# 3. Unit: XML Processing with Java (I)

The first two exercises are "typical" exercises to get into the new technologies:

# Exercise 3.1 (DOM Basics)

Parse mondial.xml into a DOM instance and implement the following query based on the DOM operations (do not apply XPath in DOM):

For all organisations that have their headquarter in the capital of a member country, output the name of the organisation and the name of the headquarter (to System.out).

## Exercise 3.2 (SAX and StAX: Basics, Comparison, Queries against Mondial)

(a)-(c) about SAX, (d) is (a-c) in StAX, (e) is StAX

Write SAX Event Handlers in Java for the following tasks.

- a) Output an HTML file that lists the names of all countries in mondial.xml.
- b) Output the name and the population of the capital of Germany via System.out.
- c) use the previous part of the exercise to output all country names, the country's capital and the country capital's population if available into an HTML table.

Example:

country	capital	capital population
Albania	Tirane	192000
Greece	Athens	885737
:	:	:

- First, simply output the table values as text to System.out.
- Create an HTML table as a JDOM object with the result during the SAX run and output it into a file.
- d) Implement parts a) -c) as given in the SAX exercise, now by using StAX. Reuse the program code as far as possible, and adapt it only to the StAX frame.
- e) Implement the following query in StAX: For all organisations that have their headquarter in the capital of a member country, output the name of the organisation and the name of the headquarter city.

# Exercise 3.3 (SAX, StAX and Streams)

- a) Extend the SAX event Handler from Exercise 3.2: For each country in mondial.xml, output an HTML table containing the names and if present the most recent population count for each city in the country. Use a 
   ul> (unordered list) environment with one list item per country. Example:
  - ...

# • Germany

Stuttgart	588482
Mannheim	316223
Karlsruhe	277011

- ...
- b) Modify the event handler of the previous part of the exercise to output the following for each country with at least 10 valid city population entries:
  - the country's name,
  - the overall number of cities,
  - each city with name and most recent population count,
  - the average city population,

- inside the city table, mark (either by color, font etc) (1) the capital city, and (2) the city whose population is closest to the average city population of the country.
- c) Modify the event handler from Exercise 3.2, Part 2 as follows:
  - stop after the population number of Germany's capital has been printed;
  - add appropriate additional output to show that the process stopped there;
  - instead of printing Germany's population number, print the whole contents of the first argument of the respective characters(char[], int, int) call. Explain the result.
- d) Playing with streams: implement a pipe such that the event handler from Exercise 3.2, Part 5 creates results of the form

```
<result>
 <organization name="European Union"/>
 <city name="Brussels"/>
 </result>
```

that are written into another XMLOutputStream. Create a second thread that reads from this stream and filters (via StAX) only the <city> elements. Let both threads log to System.out to show the concurrent processing.

# The following exercises are based on a Usecase Scenario: Some province becomes independent and forms a new country

The second group of exercises implement a "realistic" use case for an update to Mondial.

Consider the case that Catalonia leaves Spain. Then, some updates have to be executed on Mondial:

- plan first your strategy, before starting to program,
- consider especially, what kinds of data items must be changed, and how,
- write the programs as generic as possible (such that they can be applied also whenever some other province turns into a new, independent country).
- Update mondial.xml appropriately, especially:
- take as much data as possible from Mondial,
- Catalonia becomes a member of all organizations where Spain is a member.
- Ignore the airport elements at the end of Mondial.
- The below fragment contains all necessary new facts about Catalonia [Filename: catdata.xml]:

```
<country car_code='CAT'>
<name>Catalonia</name>
<population_growth>0.8</population_growth>
<infant_mortality>2.5</infant_mortality>
<gdp_total>204189</gdp_total>
<gdp_agri>3</gdp_agri>
<gdp_ind>37</gdp_ind>
 <gdp_serv>60</gdp_serv>
<inflation>1.5</inflation>
<unemployment>12.6</unemployment>
<indep_date from="E">2018-04-01</indep_date>
<ethnicgroup percentage="100">Mediterranean Nordic</ethnicgroup>
<religion percentage="52.4">Roman Catholic</religion>
<religion percentage="2.5">Protestant</religion>
<religion percentage="7.3">Muslim</religion>
<religion percentage="1.3">Buddhist</religion>
<religion percentage="1.2">Christian Orthodox</religion>
<language percentage="52">Spanish</language>
```

```
<language percentage="41.5">Catalan</language>
 <language percentage="0.1">Occitan</language>
 <border country="AND" length="65"/>
 <border country="F" length="300"/>
 <border country="E" length="320"/>
 </country>
```

The task should be solved by different approaches:

- JDOM: just updating the existing Mondial data [lengthy Java programming?],
- XSLT: as a high-level transformation [declarative maybe the shortest solution?],
- SAX/StAX: Streaming [the only way for really laaaarge XML data not just hacking, but a strategy is needed],
- JAXB: Mapped to an object-oriented model [and then same as DOM?].

#### Hints:

- For XSLT and SAX/StAX, you have also to deal with the unchanged portions. Handle this with as little effort as possible, using general rules (XSLT templates and SAX/StAX conditions). Note that these two strategies are closely related.
- Focus on the general, structural issues; don't spend too much time for the dirty details of the new values for Spain.
- Our solution in XSLT just contains 10 templates (85 lines) for the general handling, and another 8 templates (80 lines) for dealing with a reasonable computation of the new Spain values.

## Exercise 3.4 (Country Independence in JDOM)

- Read mondial.xml into a JDOM object,
- Update it using the JDOM operations (you can use XPath in the JDOM for searching for values or nodes),
- Write it out into a file,
- Manually, using e.g. xlint: Validate the result file against mondial.dtd.

# Exercise 3.5 (Country Independence in XSLT)

Solve the previous exercise with an XSLT transformation.

## Exercise 3.6 (Country Independence in SAX)

Solve the previous exercise with a SAX transformation, again writing the result out as XML into a file.

Note: as an exercise for *stream processing* consider that the data in general might be extremely large, even an infinite stream:

- write into an appropriate output stream,
- materialize as little data as possible,
- write out unchanged/changed data as soon as possible.

# Exercise 3.7 (Country Independence in StAX)

Solve the previous exercise with a StAX transformation, again writing the result out as XML into a file.

Consider the same stream processing aspects as in the SAX variant of this exercise.