Talk 12: Publishing TEI

James Cummings

28 January 2014

Publishing

What do you want to do with your TEI document(s)?

1. Make a web page facsimile edition
2. Make an ebook reading edition
3. Do some analysis of the structured content
4. Perform linguistic analysis of the words
5. Provide a searchable interface

Basic approaches

Print	Produce a printed readable version of your text
Display	Render the TEI XML in a web browser
Extract	Pull out particular parts
Transform	Turn the TEI XML into (eg) HTML5 for an eBook
Index	Make entry points into the text

Tools and skills you may need

for print understanding basics of page layout

for web display an understanding of HTML and CSS, and some design ideas

for extraction an ability to express your query (eg in XPath)

for transformation a knowledge of scripting languages like XSLT

for searching and indexing a XML-aware database system

Systems: examples

oXygen the editor has a lot of built-in transformations to make different outputs from a TEI file, and ways of searching TEI files

TEI Boilerplate displays TEI texts in a web browser directly

Omeka a web CMS which knows about TEI XML, good for discrete objects

OxGarage performs transformations to and from TEI (and pipelines conversions to other formats)

An example application: OxGarage

OxGarage is a web interface to the TEI-C XSL stylesheets and its profiles: http:/www.tei-c.org/oxgarage

OxGarage lets you:

- generate schemas using the same tools as Roma
- convert documentation to HTML, ePub, and DOCX
- convert between TEI XML and Word DOCX
- perform all the ODD tasks using web services
- chain sets of transformations together

Key features of OxGarage

- Built on EU-funded ENRICH project's EGE for converting manuscript descriptions (University of Poznan)
- Chained XSLT conversions
- Uses TEI as pivot format
- Read/write OpenOffice and Open XML
- Provides route from Word to ePub
- Supports "profiles" for variations

Matrix of OxGarage conversions

Inputs:	Compiled TEI ODD Document	DocBook Document	Microsoft Word (.doc)	Microsoft Word (.docx)	ODD Document	Open Office Text (.odt)	OpenOffice 1.0 Text (.sxw)	OpenOffice Text (.odt)	Plain Text (.txt)	Rich Text Format (.rtf)	TCP XML Document	TEI P4 XML Document	TEI P5 XML Document	TEI Tite XML Document	WordPerfect (.wpd)	xHTML
ODD documentation as TEI Lite					✓											
Comma-Separated Values (.csv)		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DTD created from ODD					✓											
ePub		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ISO Schematron constraints	✓				✓											
LaTeX		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Excel (.xls)		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Plain text		✓			✓						✓	✓	✓	✓		
Microsoft Word (.doc)		✓		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Word (.docx)		✓	✓		✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
National Library of Medicine (NLM) DTD 3.0		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ODD documentation as HTML					✓											
Open Office (.ods)		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OpenOffice 1.0 (.sxc)		✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OpenOffice 1.0 Text (.sxw)		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓

OxGarage web service example (1)

Process ODD to compiled ODD, then to TEI Lite, then to DOCX

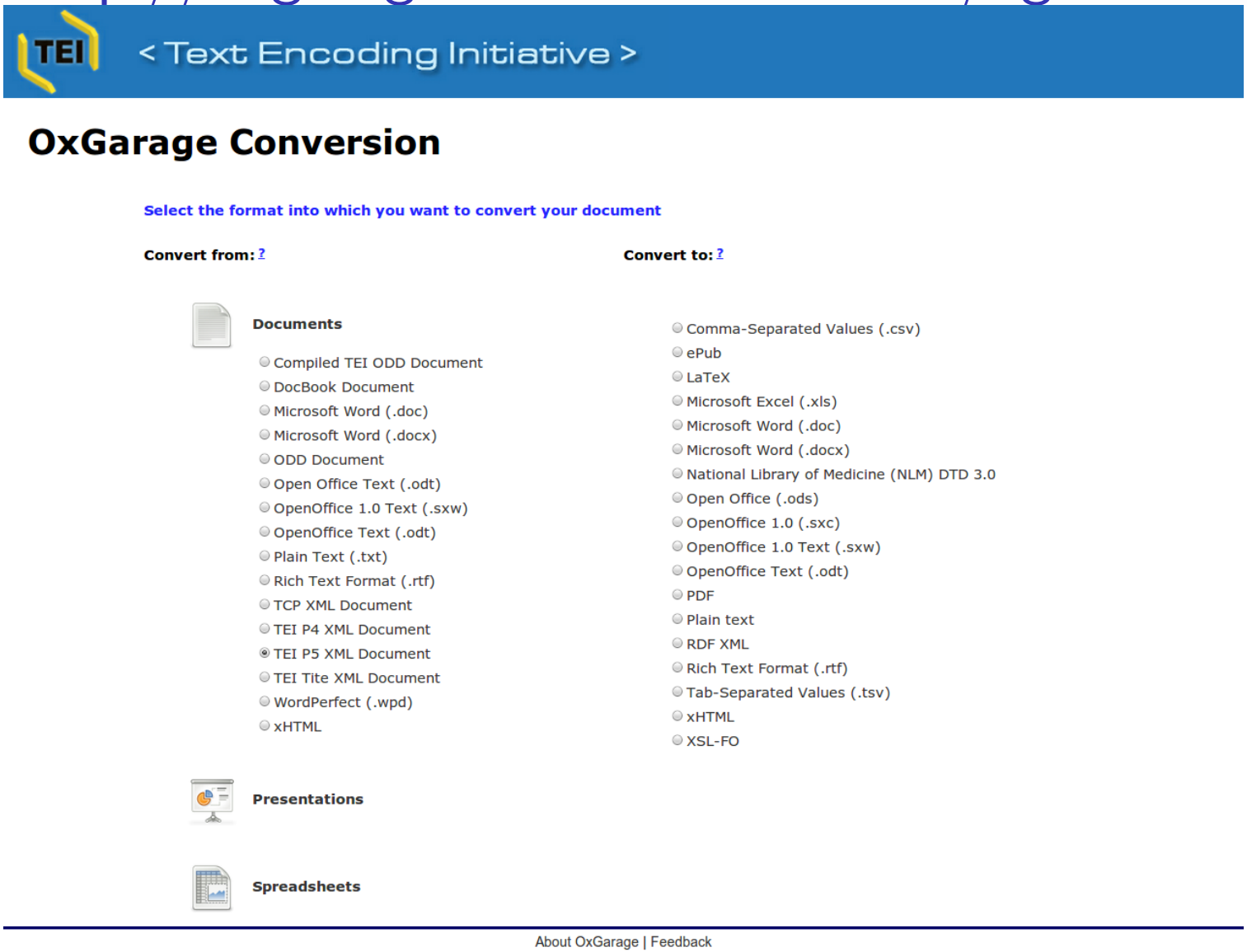
```
curl -s  -F upload=@test.odd  -o test.docx
http://oxgarage.oucs.ox.ac.uk:8080/
ege-webservice/Conversions/
ODD%3Atext%3Axml/
ODDC%3Atext%3Axml/
TEI%3Atext%3Axml/
docx%3Aapplication%3Avnd.openxmlformats-officedocument.wordprocessing
```

OxGarage web service example (2)

ODD to HTML, in French

```
curl -s  -F upload=@test.odd  -o test.html
http://oxgarage.oucs.ox.ac.uk:8080/ege-webservice/Conversions/
ODD%3Atext%3Axml/
ODDC%3Atext%3Axml/
oddhtml%3Aapplication%3Axhtml%2Bxml/
?properties=<conversions><conversion%20index='1'>
<property%20id='oxgarage.lang'>fr</property></conversion></conversion
```

http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient/



http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient/

OxGarage Conversion

Choose the file, upload images and press convert

Select file to convert: ?

Choose File No file chosen

+ Show advanced options ?

Upload images: ?
You can upload image files and .zip files containing images

Choose File No file chosen - Remove

+ Add more images

Convert Reset

Convert from: ?

Documents

- Compiled TEI ODD Document
- DocBook Document
- Microsoft Word (.doc)
- Microsoft Word (.docx)
- ODD Document
- Open Office Text (.odt)
- OpenOffice 1.0 Text (.sxw)
- OpenOffice Text (.odt)
- Plain Text (.txt)
- Rich Text Format (.rtf)
- TCP XML Document
- TEI P4 XML Document
- TEI P5 XML Document
- TEI Tite XML Document
- WordPerfect (.wpd)
- xHTML

Convert to: ?

- Comma-Separated Values (.csv)
- ePub
- LaTeX
- Microsoft Excel (.xls)
- Microsoft Word (.doc)
- Microsoft Word (.docx)
- National Library of Medicine (NLM) DTD 3.0
- Open Office (.ods)
- OpenOffice 1.0 (.sxc)
- OpenOffice 1.0 Text (.sxw)
- OpenOffice Text (.odt)
- PDF
- Plain text
- RDF XML
- Rich Text Format (.rtf)
- Tab-Separated Values (.tsv)
- xHTML

http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient/

OxGarage Conversion

Choose the file, upload images and press convert

Select file to convert: ?

Choose File No file chosen

- Hide advanced options ?

Upload images: ?
You can upload image files and .zip files containing images

Choose File No file chosen - Remove

+ Add more images

Convert text only, don't include any images from the document
Download images located on the internet
Copy and rename all provided images

Conversion: TEI P5 XML Document -> Microsoft Word (.docx)

Language en

Select conversion profile: default

URL for conversion with current properties:

http://oxgarage.oucs.ox.ac.uk:8080/ege-webservice/Conversions/TEI%3Atext%3Axml/docx%3Aapplication%3Avnd.openxmlformats-officedocument.wordprocessingml.document/conversion?properties=<conversions><conversion index="0"><property id="oxgarage.getImages">true</property><property id="oxgarage.getOnlineImages">true</property><property id="oxgarage.lang">en</property><property id="oxgarage.textOnly">false</property><property id="pl.psnc.dl.ege.tei.profileNames">default</property></conversion></conversions>

URL for conversion with default properties:

http://oxgarage.oucs.ox.ac.uk:8080/ege-webservice/Conversions/TEI%3Atext%3Axml/docx%3Aapplication%3Avnd.openxmlformats-officedocument.wordprocessingml.document/

Convert Reset

Convert from: ? **Convert to:** ?

Rolling your own: XSL

- XPath: a language for expressing paths through XML trees
- XSLT: a programming language for transforming XML
- XSL FO: an XML vocabulary for describing formatted pages

XSLT

The XSLT language is a scripting language:

- expressed in XML
- uses namespaces to distinguish output from instructions
- purely functional
- reads and writes XML trees

It was designed to generate XSL FO, but now widely used to generate HTML.

What is a transformation?

Take this:

```
<persName>
  <forename>Milo</forename>
  <surname>Casagrande</surname>
</persName>
<persName>
  <forename>Corey</forename>
  <surname>Burger</surname>
</persName>
<persName>
  <forename>Naaman</forename>
  <surname>Campbell</surname>
</persName>
```

and make this:

```
<li>Burger</li>
<li>Campbell</li>
<li>Casagrande</li>
```

ie including *subsetting*, *re-ordering* and *changing the markup*

An example

Take this

```
<div type="recipe" n="34">
 <head>Student Pasta</head>
 <list>
  <item>Pasta</item>
  <item>Grated cheese</item>
 </list>
 <p>Cook the pasta and mix with the cheese</p>
</div>
```

and make this

```
<html>
 <h1>34: Student Pasta</h1>
 <p>Ingredients: Pasta Grated cheese</p>
 <p>Cook the pasta and mix with the cheese</p>
</html>
```

How do you express that in XSL?

```
<xsl:stylesheet
  xpath-default-namespace="http://www.tei-
c.org/ns/1.0" version="2.0">
 <xsl:template match="div">
  <html>
   <h1>
    <xsl:value-of select="@n"/>:
   <xsl:value-of select="head"/>
   </h1>
   <p>Ingredients:
   <xsl:apply-templates select="list/item"/>
   </p>
   <p>
    <xsl:value-of select="p"/>
   </p>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Note: the namespace declaration linking `xsl:` to `http://www.w3.org/1999/XSL/Transform` is not shown in these examples.

Structure of an XSL file

```
<xsl:stylesheet
  xpath-default-namespace="http://www.tei-
c.org/ns/1.0" version="2.0">
 <xsl:template match="div">
<!-- .... do something with div elements....-->
 </xsl:template>
 <xsl:template match="p">
<!-- .... do something with p elements....-->
 </xsl:template>
</xsl:stylesheet>
```

The `div` and `p` are patterns which specify which bit of the document is matched by the template.

Any element not starting with xsl: in a template body is put into the output.

The Golden Rules of XSLT

1. If there is no template matching an element, we go on and process the elements inside it
2. If there are no elements to process by Rule 1, any text inside the element is output
3. Children elements are not processed by a template unless you explicitly say so
4. `xsl:apply-templates select="XX"` looks for templates which match element "XX"; `xsl:value-of select="XX"` simply gets any text from that element
5. The order of templates in your program file is immaterial
6. You can process any part of the document from any template
7. Everything is well-formed XML. Everything!

Important magic

Our examples and exercises all start with two important attributes on <stylesheet>:

```
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xpath-default-namespace="http://www.tei-c.org/ns/1.0"
 version="2.0">
 ....
```

which indicates

1. In our XPath expressions, any element name without a namespace is assumed to be in the TEI namespace
2. We want to use version 2.0 of the XSLT specification.

More complex patterns

The select attribute can point to any part of the document. Using XPath expressions, we can find:

`/`	the root of document (*outside* the root element)
`*`	any element
`text()`	
`name`	an element called 'name'
`@name`	an attribute called 'name'

Example of complete path in <value-of>:

```
<xsl:value-of
  select="/TEI/teiHeader/fileDesc/titleStmt/title"/>
```

Next

Any Questions?