**Steps to Build**

**Public Repository URL = git clone https://github.com/mayur-us/Spring.git**

**Note: Please ensure availability of Docker Runtime(in case of Windows its Docker Desktop) and also ensure its up and running.**

1. **Build the artifact and Docker image**

**Command : mvn clean install**

1. **This will build the jar in target folder**

eg: C:\InterviewsComplete\UBS\emlebi\_eval\target\**emlebi-restservices-1.0.jar**

1. **Next it will create the Docker Image**

eg: **mayurdoshi/emlebi-restservices:1.0**

Here mayurdoshi is the name of my Docker Repository.In order to tag the image with your

Docker Repository name, please update your repository name in the **Docker plugin (marked in yellow) declared in pom.xml**

*<plugin>*

*<groupId>com.spotify</groupId>*

*<artifactId>dockerfile-maven-plugin</artifactId>*

*<version>1.4.10</version>*

*<executions>*

*<execution>*

*<id>default</id>*

*<goals>*

*<goal>build</goal>*

*<!-- <goal>push</goal> -->*

*</goals>*

*</execution>*

*</executions>*

*<configuration>*

*<repository>mayurdoshi/${project.artifactId}</repository>*

*<tag>${project.version}</tag>*

*<skipDockerInfo>true</skipDockerInfo>*

*</configuration>*

1. **Deploy and Run the Image created in Step 1 above**
2. **Validate the presence of the Image created in Step 1, using the following command**

**docker images**

1. **Run the image(Please make sure to change the image tag prefix, marked in yellow to your respective repository name as explained above). Use the below command**

**docker run -p 5000:8080 mayurdoshi/emlebi-restservices:1.0**

**Steps to Test using SwaggerUI AND/OR Postman Client**

**Swagger URL**

**http://localhost:5000/swagger-ui.html**

We have used Spring Security with JWT Token Authentication and Authorization.

For the sake of simplicity and ease of testing both scenarios, we have only secured one REST Endpoint, the remaining REST Endpoints are permitted without JWT token Authentication

REST Endpoints without Authentication

1)Get Person by Id

### [GET](http://localhost:5000/swagger-ui.html#!/person-controller/getUsingGET) [/api/v1/person/getPerson/{id}](http://localhost:5000/swagger-ui.html#!/person-controller/getUsingGET)

2)Create Person

[**POST**](http://localhost:5000/swagger-ui.html#!/person-controller/createUsingPOST)[**/api/v1/person/createPerson**](http://localhost:5000/swagger-ui.html#!/person-controller/createUsingPOST)

3)Delete Person

### [DELETE](http://localhost:5000/swagger-ui.html#!/person-controller/deleteUsingDELETE) [/api/v1/person/deletePerson/{id}](http://localhost:5000/swagger-ui.html#!/person-controller/deleteUsingDELETE)

4)Update Person

### [PUT](http://localhost:5000/swagger-ui.html#!/person-controller/updateUsingPUT) [/api/v1/person/updatePerson/{id}](http://localhost:5000/swagger-ui.html#!/person-controller/updateUsingPUT)

**REST Endpoint with Authentication**

**1)Get List of All Persons**

### [GET](http://localhost:5000/swagger-ui.html#!/person-controller/listUsingGET) [/api/v1/person/getAllPersons](http://localhost:5000/swagger-ui.html#!/person-controller/listUsingGET)

Steps

1. Create a User (POST request)

**/api/v1/users/saveUser**

RequestBody

Eg:

{

"username": "meeti",

"email": "mayur.us@gmail.com",

"password": "meeti"

}

1. Login with this created user and get the JWT Token

**/api/v1/users/login**

RequestBody

Eg:

{

"username": "meeti",

"password": "meeti"

}

Response (contains JWT Token marked in yellow below)

meeti eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJtZWV0aSIsImV4cCI6MTYxMzkyNzQwMn0.6976AEhMx\_Rpj8M0-0cw7JoXfmbCfFTEdbn9TqtVUDicEngiGrF7GQD5Zikd9njRQ3eKhb7Y8b-Ey7\_3F5b9Ow

1. Use this JWT Token to Authorize for accessing the protected REST Endpoint

**/api/v1/person/getAllPersons**

**(POST Request)**

**Request Header**

Content-Type = application/json

Authorization = Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJtZWV0aSIsImV4cCI6MTYxMzkyNzY1NX0.l7V7244CK1ZVGSUgMw7D-keFTBG\_i3FKYsOeWpm03ZEQhjrZfw2pLTV0sDkbsByYiKp-IhgvA5RgBnVDooYe5A

Response

List of all Persons