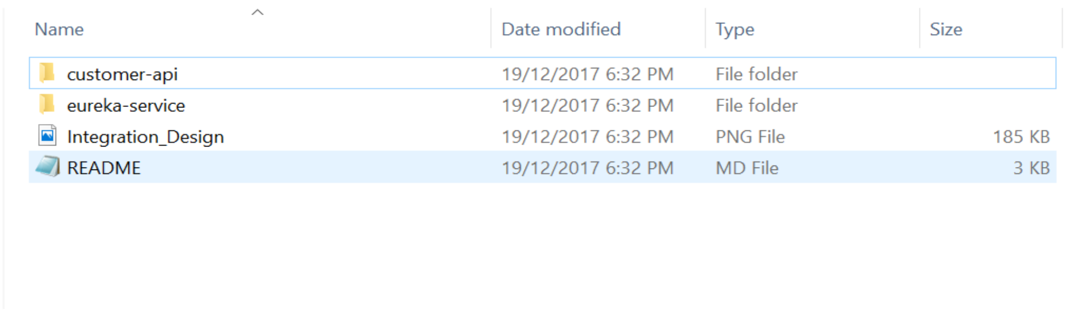
Spring Boot Customer Application using Microservices architecture

Run without IDE(Assumptions – maven is installed in your PC, java 8 is installed as well)

1. Clone this project from root folder.(https://github.com/shriramdevanathan/spring-customer-crm/tree/feature/development)
2. Unzip the file. You should see the below folder structure.



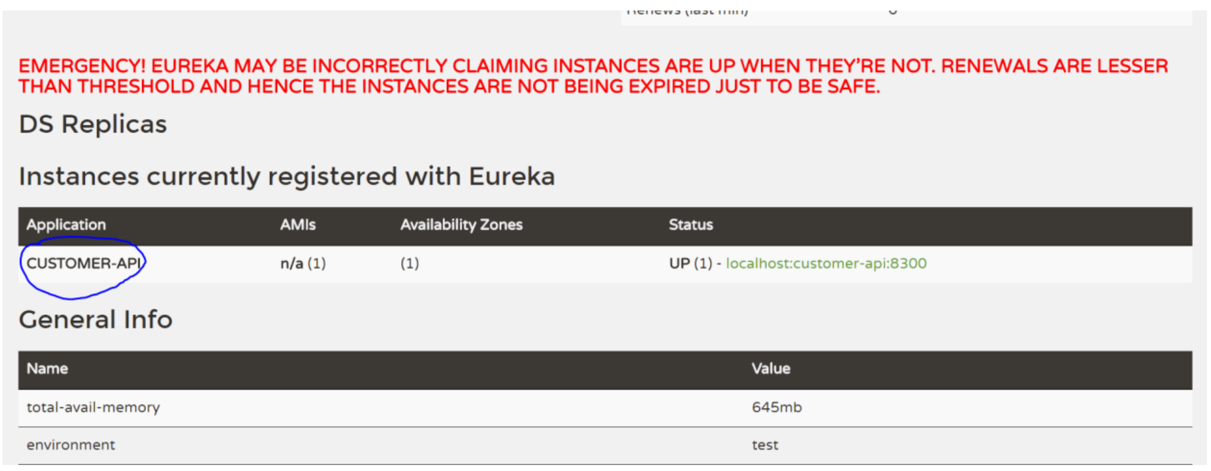
Name	Date modified	Type	Size
customer-api	19/12/2017 6:32 PM	File folder	
eureka-service	19/12/2017 6:32 PM	File folder	
Integration_Design	19/12/2017 6:32 PM	PNG File	185 KB
README	19/12/2017 6:32 PM	MD File	3 KB

3. Navigate to 'eureka-service' root and run
   - **mvn clean install**
   - *mvn package*
4. This should create the 'target' folder and also contain eureka-service-0.0.1-SNAPSHOT.jar
5. Run the following command.
   - cd target
   - java -jar eureka-service-0.0.1-SNAPSHOT.jar
6. This should startup the embedded tomcat and the entity manager stuff. The port should be **8302** as per settings.
7. Navigate to 'customer-api' root and run
   - **mvn clean install**
   - *mvn package*
8. This should create the target folder and also contain customer-api-0.0.1-SNAPSHOT.jar
9. Run the following command then.
   - cd target
   - java -jar customer-api-0.0.1-SNAPSHOT.jar
10. This should startup the embedded tomcat in port **8300**.

If the above steps were successful, it means that we have a Eureka server running, with the customer-api eureka client registered to the Eureka server.

Now from browser, access the following url to access the Eureka Dashboard

➔ **http://localhost:8302**


EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
CUSTOMER-API	n/a (1)	(1)	UP (1) - localhost:customer-api:8300

General Info

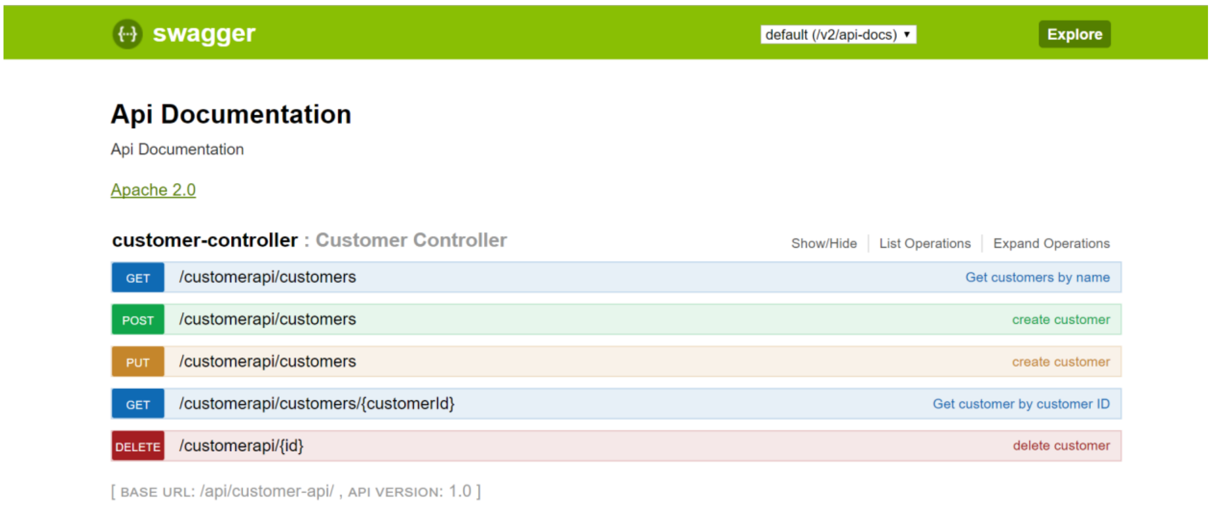
Name	Value
total-avail-memory	645mb
environment	test

You should see the above screenshot with the customer-api registered(as circled)

11. In another browser tab, access the following url which will hit the gateway and reroute to customer-api using the Eureka registry.

→localhost:8082/api/customer-api/swagger-ui.html

This will bring up the swagger ui which will show up all of the rest end points along with relevant documentation about each REST.



Api Documentation

Api Documentation

Apache 2.0

customer-controller : Customer Controller

GET	/customerapi/customers	Get customers by name
POST	/customerapi/customers	create customer
PUT	/customerapi/customers	create customer
GET	/customerapi/customers/{customerId}	Get customer by customer ID
DELETE	/customerapi/{id}	delete customer

[BASE URL: /api/customer-api/ , API VERSION: 1.0]

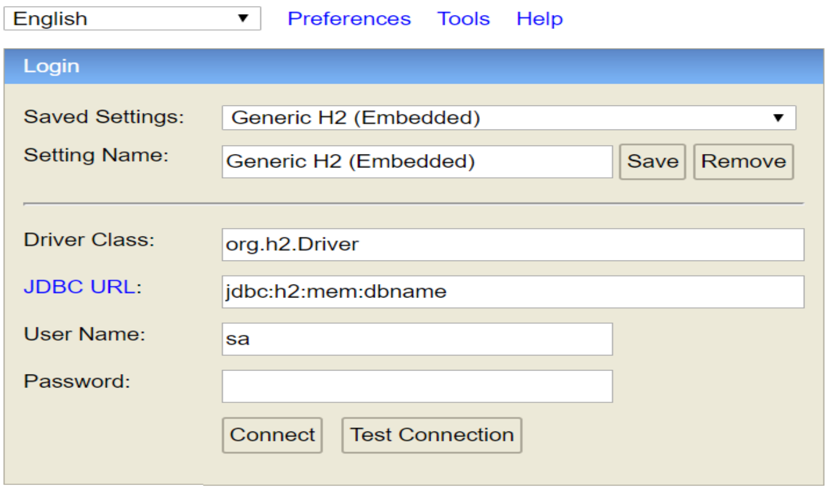
<u>Sample POST request customer:</u>

```
{
"addresses": [
{
"address_type": "Office",
"city": "North Sydney",
"country": "Australia",
"postalCode": "1000",
"street": "Miller Street",
"complexName" : "Chifley Square"
},
{
"address_type": "Home",
"city": "Parramatta",
"country": "Australia",
"postalCode": "2150",
"street": "Great Western Highway",
"unitNumber" : "20",
"apartmentNumber" : "62"

}
],
"dob": "2017-12-20T03:01:39.317Z",
"firstName": "Shriram",
"lastName": "Devanathan"
}
```

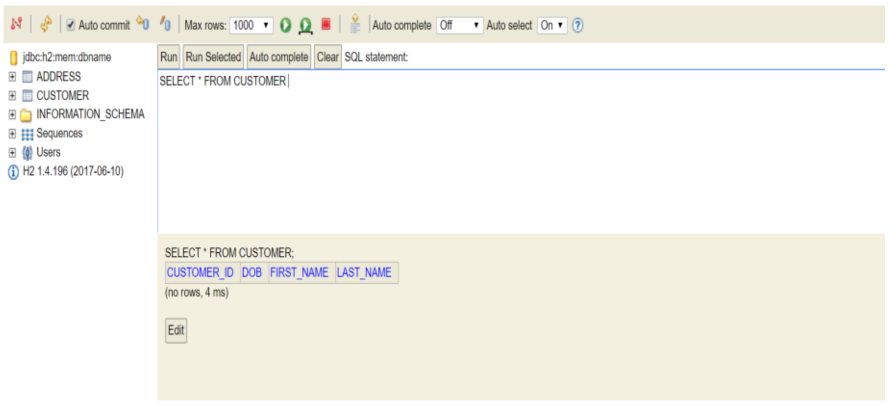
12. Access the following url to access the H2 database console :
    ➔ **localhost:8082**

    Use the following credentials:



Saved Settings:	Generic H2 (Embedded)
Setting Name:	Generic H2 (Embedded)
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:mem:dbname
User Name:	sa
Password:	

You should be able to see the next screen as follows,



Auto commit ✓ Max rows: 1000 Auto complete: Off Auto select: On

jdbc:h2:mem:dbname
ADDRESS
CUSTOMER
INFORMATION_SCHEMA
Sequences
Users
H2 1.4.196 (2017-06-10)

SQL statement:

```
SELECT * FROM CUSTOMER
```

```
SELECT * FROM CUSTOMER;
```

CUSTOMER_ID	DOB	FIRST_NAME	LAST_NAME

(no rows, 4 ms)

Run with IDE(preferably Intellij)

1. Import both projects in Intellij and let the maven imports run.
2. Run the Eureka-service first and then the customer-api.
3. Now repeat step 11 from above.

We have now established an API design via a API gateway, that can be extended to Mobile as well.