

Name:	Plex
Description:	Library used to simplify the import of Microsoft Excel files
Version:	0.1
Release-Date:	14.09.2010
Downloads:	plex.zip (complete distribution)
License:	LGPL
History:	0.1 - First official release.
Subversion:	https://www.kasisoft.com/svn/plex/trunk
JIRA:	http://www.kasisoft.com/jira/browse/PLEX

1 Motivation

Due to my vocational experience I need to frequently work with technical information specified using [Microsoft Excel](#) files. With a lot of help using the [Apache POI](#) I often managed to support such files. Using them always required to solve the following problems:

- Identify the table structures.
- Transform cell contents into the desired data types.
- Error-Handling in case the cell contents weren't supported.
- Representation of the data using a comfortable data structure.

For this reason I've developed *Plex* which is used to simplify the import of MS Excel files. Basically *Plex* consists of a generic *Importer* which is capable to translate a single workbook into a collection of Java *TableModel* instances. This import process is controlled using a declaration of the useful data.

2 Introduction

As mentioned before the import process only requires two ingredients: The *Plex* declaration and the importer which is part of *Plex*. The *Plex* declaration is a simple xml description of the desired table structures. The following generic snippet demonstrates how to import an excel file:

```
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
}
```

This simple snippet shows the creation of a *PlainExcel* instance using the *Importer* where this instance only contains the interesting data from the excel workbook. Basically a *PlainExcel* instance is a collection for *PlainSheet* instances (which is a subclass for *TableModel*). This easily allows to visualise the imported data.

The main workload is produced while setting up the *Plex* declaration and potentially the realisation of some interfaces necessary by a specified excel workbook.

3 The Plex declaration

A *Plex* declaration is a simpel xml description related to an excel file. A corresponding schema file is part of the distribution. A declaration always looks like followed:

```
<plex>

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

  <!-- At least one sheet must be declared. -->
  <sheet name="namelist" firstrow="0">
    <!-- [2] Declaration of the columns. -->
  </sheet>

</plex>
```

Part [1] contains the declaration for the used APIs. Each declared implementation must provide a unique ID for further reference in order to support the data extraction process. This snippet shows the example class *CleanupTransform* which makes sure that cell-content is either non-empty or *null*.

The second part [2] describes the columns per sheet. These columns will be imported and are used to identify the desired information within the excel workbook. All uncovered information within the excel workbook will not be considered for the import process.

3.1 The sheet declarations

Each *sheet* element describes the data for an excel sheet that has to be imported. The related excel sheet can be identified by one of the attributes *name* or *namepattern*. While *name* simply selects a specified excel sheet, the attribute *namepattern* accepts a regular expression and therefore allows to match multiple excel sheets.

The following declaration just imports one excel sheet with the name "persons" in case it exists:

```
<sheet name="persons">
  <!-- Declaration of the columns. -->
</sheet>
```

On the other hand similar excel sheets may be declared as followed:

```
<sheet namepattern="statistic-[0-9]{1,3}">
  <!-- Declaration of the columns. -->
</sheet>
```

This declaration supports all sheets that do start with the term "statistics-" and ends with a number (f.e. "statistics-76", "statistics-9").

3.2 Selecting the rows

In order to import some data it's necessary to specify the first row of the desired area. At the moment there's no possibility to support a last row so all rows until the last row with content become selected.

The simplest possibility is the use of the attribute *firstrow*:

```
<sheet name="fluffy" firstrow="20">
  <!-- Declaration of the columns. -->
</sheet>
```

Using this declaration all rows starting with row 20 will be imported. Since all numerical ranges do start with 0 the first row corresponds to row numbered 21 in excel. In case you want to import multiple and similar sheets where the first row can vary it's also possible to identify that row dynamically. This requires to specify a *RowResolver* as demonstrated in the following example:

```
<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>
    <!-- Declaration of the columns. -->
  </sheet>

</plex>
```

The *general* block declares the lookup mechanism for the first row. In this case the implementation *SimpleRowResolver* is used, which is part of the *Plex* library. Obviously it's allowed to provide a custom implementation of the interface *RowResolver*. The here mentioned example just delivers the first row which contains content. The element *firstrowdetect* makes use of this implementation while the argument '1' is an offset that will be added to the result. The supported types of arguments are dependent on the concrete implementation of an api.

Let's just assume that the data on excel sheet *namelist* are starting with the excel row 34, the function will deliver the value 33. Adding the offset causes a first row value of 34 (excel row 35).

3.3 Selecting the columns

The most important declarative element is the specification of columns. Each column must have the *title* attribute which corresponds to the title (=name) of each column within a *PlainSheet* (subclass of *TableModel*) instance.

The easiest way is the direct specification of the column within the excel sheet:

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

The numerical column numbers are also 0 based, so the mentioned declaration refers to the column 'C' in excel. Nevertheless it's legal to use the excel column names directly (case sensitivity doesn't matter):

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

This example corresponds to column 26. As shown for the row selection the columns can be selected dynamically, too:

```
<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>

```

This declaration makes use of the api *ColumnResolver* which only delivers the first column with content. The mentioned implementation is part of the *Plex* distribution.

3.4 Handling of cell content

Each cell within an excel workbook has a specific type which is mapped like followed:

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

Apart from the fact that excel types aren't necessarily consistent within a row there might be other reasons to alter the content of a cell. Therefore each column declaration is allowed to provide an unlimited number of transformation steps (order matters):

```

<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>

```

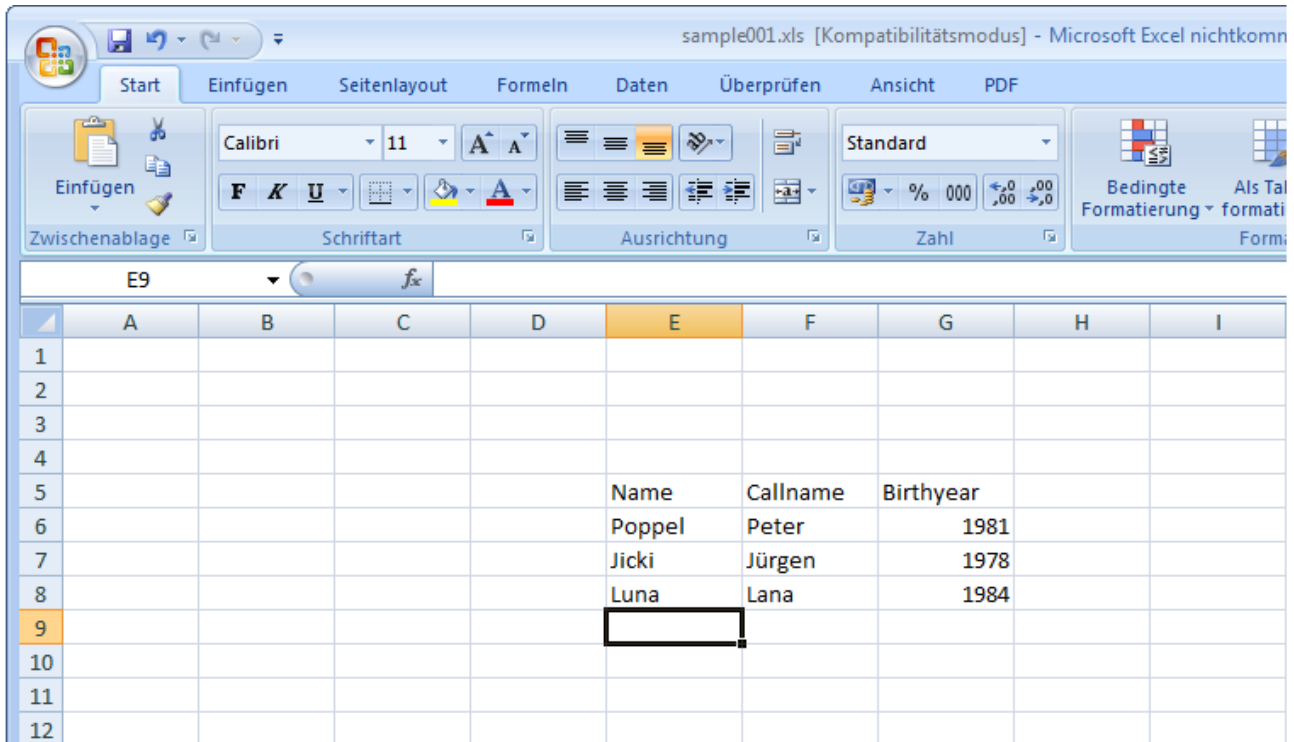
These *transformer* elements are executed in the specified order. There's no restriction regarding the content they will have to handle.

4 Examples

In order to get started with the *Plex* library I will provide some simple usecases here. These usecases are part of the *Plex* distribution so they can be verified very easily.

4.1 Simple fixed table [sample-001]

There's just one table free located within a sheet:



	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									

The following declaration allows to import the corresponding data:

```
<plex>

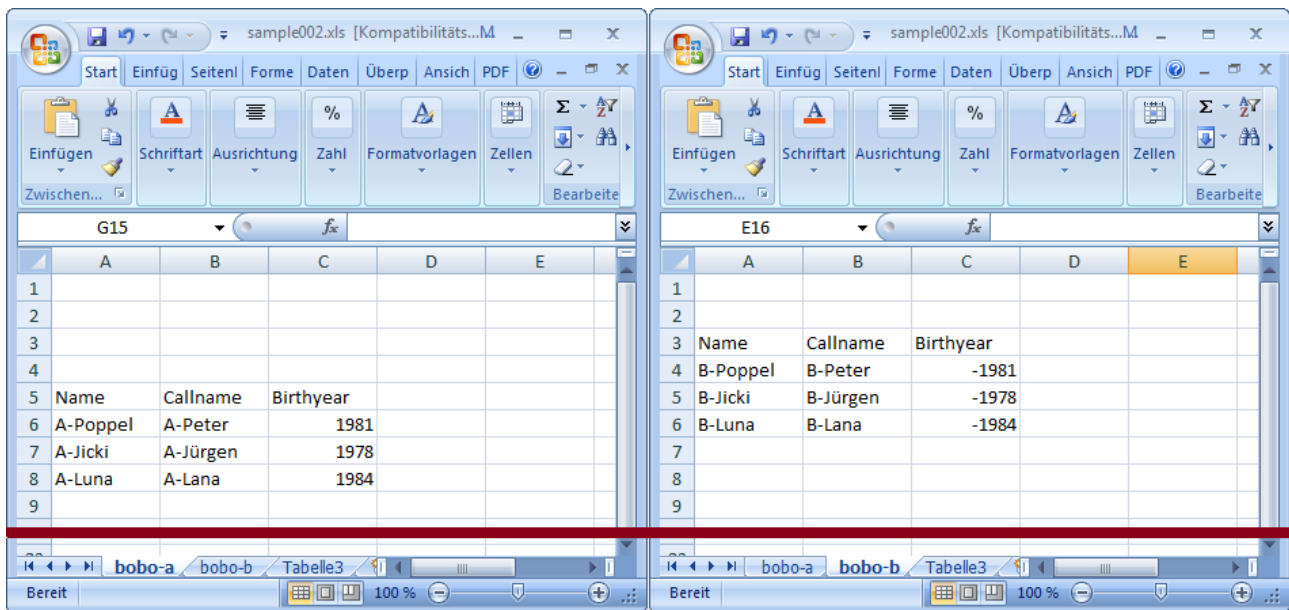
  <general>
  </general>

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

</plex>
```

4.2 Row lookup for similar tables [sample-002]

Ideally similar tables can be imported using the sample declaration. This might require to identify some parameters dynamically. The following picture shows two tables where the first row is varying:



The import can be realized using a *namepattern* and a dynamic lookup of the first row:

```
<plex>

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

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

</plex>
```