OpenDoPE Word Add-In

Introduction

The OpenDoPE Word Add-In is an Add-In for Microsoft Word 2007 which is designed to help you to:

- **bind** content controls to CustomXML parts - In this respect it is similar to the Content Control Toolkit, except that it runs from within Word
- make the contents of a content control into **conditional text**
- make the contents of a content control **repeat** a certain number of times
- insert another docx at run time in place of a content control

It implements the architecture described in [Convention for repeating/conditional content controls](#)

Once you have used the Add-In to instrument your docx, it is ready to be pre-processed to create instance documents.

You'll need software to do this.   Pre-processing implementations

Enviroment	Architecture	Status
Java	server-side	implemented in docx4j v2.6.0
Word macros	client-side	this Add-In includes a macro which it can inject into your docx
.NET	N/A	not done (could be ported from the Java quite easily)

Add-In Status

The OpenDoPE Word Add-In is currently available as a pre-alpha quality download.

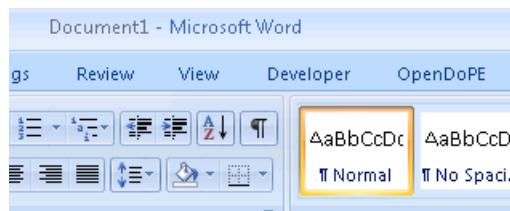It can be used to create documents which:

- the macro or a web site like fabdocx.com can process interactively
- the Java pre-processor can process in non-interactive mode

It is certainly helpful for getting a feel for how the convention works (instrument a docx using the Add-In, save it, and then unzip the resulting docx and look at it).
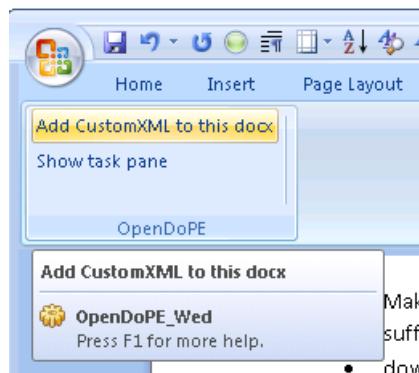
Download and Setup

- Make sure you have the full .NET 4 framework installed (the .NET 4 CP client profile is not sufficient), then
- download the OpenDoPE Word Add-In from http://dev.plutext.org/opendope/setup.exe

After a successful install, upon opening Word. you will see a new OpenDoPE menu:
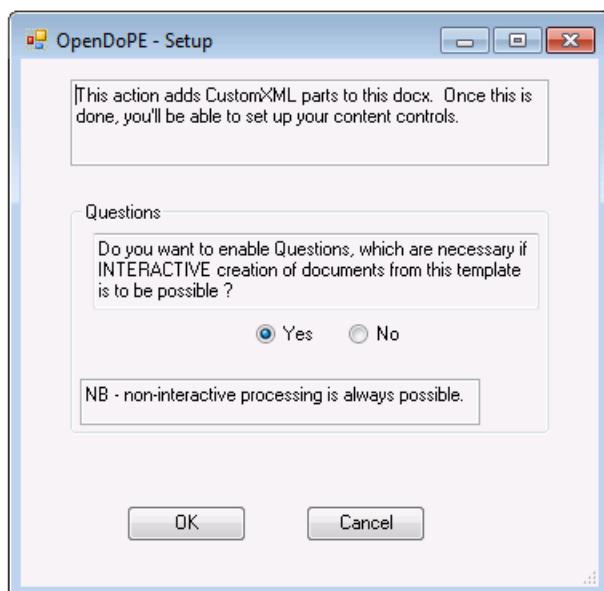
Click on it, and select "add CustomXML to this docx"

You can do that with a new empty docx, or with an existing docx (with or without content controls).

You will see:

For simplicity, answer "no". (For the "yes" case, see *Interactive Processing* on page 9 below)

This will add the necessary custom xml parts, and opens the OpenDoPE task pane.
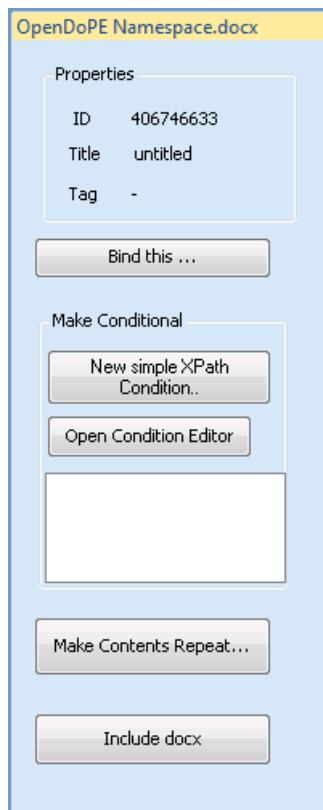
Nothing more will happen until you click into a content control in your document (either an existing one, or one you have just created).

To create a content control, use one of the following buttons:

(Alternatively, you can use the Word Developer ribbon. If it isn't visible, you can enable it in Word Options. Note: if you want to do a traditional bind, Word requires that you must use a plain text control, not a rich text control. The Add-In will offer to convert, if necessary.)

When you click inside a content control, your task pane will show something like:

The four options are

1. a traditional bind (exactly what you'd create using the Content Control Toolkit)
2. making the content control conditional

   three ways are provided to make the conditional:

   a.   adding a simple XPath

   b.   creating a more complex condition using the condition editor

   c.   selecting an existing condition from the list (empty here; not yet implemented)
3. making the content control repeat
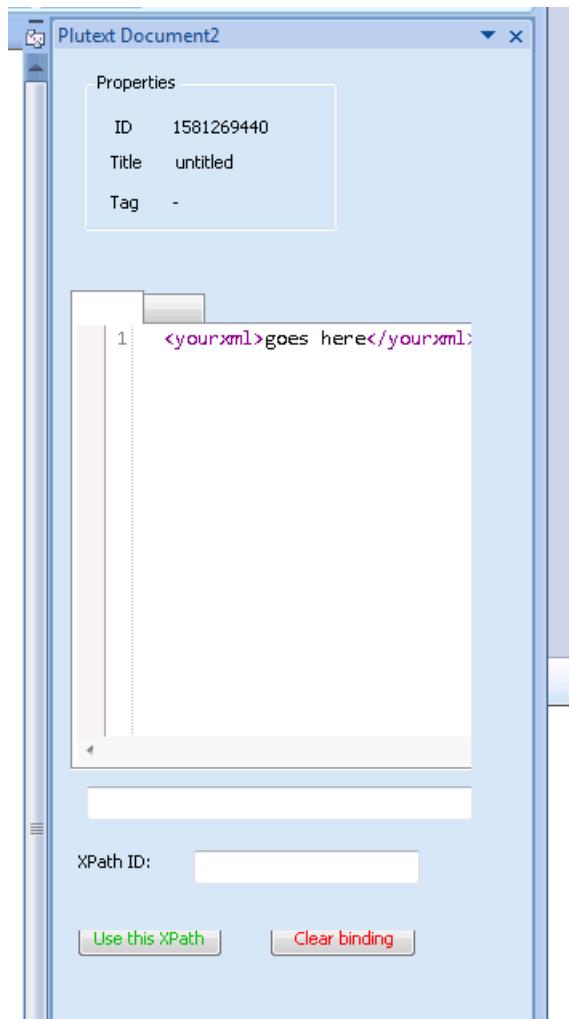4. replace the content control with another docx

We'll look quickly at each

Traditional Bind

Note: if you want to do a traditional bind, you should use a plain text control, not a rich text control. This is a restriction imposed by the OpenXML spec, and Word.

When you press the "Bind this.." button, Word may pause for a few seconds as our editors load. This will only happen once (per docx open).

You'll see:

The XML you can see is your custom xml part. The XML should match your business requirements. You might have an existing XML document you want to paste here, or you can create it as you go along.

Notice the tab. You can work in a tree view if you prefer.

If you click on an opening tag, an XPath will appear in the box below. Once you see the XPath you want (you can edit it as necessary), make up a name for it (and put that in the ID box), then click "Use this XPath".

That's all you need to do to create a traditional bind.   You should see Word put the value of the XPath into your content control.

Namespace Guidelines

Guidelines:

1. Either don't use namespaces, or if you do use them, don't use a default namespace.
2. All namespaces you use should be declared on the root element

Example - no namespace

```
<invoices>
   < invoice>an invoice<invoice>
</invoices>
```

Example - explicit namespace

```
<ns0:invoices xmlns:ns0="http://something">
   <ns0:invoice>an invoice</ns0:invoice>
</ns0:invoices>
```
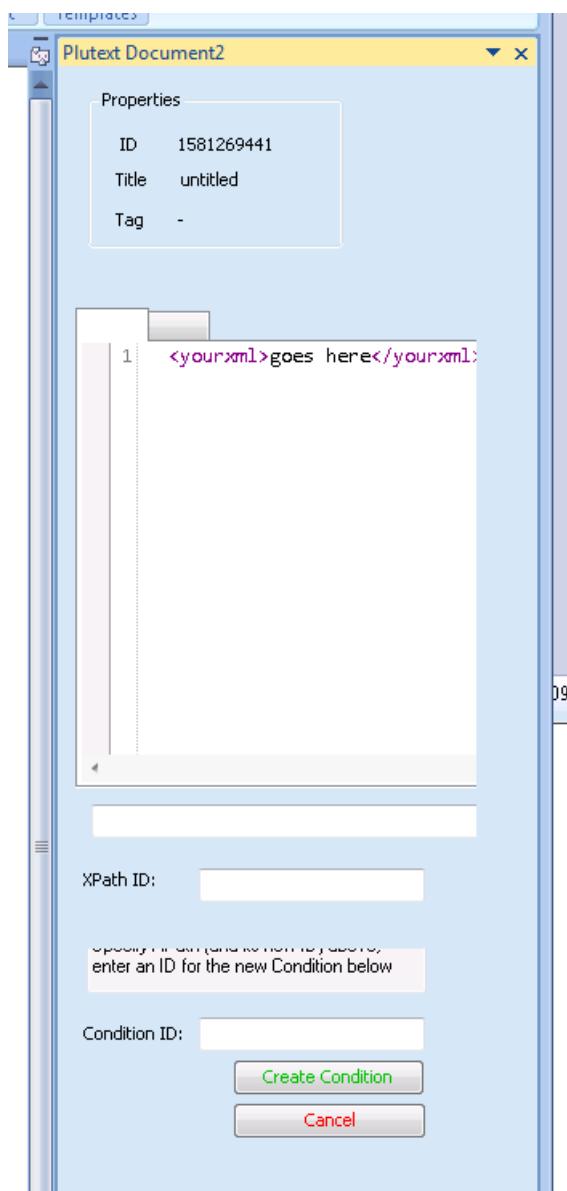
Example: default namespace

```
<invoices xmlns:ns0="http://something">
   <invoice>an invoice</invoice>
</invoices>
```

Adding a Condition - Method 1

There are currently 2 ways to add a condition.

The first is using the simple condition editor:

Using this is very similar to doing a traditional bind. The only difference is that you also need to make up an ID for your new condition (then hit the "Create Condition" button).

Adding a Condition - Method 2

More complex conditions can be created using the condition editor:

This is only really necessary if you want to make conditions based on boolean expressions. It is not discussed further here.

Repeats

To make a content control repeat, press the "Make Contents Repeat.." button.

In this example:

```
 1  <invoice>
 2      <customer>
 3          <name>Joe Bloggs</name>
 4      </customer>
 5      <items>
 6          <item>
 7              <name>apples</name>
 8              <price>$20</price>
 9          </item>
10          <item>
11              <name>bananas</name>
12              <price>$30</price>
13          </item>
14          <item>
15              <name>cherries</name>
16              <price>$40</price>
17          </item>
```

/invoice[1]/items/item

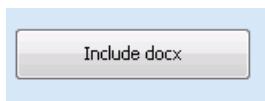XPath ID: repeatItem

Use this XPath   Clear binding

the content control will be cloned for each item.

Inside this repeating content contol, you then put the actual content you want to repeat. Typically this will include more content controls - in this example, one bound to /invoice/items/item/name and/or price.

It is the job of the pre-processor to clone the repeat control, and then change the xpaths within it to item[1]/name in the first, item[2]/name in the second etc.

Inserting another docx

To do this, insert a content control, and click:

Include docx

The task pane will let you type a URL:

URL:

OK   Remove

In the docx4j implementation, you supply a DocxFetcher object, which knows how to handle the URL (ie what protocol to use, any password required etc). The docx4j implementation will replace this with an altChunk element, unless you have the MergeDocx paid extension (in which case the docx will be merged in without the need for Word).

This docx can be a plain docx, or it can itself contain OpenDoPE content (you'd need the MergeDocx extension in order for docx4j to process it).

Finishing Up

To see the results of your work, save your docx in the usual Word way, and then inspect its XML. There are lots of ways to look at the XML:
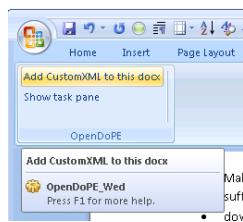
- from Word, save as ".xml", or
- unzip the docx, or
- drag the docx onto Visual Studio 2010

Have a look at each of the custom XML parts; also look at your content controls (<sdt>) in document.xml
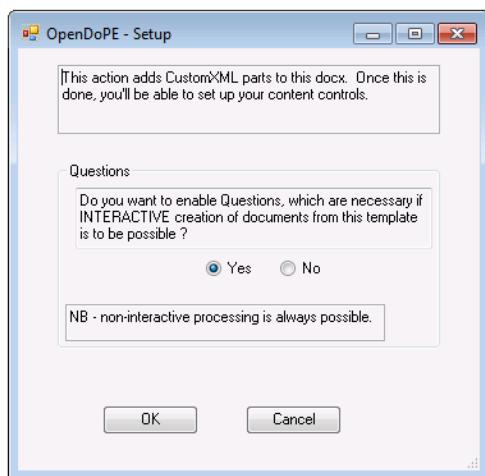
You can also process the resulting docx using one of the tools identified in the Introduction above.

Interactive Processing

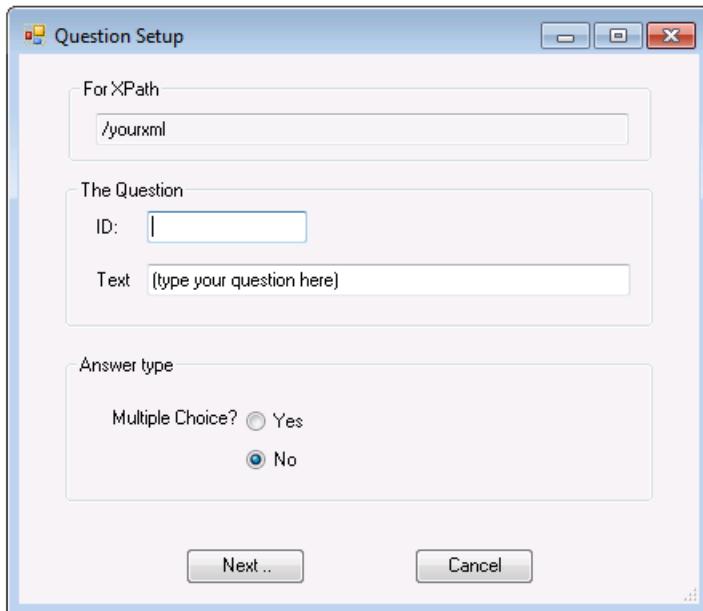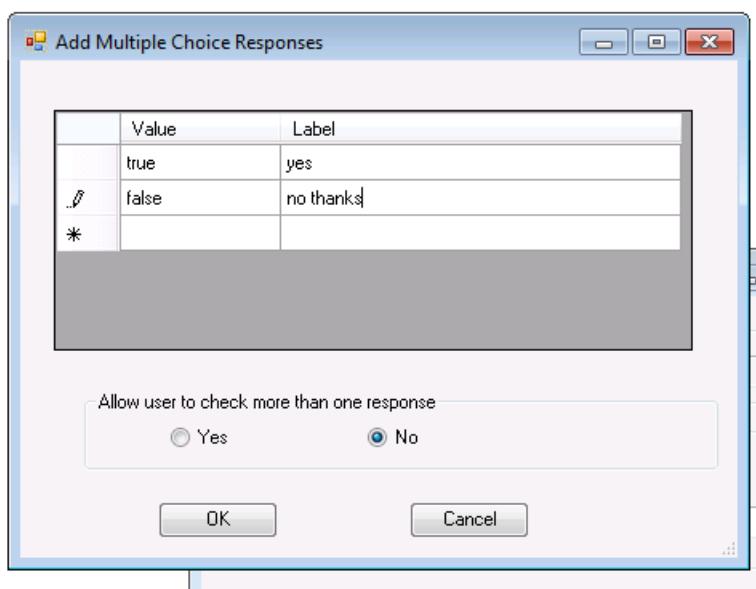When you first clicked "add CustomXML to this docx"

You saw:

To enable interactive processing, answer "yes" to this question.

If you answer "yes" to this question, a Questions part will be added to the docx, and then each time you bind a content control (either a standard bind, or as a repeat or a condition) you will be prompted to enter a corresponding question:

Answers can be multiple choice. For a condition, you will want multiple choice, with values true and false:

(If the user chooses "true", the conditional content control will be retained)

Macro Injection

A docx with questions embedded in it can be processed interactively, in various ways, for example:

- via a macro embedded in the docx
- via a web-based application

The Add-In can inject a suitable macro into the docx. This macro is set to execute when the document is opened.

To inject the macro, click the "Inject Macro" button on the toolbar:

This will prompt you to save your document as a docm (a macro-enabled docx). Note that this operation cannot be performed if the document is already a docm.

Once complete, you can distribute the docm to end users. The macro will run on opening (provided the user allows), enabling the end user to create a customised document. For further information, please see the macro_walkthrough document.

Questions/Comments

Questions/discussion belong at http://www.opendope.org/forum.html, or, for the Java implementation, the docx4j data binding sub forum.

Alternatively, you can email jason@plutext.org