PB138/07 - Modern Markup Languages and Their Applications

Lab 01 [24.02.2020]
Course instructions and Introduction to XML, DOM & JAXP

Bruno Rossi
Department of Computer Systems and Communications,
Lasaris (Lab of Software Architectures and Information Systems)
Masaryk University, Brno

Introduction

- About your lab instructor
- About your previous experience
- Content of the seminars

Evaluation

- Assignments/Tasks solved during each seminar session
  → **20 pts** (2pts x 10 seminars, starting week 3)
- Team project given at half the semester (4 students)
  → **0-40 pts**
- Final Exam
  → **0-40 pts**

- **To pass the course:**
  → **70 pts** (60 with 'zapoucet')

Attendance to seminars in **not compulsory**, but to get the points is necessary to submit the **completed task for each week**

Structure of the Seminars

- During the semester:
  - → each week an incremental task to be completed
- Second part:
  - → starting with the development of the project to be delivered

Introduction to XML, DOM & JAXP

XML = eXtensible Markup Language
DOM = Document Object Model
JAXP = Java API for XML Processing

XML

- XML is a mark-up language created to store & transport information in a structured form

```
<?xml version="1.0" encoding="utf-8"?>
<company></company>
```

What is the difference between **well-formed** and **valid** XML?
Can a **'non-well-formed'** XML be **valid**?
Can a **'non-valid'** XML be **well-formed**?

XML well-formedness

- Try yourself: create 6 files

```
<?xml version="1.0" encoding="utf-8"?>
<company></company>
```

01.xml

```
<?xml version="1.0" encoding="utf-8"?>
<company></Company>
```

02.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

03.xml

```
<?xml version="1.0" encoding="utf-8"?>
<company id="10<5"></company>
```

04.xml

```
<?xml version="1.0" encoding="utf-8"?>
<company id="10&lt;5"></company>
```

05.xml

```
<?xml version="1.0" encoding="utf-8"?>
<company>B&N</company>
```

06.xml

Before running `xmllint`, think about which should not pass the well-formedness

What happens if you add the `--valid` flag to `xmllint`?

- At the prompt

```
$> xmllint --noout *.xml
```

(if you have only those xml files in the dir)

- Try to fix the errors and re-run `xmllint` (see also next slide)

XML Escaping

"	&quot;
'	&apos;
<	&lt;
>	&gt;
&	&amp;

- `<![CDATA[ ]]>`
  → Cannot be used in attributes

- Try rewriting the following by proper escaping

```
<?xml version="1.0" encoding="utf-8"?>
<company>B&N</company>
```

Playing with encoding

- Download the file `01-enc.zip`
- Uncompress the files in a directory
- You should have two files: `enc.xml` and `enc-prob.xml`
- Try to run `xmllint` on both
- Running on `enc-prob.xml` should give you the following error:

```
enc-prob.xml:2: parser error : Input is not proper UTF-8,
indicate encoding !
Bytes: 0xE0 0x3E 0x0A 0x20
<localit>
```

- Try to fix this issue in the file (hint: you can either use a text editor or `iconv`)

Playing with encodings - hints

- In a Linux terminal, you can look at the file encoding

```
$> file --mime enc.xml
enc.xml: application/xml; charset=utf-8

$> file --mime enc-prob.xml
enc-prob.xml: application/xml; charset=iso-8859-1
```

- You can look at the “bytes dump” for the two files

```
$> hexdump enc.xml
0000000 3f3c 6d78 206c 6576 7372 6f69 3d6e 3122
0000010 302e 2022 6e65 6f63 6964 676e 223d 7475
0000020 2d66 2238 3e3f 3c0a 6f6c 6163 696c c374
0000030 3ea0 200a 6f52 656d 3c0a 6c2f 636f 6c61
0000040 7469 a0c3 003e

$> file --mime enc-prob.xml
0000000 3f3c 6d78 206c 6576 7372 6f69 3d6e 3122
0000010 302e 2022 6e65 6f63 6964 676e 223d 7475
0000020 2d66 2238 3e3f 3c0a 6f6c 6163 696c e074
0000030 0a3e 5620 6e65 6369 0a65 2f3c 6f6c 6163
0000040 696c e074 003e
```

Create a new XML file

- Create a new XML file called `continent.xml`
- Represent several continents in the file (e.g. Asia, Africa, America, Europe, Australia). Each continent has a **name** attribute
- Each continent contains one or more **cities**
- Each **city** has an **attribute: id** and **sub-elements: name, population**, and **pollution** (that can be either 'low','medium','high')
- Add several continents and cities to the file
- Check with `xmllint` the well-formedness of the xml file

- Hint: you can use `xmllint` to output a nicely formatted xml file from the command line:

```
$> cat continent.xml | xmllint --format -
```

Using JAXP (1/2)

- We will start familiarizing with JAXP in this exercise (will continue next time)
- In this lab we are interested about the DOM API:
  - org.w3c.dom
  - javax.xml.parsers

- Check the following documentation (so it will be familiar for the next lab session):

→ http://www.oracle.com/technetwork/java/intro-140052.html (intro to JAXP)
→ http://www.oracle.com/technetwork/java/dom-139036.html (DOM API)
→ https://docs.oracle.com/javase/8/docs/api/org/w3c/dom/package-summary.html(org.w3c.dom)

Using JAXP (2/2)

- Use the sample project (`01-02-xml-ex.zip`) to familiarize with the API
- Just consider the class '`XML`'
- You can modify the method `doMyXMLTransformations(org.w3c.dom.Document)`
- In the `src/` folder, there is already a `continent.xml` file with some data that you can use
- Try to run the application

Task 1: Output Element Names

- In `doMyXMLTransformations(Document document)`
  1. Get the root element
  2. Print the root element name (`'world'` in the sample file)
  3. Get all the continents in the file (you can use `getElementsByTagName() )`
  4. Iterate over the continents and print the names (`name` attribute) – output based on the sample should be (`asia, africa, europe, america`)

Task 1: Output Element Names

HINTS

- Use `getDocumentElement()` to get the root as a Node
- Cast the Node to `Element`
- Iterate over a `NodeList` by using `getElementsByTagName()`
- Use `getAttribute("name")` to print the attribute ‘name’

References

Suggested material:

- Extensible Markup Language (XML) 1.0 W3C Recommendation (annotated):
  → https://www.xml.com/axml/testaxml.htm
- Introduction to JAXP from Oracle docs:
  → http://www.oracle.com/technetwork/java/intro-140052.html(chapt1)
  → http://www.oracle.com/technetwork/java/dom-139036.html(chapt3)