**1. Emirates Spring Boot Rest API Project**

This is a sample Java / Maven / Spring Boot application that can be used to create/change flight seat and check seat availability on a given date.

**2. How to Run**

This application is packaged as a war which has Tomcat embedded. No Tomcat or JBoss installation is necessary. You run it using the java -jar command.

* Clone this repository <https://github.com/divyangpatel123/emirates.git>
* Make sure you are using JDK 1.8 and Maven 3.x.
* Configure the project with MySQL.

- This project uses an mysql database so that you have to install MYSQL database in order to run it. Since the project uses Spring and the Hibernate framework, it's even fairly easy to back the same service with any DB.

- To import sample passenger list data we need to create a database named emirates. Import sample data using sql file.

- mysql -u username -p emirates < emirates.sql

Note : emirates.sql file found in source code of github.

\*\*\*\*\*\* Change this config in application.properties: \*\*\*\*\*\*

---

# Connection url for the database "emirates"

spring.datasource.url = jdbc:mysql://localhost:3306/emirates?useSSL=false

# Username and password

spring.datasource.username = <your\_mysql\_username>

spring.datasource.password = <your\_mysql\_password>

* You can build the project and run the tests by running mvn clean package
* Once successfully built, you can run the service by one of these two methods:

java -jar target/emirates-rest-api-0.0.1-SNAPSHOT.war

Or

mvn spring-boot:run

* Check the std out to make sure no exceptions are thrown
* Once the application runs you should see something like this

2017-06-04 11:15:00.793 INFO 13444 --- [ main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 7090 (http)

2017-06-04 11:15:00.808 INFO 13444 --- [ main] c.e.flight.SpringBootWebApplication : Started SpringBootWebApplication in 28.694 seconds (JVM running for 30.718)

1. **About the Service**

The service is just a simple flight booking REST service. It uses a MySQL database to store the data. If your database connection properties work, you can call some REST end points defined in com.emirates.flight.controllers.FlightController on **\*\*port 7090\*\***. (see below)

Here is what this little application demonstrates:

* Full integration with the latest **Spring**  and **Spring Boot** Framework: dependency injection, REST Web Service, Hibernate/JPA Integration, Swagger etc.
* Generated API documentation with swagger.Swagger is a specification for documenting REST API. It specifies the format (URL, method, and representation) to describe REST web services.

- <http://localhost:7090/swagger-ui.html>

- <http://localhost:7090/v2/api-docs>

* Packaging as a single war with embedded container (tomcat): No need to install a container separately on the host just run using the java -jar command.
* Writing a RESTful service using annotation: JSON request / response; simply use desired Accept header in your request
* Exception mapping from application exceptions to the right HTTP response with exception details in the body
* Spring Integration with JPA/Hibernate with just a few lines of configuration and familiar annotations.
* Demonstrates Spring Boot test framework with associated libraries
* All APIs are "self-documented" by Swagger using annotations
* Here are some endpoints you can call:

**### Retrieve a available list of seats**

```

GET http://localhost:7090/flight/seat/available-list/{date}

[i.e Date should be in this format : dd-MM-yyyy]

Content-Type: application/json

RESPONSE: HTTP 200 (OK)

Location header: <http://localhost:7090/flight/seat/available-list/10-06-2017>

Content : List of seats

```

**### Create a seat**

```

POST /flight/seat/create

Accept: application/json

Content-Type: application/json

{

"departureDate": "2017-06-05T06:37:24.716Z",

"passengerEmail": "divyangpatel74@gmail.com",

"passengerId": "EMIPASS26",

"passengerMobile": "+91 7698221862",

"passengerName": "Divyang Patel",

"seatId": 1

}

RESPONSE: HTTP 201 (Created)

Location header: <http://localhost:7090/flight/seat/create>

```

**### Change a seat**

```

PUT /flight/seat/change/{passengerId}

Accept: application/json

Content-Type: application/json

{

"seatId": 9

}

RESPONSE: HTTP 200 (OK)

Location header: <http://localhost:7090/flight/seat/change/EMIPASS26>

```

**### Retrieve a booked list of seats**

```

GET http://localhost:7090/flight/seat/booked-list/{date}

[i.e Date should be in this format : dd-MM-yyyy]

Content-Type: application/json

RESPONSE: HTTP 200 (OK)

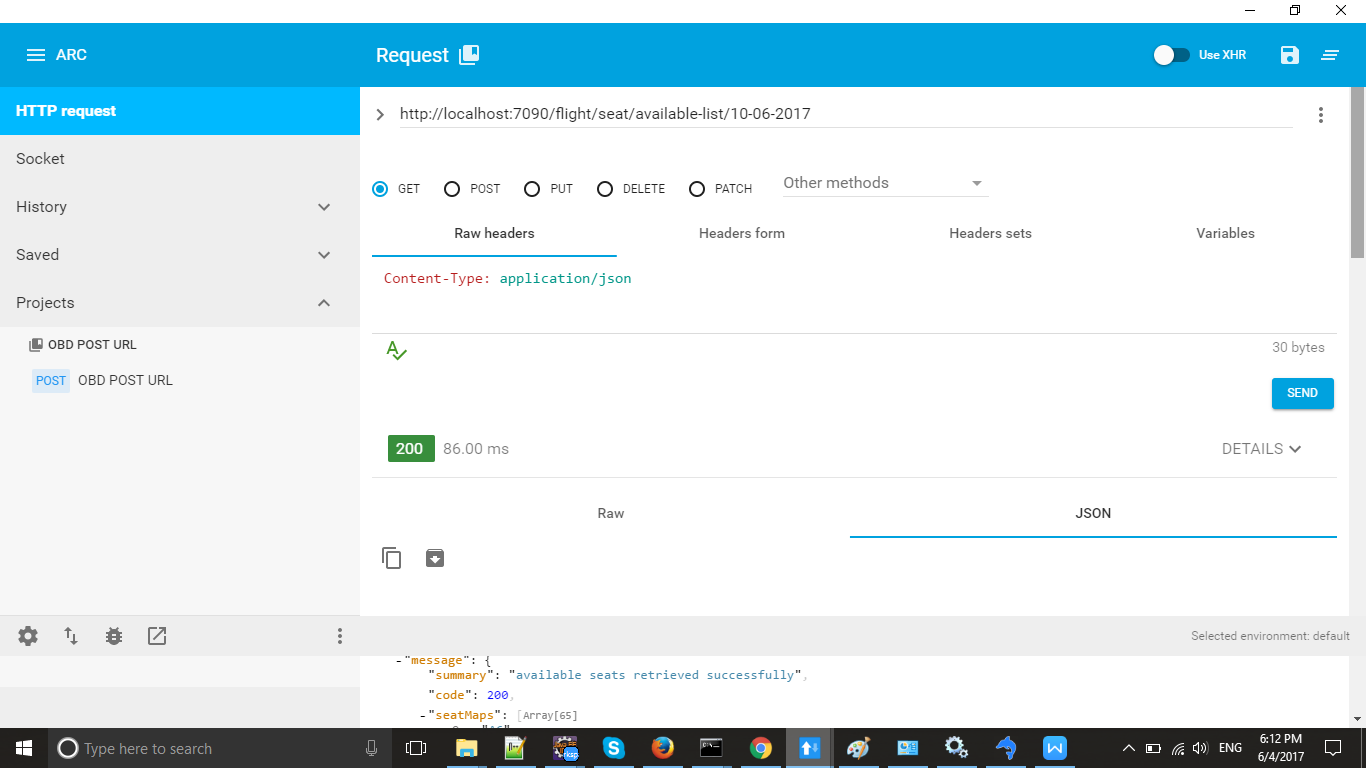
Location header: <http://localhost:7090/flight/seat/booked-list/10-06-2017>

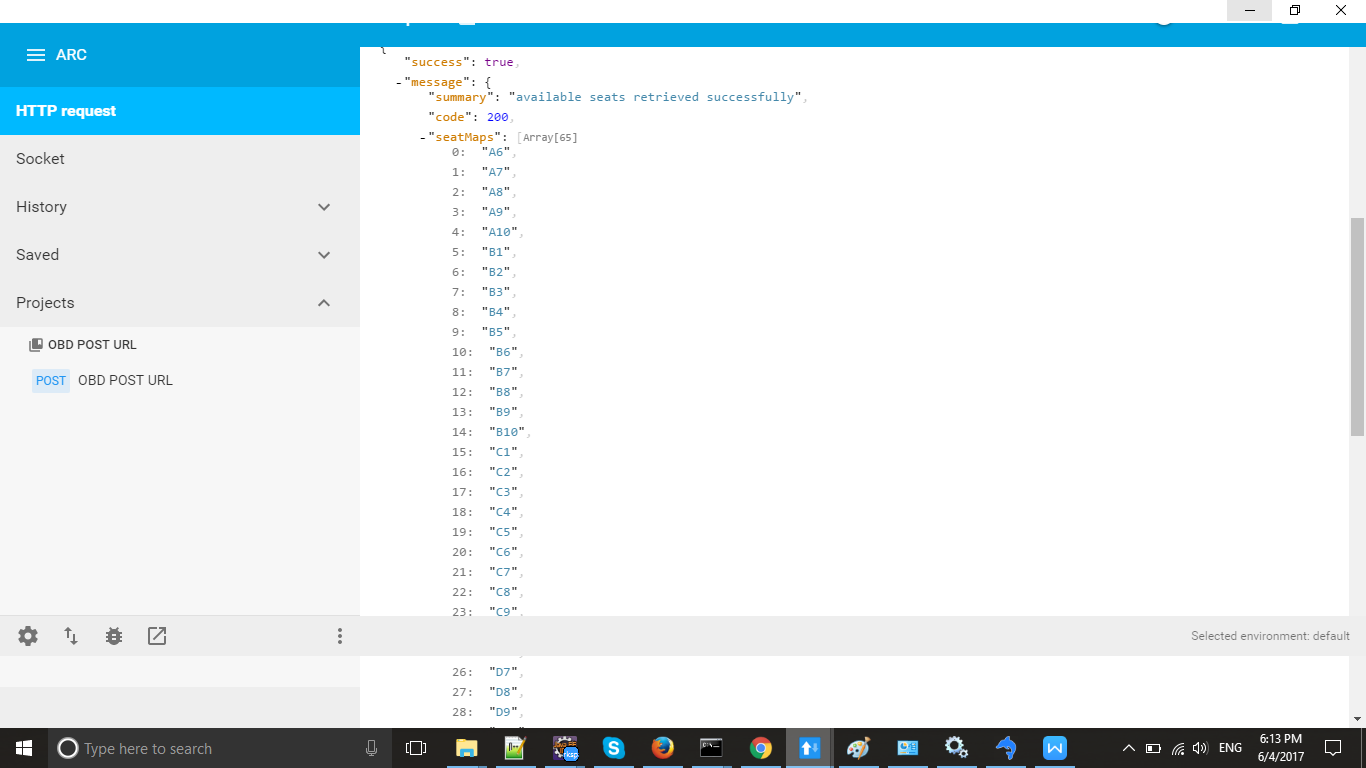
Content : List of seats

**4. About Spring Boot**

Spring Boot is an "opinionated" application bootstrapping framework that makes it easy to create new RESTful services (among other types of applications). It provides many of the usual Spring facilities that can be configured easily usually without any XML. In addition to easy set up of Spring Controllers, Spring Data, etc.

1. **Screenshots**

****

****

