

Event Reservation REST API: Take Home Challenge

Friday, October 19, 2018 5:24 PM

Introduction

1. Java Version: 1.8
2. Framework: Spring Boot
3. Database: H2 (In-memory)
4. URLs
 - a. REST API hosted at <http://localhost:8080/>
 - b. H2 DB Console: <http://localhost:8080/h2-console>
 - c. Swagger JSON: <http://localhost:8080/v2/api-docs>
5. Unit Tests for the Repositories, Service & Controller are written. See Testing section for additional details.
6. Integration Tests to verify appropriate handling of Race Condition(s) are written. See Testing section for additional details.

API Documentation

1. Open the online Swagger Editor by navigating to the following URL <https://editor.swagger.io/>
2. Import the REST API Swagger v2 JSON (event-reservation-api-swagger-v2.json), included in the codebase and view the Documentation UI as per following screenshot.

The screenshot displays the Swagger Editor interface. On the left, the Swagger JSON definition is visible, detailing the API's metadata and endpoints. On the right, the generated API documentation UI is shown, featuring a list of endpoints with their respective HTTP methods and descriptions.

```
1 swagger: '2.0'
2 info:
3   description: Api Documentation
4   version: '1.0'
5   title: Api Documentation
6   termsOfService: 'urn:tos'
7   contact: {}
8   license:
9     name: Apache 2.0
10    url: 'http://www.apache.org/licenses/LICENSE-2.0'
11 host: 'localhost:8080'
12 basePath: /
13 tags:
14 - name: basic-error-controller
15   description: Basic Error Controller
16 - name: reservation-controller
17   description: Reservation Controller
18 paths:
19   /error:
20     get:
21       tags:
22         - basic-error-controller
23       summary: errorHtml
24       operationId: errorHtmlUsingGET
25       consumes:
26         - application/json
27       produces:
28         - text/html
29       responses:
30         '200':
31           description: OK
32           schema:
33             $ref: '#/definitions/ModelAndView'
34         '401':
35           description: Unauthorized
36         '403':
37           description: Forbidden
38         '404':
39           description: Not Found
40     head:
41       tags:
42         - basic-error-controller
43       summary: errorHtml
44       operationId: errorHtmlUsingHEAD
45       consumes:
46         - application/json
47       produces:
48         - text/html
```

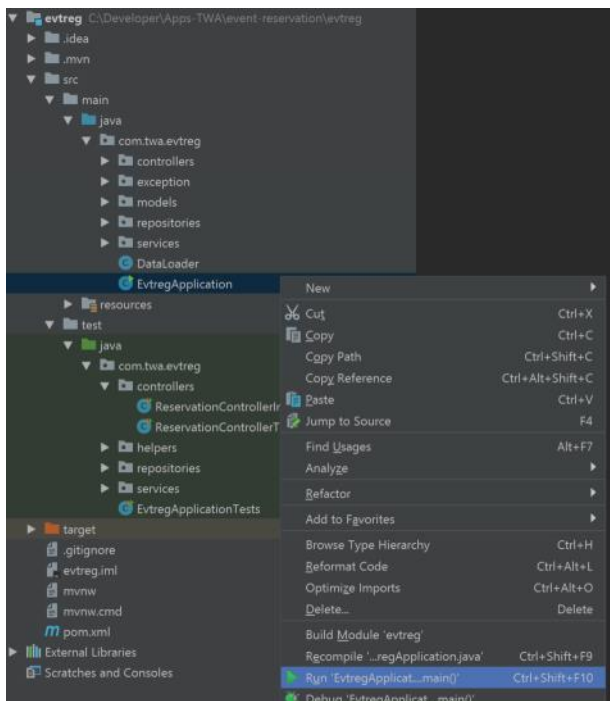
The right panel shows the generated API documentation for the **reservation-controller**. The endpoints listed are:

- DELETE** `/error` `errorHtml`
- OPTIONS** `/error` `errorHtml`
- PATCH** `/error` `errorHtml`
- GET** `/reservation/available-dates` `getAvailableDatesFor`
- POST** `/reservation/book` `book`
- POST** `/reservation/book-faulty` `bookFaulty`
- DELETE** `/reservation/cancel/{id}` `cancel`
- PUT** `/reservation/change/{id}` `change`
- GET** `/reservation/list` `list`
- GET** `/reservation/lock` `lock`
- GET** `/reservation/unlock` `unlock`

At the bottom, there is a section for **Models** with a right arrow indicating further details.

Launch Application

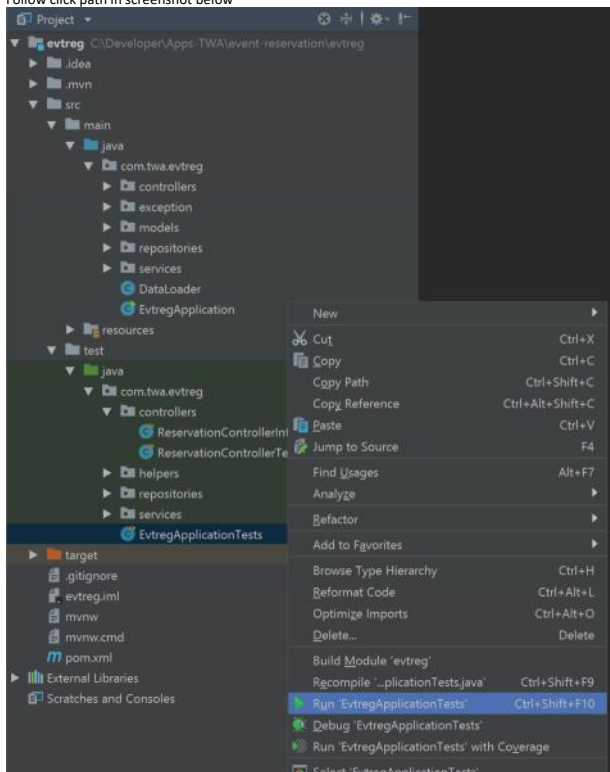
1. Open the project in IntelliJ
2. Follow the click path shown in the screenshot below to launch the application
3. This will launch the application on <http://localhost:8080/>
4. In addition, you can access the H2 console at <http://localhost:8080/h2-console/>



Running Testing

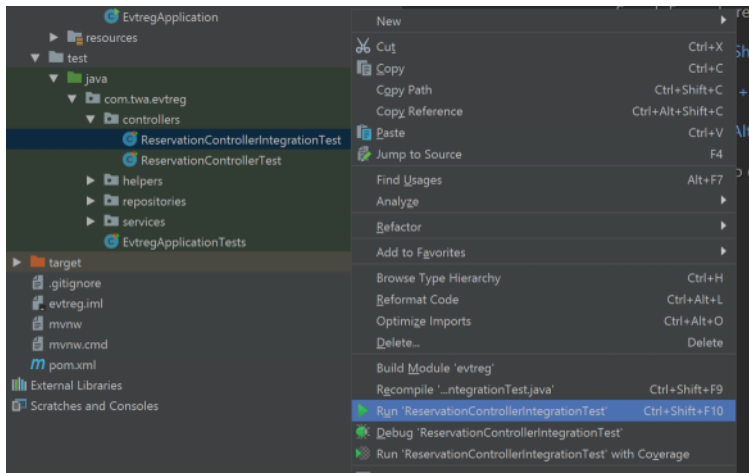
How to run Unit Tests

1. Follow click path in screenshot below



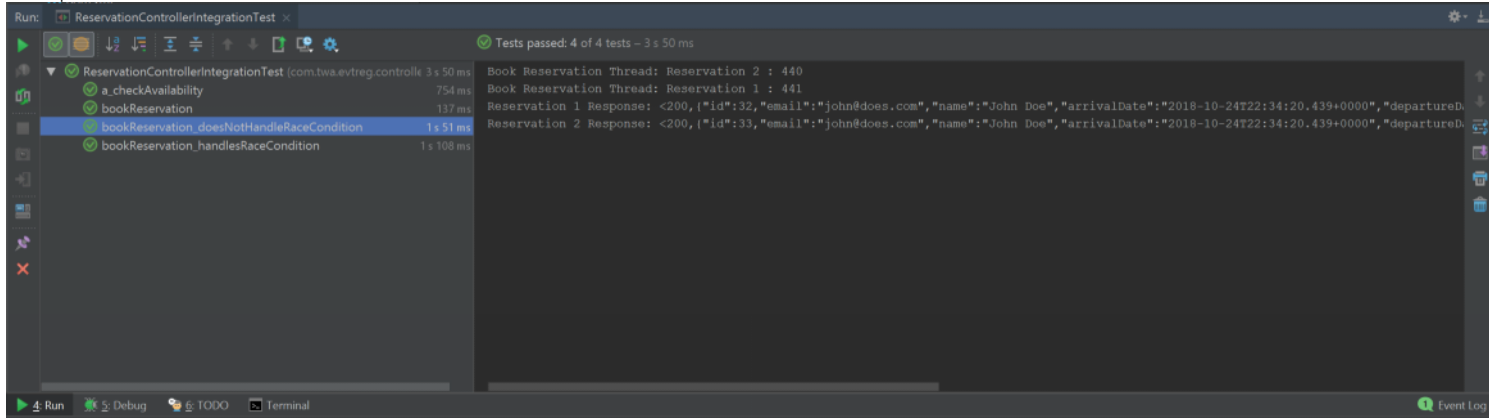
How To Test Race Condition

1. IDE: IntelliJ
2. The Integration Tests are defined in `com.twa.evreg.ReservationControllerIntegrationTest`
3. Right Click on the above mentioned Java Class and choose "Run" from the Context Menu
4. This should trigger the Tests as shown in the below
5. Then click on the appropriate Test to see the corresponding **Response** logged in the **Console**.



Race Condition Not Handled

- Both Response 1 & 2 return HTTP STATUS CODE 200



Race Condition Handled

- Response 2 returns HTTP STATUS CODE 200
- Response 1 returns HTTP STATUS CODE 404

