****

**Docx4j**

Docx4j is an open source Java library. It is available under the Apache License (v2). Docx4j is used for creating and manipulating Microsoft Open XML (Word-docx, Powerpoint-pptx, and Excel-xlsx) files. It is similar to Microsoft's OpenXML SDK (It is a zipped, XML-based file format developed by Microsoft for .NET for representing spreadsheets, charts, presentations and word processing documents.) but for Java. Docx4j uses JAXB (**Java Architecture for XML Binding**, it allows Java developers to map Java classes to XML representations) to create the in-memory object representation.

Docx4j was created by Plutext Pty Ltd in 2008 - using OpenXML4J for the OPC piece. Plutext still drives the project, but since then docx4j has benefited from contributions from many individuals. The contributors are listed in docx4j's pom.xml. Latest version of Docx4j is 3.2.2 released on 28 Dec 2014.

Docx4j can be used in a .NET environment also via IKVM (It is an implementation of java for .Net), and there are several reasons you might wish to do this:

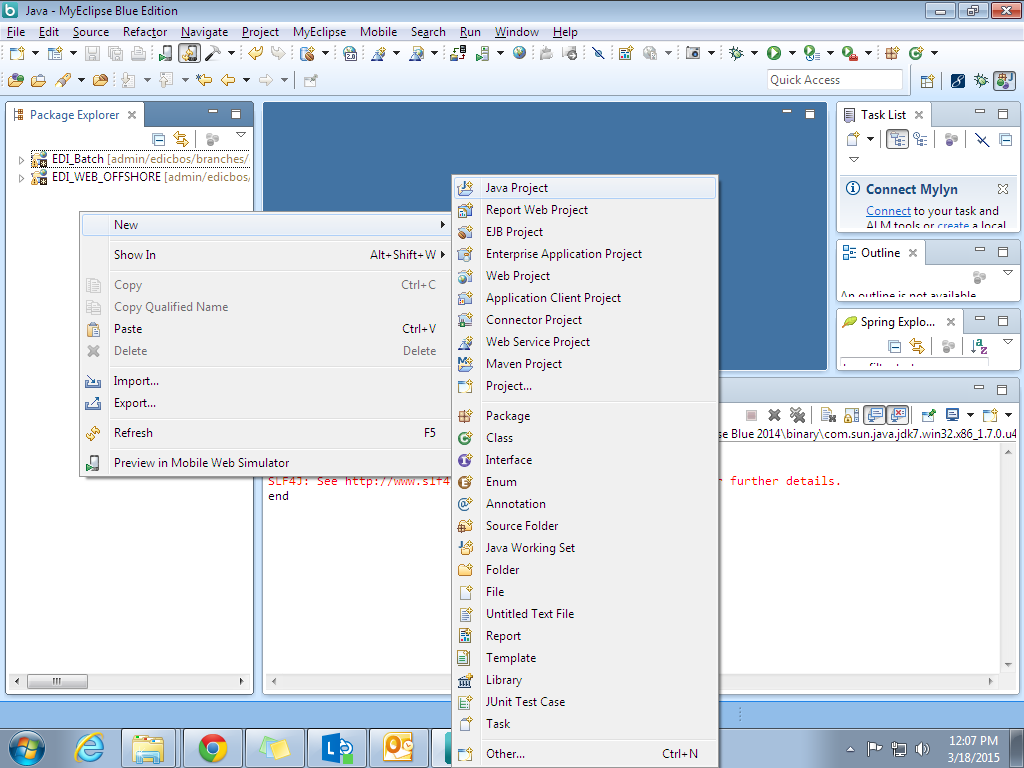
* Where you need docx4j’s capabilities, for example:
  + XHTML import
  + PDF or XHTML export
  + OpenDoPE processing
* Capabilities provided by docx4j commercial extensions, for example:
  + Merging documents or presentations
  + OLE embedding
  + TOC generation/updating
* Where you need to work in both Java and .NET, and want to use a single API in both environments
* Where you need the source code (Microsoft doesn’t provide that)

**What docx4j can do?**

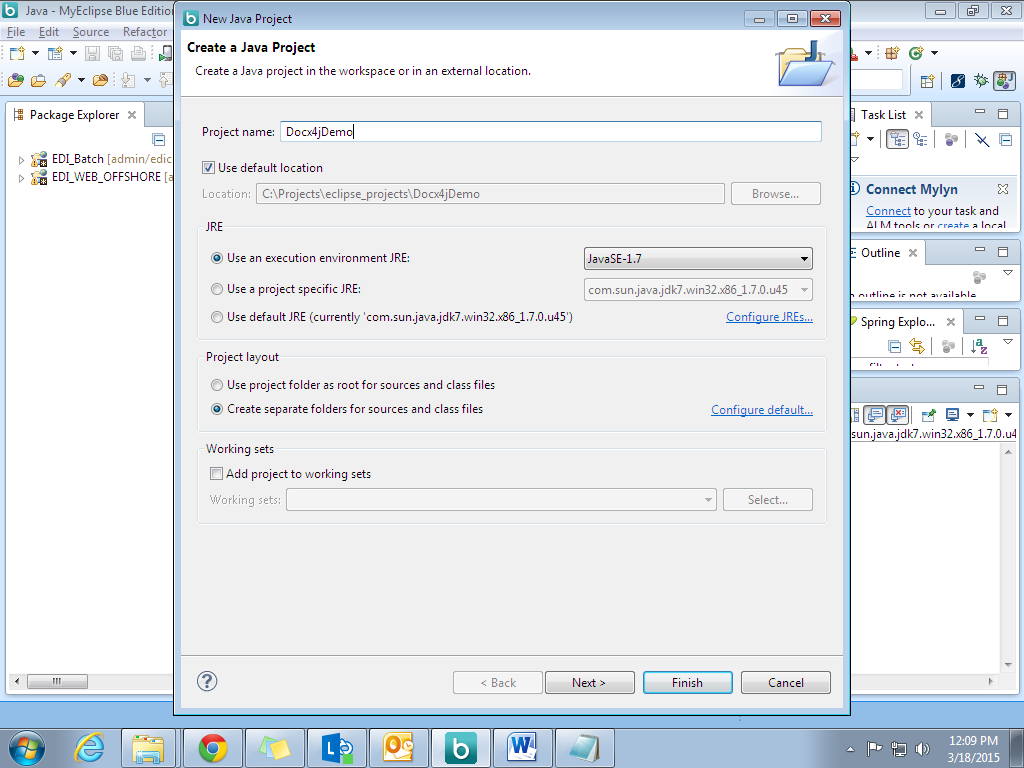
* Open existing docx (from filesystem, SMB/CIFS, WebDAV using VFS), pptx, xlsx
* Create new docx, pptx, xlsx
* Programmatically manipulate the above
* Save to various media zipped, or unzipped
* Import XHTML
* Export as (X)HTML or PDF
* Template substitution; CustomXML binding
* Mail merge
* Produce/consume Word 2007's xmlPackage (pkg) format
* Apply transforms, including common filters
* Diff/compare documents, paragraphs or sdt (content controls)
* Font support (font substitution, and use of any fonts embedded in the document).

**How to configure Docx4j in your project.**

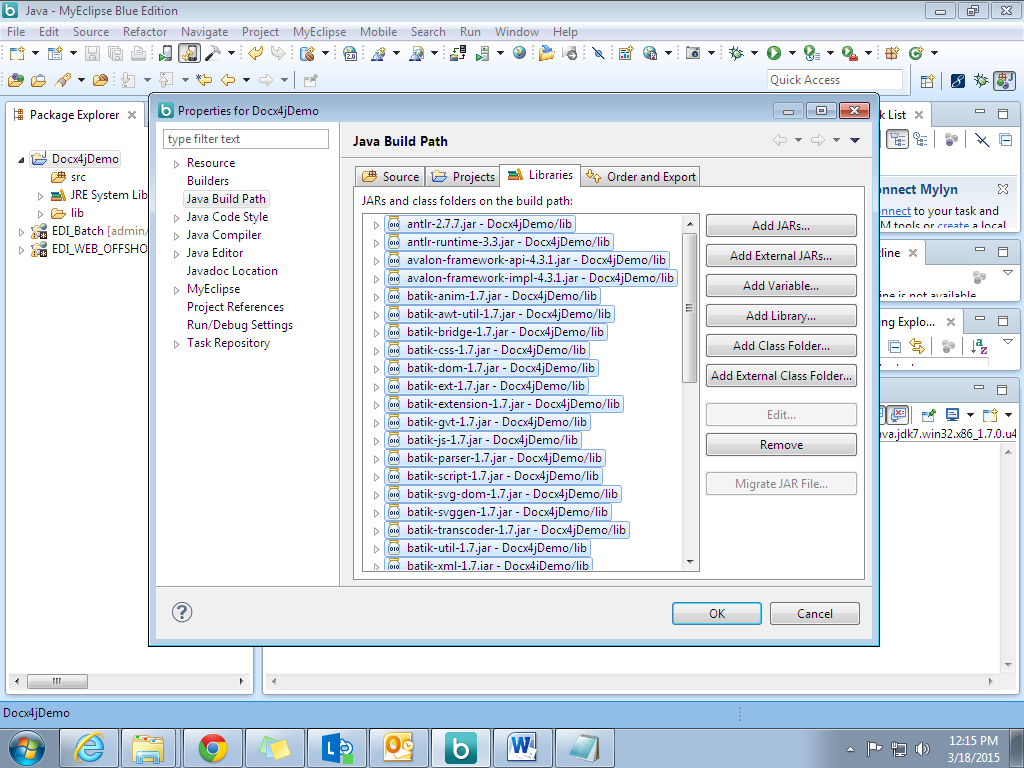
1. Open any java IDE (MyEclipse in this case).
2. Go to project navigator, Right-Click 🡪 New 🡪 Create 🡪 Java Project.



1. Name your project and click Finish.



1. Now Right click your project 🡪New 🡪 Folder, Name it something let say **Lib.**
2. In Lib folder add Docx4j and their dependent jars which can be download from [here](http://www.docx4java.org/downloads.html).
3. Further add these into your project build path.



**How to create a word file.**

Create a java main class in your project and write the following code-

**import** org.docx4j.openpackaging.exceptions.Docx4JException;

**import** org.docx4j.openpackaging.exceptions.InvalidFormatException;

**import** org.docx4j.openpackaging.packages.WordprocessingMLPackage;

**public** **class** DocxTest {

**public** **static** **void** main(String[] args) **throws** Docx4JException {

WordprocessingMLPackage wordMLPackage = WordprocessingMLPackage.*createPackage*();

wordMLPackage.save(**new** java.io.File("HelloWorld.docx"));

}

}

This will create a word file with the name **HelloWorld** in your project folder. **WordprocessingMLPackage** represents a word document in similar way-

|  |  |
| --- | --- |
| ppt,pptx | org.docx4j.openpackaging.packages.**PresentationMLPackage** |
| xls,xlsx | org.docx4j.openpackaging.packages.**SpreadsheetMLPackage** |

We can use above packages for presentation and excel files.

**How to write in a word file.**

The following code snippet will write in a word file and save it.

**import** org.docx4j.openpackaging.exceptions.Docx4JException;

**import** org.docx4j.openpackaging.exceptions.InvalidFormatException;

**import** org.docx4j.openpackaging.packages.WordprocessingMLPackage;

**public** **class** DocxTest {

**public** **static** **void** main(String[] args) **throws** Docx4JException {

WordprocessingMLPackage wordMLPackage = WordprocessingMLPackage.*createPackage*();

wordMLPackage.getMainDocumentPart().addParagraphOfText("Hello Word!");

wordMLPackage.getMainDocumentPart().addParagraphOfText("This is written using Docx4j");

wordMLPackage.save(**new** java.io.File("HelloWorld.docx"));

}

}

**Open an existing docx/pptx/xlsx document**

The following code will open a word file e.g. HelloWorld.docx in our case, append the text and save it.

**import** org.docx4j.openpackaging.exceptions.Docx4JException;

**import** org.docx4j.openpackaging.exceptions.InvalidFormatException;

**import** org.docx4j.openpackaging.packages.WordprocessingMLPackage;

**public** **class** DocxTest {

**public** **static** **void** main(String[] args) **throws** Docx4JException {

WordprocessingMLPackage wordMLPackage = WordprocessingMLPackage.*load*(**new** java.io.File("HelloWorld.docx"));

wordMLPackage.getMainDocumentPart().addStyledParagraphOfText("Heading1", " This text is further added using Docx4j with formatting");

wordMLPackage.save(**new** java.io.File("HelloWorld.docx"));

}

}

**Converting Doc to Pdf-**

With the help of Docx4j we can also convert a word file in Pdf, To do so we use **convert.out.pdf** package, following code will convert the given word file into pdf file.

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.FileOutputStream;

**import** java.io.OutputStream;

**import** org.docx4j.convert.out.pdf.viaXSLFO.~~PdfSettings~~;

**import** org.docx4j.openpackaging.exceptions.Docx4JException;

**import** org.docx4j.openpackaging.packages.WordprocessingMLPackage;

**public** **class** DocxTest {

@SuppressWarnings("deprecation")

**public** **static** **void** main(String[] args) **throws** Docx4JException, FileNotFoundException {

WordprocessingMLPackage wordMLPackage = WordprocessingMLPackage.*load*(**new** java.io.File("HelloWorld.docx"));

~~PdfSettings~~ pdfSettings = **new** PdfSettings();

org.docx4j.convert.out .pdf.~~PdfConversion~~ conversion = **new** org.docx4j.convert.out .pdf.viaXSLFO.Conversion(wordMLPackage);

OutputStream os =**new** FileOutputStream(**new** File("output.pdf"));

conversion.~~output~~(os, **new** PdfSettings() );

}

}

**Other useful packages,classes and methods-**

* **public WordprocessingMLPackage merge(List<WordprocessingMLPackage> wmlPkgs) –** It will merge multiple documents into single one.
* **BinaryPartAbstractImage-** Thisclass contains methods to allow you to add both embedded and linked images
* **PhysicalFont** and **FontMapper-** Used to change text font.

**References**

* <http://www.docx4java.org/>
* <http://htmlpreview.github.io/?https://github.com/plutext/docx4j/blob/master/docs/Docx4j_GettingStarted.html>