CS410J Project 4: A REST-ful Appointment Book Web Service (13 points[1])

In this project you will extend your appointment book application to support an appointment book server that provides REST-ful web services to an appointment book client.

Goals:

- Write a web application in Java
- Work with HTTP-based network communication

For this project you will implement the following classes in the `edu.pdx.cs410J.`*loginid* package:

- An `AppointmentBookServlet` that provides REST access to an `AppointmentBook`. The servlet should be deployed in a web application named `apptbook` and should support the following URLs:
  - `http://host:port/apptbook/appointments?owner=name`
    - `GET` returns all appointments in the appointment book formatted using the `PrettyPrinter`
    - `POST` creates a new appointment from the HTTP request parameters `owner`, `description`, `beginTime`, and `endTime`. If the appointment book does not exist, a new one should be created.
  - `http://host:port/apptbook/appointments?owner=name&beginTime=start&endTime=end`
    - `GET` returns all of given appointment book's appointments that occurred between the `startTime` and the `endTime`.
- Class `Project4` is a client program that sends HTTP requests to the server. Dates and times should be specified using the same format as previous project (AM/PM, not 24-hour).

  If the `-search` option is provided, only the `owner`, `beginTime` and `endTime` are required. The client should pretty print to standard out all of the appointments that begin between those two times.

```
usage: java edu.pdx.cs410J.<login-id>.Project4 [options] <args>
  args are (in this order):
    owner                  The person whose owns the appt book
    description            A description of the appointment
    beginTime              When the appt begins
    endTime                When the appt ends
  options are (options may appear in any order):
    -host hostname         Host computer on which the server runs
    -port port             Port on which the server is listening
    -search                Appointments should be searched for
    -print                 Prints a description of the new appointment
    -README                Prints a README for this project and exits
```

[1] 12 for code, 1 for POA

It is an error to specify a host without a port and vice versa.

The client can perform several functions:

- Add an appointment to the server:

```
$ java edu.---.Project4 -host cs.pdx.edu -port 12345 "Dave" \
  "Teach Java Class" 10/19/2016 6:00pm 10/19/2016 9:30 pm
```

- Search for an appointment between two times. The below command line should pretty-print all appointments that begin on or after November 1st and end on or before November 30th A message should be printed if there is no appointment between those two times.

```
$ java edu.---.Project4 -host cs.pdx.edu -port 12345 -search "Dave" \
  11/01/2016 12:00 am 11/30/2016 11:59 pm
```

Error handling: Your program should exit "gracefully" with a user-friendly error message under all reasonable error conditions. Examples of such conditions include

- The syntax of the command line is invalid
- The format of the day or time is incorrect
- A connection to the server cannot be established

To get you started working with web applications, I put together a Maven archetype that creates a skeleton project[2].

```
mvn archetype:generate \
  -DarchetypeCatalog=https://dl.bintray.com/davidwhitlock/maven/ \
  -DarchetypeGroupId=edu.pdx.cs410J \
  -DarchetypeArtifactId=apptbook-web-archetype \
  -DgroupId=edu.pdx.cs410J.<login> \
  -DartifactId=apptbook \
  -Dversion=1.0-SNAPSHOT
```

(Note that the `artifactId` is `apptbook` which is probably the same as your Project 1 Maven project. Since the name of the artifact is the name of the web application (which is part of the URL), it really needs to be `apptbook`. Rename the directory that you created for Project 1 before creating the Maven project for this assignment.)

The archetype project contains an `AppointmentBookServlet` and a `Project4` class. You can run the servlet with Jetty:

```
$ mvn jetty:run
```

You can run the main class as an "executable" jar:

[2]Note that if you copy and paste this command line into your shell, you might need to replace the tilde ( ) character copied from the PDF with an actual tidle character.

```
$ java -jar target/apptbook.jar
```

The archetype also creates an integration test that drives the `Project4` main class. Since it requires that the server is running, it is run in the "integration test" phase of the Maven build. If you run this command, it will build and start the web container, run all unit tests, and shut the server down again.

```
$ mvn integration-test verify
```

More notes:

- Take a look at the `edu.pdx.cs410J.servlets.FamilyTreeServlet` class for ideas on how you could implement the REST functionality.
- The client uses the `edu.pdx.cs410J.web.HttpRequestHelper` class in the `examples.jar` to make requests on the server.
- You can run your Jetty server on a port other than 8080 by setting the `http.port` variable in Maven:
  - `$ mvn jetty:run -Dhttp.port=8888`
- You only need to submit your source code (`.java` files) for this assignment. I will generate a Maven project containing the appropriate `pom.xml` and `web.xml` into which I will copy your source.