OpenDoPE XHTML support

22 May 2012

docx4j **2.8.0**'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.

It relies on docx4j’s underlying XHTMLImporter.

# 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

Supported:

* paragraphs
* tables
* images (no scaling etc though)

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

No attempt is made yet to create reusable docx styles out CSS definitions - all formatting is applied ad hoc in the converted content. However, existing docx styles can be used (if @class value matches an existing style).

h1, h2 etc are styled as per the CSS. Since 3.2.0, there is the option to convert h1, h2 etc to corresponding Word style:

# If true (default is false), h1, h2 etc will be converted to corresponding Word style,

# UNLESS:

# 1. we're set to use CLASS\_TO\_STYLE\_ONLY or CLASS\_PLUS\_OTHER, and

# 2. @class maps to some other paragraph style

docx4j-ImportXHTML.Element.Heading.MapToStyle=true

# Specification

This is a working document.

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