Help documentation – SCI User and Donation RESTful API 1.0.0

(*Y Haile,yidnekr@yahoo.com, +447880312339, London*)

Overview

This application is a Java RESTful API that is capable of taking user and donation information and then store that information in the data repository. The API is a Maven Java Application built with the latest version of Spring Framework (Version 2.1.1). For brevity and convenience for evaluators to test, the API consumes in-memory data and does not need to connect to external database to run.

The user and donation properties and methods are defined in SCIUser and Donation classes respectively. The business service for the API is implemented in UserService service class and the API is implemented with USerController REST controller class. The GET and POST methods are implemented in this controller class.

To test whether the API successfully fetches user/donation information, run the application and send a valid URL such as http://localhost:8080/users/ and http://localhost:8080/users/user1/donations from the browser or use other API testing method/tools of your choice such as Postman API testing tool. For example, a successful fetch via http://localhost:8080/users/ will retrieve all SCI users and their respective donations and a successful fetch via http://localhost:8080/users/user1/donations will respond the donation information in the form of an array of objects which is a set of information about donations donated by a given user (by user1 in this case):

```
[{"id":"donation1","name":"blanket","description":"Brand x blakets to be
delivered to site Y"},
{"id":"donation2","name":"Dry Food","description":"Various dry food items
including Biscuits, Porridge powder , Bread, and Sugar"},
{"id":"donation3","name":"Cash donation of £20K","description":" Cash
donation to be paid to Save the Children International in three
installements"},
{"id":"donation4","name":"miscellaneous","description":"Various types of
donations"}]
```

To test whether the API successfully stores user and donation information, you can run the application and send user/donation information such as in JSON format above to a valid URL such as http://localhost:8080/users/user1/donations using a method or tool of your choice such as using Postman API testing tool.

Business Service Implementation

UserService service class implements the services of this API. A user can donate any number of times. A donation has id, name, and description. A SCI user has id, name and profession and one or more donations they have donated to SCI. User id's are hashed for enhanced security and donation information is subjected to appropriate formatting for better readability and consistence.

The following are the main methods involved in the service:

- `public List<SCIUser> retrieveAllUser()` – Retrieves all users of Save The Children International and their respective donations from the data repository.

- `public List<Donation> retrieveDonations(String userId)` – Retrieves all donations donated by a single user identified by a given ID.

- `public Donation retrieveDonations(String userId, String donationId)` - Retrieves a single donation identified by a given donationID by a given SCI user identified by userID.

- `public Donation addDonation(String userId, Donation donation)` – saves donations donated by a given donor identified by userID to the data repository.

Service Controller Implementation

The REST Controller UserController controller class exposes some example GET and POST services.

`UserService` service class and `UserController` controller class have been wired together with the help of Spring auto-wiring facility.

```
@Autowired
 private UserService userService;
 private String userId;
```

A GET service is exposed to users to retrieve list of users.

```
@GetMapping ("/users")
public List<SCIUser> retrieveAllSCIUsers ()
```

A GET service is exposed with userId as a path variable

```
@GetMapping("/users/{userId}/donations")
public List<Donation> retrieveDonationsForUser(@PathVariable String userId)
```

A GET service is exposed with userId and donationId as path variables.

```
@GetMapping("/users/{userId}/donations/{donationId}")
public Donation retrieveDetailsForDonation(@PathVariable String userId,
             @PathVariable String donationId)
```

A POST service is exposed with userId as path variables newDonation as a request body.

```
@PostMapping("/users/{userId}/donations")
       public ResponseEntity <Void> registerUserForDonation (@PathVariable
String  userId, @RequestBody Donation newDonation)
```

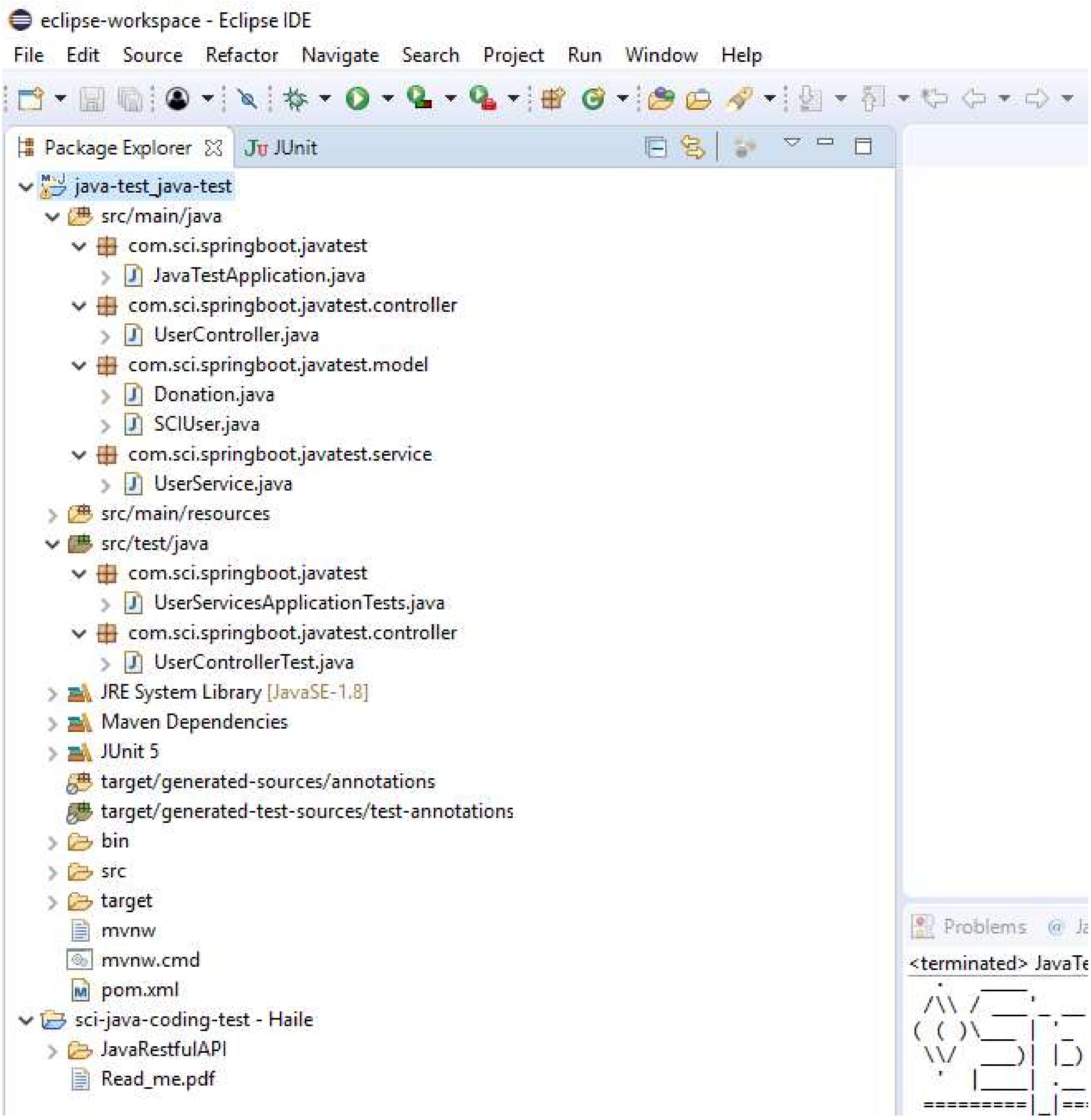
Service Controller Unit Testing

The UserControllerTest test class for the UserController controller class implements the Unit Test for the POST and GET REST Services and successfully passed all important assertion scenarios. The code snippet of the POST service Unit Test is as follows:

```
@Test
public void createUserDonation() throws Exception {
		Donation mockDonation = new Donation("2", "Even and Prime 
number", "2");
	Mockito.when(
			userService.addDonation(Mockito.anyString(), 

Mockito.any(Donation.class))).thenReturn(mockDonation);

	// Send Donation as body to the path /users/user1/donations
	RequestBuilder requestBuilder = MockMvcRequestBuilders
			.post("/users/user1/donations")

.accept(MediaType.APPLICATION_JSON).content(exampleDonationJson)
			.contentType(MediaType.APPLICATION_JSON);

	MvcResult result = mockMvc.perform(requestBuilder).andReturn();

	MockHttpServletResponse response = result.getResponse();

	assertEquals(HttpStatus.CREATED.value(), response.getStatus());

	assertEquals("http://localhost/users/user1/donations/2",
			response.getHeader(HttpHeaders.LOCATION));

}
```

General Project Structure

The general file structure of the project viewed from the Eclipse 4.10 IDE is as follows.



eclipse-workspace - Eclipse IDE

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer | JUnit

```
java-test_java-test
  src/main/java
    com.sci.springboot.javatest
      JavaTestApplication.java
    com.sci.springboot.javatest.controller
      UserController.java
    com.sci.springboot.javatest.model
      Donation.java
      SCIUser.java
    com.sci.springboot.javatest.service
      UserService.java
  src/main/resources
  src/test/java
    com.sci.springboot.javatest
      UserServicesApplicationTests.java
    com.sci.springboot.javatest.controller
      UserControllerTest.java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  JUnit 5
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  bin
  src
  target
  mvnw
  mvnw.cmd
  pom.xml
sci-java-coding-test - Haile
  JavaRestfulAPI
  Read_me.pdf
```

Problems

<terminated> JavaTe

Project Dependency

The dependency of libraries and tools of the project are as shown below in the pom.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.1.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.sci.springboot</groupId>
    <artifactId>java-test</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>java-test</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.mockito</groupId>
            <artifactId>mockito-all</artifactId>
            <version>1.9.5</version>
        </dependency>

        <dependency>
          <groupId>org.skyscreamer</groupId>
          <artifactId>jsonassert</artifactId>
          <version>1.5.0</version><!--$NO-MVN-MAN-VER$-->
         </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```