Inf1B Data and Analysis
Tutorial 5 (week 7)

Hutchins-Korte, Simpson

12 February 2008

- Please answer all questions on this worksheet in advance of the tutorial, and bring with you all work, including printouts of your `recipe-queries.xq`-file and the XML files it produces. Tutorials cannot function properly unless you do the work in advance.
- Data & Analysis tutorial exercises are not assessed, but are a compulsory and important part of the course. If you do not do the exercises then you are unlikely to pass the exam.
- Attendance at tutorials is obligatory; please let your tutor know if you cannot attend.
- *Recommended Reading:* XPath Tutorial (`http://www.w3schools.com/xpath/default.asp`), XQuery Tutorial (`http://www.w3schools.com/xquery/default.asp`).

Introduction

This week's tutorial uses the *XQuery* language, introduced in lecture note 7, as a means for querying XML documents. The goal is write queries to extract data from an on-line XML document containing food recipes, and to test these queries using the Galax program installed on all DICE machines.

Some hints with XQuery

The following information augments the lecture notes with additional ways of building predicates which may prove useful when writing the queries below. For examples illustrating the use of these features, look at the on-line tutorials mentioned above.

- Besides the equality check discussed in the lecture, you can also use comparison like `<` or `>` in a predicate. E.g.

```
doc("examples.xml")/elem[@attr < 5]
```

- Boolean operations like `and`, `or`, and `not` are allowed in predicates. E.g.

  ```
  doc("examples.xml")/elem[not(@attr1 < 5) and @attr2 = 10]
  ```

- Boolean expressions can even be combined with path expressions. For example, the following test is true only when the path-expression returns an empty list of nodes.

  `not (`*`path-expression`*`)`

- Caution needs to be used when writing a path expression *inside* square predicate brackets. We use the XML document from the lecture note as an illustration.

  (a) `doc("gaz.xml")//Country[Region/Feature[@type='lake']]`
  As expected, this returns those `Country` nodes for which the country contains at least one lake. However,

  (b) `doc("gaz.xml")//Country[//Feature[@type='lake']]`
  will return all `Country` nodes in the gazetteer, as long as there exists at least one lake in the Gazetteer. The reason for this is that XQuery interprets path expressions that start with a `/` (in the example, the path expression `//Feature[@type='lake']]`) as expressing a path from the root node of the document. In contrast, in example (a) above, the path expression `Region/Feature[@type='lake']]`, which does not start with a `/`, is interpreted as a *relative path expression* and so defines paths from the current context node. The counterintuitive interpretation of `//Feature[@type='lake']]` can be circumvented by using the path expression below instead.

  (c) `doc("gaz.xml")//Country[.//Feature[@type='lake']]`
  As in example (a), this expression returns those `Country` nodes for which the country contains at leat one lake. Here the "`.`" on the predicating path expression tells XQuery to interpret the continuation expression `//Feature[@type='lake']` from the current context node.

Setting up Galax on DICE

- Create a new directory called `galax` in your home directory.
- Download the XML file containing the recipe data to your new `galax`-directory. The XML file can be found at the following location:

  `http://www.brics.dk/~amoeller/XML/xml/recipes.xml`

- Download `recipe-queries.xq` to your new `galax`-directory. The `.xq`-file can be found on the course website:

  `http://www.inf.ed.ac.uk/teaching/courses/inf1/da/tutorials/recipes.xq`

1 XQuery: Path Expressions

Question 1

Write a path expression which retrieves the titles of all recipes and run the query on Galax. To do this you need to:

(a) Open the file `recipe-queries.xq` tn a text editor.

(b) Add your expression or query in the allocated space.

(c) Save.

(d) Open a terminal window.

(e) Change directory to your `galax`-directory.

(f) In your `galax`-directory, type the following:
`galax-run recipe-queries.xq`

Your output should look like this:

```
<title xmlns="http://recipes.org">Beef Parmesan with Garlic Angel Hair Pasta</title>,
<title xmlns="http://recipes.org">Ricotta Pie</title>,
<title xmlns="http://recipes.org">Linguine Pescadoro</title>,
<title xmlns="http://recipes.org">Zuppa Inglese</title>,
<title xmlns="http://recipes.org">Cailles en Sarcophages</title>
```

Note here the annoying attribute `xmlns="http://recipes.org"`. This declares the *namespace* of the `title` element. Namespaces are an important feature of XML, but are beyond the scope of this course.

Question 2

Write a path expression which retrieves the titles of all recipes with less than 1000 calories and run the query on Galax.

N.B. Galax can only run one query at a time, so you will need to comment out your answer to Question 1, before you are able to run your new query. In general, you should always comment out all your queries, except the one your are currently trying to run. You can comment out a line by enclosing it in mirrored (noseless) smiley-faces. E.g. the following line is commented out.

```
(: This line is commented out :)
```

Question 3

Write a path expression which retrieves all the ingredients of ricotta pie and run the query on Galax.

Question 4

Write a path expression which retrieves all compound ingredients, i.e. ingredients composed from other ingredients, that appear in the document, and run the query on Galax.

Question 5

Write a path expression which retrieves all ingredients of the 'dough' used for the ricotta pie and run the query on Galax.

2 XQuery: Outputting an XML Document

In the previous section the output always consisted of a list of elements. In this section, your queries need to produce their output in the form of an XML document. Furthermore, this output will have to satisfy a specific DTD.

Question 6 - Almost out of Flour

Specify a query which retrieves (the titles of) all recipes which use one cup of flour or less. You may assume that the unit of measurement is always 'cup'. When you run your query on Galax, the output should satisfy the following DTD:

```
<!DOCTYPE LOW-ON-FLOUR [
<!ELEMENT LOW-ON-FLOUR (title)*>
  <!ELEMENT title (#PCDATA)>
]>
```

Question 7 - No Milk

Specify a query which retrieves all recipes which do not use any milk. When you run your query on Galax, the output should satisfy the following DTD:

```
<!DOCTYPE NO-MILK [
<!ELEMENT NO-MILK (title)*>
  <!ELEMENT title (#PCDATA)>
]>
```

Question 8 - Shopping Lists

Specify a query which composes shopping lists for each recipe, by extracting all the basic ingredients in the recipe (i.e. those ingredients that are *not* composed from other ingredients). When you run your query on Galax, the output should satisfy the following DTD:

```
<!DOCTYPE GROCERY-LISTS [
<!ELEMENT GROCERY-LISTS (GROCERY-LIST)*>
  <!ELEMENT GROCERY-LIST (title,ingredient*)>
     <!ELEMENT title (#PCDATA)>
     <!ELEMENT ingredient EMPTY>
]>
```

N.B., although we do not specify the attributes of the `ingredient` element in the above DTD, it is enough to copy the `ingredient` elements from `recipes.xml`, which means the relevant attributes will automatically be copied over.