TEI XSLT Stylesheets for Transformation

TEI@Oxford

July 2011

TEI stylesheets

The family of XSL stylesheets which are delivered with the TEI on Sourceforge have three functions:

- implementation of the ODD schema meta-language, ie providing tools to take an ODD customization of the TEI and generate the appropriate schemas and documentation (what the Roma web service does)
- plausible rendering to HTML, XSL FO, Word, Open Office, ePub and LaTeX of typical born-digital TEI documents
- vehicle for managing other conversions in and out of TEI (Word, Open Office, ePub, Docbook, etc)

Although there are XSLT 1.0 and TEI P4 versions, the code is only developed in XSLT 2.0 for conformant TEI P5.

The stylesheet family is available

- within oXygen as default transformation for TEI documents
- as downloadable Debian/Ubuntu packages
- inside the online OxGarage converter

Why should you use this library of styleheets, rather than roll your own?

- They solve problems you may not have thought of you (generating the right links when making multiple output files)
- They cover a variety of output formats you may need one day
- It is better to co-operate on an open-source project than reinvent too many wheels
- They are already widely packaged and distributed

Limitations

These stylesheets only do what were designed to do!

- They do not provide a rendering of all TEI elements
- They do not implement all possible values of every *@rend* attribute
- The different output formats are not always in sync, or give the same result

but they do deal with quite a few common problems.

Output assumptions

The stylesheets attempt to work in the same way with each of the supported output formats, but note:

- The HTML output is designed to work with an associated CSS stylesheet, which takes care of much of the detailed spacing and font work; however, the HTML is in charge of features such as the numbering of sections.
- The LaTeX output is designed for people who understand how to use existing LaTeX packages and classes; it therefore tries to produce reasonably readable TeX markup, with high-level commands whose effects will be determined by LaTeX (including numbering and spacing).
- The XSL FO output produces a very detailed specification of the output layout, with all the details of fonts, numbering, vertical and horizontal spacing specified *in situ*. The FO processor is only responsible for line and page breaking, and hyphenation.

How the stylesheets work

Fundamental XSLT to understand

- import and include
- top-level parameters and variables
- named templates and hooks

Feature: `import` and `include`

`<xsl:import href="...">`: include a file of XSLT templates, overriding them as needed

`<xsl:include href="...">`: include a file of XSLT templates, but do not override them

If you want to pull in a file which has the same template as the current file:

- if you use <import>, the one in the current template has a higher priority
- if you use <include>, you will get an error, unless you manually assign a higher priority to one or the other

Top-level <param> and <variable>

- You can declare variables directly as children of <stylesheet>

```
<xsl:variable name="TEI">Text Encoding
Initiative</xsl:variable>
```

  as a convenience
- You can also declare parameters directly as children of <stylesheet>, but these can be overridden when the stylesheet is called:

```
<xsl:param name="logo">../Graphics/logo</xsl:param>
```

<import> example

```
<xsl:import
  href="/usr/share/xml/tei/stylesheet/slides2/teihtml-
slides.xsl"/>
<xsl:param name="logoFile">../Graphics/logo.png</xsl:param>
<xsl:param name="cssFile">teislides.css</xsl:param>
<xsl:param name="showNamespaceDecls">false</xsl:param>
<xsl:param name="forceWrap">true</xsl:param>
<xsl:param name="spaceCharacter"> </xsl:param>
<xsl:template name="lineBreak">
 <xsl:param name="id"/>
 <br/>
</xsl:template>
```

Feature: named `template`

Often, it is convenient to store common *code* in a named template for re-use or to make the code more readable:

```
<xsl:template match="div1|div2" mode="toc">
 <xsl:call-template name="header"/>
 <xsl:apply-templates/>
</xsl:template>
<xsl:template name="header">
 <li>
  <xsl:number level="multiple"/>
  <xsl:text/>
  <xsl:value-of select="head"/>
 </li>
</xsl:template>
```

Parameters to templates

You can also pass parameters to templates:

```
<xsl:template match="div">
 <xsl:call-template name="toc">
  <xsl:with-param name="text">
   <xsl:value-of select="head"/>
  </xsl:with-param>
 </xsl:call-template>
</xsl:template>
<xsl:template match="person">
 <xsl:call-template name="toc">
  <xsl:with-param name="text">
   <xsl:value-of select="surname"/>
   <xsl:text>, </xsl:text>
   <xsl:value-of select="forename"/>
  </xsl:with-param>
 </xsl:call-template>
</xsl:template>
<xsl:template name="toc">
 <xsl:param name="text"/>
 <li>
  <xsl:value-of select="$text"/>
 </li>
</xsl:template>
```

TEI Stylesheet family top-level layout

Directories for output formats

docx	Converting to and from Word OOXML
epub	Converting to ePub
fo2	Making XSL FO
latex2	Making LaTeX
nlm	Converting from NLM
odds2	Transforming TEI ODD specifications
odt	Converting to and from OpenOffice Writer
slides2	Making slides (HTML and PDF)
tite	Converting from TEI Tite
xhtml2	Making HTML

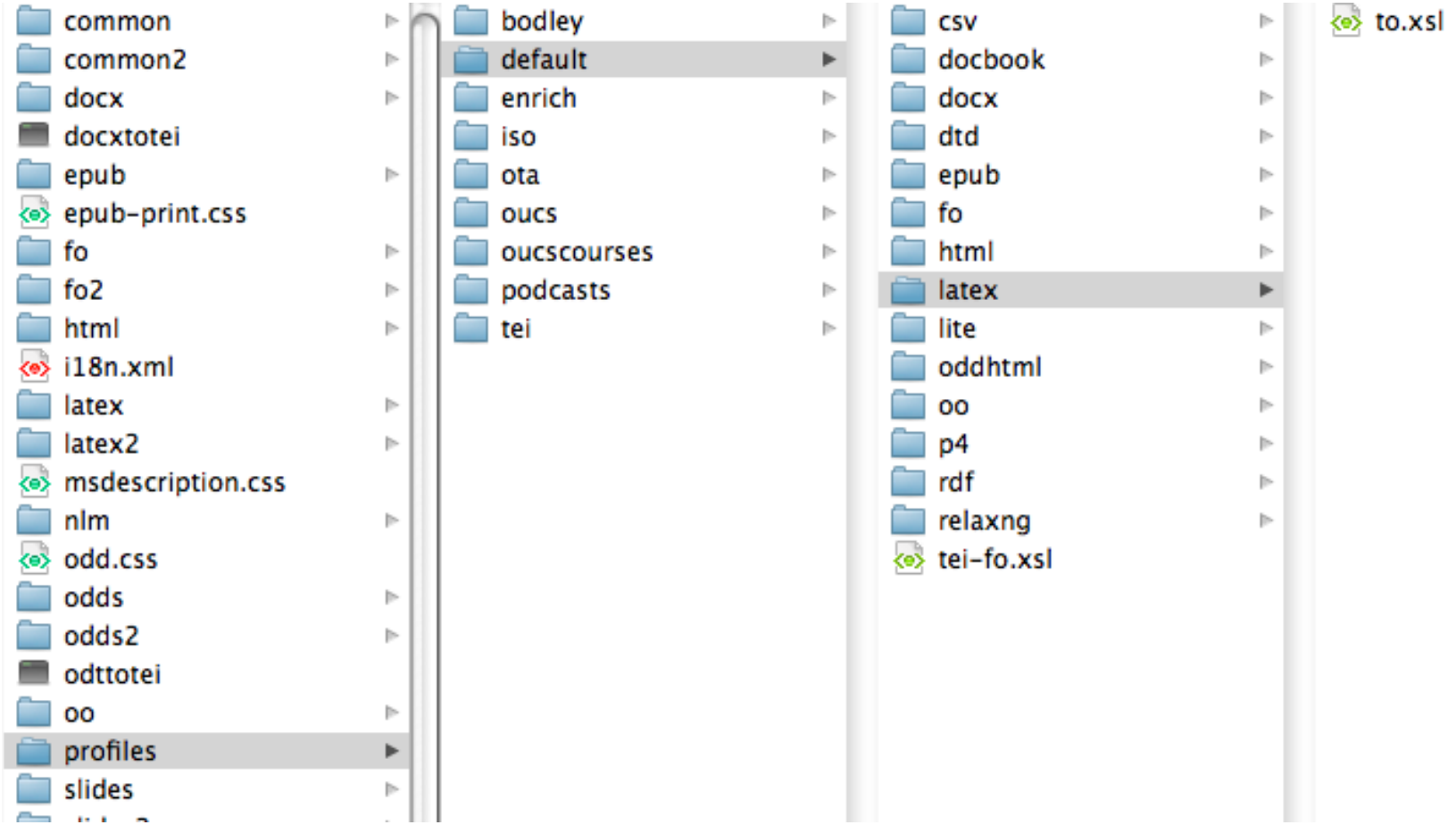
Special directories

profiles	Customizations
common2	Templates for any output format
tools2	Utilities

Content of HTML (or LaTeX or XSL FO) directory

core.xsl	Basic TEI elements
dictionaries.xsl	Dictionaries module
drama.xsl	Drama module
figures.xsl	Figures and tables module
header.xsl	Header module
linking.xsl	Linking module
namesdates.xsl	Names and Dates module
tagdocs.xsl	Processing ODDs
tei-param.xsl	Parameters
tei.xsl	Top-level wrapper
textcrit.xsl	Text critical module
textstructure.xsl	Basic structure
transcr.xsl	Transcription module
verse.xsl	Verse module

Layout of a profile directory



Profile conventions

- in a directory hierarchy of the form name/format/from.xsl or name/format/to.xsl (indicating whether it is a conversion *from* or *to* the format)
- known formats are: `csv`, `docbook`, `docx`, `epub`, `fo`, `html`, `latex`, `oo`, `p4` and (special cases for ODD processing) `lite`, `oddhtml`, `rdf`, `dtd`, and `relaxng`.
- references to the 'master' conversions should be in the form (eg)

```
<xsl:import href="../../../epub/tei-to-epub.xsl"/>
```

Areas of customization (HTML)

- Standard page features
- Layout
- Headings
- Numbering
- Output
- Table of contents generation
- Internationalization
- CSS
- Tables
- Figures and graphics
- Inline style

Remember that in HTML a lot will be done with CSS and JavaScript

Understanding the customization

There are six levels of interaction with the stylesheet family:

1. setting parameters
2. overriding templates provided for this purposed (listed in customization guide)
3. writing templates which implement the empty 'hooks' (listed in the customization guide)
4. adding new templates for elements not covered by the family
5. providing complete replacements for low-level templates

Always make changes by overriding — never hack the originals!

Many parameters

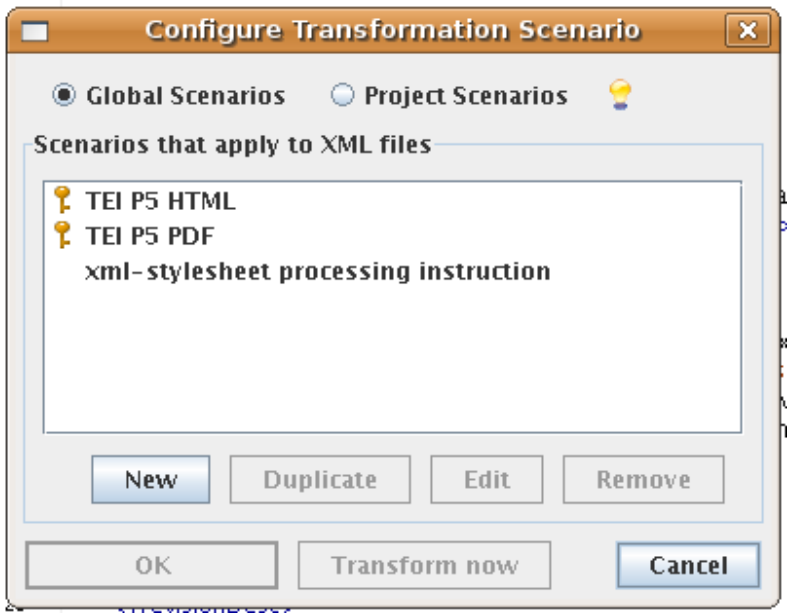
There are dozens and dozens of parameters which affect the stylesheet output; you can set values for these by

- specifying parameter names and values directly in oXygen
- setting them on a command line
- constructing a small local stylesheet which imports the public one, and adds overrides

Invoking an XSLT transform from oXygen

When you have loaded an XML file, look for the  symbol in the menu and press it.

The first time, it will ask you which transformation scenario to use:



Simple result

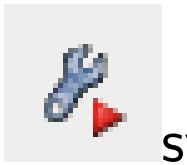
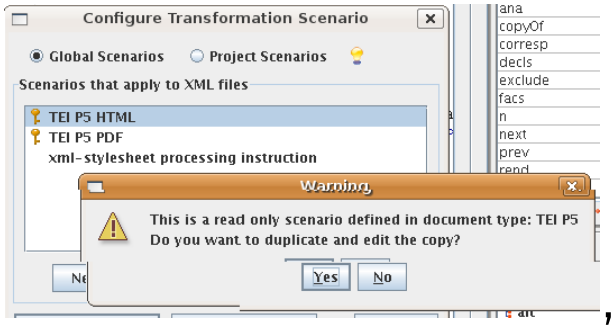
A TEI Project

Punch, or the London Charivari, Vol. 147, July 1, 1914

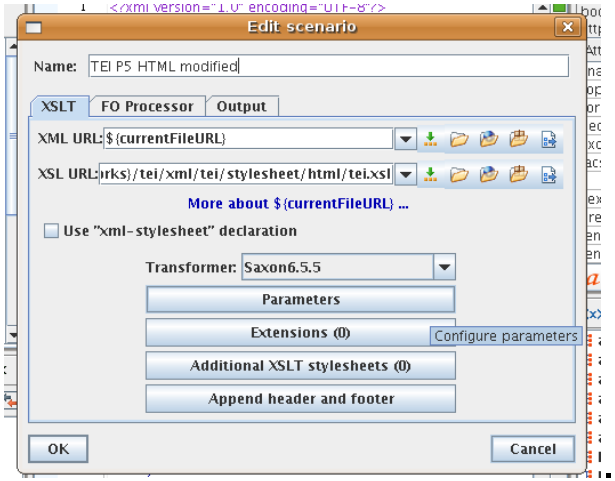
Table of contents

Find: feed Previous Next Highlight all Match case

Configuring the scenario in oXygen

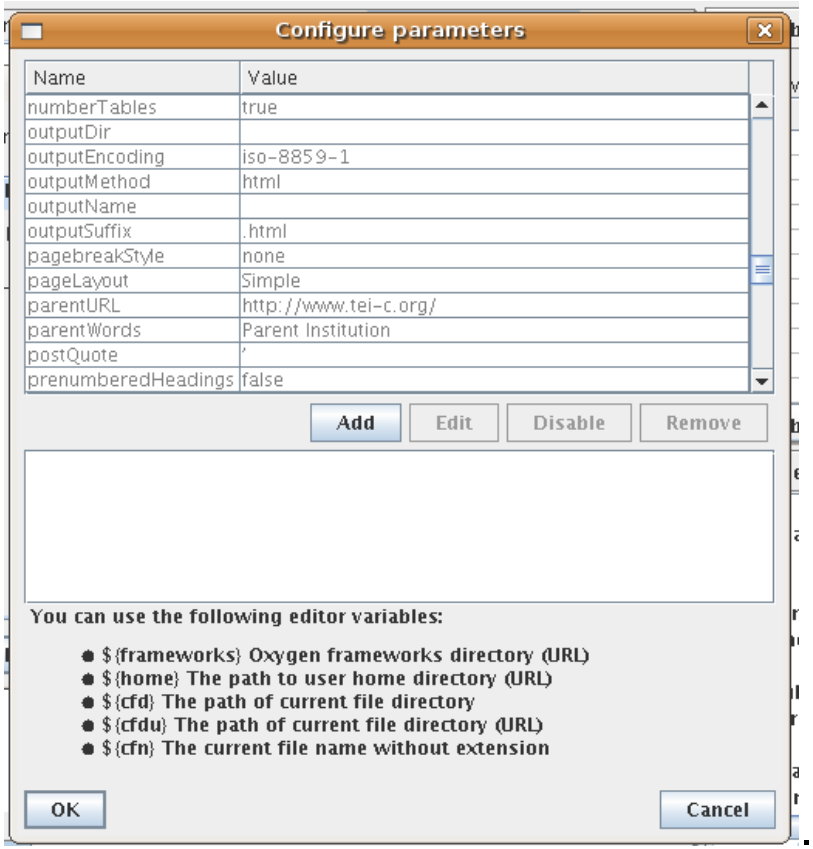
Look for the  symbol. This produces 

, asking if you want to change the setup. Choose yes, and you see

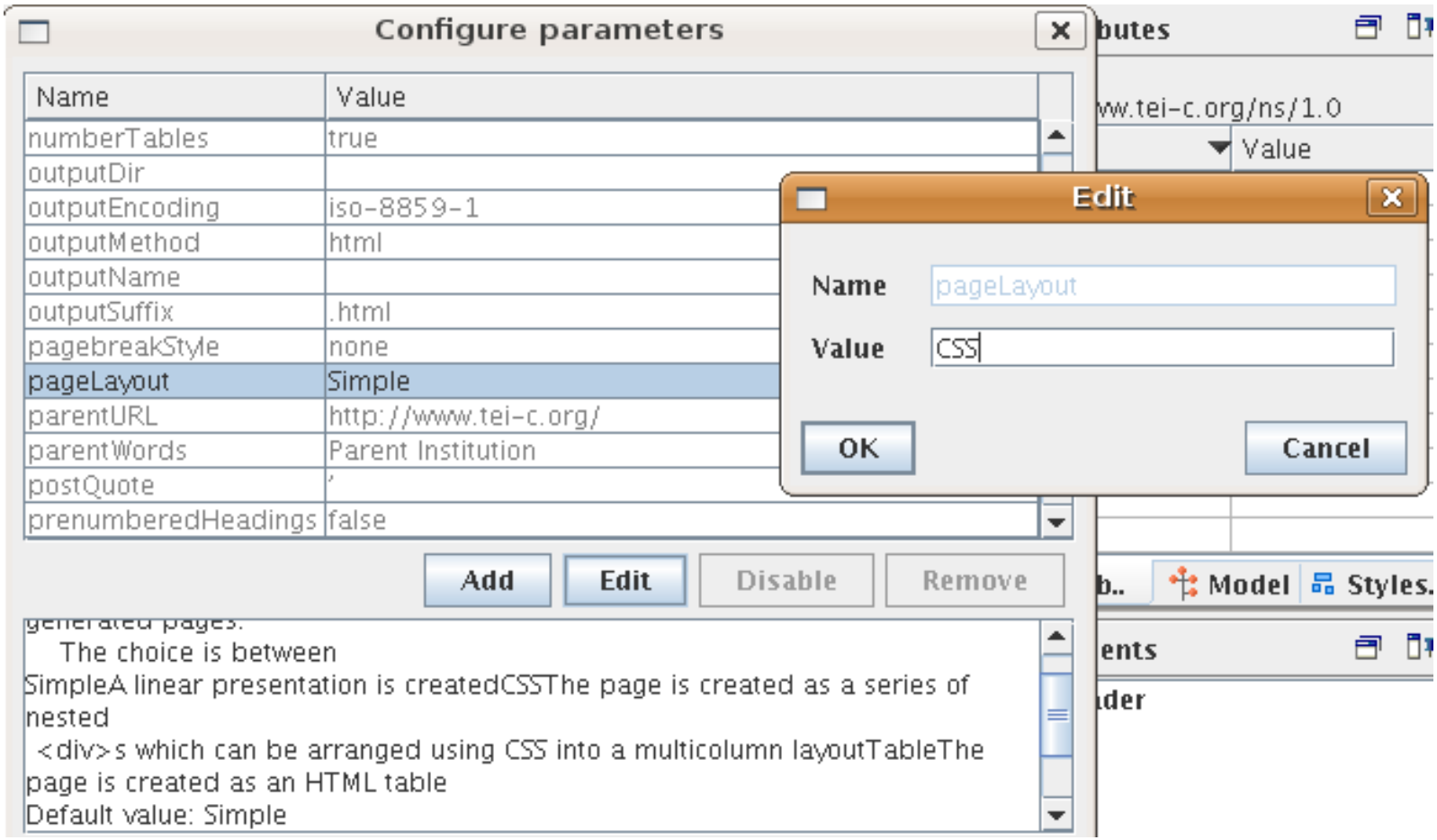


Changing parameters in oXygen

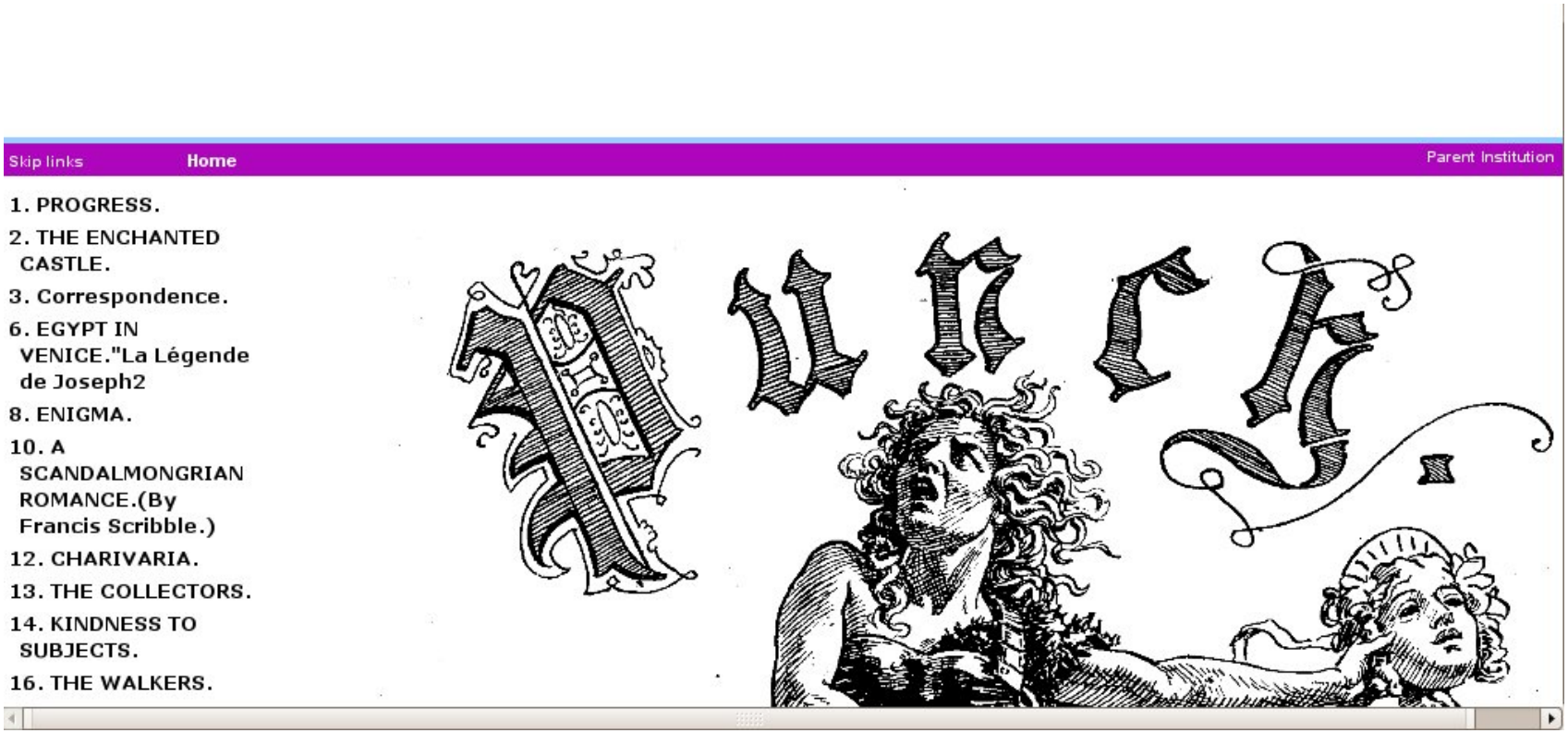
Now you can supply values for parameters:



Change pageLayout



2 column display



Changing things around a bit

Punch, or the London Charivari, Vol. 147, July 1, 1914

Using the a wrapper stylesheet

The simplest example of making a wrapper for the HTML stylesheets is:

```
<xsl:stylesheet version="2.0">
 <xsl:include
   href="http://www.tei-
c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl"/>
</xsl:stylesheet>
```

Using the a wrapper stylesheet (2)

Now you can build on it:

```
<xsl:stylesheet version="2.0">
 <xsl:include
   href="http://www.tei-
c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl"/>
 <xsl:param name="logoFile">../../logo.png</xsl:param>
 <xsl:param name="logoWidth">60</xsl:param>
 <xsl:param name="logoHeight">60</xsl:param>
 <xsl:param name="cssFile">myTEI.css</xsl:param>
 <xsl:param name="pageLayout">CSS</xsl:param>
 <xsl:param name="outputMethod">xml</xsl:param>
 <xsl:param name="parentWords">The Punch
   Project</xsl:param>
 <xsl:param name="institution">The University of
   Punch</xsl:param>
</xsl:stylesheet>
```

Using the a wrapper stylesheet (3)

And start to add your own templates:

```
<xsl:stylesheet version="2.0">
 <xsl:include
   href="http://www.tei-
c.org/release/xml/tei/stylesheet/xhtml2/tei.xsl"/>
 <xsl:param name="logoFile">../../logo.png</xsl:param>
 <xsl:param name="logoWidth">60</xsl:param>
 <xsl:param name="logoHeight">60</xsl:param>
 <xsl:param name="cssFile">myTEI.css</xsl:param>
 <xsl:param name="pageLayout">CSS</xsl:param>
 <xsl:param name="outputMethod">xml</xsl:param>
 <xsl:param name="parentWords">The Punch
   Project</xsl:param>
 <xsl:param name="parentURL">http://tei.oucs.ox.ac.uk/Punch/</xsl
 <xsl:param name="institution">The
   University of Punch</xsl:param>
 <xsl:template match="tei:hi[@rend='upsidedown']">
  <span class="upsidedown">
   <xsl:apply-templates/>
  </span>
 </xsl:template>
</xsl:stylesheet>
```

OxGarage

OxGarage is a web interface to the XSL stylesheets and its profiles: http://oxgarage.oucs.ox.ac.uk:8080/ege-webclient

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 Document (.doc)	Microsoft Word Document (.docx)	ODD Document	Open Office Text Document (.odt)	OpenOffice 1.0 Text Document (.sxw)	Plain Text (.txt)	Rich Text Format (.rtf)	TEI P4 XML Document	TEI P5 XML Document	WordPerfect Document (.wpd)	xHTML Document
ODD documentation as TEI Lite	✓				✓								
Comma-Separated Values (.csv)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DTD created from ODD	✓				✓								
ePub Document	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
LaTeX Document	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Excel Document (.xls)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
xHTML Document	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Microsoft Word Document (.doc)	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Microsoft Word Document (.docx)	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓
National Library of Medicine (NLM) DTD 3.0	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
ODD documentation as HTML	✓				✓								
Open Office Spreadsheet (.ods)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Open Office Text Document (.odt)	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
OpenOffice 1.0 Spreadsheet (.sxc)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
OpenOffice 1.0 Text Document (.sxw)	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓
PDF Document	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Plain Text (.txt)	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
RELAXNG schema created from ODD	✓				✓								
Rich Text Format (.rtf)	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
Tab-Separated Values (.tsv)	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
XSL-FO Document	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
TEI P5 XML Document		✓	✓	✓		✓	✓	✓	✓	✓		✓	✓
Compiled ODD Document					✓								

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
```