CS 548—Fall 2015
Enterprise Software Architecture and Design
Assignment Eight—Web Service Clients

In the previous assignments, you developed SOAP and REST Web services for your clinical information system. In this assignment, you will test these services using external clients.

Part 1: SOAP Client

You are provided with a command-line SOAP client in the ClinicSoapWebServiceClient project. To complete this, you should do the following:

1. Deploy your enterprise application, and download the WSDL files for both your patient and provider services from the service endpoints.
2. Copy these WSDL files to the `src/main/resources` folder of the SOAP client project.
3. Edit the POM file for the SOAP client project, and add a line in the JAXWS Maven plugin for each of the WSDL files. There should already be one for the Patient Web service, of the form:
   `<wsdlFile>PatientWebService.wsdl</wsdlFile>`
   Add a line like this for the Provider Web service.
4. Run "Maven install" to generate the jar file for the SOAP client. This should generate artifacts (such as classes for service proxy objects and DTOs) for the SOAP client from the WSDL files[1], and then generate a jar file cs548/ClinicAppSoapClient.jar in your home directory.
5. Run this client (while your service is running), e.g., in Unix by typing
   `java –jar ~/cs548/ClinicAppSoapClient.jar --input` *`inputFile`* `--output` *`outputFile`*

By default, the client will read input lines from standard input and write to standard output, but you can redirect input from a file, and output to a file, using the `--input` and `--output` options. Each input line should be of the form:

`patient` *`command arguments…`*

or

`provider` *`command arguments….`*

The first word on the line is a keyword, designating that this is a command to be executed on the patient Web service or the provider Web service. The second token on the line is the name of the operation to be invoked. The remaining tokens on the line are the arguments to the operations. Tokens are assumed to be separated by blanks, so for example do not place

[1] This needs to run the WSImport goal to generate SOAP artifacts from the WSDL. Eclipse should prompt you to search for an M2E plugin for this goal.

blanks in command arguments, use a period instead. For example, use "Joe.Bloggs" as a patient name rather than "Joe Bloggs".

You will need to edit the command processor to add the provider commands, but most of this is just a matter of copying the commands for patients. Your testing should test all of the commands, but it is okay if you only test for drug treatments. Do your testing with the Web services deployed in EC2 and the client outside the AWS network, opening up your firewall just to allow access to the Web services.

Part 2: REST Client

You should use the `curl` command-line tool to test your application. To test that your REST Web service is properly set up, you can ping it for the clinic name:

```
curl -X GET -D headers \
   http://domain-name:8080/clinic-rest/resources/patient/site
```

This puts the result headers in the file called headers. You can query a patient resource as follows:

```
curl -X GET -D headers \
   http://domain-name:8080/clinic-rest/resources/patient/17
```

This returns an XML representation of the patient resource, of the form:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<patient-rep
       xmlns:ns2="http://cs548.stevens.edu/clinic/service/web/rest/data">
   <ns2:id
       url="http://domain-name:8080/clinic-rest/resources/patient/17"
       relation="http://cs548.stevens.edu/clinic/rest/relations/patient"
       mediaType="application/xml"/>
   <ns2:patient-id>12345678</ns2:patient-id>
   <ns2:name>Joe Blow</ns2:name>
   <ns2:dob>1983-10-04-04:00</ns2:dob>
   <ns2:age>30</ns2:age>
</patient-rep>
```

You can query a treatment for a patient in a similar way:

```
curl -X GET -D headers \
   http://domain-name:8080/clinic-rest/resources/patient/17/treatment/35
```

This returns a treatment representation, e.g. of the form:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<treatment-rep>...</treatment-rep>
```

You can add a patient resource as follows:

```
curl -X POST -D headers -H "Content-Type: application/xml" \
   -d @new-patient.xml \
   http://domain-name:8080/clinic-rest/resources/patient
```

The file `new-patient.xml` contains the representation for the patient that is being added. It is similar to the patient representation above. The response headers in the file headers have the following form:

```
HTTP/1.1 201 Created
X-Powered-By: Servlet/3.1 JSP/2.3 (GlassFish Server Open Source Edition  4.1
Java/Oracle Corporation/1.7)
Server: GlassFish Server Open Source Edition  4.1
Location: http://domain-name:8080/clinic-rest/resources/patient/3
Date: Sun, 16 Mar 2015 22:59:44 GMT
Content-Length: 0
```

Use curl to test the other operations, including the operations for adding providers and treatments, and for querying provider and treatment resources.

Submission

Your solutions should be developed for Java EE 7. You should use Java 8, Glassfish 4.1 and Eclipselink 2.5.x, and Eclipse Mars. In addition, record short flash, mpeg, avi or Quicktime videos of a demonstration of your assignment working (deploy the app and run your testing, showing both input and output for client-side testing). Make sure that your name appears at the beginning of the video. *Do not provide private information such as your email or cwid in the video*. You can upload this video to the folder you are provided with in Google Drive. The video must be uploaded on time for the assignment as a whole to be considered as on time. The time of submission is based on the latest of the time you submitted your code and report to Moodle, and the time you uploaded your videos to Google Drive.

Your solution should be uploaded via the Canvas classroom, as a zip file. This zip file should have the same name as your Canvas userid. It should unzip to a folder with this same name, which should contain the files and subfolders with your submission.

It is important that you provide a document that documents your submission, included as a PDF document in your submission root folder. Name this document README.pdf. As part of your submission, export your Eclipse projects to your file system, and then include those folders as part of your archive file. Your written report should describe the test cases that you used for testing.