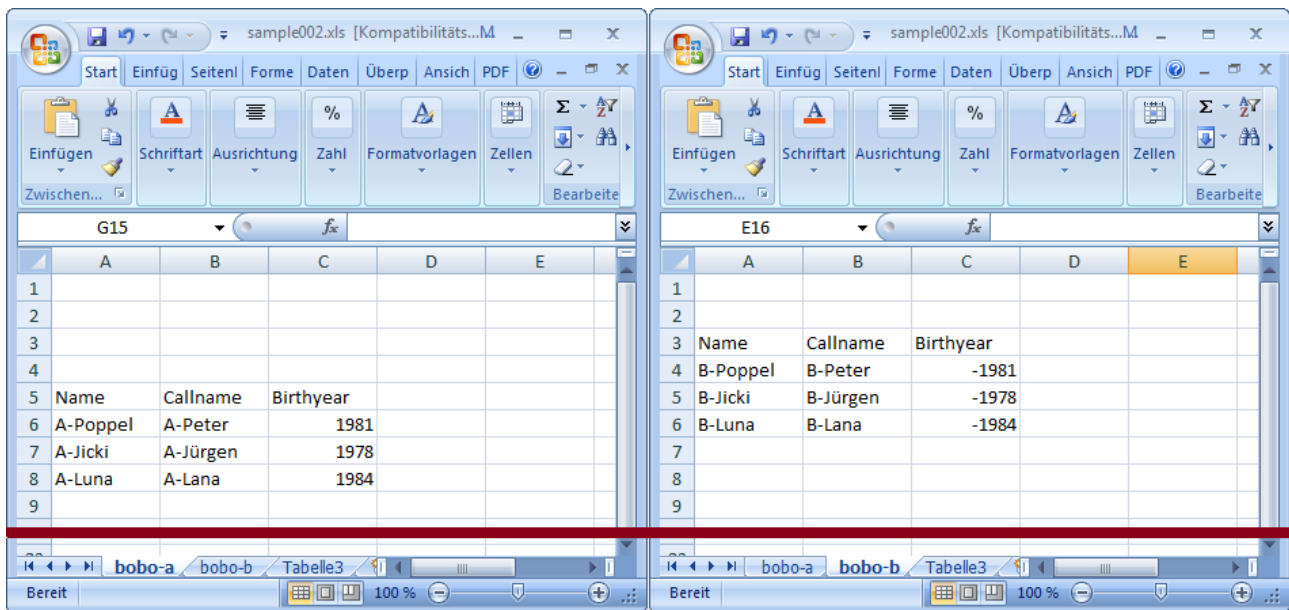
  <general>
  </general>

  <sheet name="sample" firstrow="5">
    <column title="name" column="E"/>
    <column title="callname" column="F"/>
    <column title="birthyear" column="G"/>
  </sheet>

</plex>
```

4.2 Zeilen-Lookup für gleichartige Tabellen [sample-002]

Wenn man gleichartige Tabellen hat will man idealerweise die gleiche Deklaration verwenden und die zu importierenden Datenbereiche dynamisch ermitteln lassen. Folgende Abbildung zeigt zwei solcher Tabellen in denen lediglich die erste Zeile variiert:



Für den Import-Prozess kann man ein Namensmuster angeben und die erste Zeile dynamisch ermitteln lassen:

```
<plex>

  <general>
    <interface
      api="row"
      id="rowresolver"
      classname="com.kasisoft.lgpl.libs.plex.impl.SimpleRowResolver"
    />
  </general>

  <sheet namepattern="bobo-.*">
    <firstrowdetect refid="rowresolver">
      <arg>1</arg>
    </firstrowdetect>
    <column title="name" column="A"/>
    <column title="callname" column="B"/>
    <column title="birthyear" column="C"/>
  </sheet>

</plex>
```