

COMPS203F

Topic 07: XML, JSON and Maven

Kelvin Lee

XML

- XML (eXtensible Markup Language)
- markup language for documents, similar to HTML with pairs of tags, but
 - meaning of tags not predefined
 - user can use their own tags (e.g., <phone>, </phone>)
- designed for
 - data storage/transmission
 - machine-readable, and also human-readable
- reasons to study:
 - used in many documents/systems (e.g., Word .docx files)
 - you may need to processing it

3/29/2019

2

Example

- Student information (file: university.xml)

```
<university>
  <student>
    <name>Ben Wong</name>
    <phone>99887766</phone>
  </student>
  <student>
    <name>Cat Lee</name>
    <phone>99112233</phone>
  </student>
</university>
```
- It has a tree-like structure with a root (draw diagram)

3/29/2019

3

More Details

- An XML file can **optionally** have an XML prolog as in its first line (and no space before "<?")

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- this is a comment -->
<university>
  <student>
    <name>Ben Wong</name>
    <phone>99887766</phone>
  </student>
</university>
```
- Comment: same as HTML
- Case sensitive
- 5 reserved characters (<, >, &, ', "): need to use "<", ">", "&", "'", and """ instead

3/29/2019

4

Processing Using Java

- We use JDOM (Java Document Object Model), an open source Java library
- Whole document is stored in memory
- needs the jar file (e.g., jdom.jar) in your classpath, please replace "jdom.jar" with your actual file name
- In Crimson, add "-cp jdom.jar" in "Tools > Conf. User Tools > Argument" part
- Web site: <http://www.jdom.org>
- Other alternatives: SAX, DOM, StAX, ...

3/29/2019

5

Useful classes

- Useful classes Document and Element

```
import org.jdom2.*;

// To create an XML document xmlDoc with root element university:
Element root = new Element("university");
Document xmlDoc = new Document(root);
<university></university>

// To add an element student under root:
Element student = new Element("student");
root.addContent(student);
<university>
  <student>
</student>
</university>
```

3/29/2019

6

Useful classes

- To add element and data to student :

```
student.addContent(new Element("name")
    .setText("Ben Wong"));

<university>
  <student>
    <name>Ben Wong</name>
  </student>
</university>

student.addContent(new Element("phone")
    .setText("99887766"));
...
<phone>99887766</phone>
...
```

3/29/2019

7

Exercise

- Create the following XML document

```
<shop>
  <item>
    <name>Apple</name>
    <price>5.5</price>
  </item>
</shop>
```

3/29/2019

8

Create XML class

```
import org.jdom2.*;
import org.jdom2.output.*;

public class CreateXML {
    public static void main(String[] args) {
        try {
            // for printing in pretty format
            XMLOutputter xmlOutput = new XMLOutputter();
            Format format = Format.getPrettyFormat();
            format.setExpandEmptyElements(true);
            xmlOutput.setFormat(format);

            Element root = new Element("university");
            Document xmlDoc = new Document(root);
            xmlOutput.output(xmlDoc, System.out);
        }
    }
}
```

3/29/2019

9

Create XML class

```
Element student = new Element("student");
root.addContent(student);
xmlOutput.output(xmlDoc, System.out);

student.addContent(new Element("name")
    .setText("Ben Wong"));
student.addContent(new Element("phone")
    .setText("99887766"));
xmlOutput.output(xmlDoc, System.out);
} catch (IOException e) {
    e.printStackTrace();
}
}
```

3/29/2019

10

Output

```
<?xml version="1.0" encoding="UTF-8"?>
<university></university>
<?xml version="1.0" encoding="UTF-8"?>
<university>
  <student></student>
</university>
<?xml version="1.0" encoding="UTF-8"?>
<university>
  <student>
    <name>Ben Wong</name>
    <phone>99887766</phone>
  </student>
</university>
```

3/29/2019

11

Useful Methods

- the class Element representing a node in the document tree

returnType	methodName(parameters)	descriptions
	Element(String tag)	create an element (node) containing <tag></tag>

12

Processing XML

```
import java.io.*;
import java.util.*;
import org.jdom2.*;
import org.jdom2.input.*;
import org.jdom2.output.*;

public class ProcessXML {
    private Document document;
    private List<Element> studentList;

    public List<Element> getElementList() {
        return elementList;
    }
    public void setElementList(List<Element> elementList) {
        this.elementList = elementList;
    }
    // put other methods here
}
```

3/29/2019

13

Importing and Printing XML

```
public List<Element> importXML(String filename)
    throws Exception {
    SAXBuilder saxBuilder = new SAXBuilder();
    File inFile = new File(filename);
    document = saxBuilder.build(inFile);
    Element rootElement = document.getRootElement();
    elementList = rootElement.getChildren();
    return elementList;
}
public void printList() throws Exception {
    for (Element element : elementList) {
        System.out.println(" Name: "
            + element.getChild("name").getText());
        System.out.println(" Phone: "
            + element.getChild("phone").getText());
    }
}
```

3/29/2019

14

Finding and Changing

```
public String findPhone(String name) {
    for (Element element : elementList) {
        if (element.getChild("name").getText().equals(name))
            return element.getChild("phone").getText();
    }
    return null;
}
public boolean changePhone(String name, String newPhone) {
    for (Element element : elementList) {
        if (element.getChild("name").getText().equals(name)) {
            element.getChild("phone").setText(newPhone);
            return true;
        }
    }
    return false;
}
```

3/29/2019

15

Deleting

```
public boolean deleteStudent(String name) {
    for (Element element : elementList) {
        if (element.getChild("name").getText().equals(name)) {
            elementList.remove(element);
            return true;
        }
    }
    return false;
}
```

3/29/2019

16

Adding

```
public List<Element> addStudent(String name, String phone) {
    Element element = new Element("student");
    element.addContent(new Element("name").setText(name));
    element.addContent(new Element("phone").setText(phone));
    elementList.add(element);
    return elementList;
}
```

3/29/2019

17

Exporting

```
//import org.jdom2.output.*;

public void exportXML(String filename) throws Exception {
    PrintStream outStream = new PrintStream(
        new File(filename));
    XMLOutputter xmlOutput = new XMLOutputter();
    xmlOutput.setFormat(Format.getPrettyFormat());
    xmlOutput.output(document, outStream);
    outStream.close();
}
```

3/29/2019

18

Full Testing Code

```
public static void main(String[] args) {
    ProcessXML xml = new ProcessXML();
    try {
        String xmlFile = "university.xml";
        System.out.println("import xml file: "+xmlFile);
        xml.importXML(xmlFile);
        System.out.println("print element list...");
        xml.printList();
        String aName = "Cat Lee";
        System.out.println("find phone number of "+aName);
        System.out.println("  "+aName + "'s Phone no.: "
            + xml.findPhone(aName));
        System.out.println("change phone number of Cat...");
        xml.changePhone(aName, "66778899");
        xml.printList();
    }
}
```

3/29/2019

19

Full Testing Code

```
System.out.println("delete student Ben...");
xml.deleteStudent("Ben Wong");
xml.printList();
System.out.println("add student Dick...");
xml.addStudent("Dick Chan", "99001122");
xml.printList();
String outFile = "out.xml";
System.out.println("export to XML file: "+outFile);
xml.exportXML(outFile);
//xml.exportXML(xmlFile);
} catch(JDOMException e) {
    System.out.println("JDOM exception: "+e.getMessage());
    e.printStackTrace();
} catch(Exception e) {
    System.out.println(e.getMessage());
}
}
```

3/29/2019

20

Output

```
import xml file: university.xml
print element list...
Name: Ben Wong
Phone: 99887766
Name: Cat Lee
Phone: 99112233
find phone number of Cat Lee
Cat Lee's Phone no.: 99112233
change phone number of Cat...
Name: Ben Wong
Phone: 99887766
Name: Cat Lee
Phone: 66778899
```

3/29/2019

21

Output

```
delete student Ben...
Name: Cat Lee
Phone: 66778899
add student Dick...
Name: Cat Lee
Phone: 66778899
Name: Dick Chan
Phone: 99001122
export to XML file: out.xml
```

3/29/2019

22

Exercise

- Write a method `findName(String phone)` to return the name of the student having phone number `phone`

3/29/2019

23

More...

- Each element of XML can have attributes
- they are usually for storing meta-data
- If student has an attribute "id", the following statement can get its value:
`student.getAttribute("id");`
- See
https://www.tutorialspoint.com/java_xml/java_jdom_parse_document.htm

3/29/2019

24

JSON

- JSON (JavaScript Object Notation)
- contains text written in certain format
- for data storage and exchange
- light-weight
- both human and machine readable
- mainly use to transmit information between server and Web applications
- needs a jar file, get it at
<http://central.maven.org/maven2/org/json/json/20180813/json-20180813.jar>

3/29/2019

25

JSON Data Types

- JSON has six data types
- string: "JSON", "new line\n"
- number: 12, 3.14
- boolean: true, false
- null: null
- array (in square brackets): [5, 6, 7], [8, "nine", true]
- object (in curly brackets): {"id": 23, "name": "John", "hobbies": ["cycling", "reading"]}

3/29/2019

26

XML Example Recall

- Student information (file: university.xml)

```
<university>
  <student>
    <name>Ben Wong</name>
    <phone>99887766</phone>
  </student>
  <student>
    <name>Cat Lee</name>
    <phone>99112233</phone>
  </student>
</university>
```

3/29/2019

27

Example

- To write the same data in JSON (file: university.json)

```
{ "university" : {
  "student": [
    { "name": "Ben Wong",
      "phone": "99887766"
    },
    { "name": "Cat Lee",
      "phone": "99112233"
    }
  ]
}
```

- 2 key-value pairs of data form a student object

3/29/2019

28

Processing Using Java

- needs a jar file (e.g., json.jar) in your classpath, please replace "json.jar" with your actual file name
- In Crimson, add "-cp json.jar" in "Tools > Conf. User Tools > Argument" part
- Web site (org.json):
<http://github.com/stleary/JSON-java>
- jar file:
<http://central.maven.org/maven2/org/json/json/20180813/json-20180813.jar>
- An alternative: org.json.simple

3/29/2019

29

Useful classes

- Useful classes JSONObject and JSONArray
- ```
import org.json.*;

// To create a JSONArray and add a JSONObject with data:
JSONArray studentArray = new JSONArray();
studentArray.put(new JSONObject()
 .put("name", "Ben Wong")
 .put("phone", "99887766"));

[{"name": "Ben Wong", "phone": "99887766"}]

studentArray.put(new JSONObject()
 .put("name", "Cat Lee")
 .put("phone", "99112233"));

[{"name": "Ben Wong", "phone": "99887766"},
 {"name": "Cat Lee", "phone": "99112233"}]
```

3/29/2019

30

## Useful classes

- To add the array to a JSONObject :

```
JSONObject student = new JSONObject()
 .put("student", studentArray);
{student:[{"name": "Ben Wong","phone": "99887766"},
 {"name": "Cat Lee","phone": "99112233"}]
}
```

- To add the array to another JSONObject :

```
JSONObject university = new JSONObject()
 .put("university", student);
{university:
 {student:[{"name": "Ben Wong","phone": "99887766"},
 {"name": "Cat Lee","phone": "99112233"}]
 }
}
```

3/29/2019

31

## Exercise

- Create the following JSONObject

```
{ "shop":
 { "item":
 [{ "name": "Apple",
 "price": 5.5 }]
 }
}
```

3/29/2019

32

## Create JSON class

```
import org.json.*;

public class CreateJSON {
 public static void main(String[] args) {
 JSONArray studentArray = new JSONArray();
 studentArray.put(new JSONObject()
 .put("name", "Ben Wong")
 .put("phone", "99887766"));
 System.out.println(studentArray);
 studentArray.put(new JSONObject()
 .put("name", "Cat Lee")
 .put("phone", "99112233"));
 System.out.println(studentArray);
 JSONObject student = new JSONObject().put("student", studentArray);
 System.out.println(student);
 JSONObject json = new JSONObject().put("university", student);
 System.out.println(json);
 }
}
```

3/29/2019

33

## Output

```
[{"phone": "99887766", "name": "Ben Wong"}]
[{"phone": "99887766", "name": "Ben
Wong"}, {"phone": "99112233", "name": "Cat Lee"}]
{"student": [{"phone": "99887766", "name": "Ben
Wong"}, {"phone": "99112233", "name": "Cat Lee"}]}
{"university": {"student": [{"phone": "99887766", "name": "
Ben Wong"}, {"phone": "99112233", "name": "Cat Lee"}]}}
```

3/29/2019

34

## Useful Methods

- ...

| returnType | methodName(parameters) | descriptions |
|------------|------------------------|--------------|
|            |                        |              |
|            |                        |              |
|            |                        |              |
|            |                        |              |
|            |                        |              |
|            |                        |              |
|            |                        |              |

35

## Processing JSON

```
import java.io.*;
import java.util.*;
import org.json.*;

public class ProcessJSON {
 private JSONArray jsonArray;

 public JSONArray getJsonArray() { return jsonArray; }
 public void setJsonArray(JSONArray jsonArray) {
 this.jsonArray = jsonArray;
 }

 // put other methods here
}
```

3/29/2019

36

## Importing JSON

```
public JSONObject importJSON(String filename) throws Exception {
 try {
 String content = readFile(filename);
 if (content == null) return null;
 return new JSONObject(content);
 } catch (JSONException e) {
 System.out.println("Import JSON problem: "
 + e.getMessage());
 }
 return null;
}
```

3/29/2019

37

## Importing JSON

```
public String readFile(String filename) {
 String content = "";
 try {
 BufferedReader in = new BufferedReader(
 new FileReader(filename));

 String line;
 while ((line = in.readLine()) != null)
 content += line;
 in.close();
 } catch (IOException e) {
 System.out.println("File read problem: " + e.getMessage());
 return null;
 }
 return content;
}
```

3/29/2019

38

## Printing

```
public void printArray() {
 JSONObject element;
 for (int i=0; i<jsonArray.length(); i++) {
 element = (JSONObject) jsonArray.get(i);
 System.out.println(" name: " + element.get("name"));
 System.out.println(" phone: " + element.get("phone"));
 }
}
```

3/29/2019

39

## Finding

```
public String findPhone(String name) {
 JSONObject element;
 for (int i=0; i<jsonArray.length(); i++) {
 element = (JSONObject) jsonArray.get(i);
 if (element.get("name").equals(name))
 return (String) element.get("phone");
 }
 return null;
}
```

3/29/2019

40

## Changing

```
public boolean changePhone(String name,
 String newPhone) {
 JSONObject element;
 for (int i=0; i<jsonArray.length(); i++) {
 element = (JSONObject) jsonArray.get(i);
 if (element.get("name").equals(name)) {
 element.put("phone", newPhone);
 return true;
 }
 }
 return false;
}
```

3/29/2019

41

## Deleting

```
public JSONObject deleteStudent(String name) {
 JSONObject element;
 for (int i=0; i<jsonArray.length(); i++) {
 element = (JSONObject) jsonArray.get(i);
 if (element.get("name").equals(name)) {
 return (JSONObject) jsonArray.remove(i);
 }
 }
 return null;
}
```

3/29/2019

42

## Adding

```
public JSONArray addStudent(String name,
 String phone) {
 return jsonArray.put(new JSONObject()
 .put("name", name)
 .put("phone", phone)
);
}
```

3/29/2019

43

## Exporting

```
public JSONObject exportJSON(String filename) {
 try {
 PrintWriter out = new PrintWriter(new BufferedWriter(
 new FileWriter(filename)));
 JSONObject jsonObject = new JSONObject()
 .put("university", new JSONObject()
 .put("student", jsonArray));
 out.println(jsonObject);
 out.close();
 return jsonObject;
 } catch (IOException e) {
 System.out.println("Export JSON problem: "
 + e.getMessage());
 }
 return null;
}
```

3/29/2019

44

## Full Testing Code

```
public static void main(String[] args) {
 ProcessJSON json = new ProcessJSON();
 String jsonFile = "university.json";
 System.out.println("import json file: "+jsonFile);
 JSONObject jsonObject = null;
 try {
 jsonObject = json.importJSON(jsonFile);
 } catch (Exception e) {
 System.out.println(e.getMessage());
 }
 System.out.println("All: \n " +jsonObject);
 JSONObject university = (JSONObject)
 jsonObject.get("university");
 System.out.println("university: \n " +university);
 JSONArray student = (JSONArray) university.get("student");
 json.setJSONArray(student);
 json.printArray();
}
```

3/29/2019

45

## Full Testing Code

```
String aName = "Cat Lee";
System.out.println("find phone number of "+aName);
System.out.println(" " +aName + "'s Phone no.: "
 + json.findPhone(aName));
System.out.println("change phone number of Cat...");
json.changePhone(aName, "66778899");
json.printArray();
System.out.println("delete student Ben: "
 +json.deleteStudent("Ben Wong"));
json.printArray();
System.out.println("add student Dick...");
json.addStudent("Dick Chan", "99001122");
json.printArray();
String newJsonFile = "out.json";
System.out.println("exporting to "+newJsonFile);
System.out.println(" " +json.exportJSON(newJsonFile));
}
```

3/29/2019

46

## Output

```
import json file: university.json
All:
{"university":{"student":[{"phone":"99887766","name":"Ben Wong"}, {"phone":"99112233","name":"Cat Lee"}]}}
university:
{"student":[{"phone":"99887766","name":"Ben Wong"}, {"phone":"99112233","name":"Cat Lee"}]}
name: Ben Wong
phone: 99887766
name: Cat Lee
phone: 99112233
```

3/29/2019

47

## Output

```
find phone number of Cat Lee
Cat Lee's Phone no.: 99112233
change phone number of Cat...
name: Ben Wong
phone: 99887766
name: Cat Lee
phone: 66778899
delete student Ben: {"phone":"99887766","name":"Ben Wong"}
name: Cat Lee
phone: 66778899
add student Dick...
name: Cat Lee
phone: 66778899
name: Dick Chan
phone: 99001122
exporting to out.json
{"university":{"student":[{"phone":"66778899","name":"Cat Lee"}, {"phone":"99001122","name":"Dick Chan"}]}}
```

3/29/2019

48



## Exercise

- Write a method `findName(String phone)` to return the name of the student having phone number `phone`

3/29/2019

49

## Maven

- Maven
  - from Apache
  - project management tool in Java
- can automate project build and other tasks
- project details written in XML file `pom.xml` (Project Object Model)
- by default, creates sensible project structure automatically

3/29/2019

50

## Maven

- Download
  - <https://maven.apache.org/download.cgi>
  - needs JDK 1.7 or above (use **JDK 1.8** to avoid problems!)
  - install instructions: <https://maven.apache.org/install.html>
- check installation successful:

```
D:\>mvn -v
D:\
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5fffb20918da4f719f3; 2018-10-25T02:41:7+08:00)
Maven home: C:\Program Files\Java\apache-maven-3.6.0\bin\..
Java version: 1.8.0_65, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_65\jre
Default locale: en_US, platform encoding: MS950_HKSCS
OS name: "windows 7", version: "6.1", arch: "amd64", family: "windows"
3/29/2019
```

51

## Maven Repositories

- directory storing Maven jars and other things
- search order
  - local (created when first run, `C:\Users\<userName>\.m2`)
  - central (if found, downloaded to local)
  - remote (optional)

3/29/2019

52

## OpenIMAJ Installation

- Set of Java libraries for image analysis
- Create OpenIMAJ project in folder `openimaj-01`:

```
D:\> mvn -DarchetypeGroupId=org.openimaj -DarchetypeArtifactId=openimaj-quickstart-archetype -DarchetypeVersion=1.3.8 archetype:generate
```

```
Downloading...
...
Define value for property 'groupId': s358f
Define value for property 'artifactId': openimaj-01
[INFO] Using property: version = 1.0-SNAPSHOT
Define value for property 'package' s358f: : <enter key>
[INFO] Using property: openimajVersion = 1.3.8
```

3/29/2019

53

## OpenIMAJ

Confirm properties configuration:

```
groupId: s358f
artifactId: openimaj-01
version: 1.0-SNAPSHOT
package: s358f
openimajVersion: 1.3.8
Y: : y
```

3/29/2019

54

## OpenIMAJ

```
[INFO] Parameter: groupId, Value: s358f
[INFO] Parameter: artifactId, Value: openimaj-01
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: package, Value: s358f
[INFO] Parameter: packageInPathFormat, Value: s358f
[INFO] Parameter: package, Value: s358f
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: groupId, Value: s358f
[INFO] Parameter: openimajVersion, Value: 1.3.8
[INFO] Parameter: artifactId, Value: openimaj-01
[INFO] Project created from Archetype in dir: D:\openimaj-01
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
```

3/29/2019

55

## (Further installation and) Compile

- A folder openimaj-01 has been created
- Type the commands to **compile** and wait for it to finish (may take 10-50 minutes):

```
D:\> cd openimaj-01
D:\openimaj-01> mvn assembly:assembly
```

Downloading...

```
...
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 37:27 min
[INFO] Finished at: 2019-03-25T17:22:58+08:00
[INFO] -----
3/29/2019
```

56

## OpenIMAJ

- Type the command to **execute** and the window below will be displayed:

```
D:\openimaj-01> java -jar target\openimaj-01-1.0-SNAPSHOT-jar-with-dependencies.jar
```



3/29/2019

57

## Source Code

```
// file: D:\openimaj-01\src\main\java\s358f\App.java
// package and import statements not shown
public class App {
 public static void main(String[] args) {
 //Create an image [with size 320 x 70]
 MBFImage image = new MBFImage(320,70, ColourSpace.RGB);

 //Fill the image with white
 image.fill(RGBColour.WHITE);

 //Render some test into the image
 image.drawText("Hello World", 10, 60,
 HersheyFont.CURSIVE, 50, RGBColour.BLACK);

 //Apply a Gaussian blur
 image.processInplace(new FGaussianConvolve(2f));

 //Display the image
 DisplayUtilities.display(image);
 }
}
3/29/2019
```

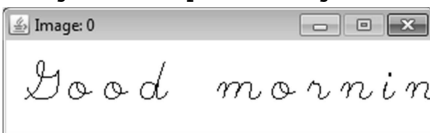
58

## Exercise

- Edit the source code with Crimson and change the text to "Good morning" and without any blur.

- Type the command and it takes around 2 minutes

```
D:\openimaj-01> mvn assembly:assembly
D:\openimaj-01> java -jar target\openimaj-01-1.0-SNAPSHOT-jar-with-dependencies.jar
```



- Try to fix any issue of missing character.

3/29/2019

59

## Exercise

- To be added

3/29/2019

60