Lab: Interact with a RESTful service

Objective

To explore and interact with a RESTful service.

Overview

At the end of this lab exercise, the student will be familiar with how to:

- Use POSTMAN to test REST endpoints
- REST interactions using GET, POST, PUT and DELETE
- The use of Hypertest Application Language (HAL)

Step by Step Guide

Binary File: **C:\WDXM-361\customerservice-0.0.1-SNAPSHOT.jar**
Source File: **C:\WDXM-361\customerservice.zip**
Time –60 Minutes

Review The Source

1. Your Instructor will **briefly** review the source code with you. In this particular case, you will see almost no explicit REST code. There are two reasons for this: one philosophical, one technical.

   The philosophical reason is that we are focusing on the REST concept, not programming. We have not formally taught the Java programming language as yet.

   The technical reason is that the solution uses a technology called *Spring Data REST*, where the runtime has automatically generated a REST (with HAL) endpoint for a repository. The entire solution consists of a boilerplate launching application, some code to pre-load data, a simple Java class representing a Customer, and a repository interface (literally, just a Java interface). In the real world, this is often a perfectly adequate bridge between a REST client and storage. When you want to customize the REST interaction, including adding behavior unrelated to persistence, you would provide your own REST code. That will be covered some weeks hence.

Launch the Service To Explore

2. From a command window, in the C:\WDXM-361 folder, run the command:

   java -jar customerservice-0.0.1-SNAPSHOT.jar

   Technically, you could doubleclick the file from the Explorer, but then you would have no console window to see any console output.

3. When the program finishes initializing, you should see the following output at the end of the console:

```
Customer Cale Tucker
Customer Akima Kunimoto
Customer Gune
Customer Stith
```

Browse to the REST Endpoint

4. Point your browser to http://localhost:8080/api/hal. This is the endpoint that was configured to automatically generated REST repository.

   The HAL output to the browser shows us additional URLs that we can access. Had we additional repositories, their URLs would appear here, as well.
5. Based on the information from the base REST endpoint, point your browser to http://localhost:8080/api/hal/customers. You should see a list of customers, matching the list from Step 3, in HAL format.
6. Finally, browse http://localhost:8080/api/hal/customers/1. That should present you with with the information for a single customer. Feel free to try any of the other customers that you saw from the list in Step 5.

Access Custom Queries

In addition to the default find all and find by id queries, which we explored in the prior section, the CustomerRepository interface has custom queries. We have not tried to use them, yet, so that is what we will do now.

7. Browse http://localhost:8080/api/hal/customers/search, which the customer resource had listed in its set of links. You should see the following:

```
{
  "_links" : {
    "findByFirstName" : {
      "href" :
"http://localhost:8080/api/hal/customers/search/findByFirstName{?
firstName}",
      "templated" : true
    },
```

```
    "findByLastName" : {
      "href" :
"http://localhost:8080/api/hal/customers/search/findByLastName{?l
astName}",
      "templated" : true
    },
    "self" : {
      "href" : http://localhost:8080/api/hal/customers/search
    }
  }
}
```

This is providing us a lisrt of available custom search endpoints.

8. Test one of the custom searches, *e.g.*,

   http://localhost:8080/api/hal/customers/search/findByFirstName?firstName=Cale

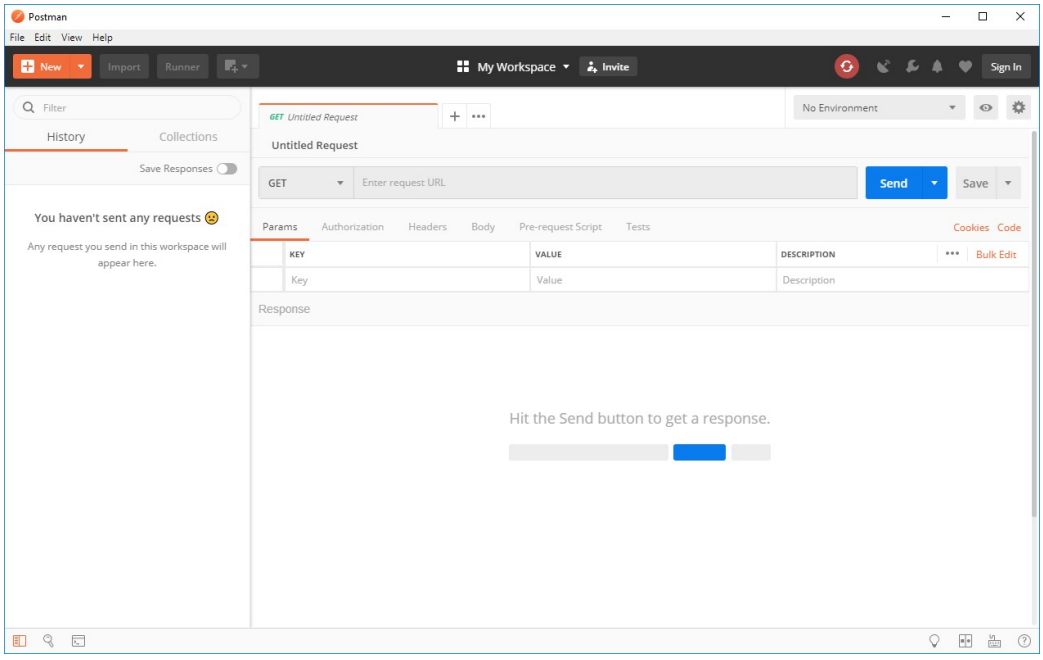
   The result of that query should be matching Customer resource(s).

Without JavaScript programming in the browser, this is as far as we can take REST in a Browser, so we'll use a different tool for the rest of the lab.

Testing REST Using Postman

Postman (www.getpostman.com) is an application with support for testing REST services.

9. Launch POSTMAN from the Desktop, and close any startup dialog. The application window should look like this screen capture, with a history pane on the left, and the main pane on the right.



GET The Customers Resource

10. Ensure that the HTTP verb is `GET`.
11. Enter `http://localhost:8080/api/hal/customers` for the URL.
12. Click SEND.

In the response portion of the screen, you should see a Status of `200 OK`, and a scrollable panel with the JSON representation of the `Customer` collection.

GET One Customer Resource

13. Click the + symbol on the tab bar to create a new request tab
14. Ensure that the HTTP verb is `GET`.
15. Enter `http://localhost:8080/api/hal/customers/1` for the URL.
16. Click SEND.

In the response portion of the screen, you should see a Status of `200 OK`, and a scrollable panel with the JSON representation of the `Customer` whose ID is 1.

Create (POST) a New Customer Resource

17. Click the + symbol on the tab bar to create a new request tab
18. Ensure that the HTTP verb is `POST`.
19. Enter `http://localhost:8080/api/hal/customers` for the URL.
20. Click on the `Body` tab below the request URL.
21. Select the `raw` radio button, and `JSON (application/json)` from the drop-down menu.
22. In the editor for the body, enter a new `Customer` in JSON format, *e.g.*,

```
{
    "firstName": "Sam",
    "lastName": "Tucker",
    "email": "tucker@titan-project.org",
    "phoneNumber": "000-555-1212"
}
```

23. Click SEND.

    In the response portion of the screen, you should see a Status of `201 Created`.

24. Click on `Headers` in the response. Look for the `Location` header, which should have a URL to the newly created resource. Highlight and copy the URL, to use in the next section.

GET The New Customer

25. Click the + symbol on the tab bar to create a new request tab
26. Ensure that the HTTP verb is `GET`.
27. Enter the URL copied from the `Location` header in the `POST` section.
28. Click SEND.

In the response portion of the screen, you should see a Status of `200 OK`, and a scrollable panel with the JSON representation of the `Customer` you just created.

Update (PUT) The Customer

29. Click the + symbol on the tab bar to create a new request tab
30. Ensure that the HTTP verb is `PUT`.
31. Enter the URL copied from the `Location` header in the POST section.
32. Click on the `Body` tab below the request URL.
33. Select the `raw` radio button, and `JSON (application/json)` from the drop-down menu.
34. In the editor for the body, enter an replacement `Customer` in JSON format, *e.g.*,

```
{
    "firstName": "Sam",
    "lastName": "Tucker",
    "email": "tucker@titan-ae.org",
    "phoneNumber": "000-555-1212"
}
```

35. Click SEND.

    In the response portion, you should see a Status of `200 OK`. This implementation chooses to return the updated body. Otherwise, it could have returned `204 No Content`.

GET The Updated Customer

We're going to repeat the steps that we used get the new `Customer` after the `POST` operation, and verify that the `PUT` worked as expected.

36. Click the + symbol on the tab bar to create a new request tab
37. Ensure that the HTTP verb is `GET`.
38. Enter the URL copied from the `Location` header in the `POST` section.
39. Click SEND.

In the response portion of the screen, you should see a Status of `200 OK`, and a scrollable panel with the JSON representation of the `Customer` you updated.

DELETE The Customer

40. Click the + symbol on the tab bar to create a new request tab
41. Ensure that the HTTP verb is `DELETE`.
42. Enter the URL copied from the `Location` header in the `POST` section.
43. Click SEND.

In the response portion, you should see a Status of `204 No Content`.

GET The Deleted Customer

We're going to repeat the steps that we used get the new `Customer` after the `POST` operation, and verify that the `DELETE` worked as expected.

44. Click the + symbol on the tab bar to create a new request tab
45. Ensure that the HTTP verb is `GET`.
46. Enter the URL copied from the `Location` header in the `POST` section.
47. Click SEND.

In the response area, you should see a Status of `404 Not Found`.

End of Lab