Distributed Systems COMP3211 - Coursework 2
Submission Deadline: 10.00, 17 November 2021

1 Introduction

This exercise aims to give you some practical experience of writing server and clients to deploy and consume Web Services. The students should work together in **pairs**. The application to be developed can be implemented in the programming language of your choice.

2 Useful Resources

- Web Services examples: available on COMP3211 area on Minerva `http://minerva.leeds.ac.uk`
- Documentation for Jersey `http://jersey.java.net`
- Documentation for Flask-Restful `https://flask-restful.readthedocs.io/en/latest/`
- Servlets and JSP examples if you choose to develop the full, Web-based user interface to your application.

3 Preparation

Review the material covered in the lectures on SOAs, Web services and REST.

4 The Task

Your main task is to build an application supported by a Web service composition, which consists of:

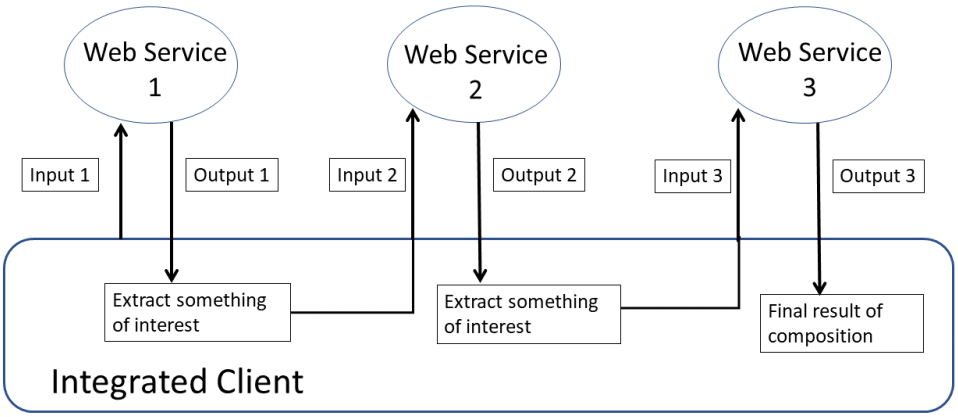
- your own RESTful services which you need to develop, e.g. using Jersey or Flask-RESTful;
- existing RESTful services, e.g. Google, Amazon, Twitter, or any other public directories;
- a combination of the above.

The composition combines THREE Web services, TWO of these must be YOUR own, which you will need to design and implement as a server as well as the associated client to test the server. The third Web service is external. Here is an imaginary example:

Quote of the Day. The invocation of the 1st Web service returns the quote of the day and its author. The name of the author is then used to do a search by invoking the 2nd Web service and output the list of books published by the author. One of the book from the list is then used to do a search for a library where the book is available for purchase following the invocation of the 3rd Web service.

Feel free to investigate the use of Web services provided on:

- `https://developers.google.com/`
- `https://aws.amazon.com/`



- `https://dev.twitter.com/docs/api`
- `http://www.programmableweb.com`
- `http://mymemory.translated.net/doc/spec.php`

As you need to choose your own services, look at the services that you may find interesting, taking note of the implementation platform and examining the REST API for each service.

Note that the order of difficulty varies. Some services are easy to handle because results are returned as simple strings of plain text; others are slightly more challenging because the returned string is XML or JSON rather than plain text (albeit simple XML/JSON), therefore requires parsing.

There are various ways to develop your own services. The steps to follow to write server and client-side code are provided on Minerva. More marks are available if you tackle *original* and *challenging* services.

The work is carried as follows:

1. Student A should be paired with another student B
2. Each student should design and implement ONE Web service
3. The students work together to integrate the external Web service
4. The students work together to integrate and test the Web services composition
5. There is no requirement with regard to the order the Web services are invoked
6. The students submit their individual and joint work together (ONE joint submission only).

Your must report on the following tasks:

Task 1. The description of your OWN Web services: what they do, their type, design, implementation and testing of the server and client.

Task 2. The description of the external Web service: is it RESTful, what is does, the name of the publisher, and how the client to test it is developed.

Task 3. Once you have all services running and invoked through appropriate interfaces, measure the time it takes to invoke these individual services. To get these measurements you are expected to run the experiments n times (e.g. n = 5). A statistical analysis (average, standard deviation) is expected. Discuss your performance results.

Task 4. Implement the integrated client application that combines the THREE Web services and provides an appropriate user interface for the collection of input data and presentation of results. There are two options:

- **Option 1:** Make the application console-based, with input data coming from the command line if required and results being written to a file, possibly as a static page of HTML if the final result contains links.
- **Option 2:** Implement the application as Web based. You may use servlets, JSP, or any other technology as appropriate. If you are following this option, you will need to package your application in the conventional way, as a WAR file. Feel free to consider other frameworks such as Flask, Spring, GWT, Struts or Spark.

5 Submission

- Use Gradescope to submit your answers to the questions through the **Report Submission** link.
- To submit your code for the exercise, you can either: 1) provide a link to Git, or 2) create a Zip or tar archive of the files which make up your system and submit the file through the link **Code Submission** on Minerva. If you have organised your files into a directory hierarchy, then please package this as a single Zip or tar archive. If your application is packaged as a WAR file, please include this file in your submission.
- Produce a short video (maximum 3 minutes long) to demonstrate your integrated client. You can either: 1) upload it on Minerva through the link **Video Submission**, or 2) upload on any cloud platform of your choice, e.g. Youtube and provide the link.

Important: Full instructions on how to install, compile and run your integrated client should be provided as part of your written submission *and* in the file named `README` that you provide along with your code.

6 Distribution of Marks

Composition originality	10
Web Service 1	20
Web Service 2	20
External Web Service	15
Web Services Integration	10
Web user interface	10
Successful execution	10
Coding style / comments	5
	100

Weight: this coursework accounts for 20% of the assessment.
Lateness Penalty: 5% of the maximum available mark per day will be applied.