Spring RESTful API Lab-01-Review

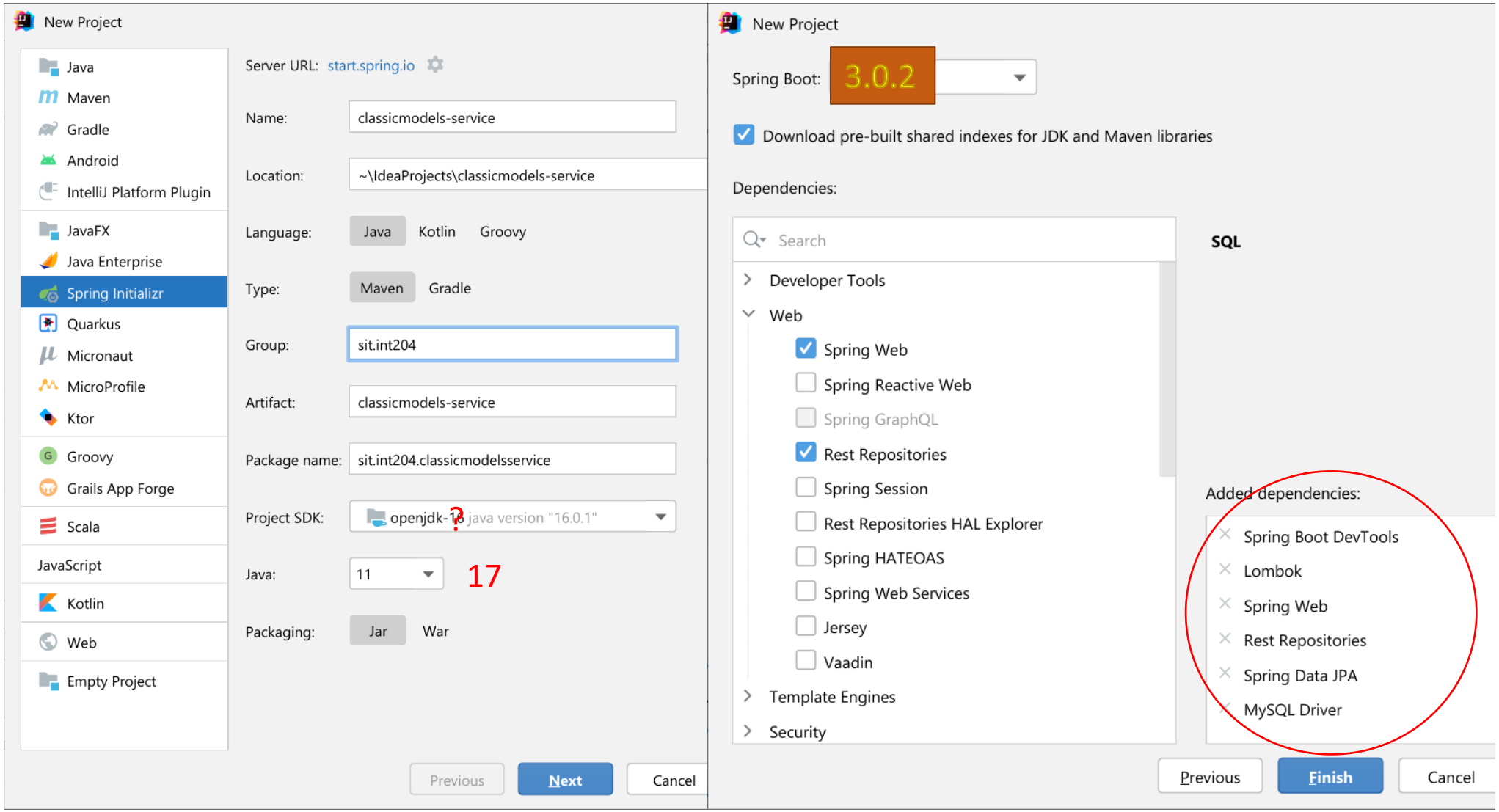
By

Pichet Limvajiranan

RESTful Resource

URI	HTTP verb	Description
api/offices	GET	Get all office
api/offices/1	GET	Get an office with id = 1
api/offices/1/employees	GET	Get all employee for office id = 1
api/offices	POST	Add new office
api/offices/1	PUT	Update an office with id = 1
api/offices/1	DELETE	Delete an office with id = 1

Step 1: Initializing a Spring Boot Project



Step 2: Connecting Spring Boot to the Database

ceController.java | application.properties | Office.java

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.password=143900
spring.datasource.username=root
spring.datasource.url=jdbc:mysql://localhost:3306/classicmodels
spring.jpa.hibernate.ddl-auto=none
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.nam
```

```
org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

ถ้าไม่กำหนด spring.jpa.hibernate.naming.physical-strategy

จะใช้ convention ดังนี้

การระบุ `@Column(name)` ใน `enitity clase` ต้องพิมพ์เป็นตัวเล็กหมด หรือถ้าระบุคอลัมน์เป็น `camel case` ชื่อฟิลด์ ใน ตารางต้องแยกคำด้วย ขีดล่าง (_)

Step 3: Creating an Office Model (1)

```
@Getter  @Setter
@Entity @Table(name = "offices")
public class Office {
   @Id
   @Column(name = "officeCode", nullable = false, length = 10)
   private String id;
   @Column(name = "city", nullable = false, length = 50)
   private String city;
   @Column(name = "phone", nullable = false, length = 50)
   private String phone;
   @Column(name = "addressLine1", nullable = false, length = 50)
   private String addressLine1;
```

Step 3: Creating an Office Model (2)

```java
@Column(name = "addressLine2", length = 50)
private String addressLine2;
@Column(name = "state", length = 50)
private String state;
@Column(name = "country", nullable = false, length = 50)
private String country;
@Column(name = "postalCode", nullable = false, length = 15)
private String postalCode;
@Column(name = "territory", nullable = false, length = 10)
private String territory;

@JsonIgnore
@OneToMany(mappedBy = "office")
private Set<Employee> employees = new LinkedHashSet<>();
```

```
{
  "id": "1",
  "city": "San Francisco",
    :
  "territory": "NA",
  "employees": [
   {
     "id": 1002,
     "lastName": "Murphy",
     :
   },
   {
     "id": 1166,
     "lastName": "Thompson",
   },
```

```
private String lastName;
@Column(name = "firstName", nullable = false, length = 50)
private String firstName;
@Column(name = "extension", nullable = false, length = 10)
private String extension;
@Column(name = "email", nullable = false, length = 100)
private String email;

@JsonIgnore
@ManyToOne
@JoinColumn(name = "reportsTo")
private Employee employees;

@Column(name = "jobTitle", nullable = false, length = 50)
private String jobTitle;
```

Step 3: Creating an Employee Model (1)

```java
@Getter @Setter
@Entity @Table(name = "employees")
public class Employee {
   @Id
   @Column(name = "employeeNumber", nullable = false)
   private Integer id;

   @JsonIgnore
   @ManyToOne
   @JoinColumn(name = "office")
   private Office office;

   @Column(name = "lastName", nullable = false, length = 50)
```

```
{
 "id": 1076,
 "lastName": "Firrelli",
 "firstName": "Jeff",
   :
 "office": {
    "id": "1",
    "city": "San Francisco",
    "phone": "+1 650 219 4782",
      :
    "territory": "NA"
  },
 "jobTitle": "VP Marketing“
},
```

Step 4: Creating Repository Classes

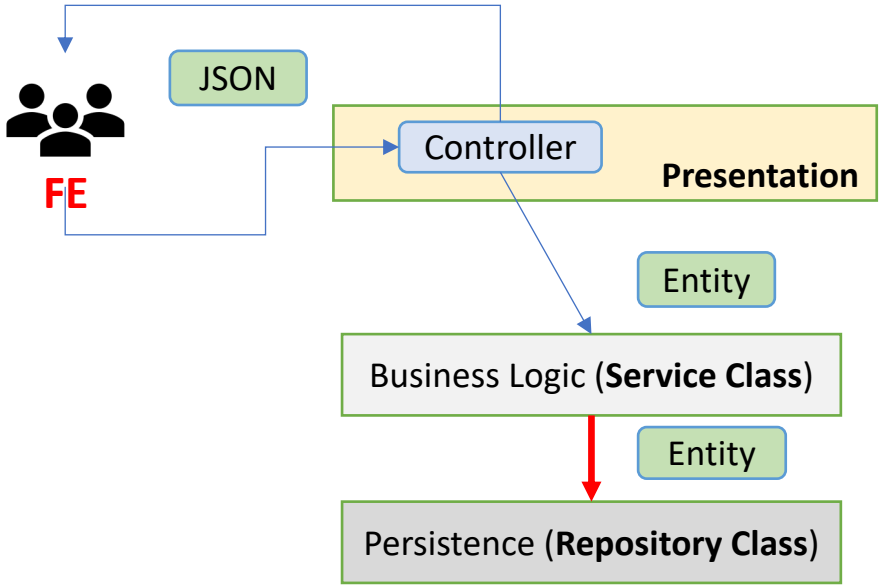
```
public interface OfficeRepository extends JpaRepository<Office, String> {

}
```

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}
```

```
public interface CustomerRepository extends JpaRepository<Customer, Integer> {

}
```

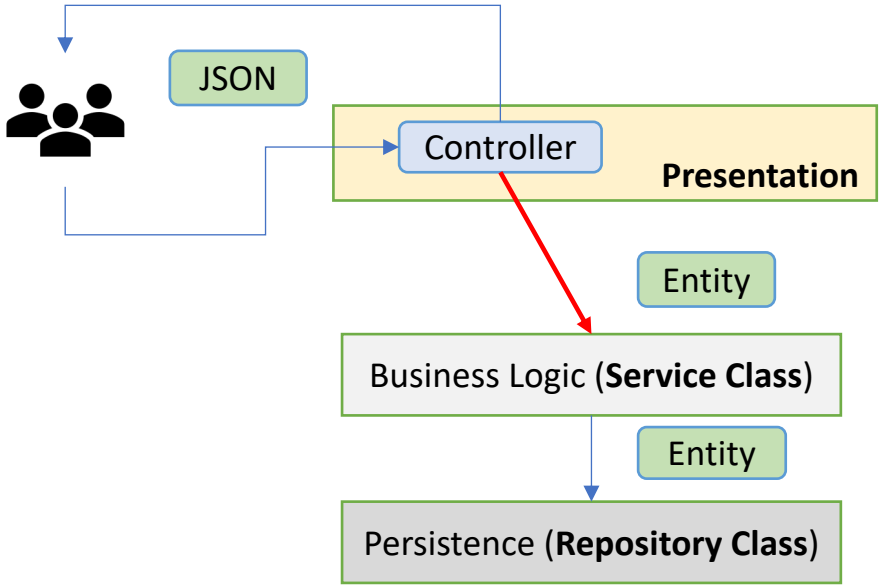
Step 5: Creating Service Class

```
@Service

public class OfficeService {
    @Autowired
    private OfficeRepository repository;

    public List<Office> getAllOffice() {
        return repository.findAll();
    }

    public Office getOffice(String officeCode) {
        return repository.findById(officeCode).orElseThrow(
                () -> new HttpClientErrorException(HttpStatus.NOT_FOUND,
                        "Office Id " + officeCode + " DOES NOT EXIST !!!") {
                }
        );
    }


    public Office createNewOffice(Office office) {
        return repository.saveAndFlush(office);
    }
```



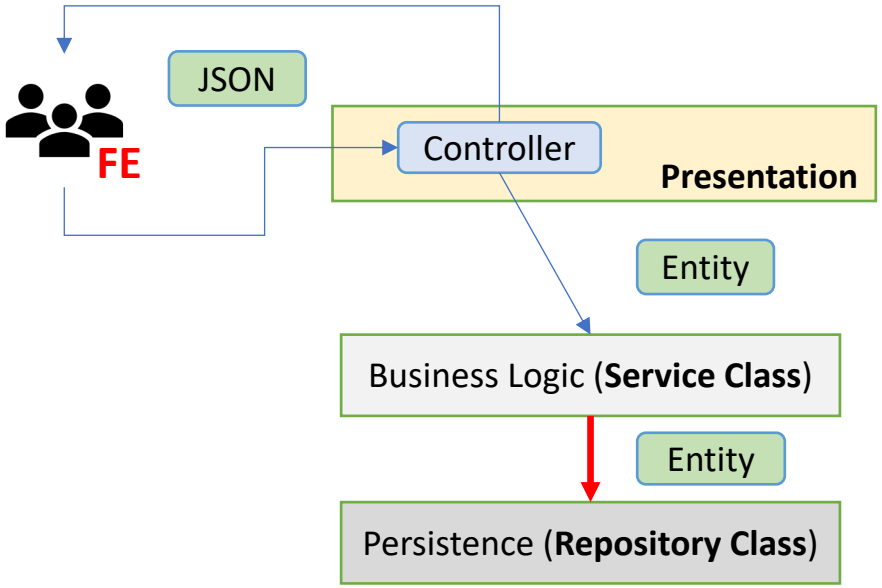
Step 6: Creating Controller

```
@RestController
@RequestMapping("/api/offices")

public class OfficeController {
    @Autowired
    private OfficeService service;

    @GetMapping("")
    public List<Office> getAllOffices() {
        return service.getAllOffice();
    }

    @GetMapping("/{officeCode}")
    public Office getOfficeById(@PathVariable String officeCode) {
        return service.getOffice(officeCode);
    }


    @PostMapping("")
    public Office addNewOffice(@RequestBody Office office) {
        return service.createNewOffice(office);
    }
```



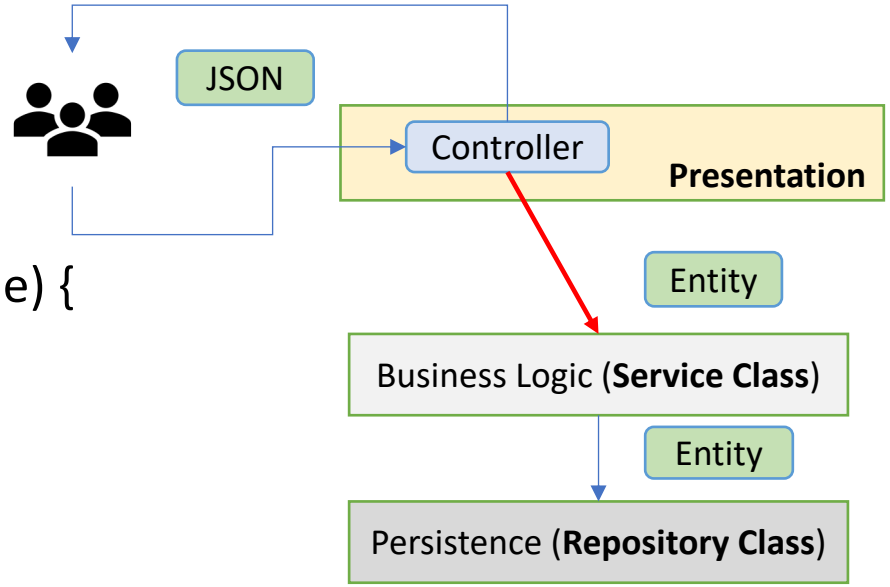
Step 5: Creating Service Class (cont.)

```
    public void removeOffice(String officeCode) {
        Office office = repository.findById(officeCode).orElseThrow(
                () -> new HttpClientErrorException(HttpStatus.NOT_FOUND, "Office Id " + officeCode + " DOES NOT EXIST !!!")
        );
        repository.delete(office);
    }

    public Office updateOffice(String officeCode, Office office) {
        Office existingOffice = repository.findById(officeCode).orElseThrow(
                () -> new HttpClientErrorException(HttpStatus.NOT_FOUND,
                        "Office Id " + officeCode + " DOES NOT EXIST !!!")
        );
        existingOffice.setCountry(office.getCountry());
        existingOffice.setAddressLine1(office.getAddressLine1());
        existingOffice.setAddressLine2(office.getAddressLine2());
        existingOffice.setPhone(office.getPhone());
        return repository.saveAndFlush(existingOffice);
    }
}
```



Step 6: Creating Controller (cont.)

```
    @PutMapping("/{officeCode}")
    public Office updateOffice(@RequestBody Office office, @PathVariable String officeCode) {
        return service.updateOffice(officeCode, office);
    }


    @DeleteMapping("/{officeCode}")
     public void removeOffice(@PathVariable String officeCode) {
         service.removeOffice(officeCode);
     }
}
```



Step 8: Testing the APIs (GET)

GET localhost:8080/api/offices

Params Authorization Headers (7) Body Pre-request Script

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": "1",
        "city": "San Francisco",
        "phone": "+1 650 219 4782",
        "addressLine1": "100 Market Street",
        "addressLine2": "Suite 300",
        "state": "CA",
        "country": "USA",
        "postalCode": "94080",
        "territory": "NA"
    },
    {
        "id": "2",
        "city": "Boston",
        "phone": "+1 215 837 0825",
        "addressLine1": "1550 Court Place",
```


GET localhost:8080/api/offices/7

Params Authorization Headers (7) Body Pre-rec

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON


```
{
    "id": "7",
    "city": "London",
    "phone": "+44 20 7877 2041",
    "addressLine1": "25 Old Broad Street",
    "addressLine2": "Level 7",
    "state": null,
    "country": "UK",
    "postalCode": "EC2N 1HN",
    "territory": "EMEA"
}
```

Step 8: Testing the APIs (POST)

POST localhost:8080/api/offices/

Params Authorization Headers (10) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1  {
2      "id": "8",
3      "city": "Bangkok",
4      "phone": "+44 20 7877 2041",
5      "addressLine1": "25 Old Broad Street",
6      "addressLine2": "Level 7",
7      "state": "",
8      "country": "UK",
9      "postalCode": "EC2N 1HN",
10     "territory": "EMEA"
11 }
```

Step 8: Testing the APIs (DELETE)

DELETE localhost:8080/api/offices/9

Params Authorization Headers (7) Body Pre-request Scri

Status: 200 OK Time: 49 ms Size: 123 B

DELETE localhost:8080/api/offices/11

Params Authorization Headers (7) Body Pre-request Script Tests

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON


```
1 {
2     "timestamp": "2022-02-20T15:37:22.683+00:00",
3     "status": 500,
4     "error": "Internal Server Error",
5     "trace": "java.lang.RuntimeException: 11 does not exist !!!\r
```

Step 8: Testing the APIs (PUT)

PUT localhost:8080/api/offices/11

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1  {
2      "id": null,
3      "city": "Songkhla",
4      "phone": "+44 20 7877 2041",
5      "addressLine1": "25 Old Broad Street",
6      "addressLine2": "Level 7",
7      "state": null,
8      "country": "UK",
9      "postalCode": "EC2N 1HN",
10     "territory": "EMEA"
11 }
```

Do It Yourself

- Create Services, Controllers for resources:

URI	HTTP verb	Description
api/offices/1/employees	GET	Get all employee for office id = 1
api/employees	GET	Get all employees
api/employees/1	GET	Get an employee with id = 1
api/offices/1/employees	POST	Add new employee for office id = 1
api/employees/1	PUT	Update an employee with id = 1
api/employees/1	DELETE	Delete an employee with id = 1