CO3102/CO7102 Mobile and Web Applications

Coursework 1

Web Data Interchange formats - XML and JSON

Important Dates:

Deadline: 7-Nov-2024 at 12:00 GMT

Please ensure that you submit your work on time.

- This coursework counts as 15% of your final mark.
- Please read the guidelines on plagiarism in the study guide and course documentation.
- This coursework requires knowledge of 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 (**WebServiceProvider.xml**):

```
<?xml version="1.0" encoding="UTF-8"?>
<service id="RESTInterface">
    <package>uk.ac.le.cs.CO3102</package>
    <extends>
       <class>Remote</class>
    </extends>
    <import>java.rmi.Remote</import>
    <import>java.rmi.RemoteException</import>
    <import>java.net.*</import>
    <abstract_method name="authenticateUser">
          <visibility>public</visibility>
          <arguments>
                <parameter type="String">account</parameter>
                <parameter type="String">password</parameter>
                <parameter type="URL">url</parameter>
          </arguments>
          <exceptions>
                <exception>RemoteException</exception>
                <exception>MalformedURLException</exception>
          </exceptions>
          <return>boolean</return>
      </abstract_method>
      <abstract_method name="deleteUser">
         <visibility>private</visibility>
         <arguments>
               <parameter type="String">user</parameter>
         </arguments>
         <return>void</return>
     </abstract_method>
</service>
```

This XML document describes a Java interface below:

```
package uk.ac.le.cs.CO3102;

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

interface RESTInterface extends Remote{

public boolean authenticateUser(String account, String password, URL url)
      throws RemoteException, MalformedURLException;
private void deleteUser (String user);
}
```

Tasks:

Task 1: [40 Marks]

Write an XML Schema (**WebServiceProvider.xsd**) to validate the provided XML document. Note that the schema defined must adhere to the following Java rules:

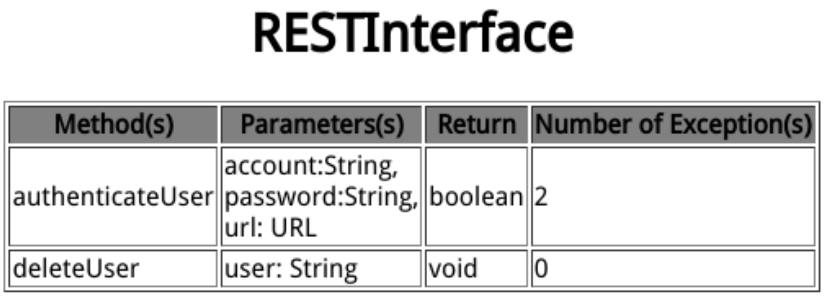
- An interface **may** extend **one or many** interfaces.
- There can be **only one** package statement *
- There can be **zero to many** import statements
- An interface can have **zero to many** abstract methods
- A method only allows **one** value to be returned (or **void**)
- A method can only have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

* Note: If a method does not explicitly have a visibility modifier (e.g., “public”), it is accessible only within its own package (default visibility). This assume all methods are non-static. If the package statement is omitted, the class belongs to the “default” package.

(**Hint**: refer to lab exercise – Week 1)

Task 2: [30 Marks]

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



RESTInterface

Method(s)	Parameters(s)	Return	Number of Exception(s)
authenticateUser	account:String, password:String, url: URL	boolean	2
deleteUser	user: String	void	0

Note: You may show the result in a different page layout or colour scheme. (Hint: Lab Week 2/3)

Task 3: [30 Marks]

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

```
{
  "abstract_method": [
    {
      "method_name": "authenticateUser",
      "visibility": "public",
      "arguments": {
        "parameter": [
          {
            "datatype": "String",
            "label": "account"
          },
          {
            "datatype": "String",
            "label": "password"
          },
          {
            "datatype": "URL",
            "label": "url"
          }
        ]
      },
      "exceptions": {
        "exception": [
          "RemoteException",
          "MalformedURLException"
        ]
      },
      "return": "boolean"
    },
    {
      "method_name": "deleteUser",
      "visibility": "private",
      "arguments": {
        "parameter": [
          {
            "datatype": "String",
            "label": "user"
          }
        ]
      },
      "return": "void"
    }
  ]
}
```

For Task 3, a Java template (**WebServiceParser**.java) is provided, and you'll require xercesImpl.jar available on Blackboard. However, you can opt for alternative programming languages (e.g., Python, C++, C#, JavaScript, etc.) and parsers for this task. In such cases, please provide a README.txt file detailing the execution steps for your code.

It's important to note that the intentional formatting of the JSON output aims to prevent the use of pre-built XML to JSON functions from specific APIs. While you're free to choose parsers, the parsing technique's implementation must be in your code. For example, the program should generate the specified JSON output without including elements like "package," "import," etc. Furthermore, use of any built-in XML-to-JSON libraries for XML to JSON transformation (such as "toJSONObject" in Python etc), is not permitted.

Submission

- Zip all files in a single zip file for submission.
    - **WebServiceInterface**.xsd
    - **WebServiceInterface**.xslt
    - **WebServiceInterface.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 resubmit 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, 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.