docx4j 8 cheat sheet

Apri 2019

Instantiate WordprocessingMLPackage **wordMLPackage**

Load

```
=Docx4J.load(java.io.File
docxFile)
OR
=WordprocessingMLPackage.
load(File|InputStream)
OR
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

Docx4j Enterprise only
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 = part.getContent();
// there are also binary, and a few non JAXB XML parts,
// not covered here
String xml = part.getXML();
```

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
```

templating

OR MERGEFIELD processing

OR Variable replace

OpenDoPE templates *(recommended!)*

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/
or our Word AddIn
// Then, you typically add
// the object to the
List<Object> contents
```

Hints and tips

Maven

Select the artifact corresponding to your preferred JAXB implementation:

```
<dependency>
    <groupId>org.docx4j</groupId>
    <artifactId>docx4j-JAXB-Internal</artifactId>
    <version>8.0.0</version>
</dependency>
```

XOR:

```
<dependency>
    <groupId>org.docx4j</groupId>
    <artifactId>docx4j-JAXB-MOXy
                          </artifactId>
    <version>8.0.0</version>
</dependency>
```

XOR:

```
<dependency>
    <groupId>org.docx4j</groupId>
    <artifactId>docx4j-JAXB-ReferenceImpl
                          </artifactId>
    <version>8.0.0</version>
</dependency>
```

docx4j source code

Sample code or GitHub

Logging

docx4j uses slf4j. Sample log4j2.xml and logback configs.

docx4j.properties

Your docx

Explore it and generate code at webapp.docx4java.org or our Word AddIn

JAXB concepts

Marshalling, Unmarshalling

Your code to XML

`XmlUtils.marshaltoString` or `part.getXML()`

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

```
Docx4J.save
OR
=WordprocessingMLPackage.save
OR
use io3.Save with your own
PartStore implementation
```

PDF

Either:

```
<artifactId>docx4j-export-fo</artifactId>
```

or for higher quality & performance:
https://converter-eval.plutext.com

(X)HTML

```
Docx4J.toHTML
```

OR see

https://github.com/plutext/docx4j/blob/master/docx4j-samples-docx4j/src/main/java/org/docx4j/samples/ConvertOutHtml.java

Extract text

```
TextUtils.getText(
   Object o);
```