OpenDoPE XHTML support

Initial draft: 13 Nov 2011;  
this version: 18 Nov (improvements to ol, ul and img support)

With forthcoming docx4j 2.8.0 (probably January 2012), docx4j's BindingHandler has a method convertXHTML which can take a snippet of XHTML, convert it to Office Open XML, and insert it into a content control.

# Example

For example, given the following CustomXML part:

<pkg:part pkg:name="/customXml/item3.xml" pkg:contentType="application/xml" pkg:padding="32">

<pkg:xmlData>

<yourxml>

<case1> &lt;html&gt;&lt;body&gt; &lt;p&gt;hello &lt;/p&gt; &lt;/body&gt;&lt;/html&gt; </case1>

<case2> &lt;body&gt; &lt;p&gt;hello &lt;/p&gt; &lt;/body&gt; </case2>

**<case3> &lt;p&gt;hello &lt;/p&gt; </case3>**

<case4>

&lt;body&gt;

&lt;p&gt;hello 1&lt;/p&gt;

&lt;p&gt;hello 2&lt;/p&gt;

&lt;/body&gt;

</case4>

<case4B>

&lt;div&gt;

&lt;p&gt;hello 1&lt;/p&gt;

&lt;p&gt;hello 2&lt;/p&gt;

&lt;/div&gt;

</case4B>

<case5> &lt;span&gt;Please &lt;a href="mailto:foo@bar"&gt;click to email&lt;/a&gt; &lt;b&gt;now!&lt;/b&gt; &lt;/span&gt;</case5>

</yourxml>

</pkg:xmlData>

</pkg:part>

and a content control:

<w:sdt>

<w:sdtPr>

**<w:tag w:val="od:xpath=case3&amp;od:ContentType=application/xhtml+xml"/>**

<w:dataBinding w:xpath="/yourxml/case3" w:storeItemID="{20B8AAA6-..-D96D3E3AA22F}"/>

<w:text/>

</w:sdtPr>

<w:sdtContent>

<w:p>

<w:r>

<w:t xml:space="preserve"> [will be replaced] </w:t>

</w:r>

</w:p>

</w:sdtContent>

</w:sdt>

the contents of the element

**<case3> &lt;p&gt;hello &lt;/p&gt; </case3>**

will be converted to something like:

<w:p>

<w:r>

<w:t>hello</w:t>

</w:r>

</w:p>

and injected into <w:sdtContent>.

# od:ContentType=application/xhtml+xml

The behaviour described here is triggered by the presence of the following od:ContentType:

<w:sdt>

<w:sdtPr>

**<w:tag w:val="od:xpath=case3&amp;od:ContentType=application/xhtml+xml"/>**

The XHTML must be well-formed XML, escaped as per the example. There are several Java libraries you can use to convert HTML to suitable XHTML.

These include:

* <http://htmlcleaner.sourceforge.net/>
* <http://ccil.org/~cowan/XML/tagsoup/>
* <http://jtidy.sourceforge.net/>

It must have a single root element. For example, if you want to include 2 paragraphs, wrap them in a div:

&lt;div&gt;

&lt;p&gt;hello 1&lt;/p&gt;

&lt;p&gt;hello 2&lt;/p&gt;

&lt;/div&gt;

# Content Suitability

The allowed content of a content controls depends on whether it is at paragraph-level, run-level , or within a table.

If your content control is at paragraph-level (ie contains paragraphs), it is your responsibility to ensure your XHTML will convert to OpenXML paragraphs and tables.

Similar considerations apply to the other levels.

# Stylesheets

The conversion is performed using a modifed version of Flying Saucer xhtmlrenderer.

This interprets CSS, which docx4j then converts to suitable w:pPr and w:rPr elements.

For the base stylesheet to be available, you need docx4j's src/main/resources directory on your path (in your docx4j jar is fine).

# Current Status

This is work in progress, which will not be complete until the release of docx4j 2.8.0.

To use this code, you need to build docx4j from trunk, and add the modified xhtmlrenderer to your path (this is not in Maven Central yet).

At present, there is no support for conversion of:

* tables
* fonts

Support for lists (ol, ul) is currently rudimentary: only list-style-type 'decimal' and 'disc' are supported, and CSS is ignored.

There is basic support for importing images, but no scaling etc yet.

Paragraph content should convert reasonably well, including hyperlinks, and <b>, <u>, <i>, <br>.

font-size: medium is converted to 11points; the other sizes aren't recognised yet.

No attempt is made yet to create reusable docx styles out CSS definitions - all formatting is applied ad hoc in the converted content.

h1, h2 etc are styled as per the CSS in XhtmlNamespaceHandler.css. Heading 1 etc style definitions in the target document are ignored.

# Specification

This is a working document.

The content will be migrated to the OpenDoPE specification in due course.