docx4j cheat sheet (most applies to 2.8.x)

15 March 2013

Instantiate WordprocessingMLPackage **wordMLPackage**

Load

```
=WordprocessingMLPackage
.load(java.io.File docxFile)
OR
=WordprocessingMLPackage.
load(InputStream is)
OR (v3.0)
final PartStore partLoader = new
ZipPartStore(is);
final Load3 loader = new
Load3(partLoader);
loader.get();
```

OR

Create

```
=WordprocessingMLPackage.
createPackage();
// automatically creates
// MainDocumentPart
// (/word/document.xml)
```

OR

Import from XHTML

```
=WordprocessingMLPackage.
createPackage();
// Convert the XHTML, and add it
// into the empty docx we made
wordMLPackage
.getMainDocumentPart()
.getContent().addAll(
XHTMLImporter.convert(
new File(inputfilepath), null,
wordMLPackage) );
// See further the
// ConvertInXHTML* samples
```

OR

Merge/concatenate

There is a paid extension to do this.
Try it at
http://webapp.docx4java.org/OnlineDemo/forms/upload_MergeDocx.xhtml

Manipulate **wordMLPackage** contents at part level

part-level operations

Get part

```
// Certain parts are easy
MainDocumentPart mdp =
wordMLPackage.getMainDocumentPart();
StyleDefinitionsPart stylesPart =
mdp.getStyleDefinitionsPart();

// or if you have a rel ID:
mdp.getRelationshipsPart().getPart(relId);

// or if you know the PartName
wordMLPackage.getParts().get(partName)
```

Create/add part

```
// Create a new part of the
// type you want, using its
// constructor

// Then add it to the part
// you want to attach it to
// (here, 'parent')
parent.addTargetPart(
  newPart, mode);
// returns the new Relationship
// mode is from
// enum AddPartBehaviour
```

Tip: Different parts are represented by different classes. Some are XML, some are binary. Upload an existing docx at webapp.docx4java.org to see what parts are in it.

part content (JAXB level) operations

Get the content of the part

```
part.getJaxbElement(); // For parts with JAXB content
// Some of these parts have a shortcut:
List<Object> contents = mdp.getContent();
// there are also binary, and a few non JAXB XML parts,
// not covered here
```

find content / insertion point

by traversing

```
Finder finder = new Finder(SomeObject.class); // <-- alter to suit
new TraversalUtil(mdp.getContent(), finder);

public static class Finder extends CallbackImpl {
  protected Class<?> typeToFind;
  protected Finder(Class<?> typeToFind) { this.typeToFind = typeToFind;
  public List<Object> results = new ArrayList<Object>();
  @Override public List<Object> apply(Object o) {
     // Adapt as required
     if (o.getClass().equals(typeToFind)) {
         results.add(o); }
     return null; }}
```

via XPath

```
String xpath = "//w:t[contains(text(),'scaled')]";
List<Object> list = documentPart.getJAXBNodesViaXPath(xpath, false);
// Beware using XPath.
// There are known limitations in JAXB implementations
```

Edit

```
// Now change the values in
// the object as you see fit

// You can generate code
// from a sample docx to
// help with this, at
http://webapp.docx4java.org/
```

OR

Create/Add

```
// You can generate code
// to create objects, at
http://webapp.docx4java.org/

// Then, you typically add
// the object to the
List<Object> contents
```

automation helpers

OR MERGEFIELD processing

OR Variable replace

Content control data binding *(recommended!)*

Hints and tips

Maven

```
<dependency>
    <groupId>org.docx4j</groupId>
    <artifactId>docx4j</artifactId>
    <version>2.8.1</version>
</dependency>
```

See further search.maven.org and docx4j-from-maven-central

docx4j jars

docx4j's dependencies can be had from maven. Also listed in build.xml, and downloadable from http://www.docx4java.org/docx4j/

JAXB implementation

JAXB is included in Java 6 and 7. Otherwise, install the reference implementation or EclipseLink MOXy

docx4j source code

Sample code

Logging

docx4j currently uses log4j. log4j.xml

docx4j.properties

Your docx

Explore it and generate code at webapp.docx4java.org

JAXB concepts

Marshalling, Unmarshalling

Your code to XML

```
XmlUtils.marshaltoString
```

OpenXML help

See ECMA 376 4ed, part 1 or Wouter's Open XML Explained book

Getting help

For help with docx4j, you can post in the relevant forum, xor on StackOverflow.

Production deployment

See the deployment forums for help with specific app servers, and consider purchasing production support from sales@plutext.com

Finish up

Save

```
=WordprocessingMLPackage
.save(java.io.File docxFile)
OR
use SaveToZipFile to save to
a zip file or output stream
OR
use io3.Save with your own
PartStore implementation
```

PDF

```
org.docx4j.convert.out
.pdf.PdfConversion c
= new org.docx4j.convert.out
.pdf.viaXSLFO.Conversion(
   wordMLPackage);
OutputStream os = …
c.output(os,
new PdfSettings() );
```

(X)HTML

```
AbstractHtmlExporter exporter
= new HtmlExporterNG2();
HtmlSettings htmlSettings= new HtmlSettings();
// Set as required; see ConvertOutHtml
OutputStream os = …
javax.xml.transform.stream.StreamResult result
= new StreamResult(os);
exporter.html(wordMLPackage,
  result, htmlSettings);
```

Extract text

```
org.docx4j.wml.Document wmlDoc=
mdp.getJaxbElement();
Writer out
 = new OutputStreamWriter(
     System.out); // or whatever
TextUtils.extractText(
   wmlDoc, out);
out.close();
```