Product Feature Idea – Semantic POC documentation

Client:	OECD
Project:	Semantic POC
Prepared By:	Rodney Lorrimar
Date:	4th May 2017
Version:	1.0
Document Status:	Issued

Background

During March and April 2016, a number of workshops and calls between 67 Bricks and OECD key stakeholders took place to help OECD understand what product features to build into future versions of the OECD content delivery platform. Following this work, a prioritised list of potential product features that could be prototyped was agreed.

The Semantic POC (Proof of Concept) was selected to demonstrate how publications could be enhanced by adding in-line links, relatedness, visualisations and connections to other related content assets. OECD selected two publications - 'Economic Outlook' and 'Going for Growth' - to enhance as part of this work.

Working in a time-based agile method the implementation of the selected prototype was completed over a 6-week period. The 67 Bricks team worked with the team at OECD to agree and work through as many of the high priority features as possible within the timeframe.

Scope of this document

This document contains brief information on how the POC was built, what data it used, and what else would be needed to make a fully working system.

It contains instructions on how to deploy and maintain the POC, and some notes about the implementation.

Data

Data used in the POC

1. Temis z-scores for each publication were collected into a time series dataset.
   a. This dataset is the basis of the bubble chart.
   b. The z-score for subjects and coverage areas was used in the tag clouds of each issue.

2. Agora glossary spreadsheet were converted into a SKOS taxonomy in a semi-automated process. A taxonomy term recognition service was run over the documents to annotate terms matching the taxonomy.

a. Annotated terms are highlighted in the text and display a popover when clicked. The popover contains the definition from the glossary, and sometimes a link to the OECD iLibrary.
b. Frequency of term annotation was used as the data source for the Glossary Terms tag cloud.

3. Country information spreadsheet was converted into a SKOS taxonomy and was also loaded for use when recognizing terms.
    a. Annotated country terms are highlighted in the text and display a popover when clicked.
    b. The popover contains an embedded OECD chart of GDP for the country, as well as links to other OECD resources.
    c. The popover links to the headings of the current issue which mention this country.
4. Other Narrdoc XML elements were used to produce a basic presentation of the document.
    a. Formatting elements and headings were translated into HTML equivalents.
    b. Figures and tables were presented as grey rectangles.
    c. Heading elements from each chapter/section/etc of the documents were collected into a table of contents.
5. The table of contents and term annotations of all issues of all publications were combined to produce the country/subject listings which appeared below the bubble chart.

Further enrichment required

During the work on the POC a number of further activities have been identified that would add greater enhancement to the work. This is not a complete list at this stage and is based on the features developed as part of the POC.

- Use a better tag cloud weighting formula, comparing term frequency against average.
- Generate Temis reports per section as well as per document.
- Use the same set of keywords in Temis as in the glossary
- Add more countries in Temis report.

Additional data needed for full product

For the proof of concept, some of the data required was manually set up. Additional source data would be needed to support creating a full product:
Publication metadata. For the POC, the publication blurb, cover images, list of issues, etc, were manually set up.

1. Publication metadata. For the POC, the publication blurb, cover images, list of issues, etc, were manually set up.
2. iLibrary links for topics. A small number of iLibrary links (e.g. OECD Productivity Statistics) were manually entered for the POC.
3. Mapping between Luxid taxonomy (skill cartridge) and glossary terms. These are separate datasets covering more or less the same thing. This is why there were two tag clouds for subjects and glossary terms. The mapping between the two datasets was not entirely accurate in the POC.
4. Glossary translations. If translations of glossary terms are available, then these could also be provided in the popovers.

5. Document translations. The POC only shows the English version of the two documents.

Narrdoc Schema

Some elements of the Narrdoc schema which could be exploited further are:

1. Some tables and figures can be detached from their documents and viewed as standalone pieces of information or in different contexts.
2. Rather than displaying the entire document, the documents can be chunked into individual sections. Sections could be linked and viewed alongside sections from other issues and publications
3. Chapter bibliographies. Cited articles can form part of a list of related OECD publications.
4. Footnotes and unlabelled cross references are not presented in the POC. A popover or similar user interface element could be used to show this information without interrupting the user's reading flow.
5. The geocontainer="true" attribute of chapters and sections could be used, but usually the country/region name in a heading is enough to identify it.

System Maintenance

Supported browsers

The site has been primarily developed and tested against the Google Chrome desktop browser. Supporting smaller screens or other web browsers was not a priority for the POC, and so these were not tested. The layout might look odd if the browser window is too narrow.
Because the site uses the Bootstrap grid layout framework, it is a relatively straightforward matter to add support for smaller screens such as tablets.
User interactions such as mouse pointer hovering require a desktop web browser, and adjustments would need to be made to support touch screens e.g. on mobile devices.

Deployment

The POC is a static web site and doesn't have any application server component.
It can therefore be served by a standard web server such as Apache or NGINX.
The HTML files are checked in to git as data/_build/html. These can simply be copied to the web server document root.
When serving with NGINX, access logs are written to /var/log/nginx. This web server can be restarted with “/etc/init.d/nginx restart”.

67 Bricks hosting during development

Through the development of the POC, the site has been hosted on an Amazon EC2 instance.

Implementation

Building the code

The static HTML files are built from source documents and other data files using a Makefile-like build script.

Install software dependencies

Use Docker to install a standard base image with necessary packages.

```
cd data
PROJ=`pwd`
docker build -t oecd-poc:latest .
```

Run the build command

Start a shell in the Docker image:

```
docker run --name oecd-poc --volume $PROJ:/src -p 127.0.0.1:8000:8000 -ti oecd-poc:latest
```

Inside the docker image:

```
cd /src
./build.sh -j
```

To quickly view the generated files, you can use the embedded web server.

```
./server.sh 0.0.0.0
```

Then open http://localhost:8000/ in a web browser.

Note: the enrichment software used for identifying taxonomy terms in the annotated content is not included. Therefore, the XMLs in data/_build/annotated can't be rebuilt.
Everything else can be rebuilt with the software configured in the Dockerfile.

How the site works

The HTML files are generated using a XSLT from the document XMLs. The publication main pages are also applied to the template.
There is a single script called oecd.ts which loads data and renders the dynamic elements. The script loads datasets from external JSON files.

How the build works

Source files are placed at the following locations:

- temis.nobib/ – source documents and Temis reports.
- csv/ – Agora glossary and country data.

- taxonomies/ – inputs to bricksTermRecognition.

There are various scripts in scripts for processing the data:

- collect_reports.py – collate Temis report XML for each issue into a single JSON file for presentation of time-series data in a chart.
- luxid2json.py – routines to convert Temis report XML into JSON (used by collect_reports.py).
- countries2json.py – generates a JSON file with data for the country popovers.
- get_words.py – calculates a histogram of recognized glossary terms, for the publication issue tag clouds.
- terms2html.py – converts the Agora glossary spreadsheet into a HTML file which is loaded in the annotation popovers.
- terms2skos.py – converts the Agora glossary spreadsheet into a Turtle format taxonomy, for loading into bricksTermRecognition.
- terms.py – routines for loading the Agora glossary spreadsheet (used by terms2html.py and terms2skos.py).
- toc_concepts.py – scans the headings of annotated publication issues and maps the recognized terms which they contain.

XSL transformations are in the xsl directory.

- base.xsl – common HTML skeleton for site.
- issue.xsl – transformation from Narrdoc to HTML.
- publication.xsl – embeds the publication page content in the template.

Other materials for the web site are in the assets directory.

- index.html – front page.
- growth.html – publication page content.
- eco_outlook.html - publication page content.
- oecd.ts – the browser script.
- oecd.scss – the browser stylesheet.
- Other files are here such as cover images and vendored library code.

Frontend libraries used

- Bootstrap 4 - flexbox grid layout system, visual styling, popovers, modal dialog elements, etc.
- JQuery - data loading and some dynamic elements.
- VueJS - dynamic elements.
- ChartJS - bubble chart.
- jqCloud - tag cloud.

Build tools used

- Typescript compiler – adds extra error checking to javascript.
- SCSS – preprocessor for simpler CSS.
- Shake build system – flexible declaration of build dependencies.
- Saxon – XSLT 2.0 processor.
- Docker – common tool for setting up build environments.