Report Project 2 : XSLT

Rémy Detobel & Nathan Liccardo

April 7, 2018

1 Introduction

This document aims to detail all the choices that we made during our implementation. As a reminder, the main goal of this project was to create an xslt file. This file had to be able to transform a part of the dblp database into html files. Our xslt file has been separated into two parts : functions and template. This report will follow the same structure.

2 Functions

There exist five complex operations which have necessitated to create functions. This first section will be focused on describing those functions (`removeSpecialChar`, `lastName`, `firstName`, `firstLetter` and `nameToPath`).

Remove Special Char Is used to replace all the non-alphanumeric characters. It takes (as input) a person name (like "C. B. Williams") and replace each characters which matches with the following regex : `[^0-9a-zA-Z_]`. Matching characters are replaced by the equal character (=). Substitute a specific characters can be realised thanks to the `replace(input, regex, char)` function. This function is defined in xslt 2.0.

Last Name Can extract the string corresponding to the last name. To be able to extract and format this element from the given input, we used two functions :

1. `tokenize` : Split the input into an array (using white spaces). Last name is corresponding to the last element. This function is defined in xslt 2.0.
2. `removeSpecialChar` : Replace (eventual) special characters from the last name. The exact definition has been given previously.

First Name Can extract the first name from the given input string. Formally, first name is corresponding to all the input string minus the last name (defined before). To extract and format this element as requested, we used four functions :

1. `tokenize` : Split the input string (using white spaces) and get the last name (last element).
2. `substring-before` : Get all the string before the last name.
3. `replace` : Replace each white spaces by equal characters in the sub-string result.
4. `removeSpecialChar` : Used as defined before on the replace result.

First Letter Is used to extract first letter of the last name. The function use `substring` to select the first letter (as with substring-before), `tokenize` to split the input string) and `removeSpecialChar` to replace the character (if it is non-alphanumeric). In addition, we use the `lower-case` function defined in xslt 2.0.

Name To Path Is used to properly format each html file name. Thanks to each previous functions, we are now able to get the first letter of the last name, the first name and the last name. Each of those strings will be concatenated as follow :

```
"firstLetter"/"lastName":"firstName"
```

Where ”firstLetter”, ”lastName” and ”firstName” are the output strings given by each previous functions.

3 Template

Templates are used to define set of rules. Our project only use one template. This element will, in fact, contain each rules corresponding to the transformation from XML to HTML files. This sections will focus on two main elements : Xpath and HTML file structure. Xpath is used to describe path in XML. To correctly understand the structure of the template, we will describe each elements in sequence.

3.1 Creation of HTML pages

Each author or editor must have an HTML page. This means that we must iterate on the dblp database and get each different authors or editors. To do this, we used the `for-each` tag. This tag use an Xpath which is defined as follow :

```
select="distinct-values(//author|//editor)"
```

For each element (author or editor) contained in this set we create an HTML file. Those file are created using the `result-document` tag as follow :

```
<xsl:result-document href="{func:nameToPath(.)}.html">
```

We can see that we use the `nameToPath` function previously defined. The string given as parameter is corresponding to the current element in the set of authors and editors. For each pages, we generate different body that we will describe in the next section.

3.2 Title and articles table

For each page, we must have the author name as title followed by a table which contain each articles of the author. Regarding to the title, this is realised as below :

```
<h1> <xsl:value-of select="."/> </h1>
```

We must now create a table which contain each article corresponding to the current author. This means that we must :

- Obtain each publications. This is realised using a variable declared after the body tag. This variable contains each articles which contains the current author.
- Order each publications in the descending order. This is realised using the `sort` tag with the `descending` parameter. Both are defined in xslt 2.0.
- Create a subsection for each year. This is realised by using a `test` tag and the following condition : `not(preceding-sibling::*[1]/year=./year) or position()=1`. Indeed, this condition checks if the previous item as the same value for the node `year`.
- Add the publication number. As we must count publications in the reverse order, we use a variable (`indexPub`, declared after the `sort`). Index value is obtained by : `last()-position()+1`.
- Add, each authors or editors of the current publication and link all of them to their personal web page. This is also realised using the `nameToPath` function.

3.3 Co-author index

Finally, for each HTML page, we must have a table which contain all other persons with whom P has jointly published. To realise this, we use two iterations :

- On each authors which have jointly published with the current author. This is realised using an `for-each` tag and the following request :

  ```
  select="$root//*[./author=$author]/author[not(.=$author)]"
  ```

- If we call the author of this new iteration A2 and the initial author A1, we iterate on all the publication of A1 and check if A2 is also an author. If it is the case, then we can add the publication :

  ```
  <xsl:for-each select="$publications">
      <xsl:if test="$coAuthor=./author" >
          <xsl:variable name="linkPublication" select="last()-position()+1" />
           [<a href="#p{$linkPublication}">
            <xsl:value-of select="$linkPublication" />
            </a>]
      </xsl:if>
  </xsl:for-each>
  ```