

# AWS HandsOn

## EC2 HandsOn

The screenshot shows the AWS Management Console for the 'us-east-2' region. The left sidebar contains navigation links for 'EC2 Dashboard', 'Events', 'Tags', 'Limits', 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations', 'Images', 'AMIs', 'Elastic Block Store', 'Volumes', 'Snapshots', and 'Lifecycle Manager'. The main content area displays a list of instances. The 'Siddhartha Instance' (i-0e07d8181fcb5db6d) is highlighted, showing its state as 'Running'. Below the list, the details for the 'Siddhartha Instance' are shown, including its Public IPv4 address (3.143.7.203) and Public IPv4 DNS (ec2-3-143-7-203.us-east-2.compute.amazonaws.com). The console also shows a notification about the new EC2 Experience and a success message for starting the instance.

The screenshot shows a web browser displaying a 'Test Page' for the Apache HTTP server. The page contains instructions for testing the server and a terminal window showing the command to start the service.

**Test Page**

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page, it means that the Apache HTTP server installed at this site is working properly.

**If you are a member of the general public:**

The fact that you are seeing this page indicates that the website you just visited is either experiencing problems, or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting `www.example.com`, you should send e-mail to "webmaster@example.com".

**If you are the website administrator:**

You may now add content to the directory `/var/www/html/`. Note that until you do so, people visiting your website will see this page, and not your content. To prevent this page from ever being used, follow the instructions in the file `/etc/httpd/conf.d/welcome.conf`.

You are free to use the image below on web sites powered by the Apache HTTP Server.

**Powered by 2.4**

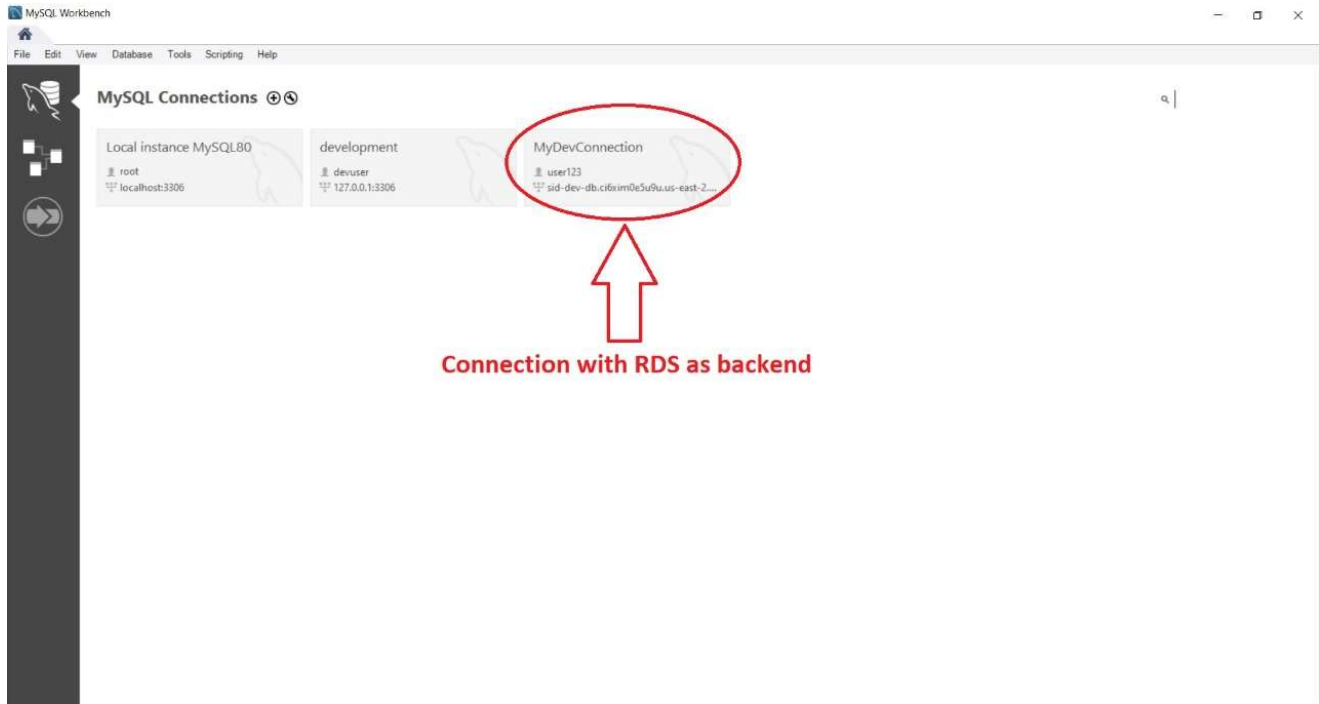
**Amazon Linux 2 AMI**

```
https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-33-205 ~]$ sudo service httpd status
Redirecting to /bin/systemctl status httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2021-04-20 07:21:29 UTC; 24min ago
     Docs: man:httpd.service(8)
  Main PID: 32581 (httpd)
   Status: "Total requests: 0; Idle/Busy workers 100/0; Requests/sec: 0; Bytes served/sec: 0 B/sec"
    CGroup: /system.slice/httpd.service
            └─32581 /usr/sbin/httpd -DFOREGROUND
              └─32582 /usr/sbin/httpd -DFOREGROUND
                └─32583 /usr/sbin/httpd -DFOREGROUND
                  └─32584 /usr/sbin/httpd -DFOREGROUND
                    └─32585 /usr/sbin/httpd -DFOREGROUND
                      └─32586 /usr/sbin/httpd -DFOREGROUND

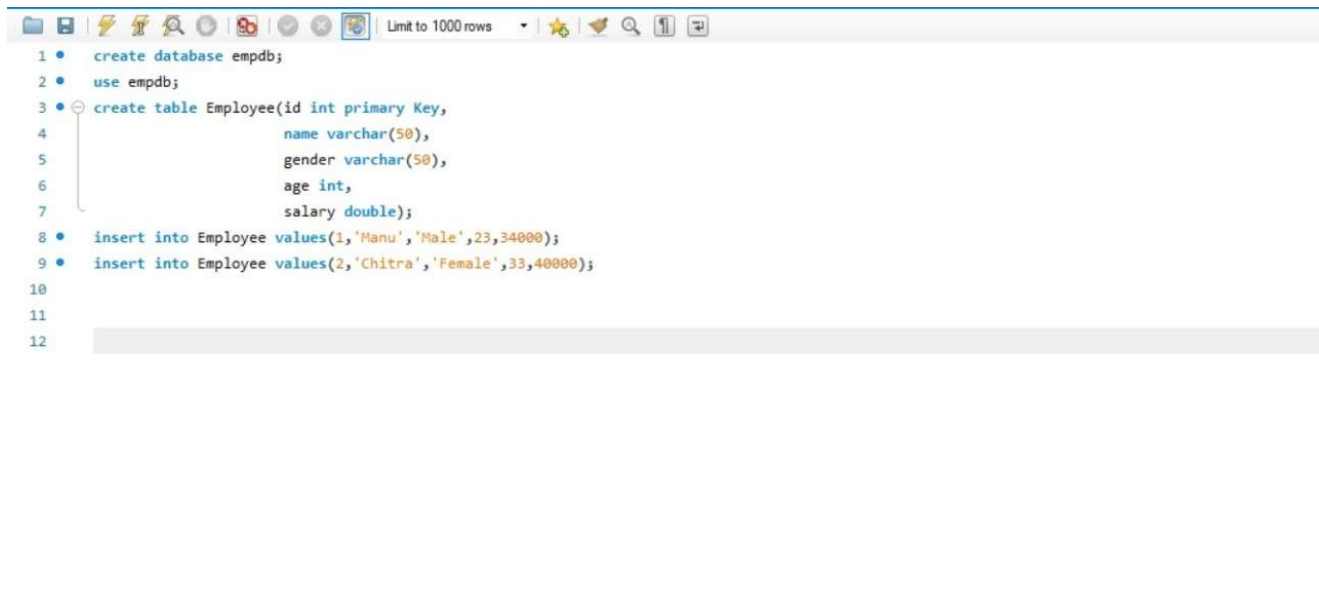
Apr 20 07:21:29 ip-172-31-33-205.us-east-2.compute.internal systemd[1]: Start...
Apr 20 07:21:29 ip-172-31-33-205.us-east-2.compute.internal systemd[1]: Start...
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-33-205 ~]$ 103.103.215.163
```

Type the public IPV4 address on the browser url bar and we should get the above shown screen

# RDS HandsOn

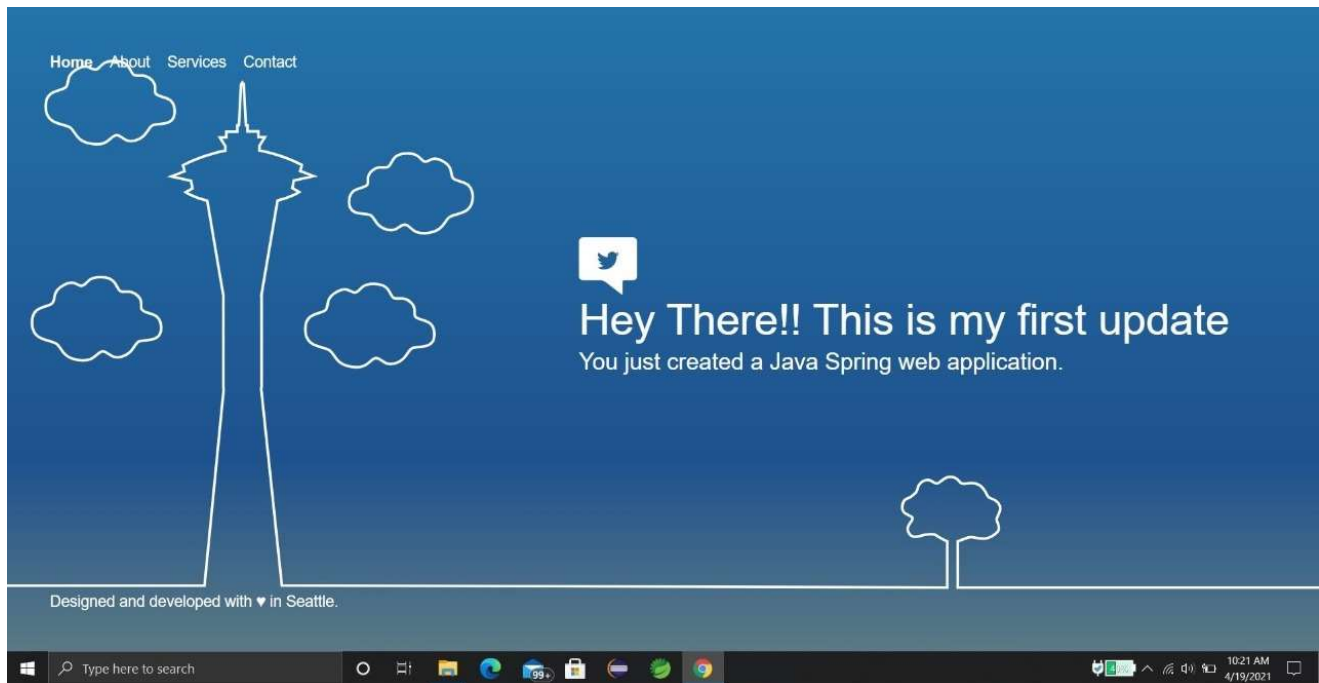


## Data in RDS

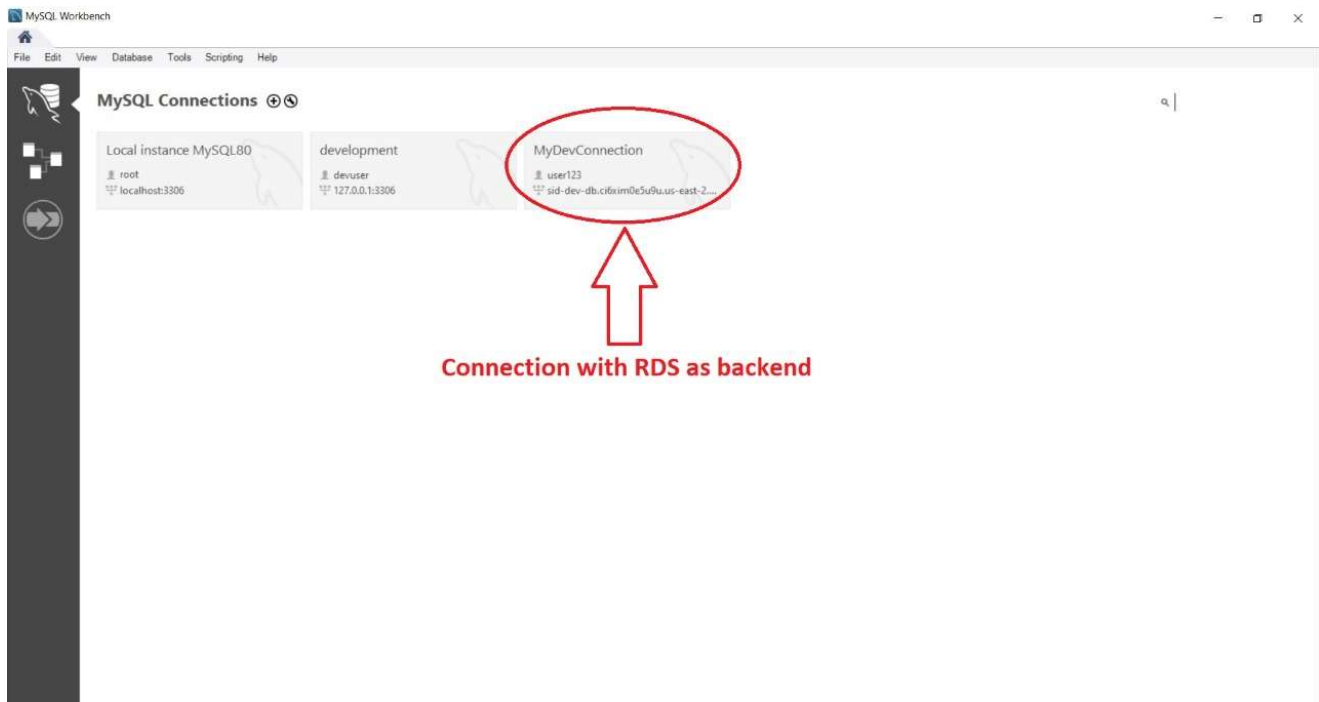


## CI/CD HandsOn

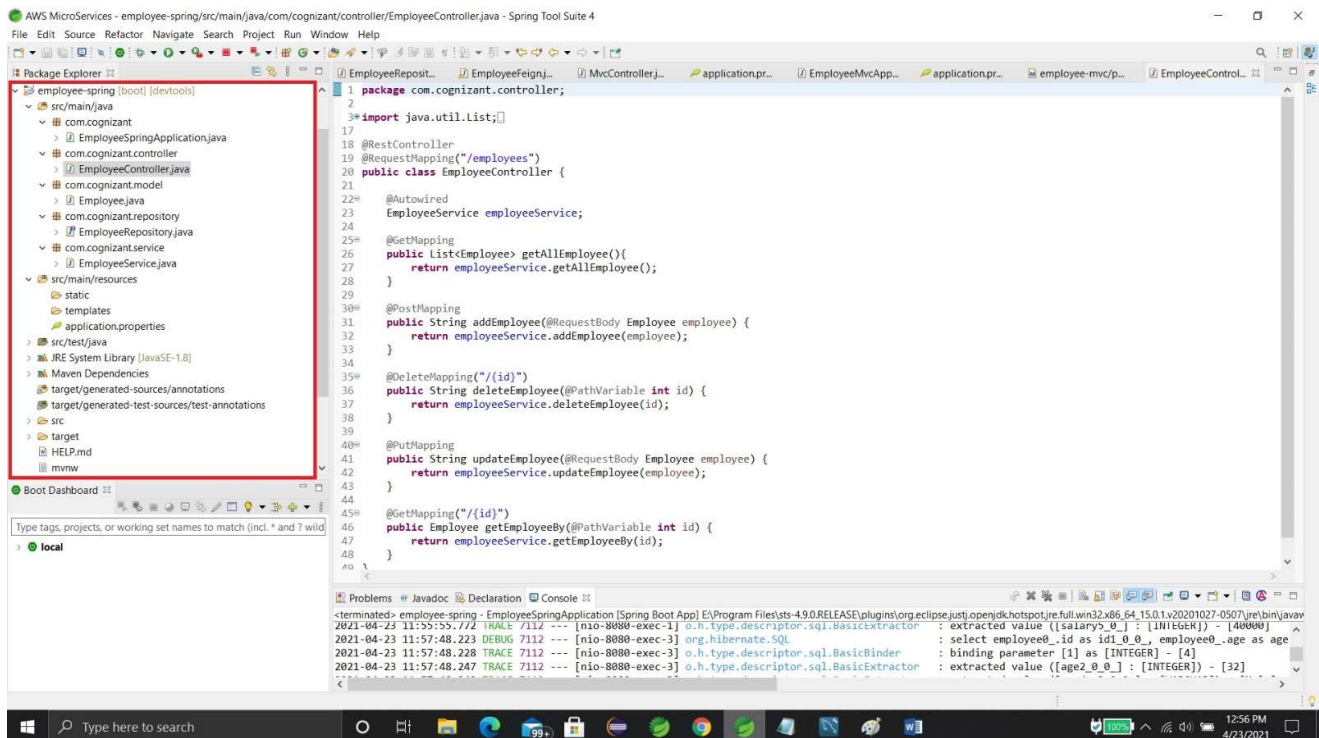
Output:-



## Spring-Rest-with-RDS-Backend



```
1 • create database empdb;
2 • use empdb;
3 • create table Employee(id int primary Key,
4                       name varchar(50),
5                       gender varchar(50),
6                       age int,
7                       salary double);
8 • insert into Employee values(1,'Manu','Male',23,34000);
9 • insert into Employee values(2,'Chitra','Female',33,40000);
10
11
12
```



We have created a “employee” microservice to test the RDS Database.

# Swagger HandsOn

## Step-1:- Add Dependencies

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>3.0.0</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>3.0.0</version>
</dependency>
</dependencies>

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

← Swagger Dependencies

## Step 2:- Create a Swagger configuration class

The screenshot shows an IDE (Spring Tool Suite 4) with the following components:

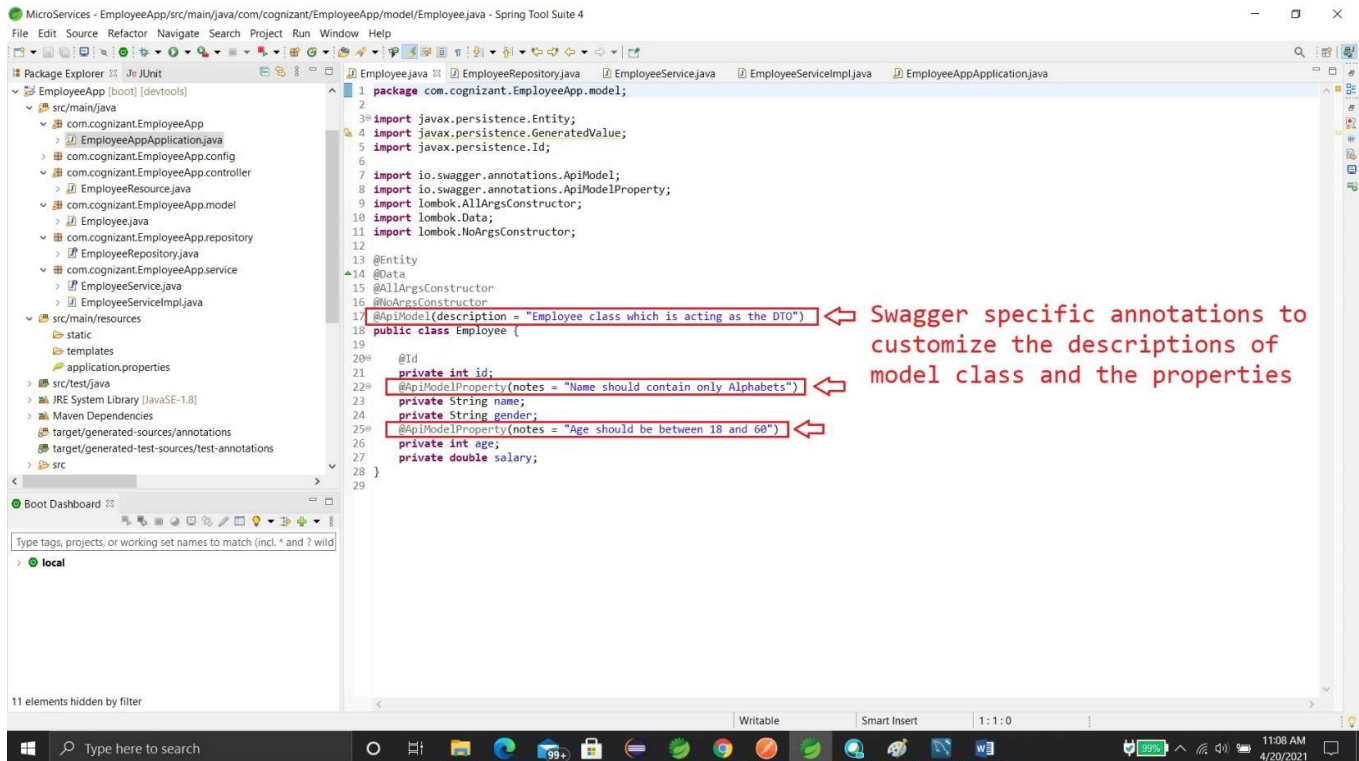
- Package Explorer:** Shows the project structure with packages like `com.cognizant.EmployeeApp`, `com.cognizant.EmployeeApp.config`, `com.cognizant.EmployeeApp.controller`, `com.cognizant.EmployeeApp.model`, `com.cognizant.EmployeeApp.repository`, and `com.cognizant.EmployeeApp.service`.
- Source Editor:** Displays the `SwaggerConfig.java` file with the following code:

```
1 package com.cognizant.EmployeeApp.config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.Configuration;
5
6 import springfox.documentation.builders.RequestHandlerSelectors;
7 import springfox.documentation.spi.DocumentationType;
8 import springfox.documentation.spring.web.plugins.Docket;
9 import springfox.documentation.swagger2.annotations.EnableSwagger2;
10
11 @Configuration
12 @EnableSwagger2
13 public class SwaggerConfig {
14
15     @Bean
16     public Docket api() {
17         return new Docket(DocumentationType.SWAGGER_2).select()
18             .apis(RequestHandlerSelectors.basePackage("com.cognizant.EmployeeApp")).build();
19     }
20 }
21
22
23
24
```
- Console:** Shows the application running successfully with the following output:

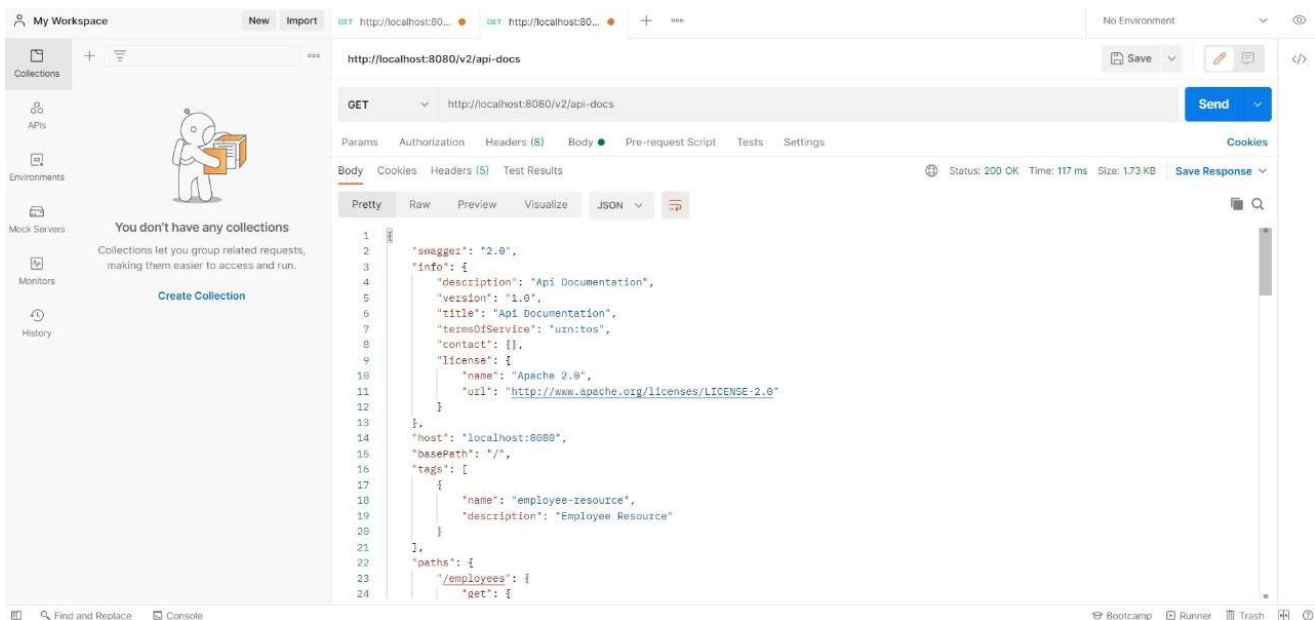
```
2021-04-20 09:48:07.626 INFO 18696 --- [ restartedMain] org.hibernate.Version : HH#000412: Hibernate ORM core version 5.4.30.Final
2021-04-20 09:48:08.247 INFO 18696 --- [ restartedMain] o.hibernate.annotations.common.Version : HC#00000001: Hibernate Commons Annotations {5.1.2.Fi
2021-04-20 09:48:08.595 INFO 18696 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-04-20 09:48:10.793 INFO 18696 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-04-20 09:48:10.886 INFO 18696 --- [ restartedMain] org.hibernate.dialect.Dialect : HH#000400: Using dialect: org.hibernate.dialect.MySQ
2021-04-20 09:48:13.993 INFO 18696 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HH#000490: Using JtaPlatform implementation: [org.hi
2021-04-20 09:48:14.022 INFO 18696 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
2021-04-20 09:48:14.201 INFO 18696 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2021-04-20 09:48:16.417 WARN 18696 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
2021-04-20 09:48:17.465 INFO 18696 --- [ restartedMain] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecuto
2021-04-20 09:48:18.179 INFO 18696 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context
2021-04-20 09:48:19.046 INFO 18696 --- [ restartedMain] c.c.EmployeeApp.EmployeeAppApplication : Started EmployeeAppApplication in 20.848 seconds (JV
2021-04-20 09:48:32.542 INFO 18696 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherSer
2021-04-20 09:48:32.542 INFO 18696 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-04-20 09:48:32.548 INFO 18696 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 4 ms
```



Step 3:- use Swagger specific annotations to customize the descriptions of model class and the properties.

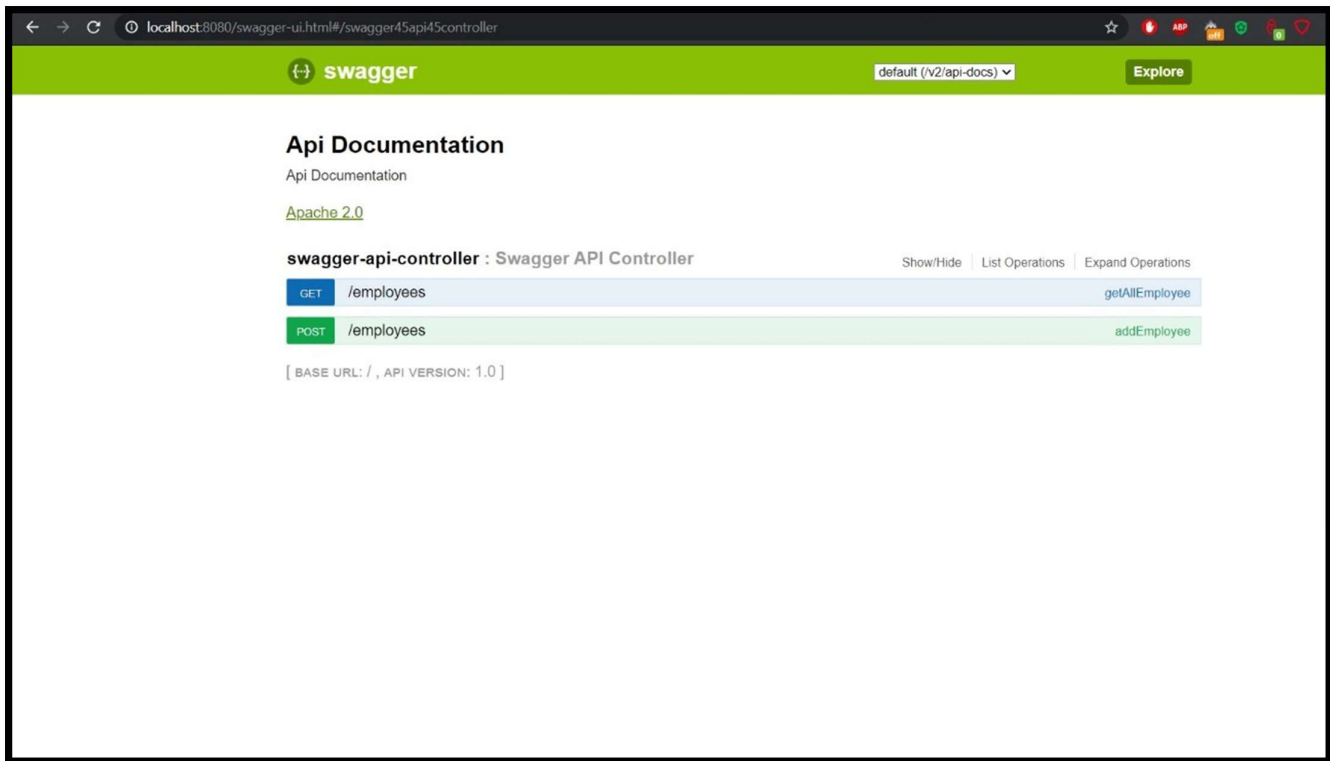


"localhost:8080/v2/api-docs" and you can see the complete API documentation of your service.



Now, hit the URL in your web browser and see the Swagger API functionalities.

**<http://localhost:8080/swagger-ui.html>**



## Spring MVC Client For Spring REST Service

Note:- We have already created a microservice(employee) in our local System. Now, we are just creating another microservice which will consume the rest service of our previous employee microservice.

First we have to add “openfeign” dependency

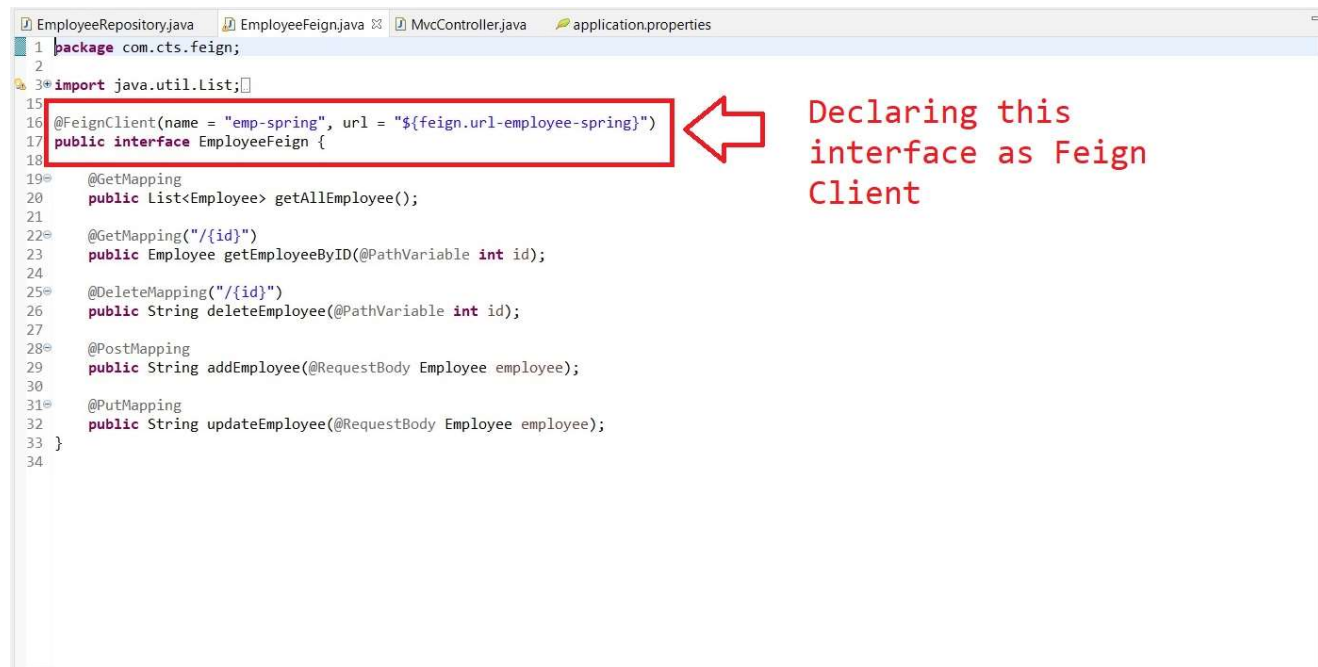
```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Feign Client dependency

## Note:-

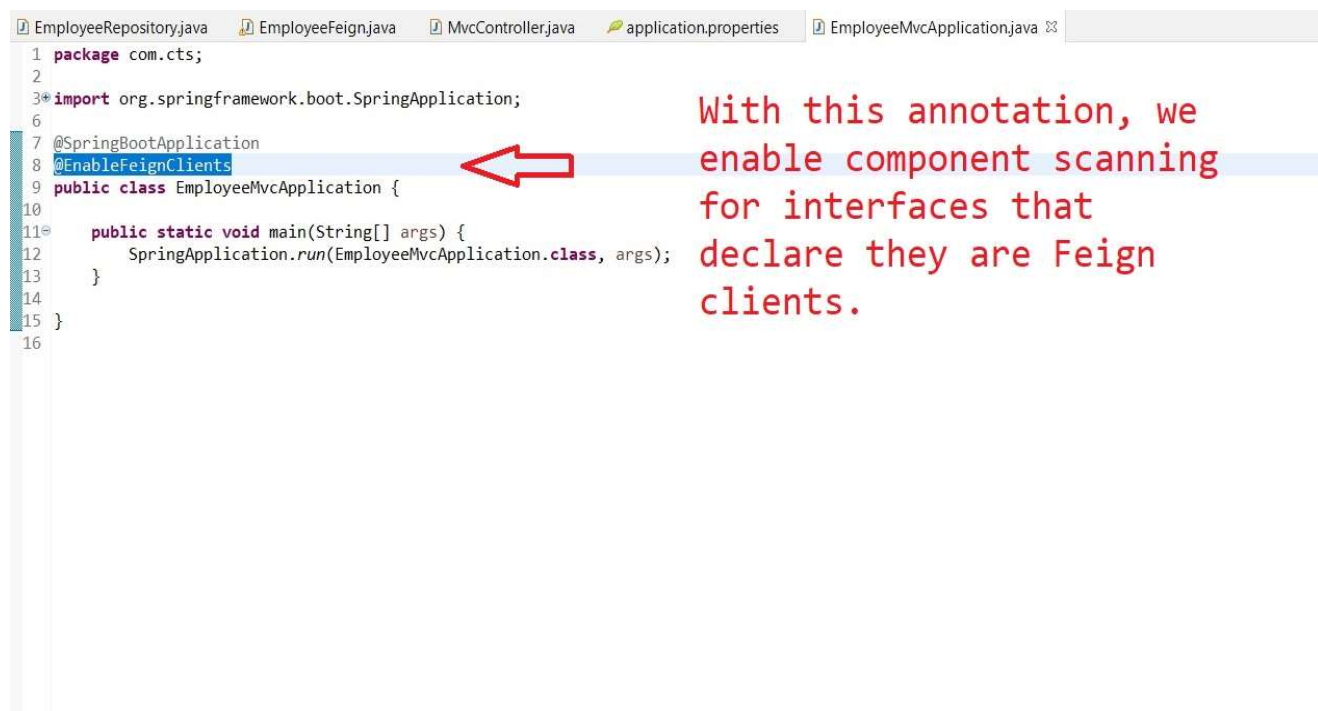
The *value* argument passed in the `@FeignClient` annotation is a mandatory, arbitrary client name, while with the *url* argument, we specify the API base URL.

Furthermore, since this interface is a Feign client, we can use the Spring Web annotations to declare the APIs that we want to reach out to.



```
1 package com.cts.feign;
2
3 import java.util.List;
4
5 @FeignClient(name = "emp-spring", url = "${feign.url-employee-spring}")
6 public interface EmployeeFeign {
7
8     @GetMapping
9     public List<Employee> getAllEmployee();
10
11     @GetMapping("/{id}")
12     public Employee getEmployeeByID(@PathVariable int id);
13
14     @DeleteMapping("/{id}")
15     public String deleteEmployee(@PathVariable int id);
16
17     @PostMapping
18     public String addEmployee(@RequestBody Employee employee);
19
20     @PutMapping
21     public String updateEmployee(@RequestBody Employee employee);
22 }
```

Declaring this interface as Feign Client



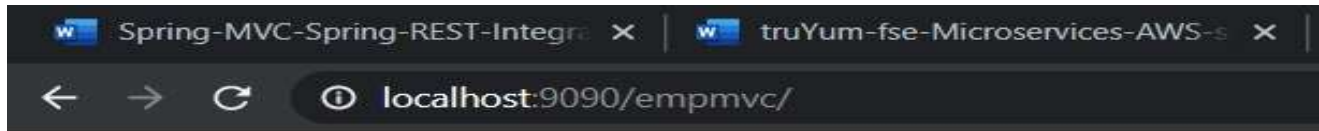
```
1 package com.cts;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 @EnableFeignClients
7 public class EmployeeMvcApplication {
8
9     public static void main(String[] args) {
10         SpringApplication.run(EmployeeMvcApplication.class, args);
11     }
12 }
```

With this annotation, we enable component scanning for interfaces that declare they are Feign clients.

With this annotation, we enable component scanning for interfaces that declare they are Feign clients.



Output:-

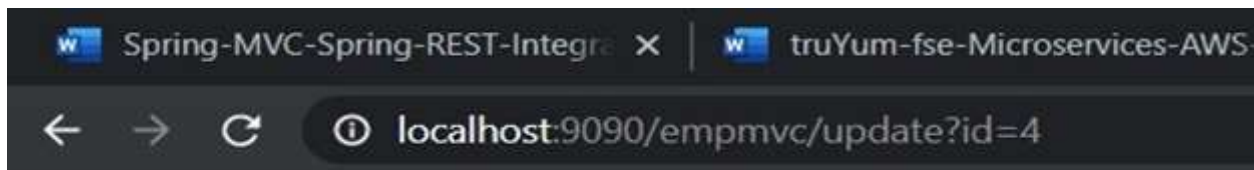


## Employee Details

Add Employee

Employee Id	Name	Gender	Age	Salary	Action
1	Manu	Male	23	34000	<a href="#">Delete</a> <a href="#">Update</a>
2	Chitra	Female	33	40000	<a href="#">Delete</a> <a href="#">Update</a>
3	Binoy	Male	27	40000	<a href="#">Delete</a> <a href="#">Update</a>
4	Anita	Male	32	41000	<a href="#">Delete</a> <a href="#">Update</a>
5	Siddhartha	Male	23	40000	<a href="#">Delete</a> <a href="#">Update</a>

output:-



Employee Name

Employee Gender  ▼

Employee age

Employee salary