XML Technologies: Exercise 3

XML Schema, XPath

This is a mandatory exercise and the result will be part of your final mark. The solution must be uploaded to OLAT by **April 1st at 15:59** (no jokes permitted). Late submissions will not be accepted.

Submit the following files in a zipped archive:

- `xwing.xsd` and `xwing.xml`
- `xpath.txt`

Make sure the archive is named [lastname]_[firstname]_3.zip (for example *mueller_mathias_3.zip*. Some parts of your submission may be automatically evaluated, so make sure to name your files *precisely* as prescribed, otherwise you might not get any points.

1 A Better Way to Validate (2 Points)

XML Schemas are used to validate XML documents. As such, they perform exactly the same task as DTDs. In this section, we will first XML Schema and apply it to a new example (so we're leaving temporarily the world of whisky).

XML Schema is itself written in XML and is far more powerful and expressive than a DTD. For instance, the XML Schema language provides a means of enforcing constraints like the following:

```
<!--There should be exactly 4 wings-->
```

Attached you'll find the documents `xwing.xml` and `xwing_incomplete.xsd`. The first document is an XML file describing a spaceship. The second document is an incomplete XML Schema for `X-wing` documents. A comment `<!--...-->` in the `xwing.xsd` file indicates a gap that must be filled.

Fill in the blanks in the `xwing.xsd` to obtain a complete XML Schema that validates the `xwing.xml`. Save the complete XML Schema as `xwing.xsd`. To facilitate evaluation, also include `xwing.xml` in your submission.

Don't let go your conscious self and rely on instinct. Instead, test your solution with a tool like `xmllint` using the `--schema` option.

2 XPath

As you now know, XML is a meta-markup language that provides a syntax to structure data in a rigorously defined way. But XML on its own does not provide a means of *finding* and *accessing* the data inside an XML document. There is a separate language for this task, **XPath**.

XPath is a query language that is meant to select specific nodes (e.g. elements or attributes) from XML content and retrieve them.

2.1 Path Expressions (3 Points)

In this exercise, we will retrieve information from the XML document `distilleries.xml`. It contains most active distilleries in Scotland, and for each of them, list the general taste profile in terms of *nuttiness*, *smokiness* and so on.

A simple example is the following XPath expression:

```
/distilleries/distillery
```

which will yield all `distillery` element nodes.

Write XPath expressions that find the following:

- **The second `distillery` element in the list. (Returns an element node.)**
- **The postcode of the distillery that is called "Linkwood". (Returns text.)**
- **All `distillery` elements that produce smoky whisky (`smoky` is 4).**
- **The number of distilleries that produce whiskies that are quite floral (`floral` is higher than 3). (Returns a number.)**
- **The average fruitiness of all the distilleries that contain *Glen* in their name. (Returns a number.)**
- **A recommendation for a whisky: We're looking for something that has a slightly medicinal flavour (1) and does not come from a region with a postcode containing *AB*. Of those whiskies, we'd like the the one that is the most floral. What's the overall taste intensity of that whisky? (Returns a number, too.)**

Save the path expressions to the file `xpath.txt`, one expression per line.

2.2 Embedding XPath into XSLT

XPath is typically not used in isolation, but embedded into a *higher-level* language that uses XPath as a query language. This is because XPath is in general limited to finding and retrieving nodes, and therefore unable to manipulate or process the results.

Examples for languages that subsume XPath are Java, Python, PHP – and XSLT. In future lectures and exercises, we will look in detail at how XPath is integrated in XSLT and at XSLT as a way to transform XML data.

Do not submit anything for this exciting spoiler section.