CO3102/CO7102 Mobile and Web Applications
Coursework 1

XML and JSON

Important Dates:

Handed out: 12-Oct-2020
Deadline: 3-Nov-2020 at 16:59 GMT

Please ensure that you submit your work on time.

- This coursework counts as 10% of your final mark.
- Please read guidelines on plagiarism in the study guide and course documentation.
- This coursework requires knowledge about XML Schema, XSLT, JSON and Document Parsing.
- Learning outcome:
  - Use data-interchange formats (XML and JSON) and techniques appropriately to create documents and handle data.

Data Description

Consider an XML document (**WebAppInterface.xml**):

```
<?xml version="1.0" encoding="UTF-8"?>
<interface id="RESTController">
          <package>uk.ac.le.cs.wt</package>
           <extends>
               <from>Remote</from>
           </extends>
           <import>java.rmi.Remote</import>
           <import>java.rmi.RemoteException</import>
           <import>java.net.*</import>
           <abstract_method name="authenticateUser">
               <access_level>public</access_level>
               <arguments>
                   <parameter type="String">user</parameter>
                   <parameter type="String">password</parameter>
               </arguments>
               <throws>
                   <exception>RemoteException</exception>
                   <exception>SecurityException</exception>
               </throws>
               <return>boolean</return>
           </abstract_method>
           <abstract_method name="activateUser">
               <access_level>public</access_level>
               <arguments>
                   <parameter type="URL">link</parameter>
               </arguments>
               <return>void</return>
           </abstract_method>
</interface>
```

This XML document describes a Java interface below:

```
package uk.ac.le.cs.wt;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.net.*;

interface RESTController extends Remote{

public boolean authenticateUser (String user, String password)
          throws RemoteException, SecurityException;
public void activateUser(URL link);
}
```

Tasks:

Task 1: [40 Marks]

Write an XML Schema (**WebAppInterface.xsd**) that allows validation of the provided XML document. Note that the schema must be consistent with these rules in Java:

- An interface **may** extend **one or many** interfaces.
- There can be **only one** package statement
- There can be **zero to many** import statements
- An interface can have **zero to many** abstract methods
- A method only allows **one** value to be returned (or **void**)
- A method can only have **zero or one** visibility modifier *
- A method can accept **zero to many** arguments
- Exception(s) **may** be thrown from a method

* If a method has no visibility modifier then it is accessible only within its own package (default visibility); assuming all methods are non-static.

(Hint: Related lab exercise – Week 1)

Task 2: [30 Marks]

Write an XSLT stylesheet (**WebAppInterface.xslt**) that takes the provided XML document as input and generates an HTML document as follows:

RESTController

Operation	Argument(s)	Return
authenticateUser	user: String, password: String	boolean
activateUser	link: URL	void

Note: You may show the result in a different page layout or colour scheme. The page must display the controller id, return type and the parameter list of each method. (Hint: Related lab exercise – Week 2/3)

Task 3: [30 Marks]

Write a program that extract some information from the XML above and generate the JSON output as follows. Please choose an appropriate parsing technology.

```
{
   "abstract_method": [
      {
         "name": "authenticateUser",
         "access_level": "public",
         "arguments": [
            {
               "type": "String",
               "variable": "user"
            },
            {
               "type": "String",
               "variable": "password"
            }
         ],
         "throws": [
            "RemoteException",
            "SecurityException"
         ],
         "return": "boolean"
      },
      {
         "name": "activateUser",
         "access_level": "public",
         "arguments": {
            "parameter": {
               "type": "URL",
               "variable": "link"
            }
         },
         "return": "void"
      }
   ]
}
```

Note: A Java template (**WebAppInterfaceParser.java,** and you will need xercesImpl.jar and xml-apis.jar which are provided on Blackboard) is provided for Task 3. However, you are allowed to use any programming languages or parsers for this task as you wish. If so, please include a text file README.txt explaining how to run your code.

Note that this JSON output was deliberately formatted in this way to prevent people from using off-the-shelf XMLtoJSON function provided by certain APIs. If there is only one parameter in the method then it should be structured as an object (e.g. parameter: {}) while if there are multiple parameters, you should print it as an JSON array using [].

(Note: You are free to use any parsers, but the implementation of the parsing technique **must be your work**. The program must generate the JSON output specified above (e.g. must not contain "package", "import"), which means you are not allowed not use any off-the-shelf XML-to-JSON libraries for the conversion e.g. toJSONObject)

Submission

- Zip all files in a single zip file for submission.
  - **WebAppInterface**.xsd
  - **WebAppInterface**.xslt
  - **WebAppInterfaceParser.java** (or in your language of choice)
- The archive should be named **CO3102_CW1_email_id.zip** or **CO7102_CW1_email_id.zip** (e.g. CO3102_CW1_yh37.zip).

Your submission should also include a completed coursework plagiarism coversheet (print and signed PDF or image). You need to submit the zip file via Blackboard and you are allowed to re-submit as many times as you like **before** the deadline. Marks for any coursework which does not have the accompanying cover sheet will be withheld till you provide one.

Marking Scheme

Note: In addition to the sample XML provided in Table 1, additional XML documents will be included in the test suite.

Part 1: XML Schema validity checking will be automated. In total, a test suite consisting of 12 XML documents will be used for the assessment. The marks will be distributed according to test results.

Part 2: XSLT will be compiled and applied to 7 test cases (XML documents) for the assessment. All HTML pages will be checked by TA(s) manually. The marks will be distributed according to test results.

Part 3: In total, 7 test cases (XML documents) will be used the assessment. Any JSON generated by your XML Parser will be checked by an JSON validator. The marks will be distributed according to test results.

Anonymous marking

We operate an anonymous marking scheme. All submitted files will be renamed using anonymous fingerprinting generated by SHA256.