# Product Java Springboot Assessment Scenario

**Name: Abhishek Kumar Gupta**
**PS No.: 61095728**
**Batch: ACE-13**
**Team: 1**

---

## Scenario case 1

Create a REST API for resource Course. The Course Entity has following fields:

- Unique ID
- Name
- Description
- Price
- CreatedAtDate

The REST API must expose all CRUD methods (GET, POST, DELETE, PUT). Following functionalities should also be exposed:

- Find a course by its name
- Search for a text contained in description
- Courses with price greater than a value.
- Change the price of course
- Find All Courses created on a specific Date
- Display courses sorted by Name

The REST API must fulfil following requirements:

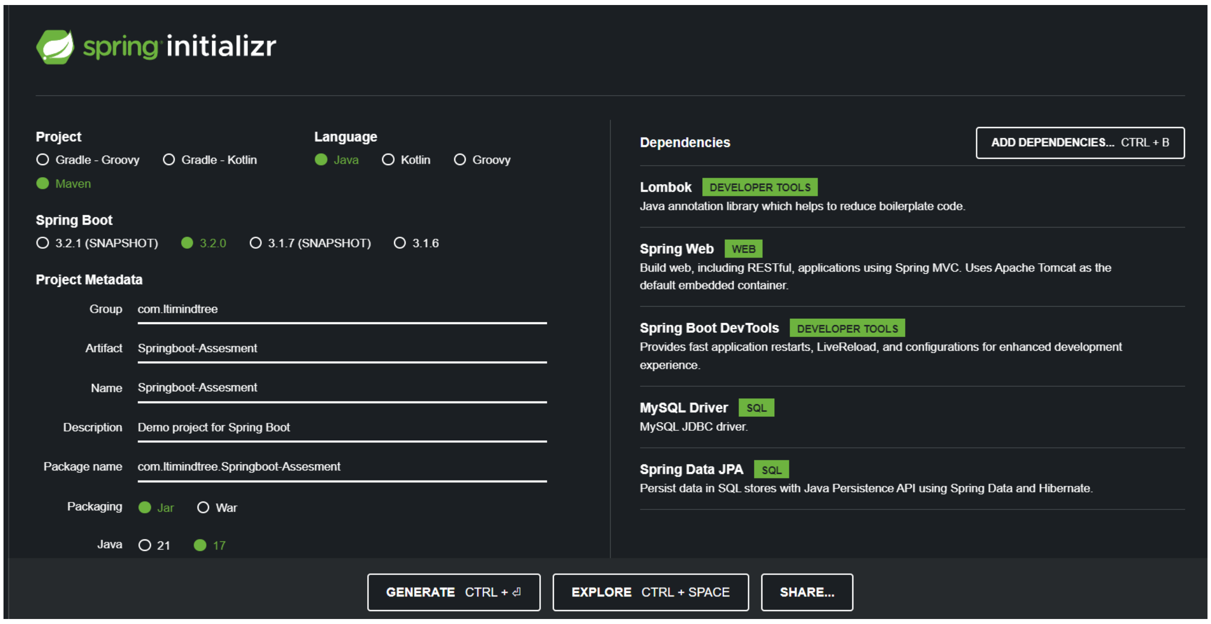
- Use appropriate status codes
- Use exception handling
- Must expose documentation using Swagger
- Create 2 unit test using Mocks
- Use separate packages for Controllers, Entities, Repositories etc.

# Solution

1. Creating a New Springboot project using Spring initializer.

   Package -> com.ltimindtree
   Name -> Springboot-Assesment

   Dependencies:
   - Lombok
   - Spring Web
   - Spring Boot Dev Tools
   - MySQL Driver
   - Spring data JPA



spring initializr

Project
○ Gradle - Groovy ○ Gradle - Kotlin ● Maven

Language
● Java ○ Kotlin ○ Groovy

Spring Boot
○ 3.2.1 (SNAPSHOT) ● 3.2.0 ○ 3.1.7 (SNAPSHOT) ○ 3.1.6

Project Metadata

| | |
|---|---|
| Group | com.ltimindtree |
| Artifact | Springboot-Assesment |
| Name | Springboot-Assesment |
| Description | Demo project for Spring Boot |
| Package name | com.ltimindtree.Springboot-Assesment |
| Packaging | ● Jar ○ War |
| Java | ○ 21 ● 17 |

Dependencies

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

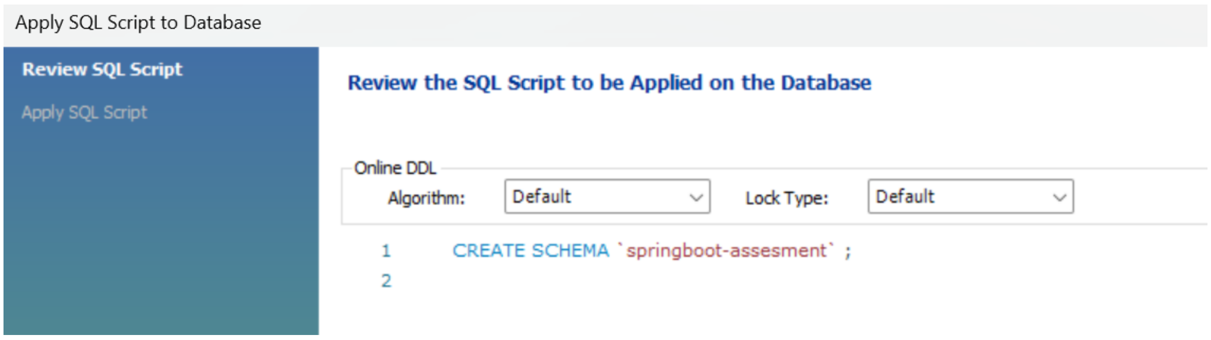
Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

MySQL Driver SQL
MySQL JDBC driver.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

2. Creating New Database in MYSQL.



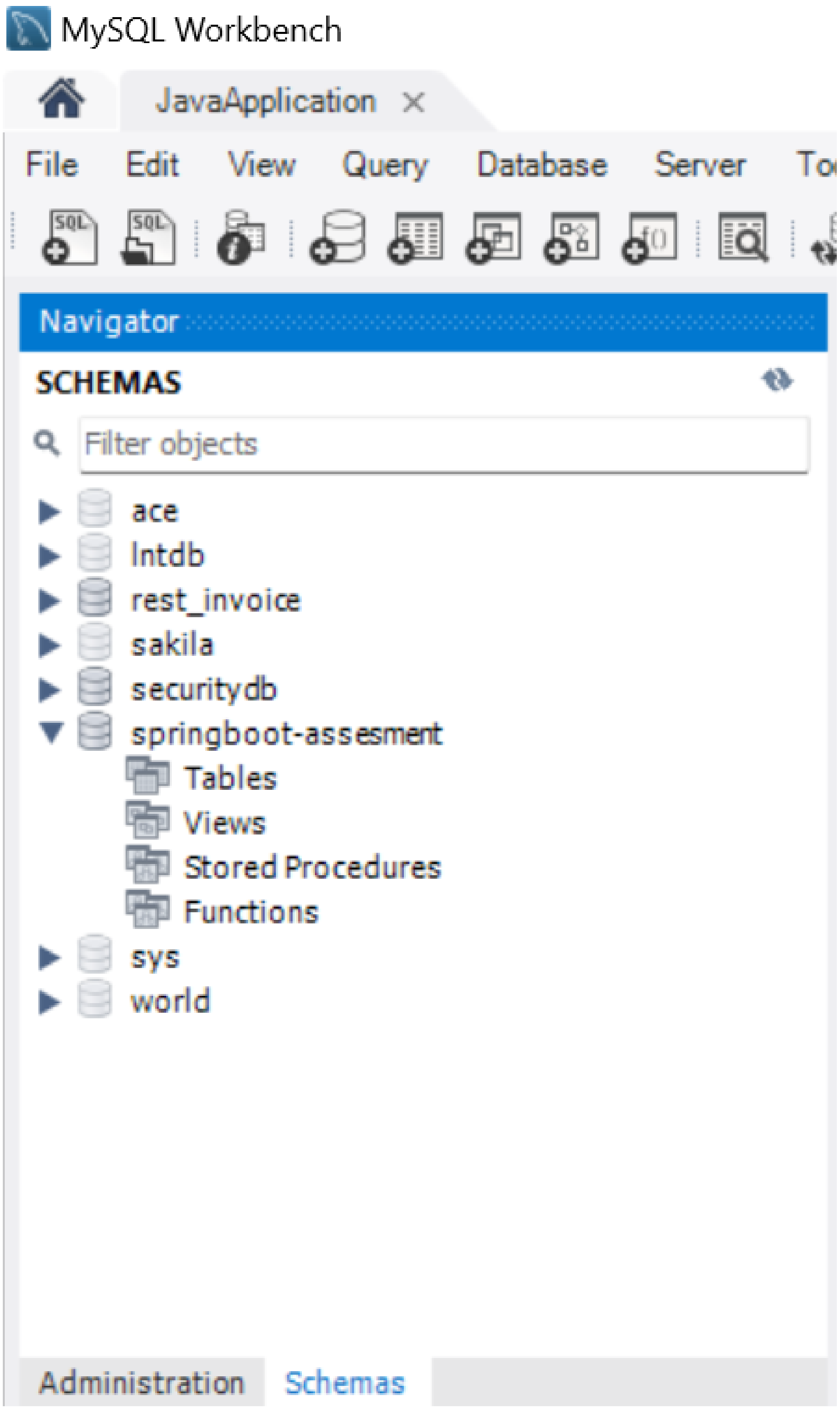
Apply SQL Script to Database

Review SQL Script
Apply SQL Script

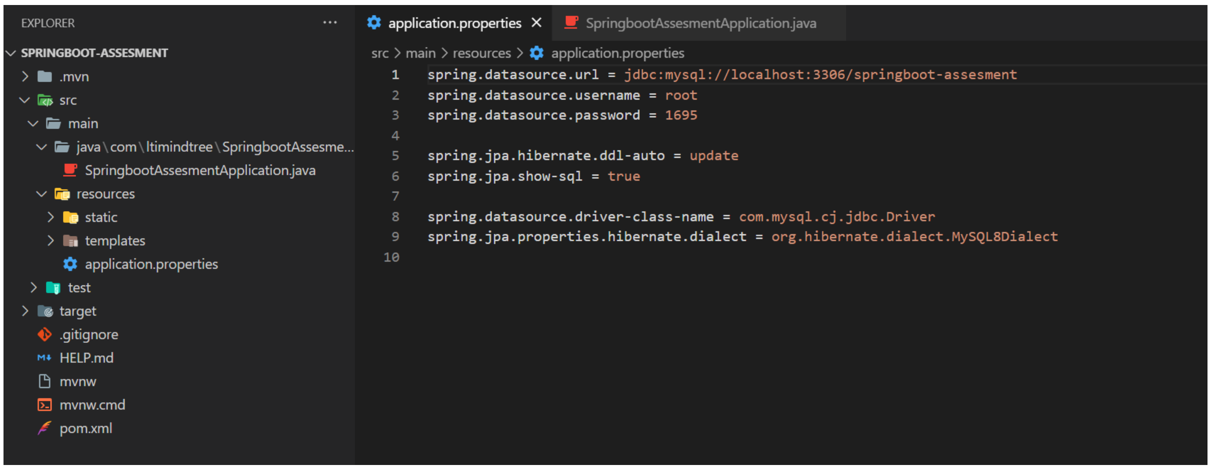
Review the SQL Script to be Applied on the Database

Online DDL
Algorithm: Default Lock Type: Default

```
CREATE SCHEMA `springboot-assesment` ;
```


MySQL Workbench
JavaApplication
File
Edit
View
Query
Database
Server
To
Navigator
SCHEMAS
Filter objects
ace
lntdb
rest_invoice
sakila
securitydb
springboot-assesment
Tables
Views
Stored Procedures
Functions
sys
world
Administration
Schemas

3. Configured "application.properties" file and connect Spring application with MySQL. Now Application is Connected with Database.


EXPLORER
SPRINGBOOT-ASSESMENT
.mvn
src
main
java \ com \ ltimindtree \ SpringbootAssesme...
SpringbootAssesmentApplication.java
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml

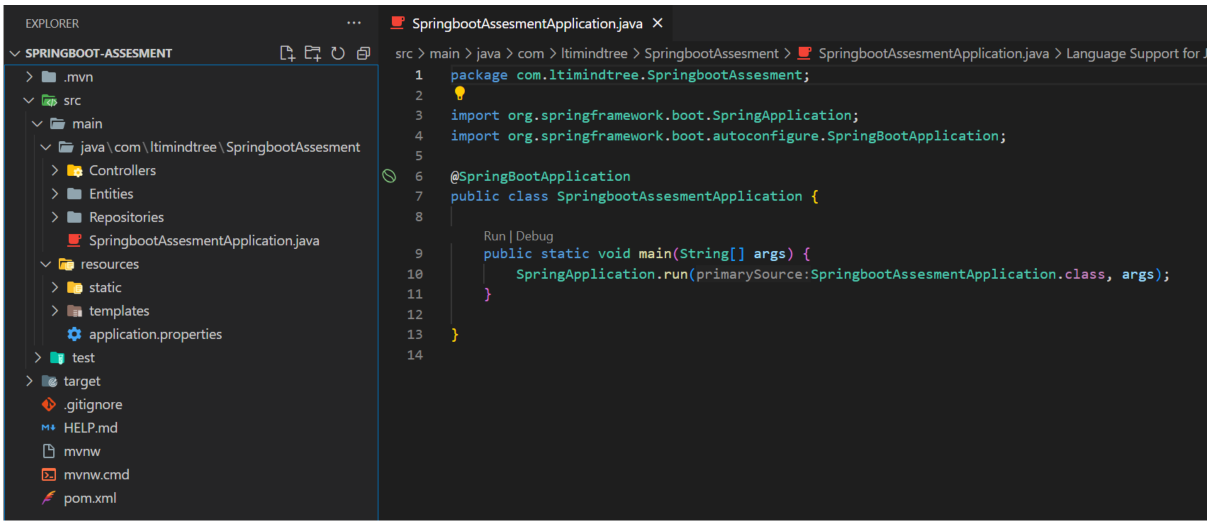
application.properties — SpringbootAssesmentApplication.java

src > main > resources > application.properties

```
spring.datasource.url = jdbc:mysql://localhost:3306/springboot-assesment
spring.datasource.username = root
spring.datasource.password = 1695

spring.jpa.hibernate.ddl-auto = update
spring.jpa.show-sql = true

spring.datasource.driver-class-name = com.mysql.cj.jdbc.Driver
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL8Dialect

```

4. Created 3 packages:

- Controllers
- Entities
- Repositories

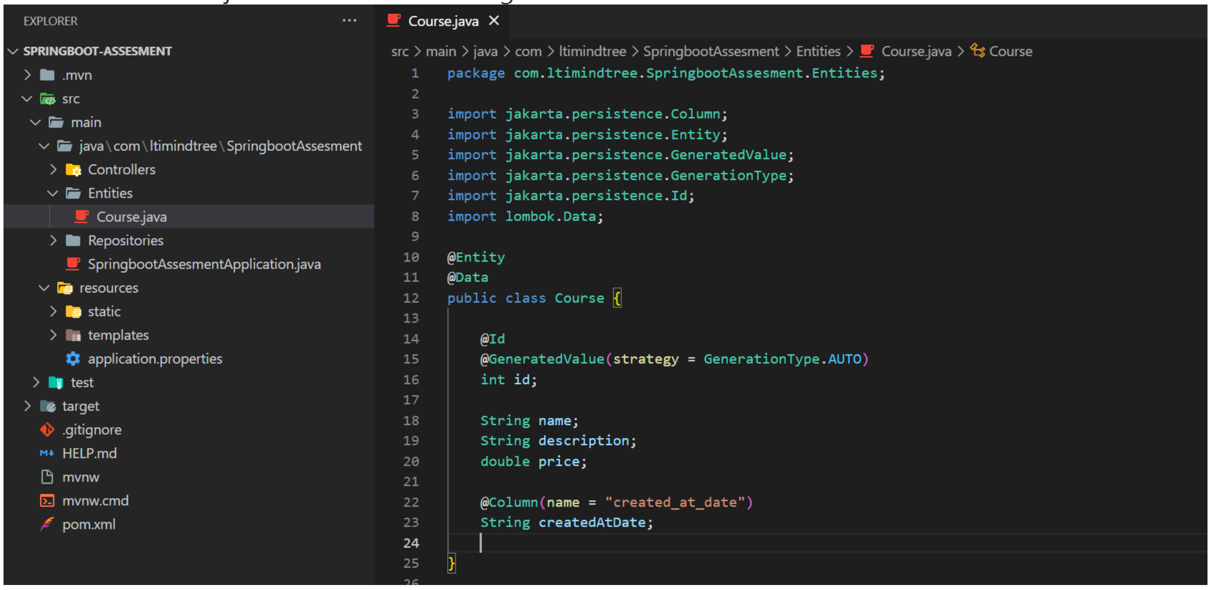

EXPLORER
SPRINGBOOT-ASSESMENT
.mvn
src
main
java \ com \ ltimindtree \ SpringbootAssesment
Controllers
Entities
Repositories
SpringbootAssesmentApplication.java
resources
static
templates
application.properties
test
target
.gitignore
HELP.md
mvnw
mvnw.cmd
pom.xml

SpringbootAssesmentApplication.java

src > main > java > com > ltimindtree > SpringbootAssesment > SpringbootAssesmentApplication.java > Language Support for J

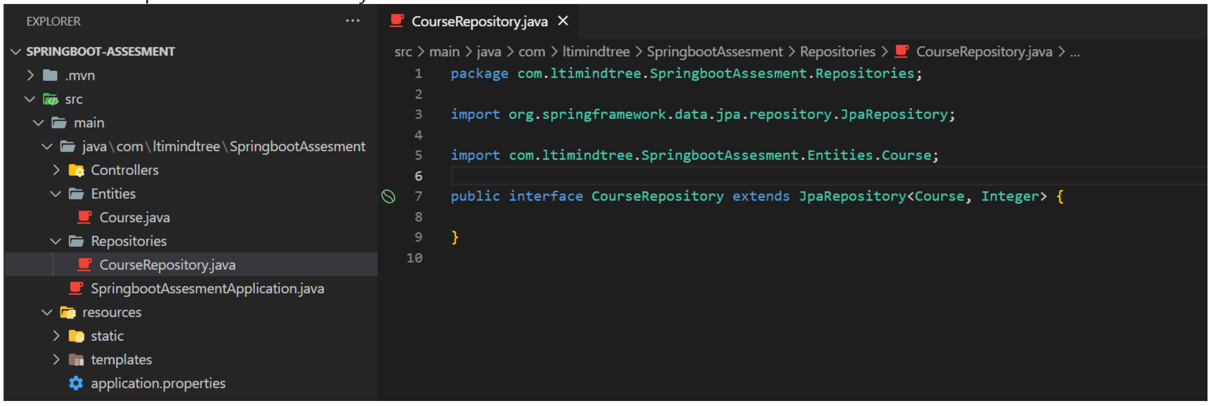
```
package com.ltimindtree.SpringbootAssesment;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class SpringbootAssesmentApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource:SpringbootAssesmentApplication.class, args);
    }

}
```

5. Created "Course.java" in Entities Package.



```java
package com.ltimindtree.SpringbootAssesment.Entities;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.Data;

@Entity
@Data
public class Course {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    int id;

    String name;
    String description;
    double price;

    @Column(name = "created_at_date")
    String createdAtDate;

}
```

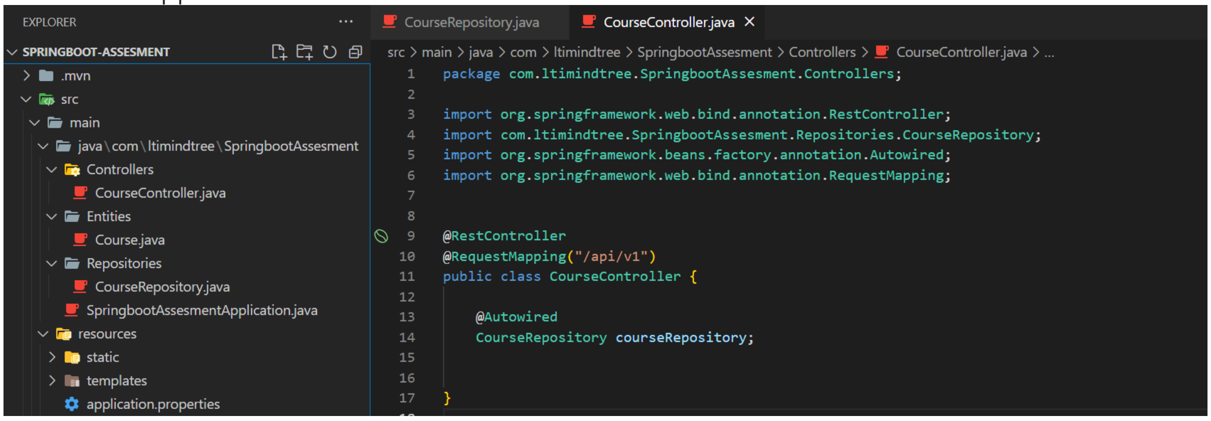
6. Created "CourseRepository.java" interface and extended "JpaRepository" to handle Database operation smoothly.



EXPLORER
SPRINGBOOT-ASSESMENT
.mvn
src
main
java \ com \ ltimindtree \ SpringbootAssesment
Controllers
Entities
Course.java
Repositories
CourseRepository.java
SpringbootAssesmentApplication.java
resources
static
templates
application.properties
CourseRepository.java
src > main > java > com > ltimindtree > SpringbootAssesment > Repositories > CourseRepository.java > ...

```java
package com.ltimindtree.SpringbootAssesment.Repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import com.ltimindtree.SpringbootAssesment.Entities.Course;

public interface CourseRepository extends JpaRepository<Course, Integer> {

}
```

7. Created "CourseController.java" file in 'Controllers' Package to create servlet, and Rest API for the application.

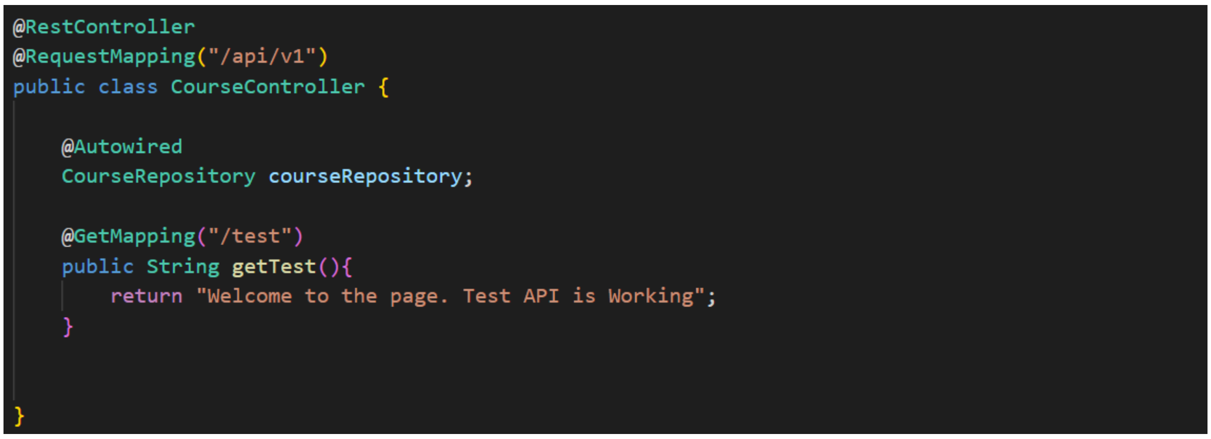


```java
package com.ltimindtree.SpringbootAssesment.Controllers;

import org.springframework.web.bind.annotation.RestController;
import com.ltimindtree.SpringbootAssesment.Repositories.CourseRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;


@RestController
@RequestMapping("/api/v1")
public class CourseController {

    @Autowired
    CourseRepository courseRepository;


}
```

8. Created a test API for testing if API is working or not.

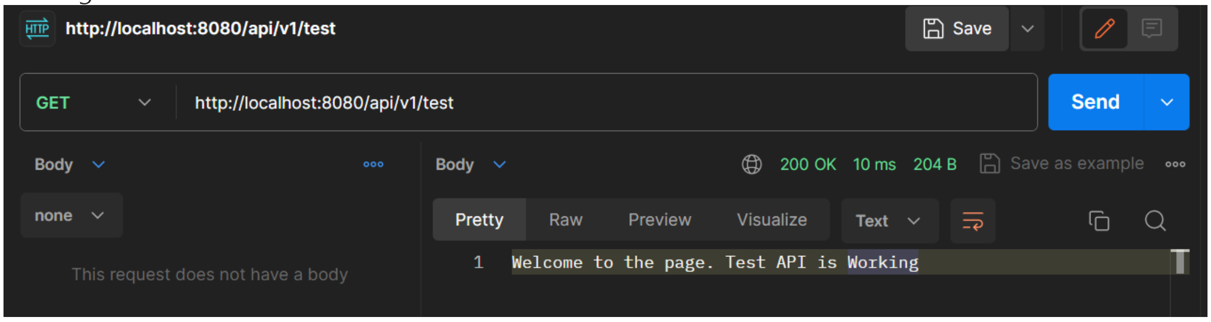
   Route -> ‘http://localhost:8080/api/v1/test’.
   Method -> GET.
   Result -> Successfull

   Java Servlet



```java
@RestController
@RequestMapping("/api/v1")
public class CourseController {

    @Autowired
    CourseRepository courseRepository;

    @GetMapping("/test")
    public String getTest(){
        return "Welcome to the page. Test API is Working";
    }


}
```

   Testing in Postman

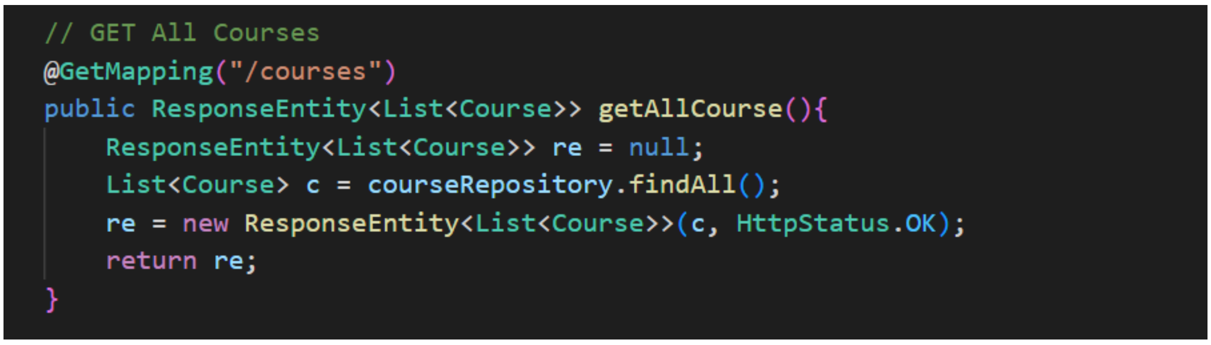


```
http://localhost:8080/api/v1/test
GET    http://localhost:8080/api/v1/test
Body: none — This request does not have a body
Body    200 OK    10 ms    204 B
Pretty | Raw | Preview | Visualize | Text
1  Welcome to the page. Test API is Working
```

9. Created ‘GET’ servlet to find all the courses.

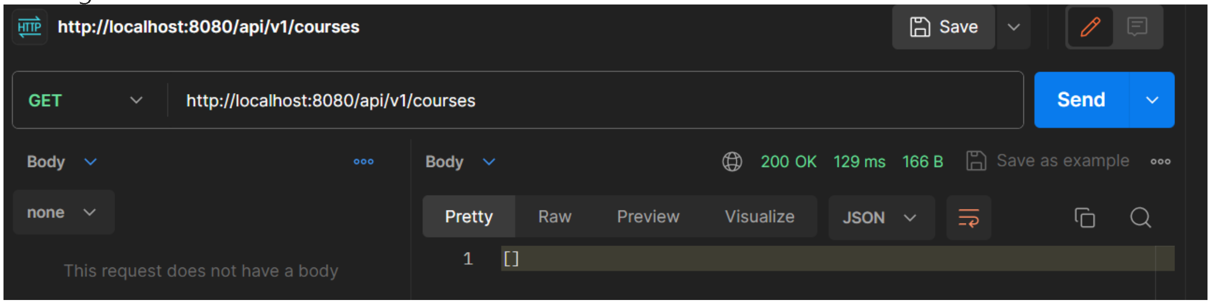
   Route -> ‘http://localhost:8080/api/v1/courses.
   Method -> GET.
   Result -> Successful.
   Response Code -> 200 (OK).

   Java Servlet



```java
// GET All Courses
@GetMapping("/courses")
public ResponseEntity<List<Course>> getAllCourse(){
    ResponseEntity<List<Course>> re = null;
    List<Course> c = courseRepository.findAll();
    re = new ResponseEntity<List<Course>>(c, HttpStatus.OK);
    return re;
}
```

Testing in Postman



GET http://localhost:8080/api/v1/courses

Body: none — This request does not have a body

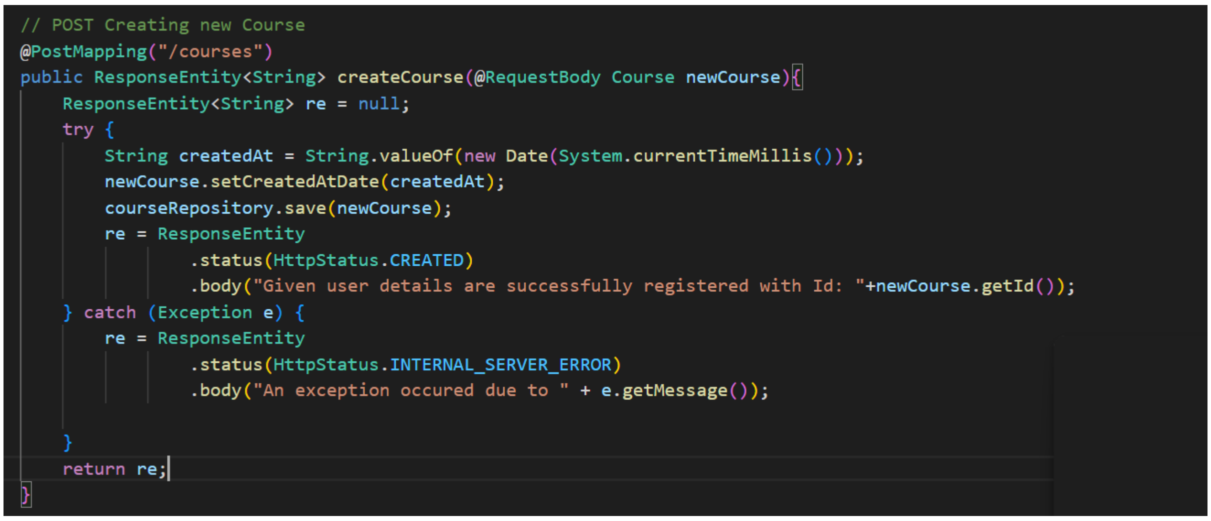
Response: 200 OK 129 ms 166 B

```
[]
```

10. Created 'POST' servlet to create a new Course.

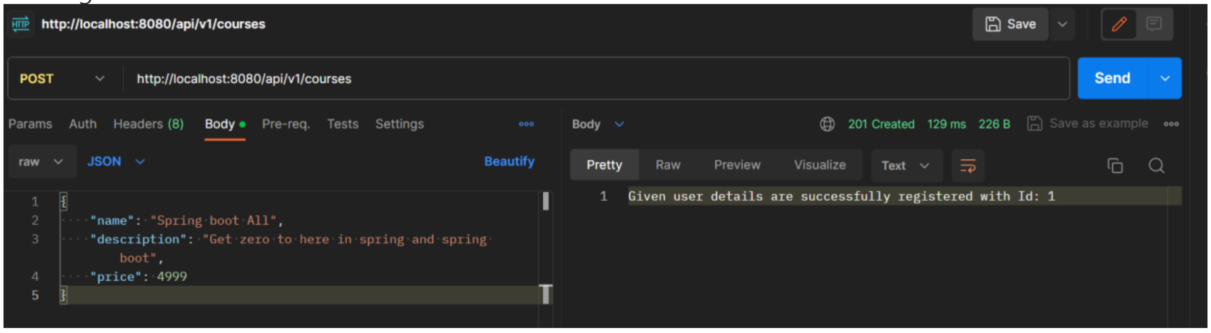
Route -> 'http://localhost:8080/api/v1/courses.
Method -> POST.
Result -> Successful.
Response Code -> 201 (CREATED).

Java Servlet



```java
// POST Creating new Course
@PostMapping("/courses")
public ResponseEntity<String> createCourse(@RequestBody Course newCourse){
    ResponseEntity<String> re = null;
    try {
        String createdAt = String.valueOf(new Date(System.currentTimeMillis()));
        newCourse.setCreatedAtDate(createdAt);
        courseRepository.save(newCourse);
        re = ResponseEntity
                .status(HttpStatus.CREATED)
                .body("Given user details are successfully registered with Id: "+newCourse.getId());
    } catch (Exception e) {
        re = ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body("An exception occured due to " + e.getMessage());

    }
    return re;
}
```

Testing in Postman



POST http://localhost:8080/api/v1/courses

```json
{
    "name": "Spring boot All",
    "description": "Get zero to here in spring and spring boot",
    "price": 4999
}
```

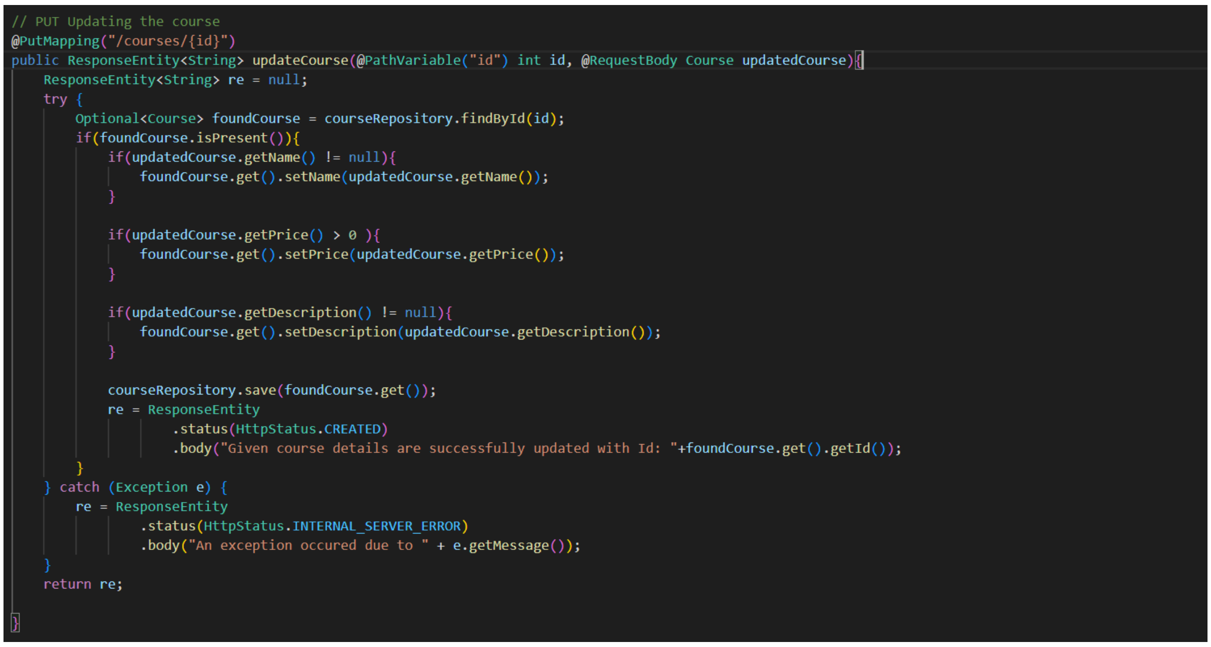
Response: 201 Created 129 ms 226 B

```
Given user details are successfully registered with Id: 1
```

11. Created ‘PUT’ servlet for updating cource by ‘id’.

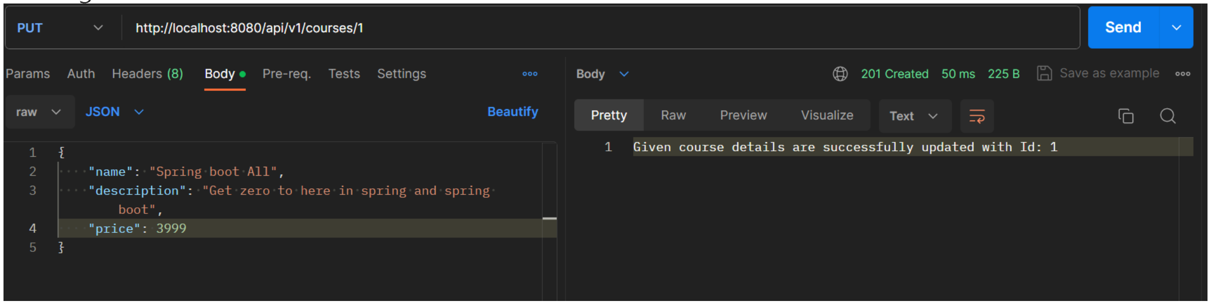
    Route -> ‘http://localhost:8080/api/v1/courses/{id}.
    Method -> PUT.
    Result -> Successful.
    Response Code -> 202 (ACCEPTED).

Java servlet



```java
// PUT Updating the course
@PutMapping("/courses/{id}")
public ResponseEntity<String> updateCourse(@PathVariable("id") int id, @RequestBody Course updatedCourse){
    ResponseEntity<String> re = null;
    try {
        Optional<Course> foundCourse = courseRepository.findById(id);
        if(foundCourse.isPresent()){
            if(updatedCourse.getName() != null){
                foundCourse.get().setName(updatedCourse.getName());
            }

            if(updatedCourse.getPrice() > 0 ){
                foundCourse.get().setPrice(updatedCourse.getPrice());
            }

            if(updatedCourse.getDescription() != null){
                foundCourse.get().setDescription(updatedCourse.getDescription());
            }

            courseRepository.save(foundCourse.get());
            re = ResponseEntity
                    .status(HttpStatus.CREATED)
                    .body("Given course details are successfully updated with Id: "+foundCourse.get().getId());
        }
    } catch (Exception e) {
        re = ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body("An exception occured due to " + e.getMessage());
    }
    return re;

}
```

Testing in Postman



PUT http://localhost:8080/api/v1/courses/1

```json
{
    "name": "Spring boot All",
    "description": "Get zero to here in spring and spring boot",
    "price": 3999
}
```

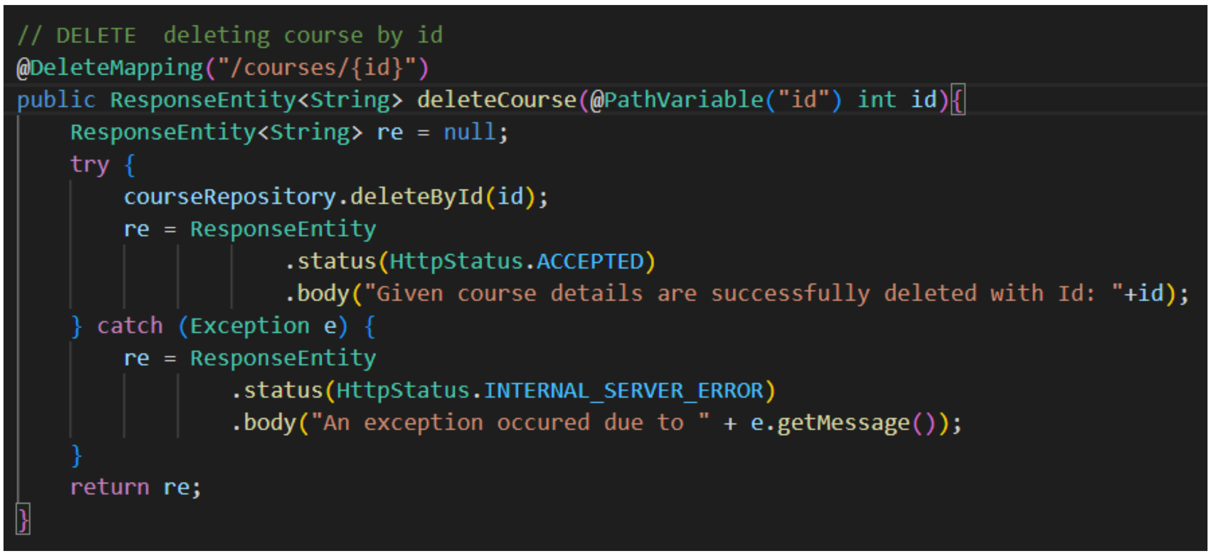
201 Created 50 ms 225 B

```
1  Given course details are successfully updated with Id: 1
```

12. Created 'DELETE' servlet for deleting cource by 'id'.

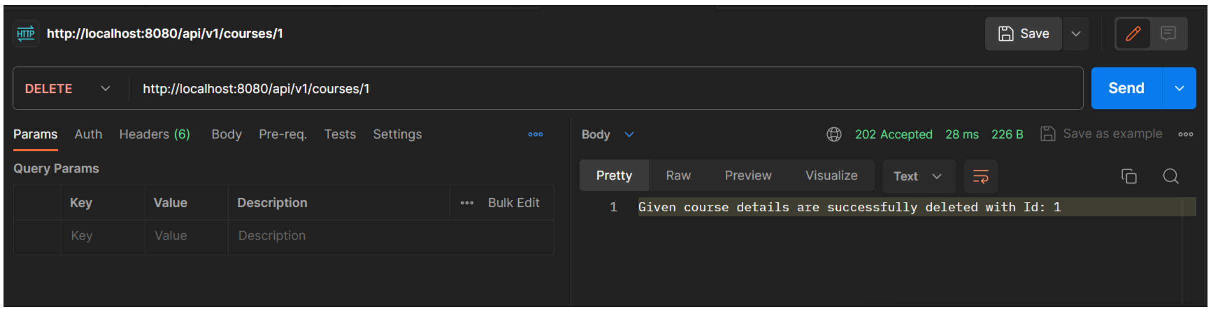
    Route -> 'http://localhost:8080/api/v1/courses/{id}.
    Method -> DELETE.
    Result -> Successful.
    Response Code -> 202 (ACCEPTED).

Java Servlet



```java
// DELETE  deleting course by id
@DeleteMapping("/courses/{id}")
public ResponseEntity<String> deleteCourse(@PathVariable("id") int id){
    ResponseEntity<String> re = null;
    try {
        courseRepository.deleteById(id);
        re = ResponseEntity
                    .status(HttpStatus.ACCEPTED)
                    .body("Given course details are successfully deleted with Id: "+id);
    } catch (Exception e) {
        re = ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body("An exception occured due to " + e.getMessage());
    }
    return re;
}
```

Testing in Postman

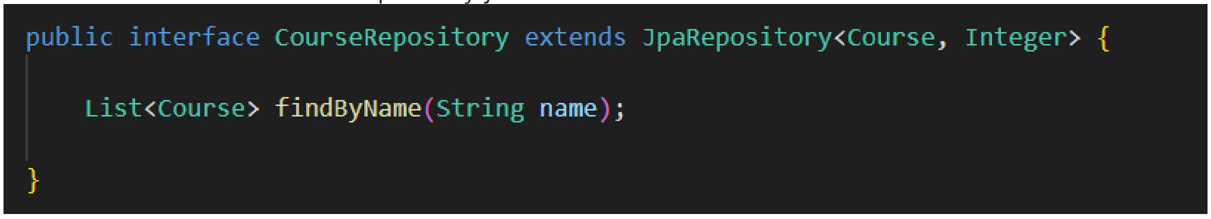


```
1  Given course details are successfully deleted with Id: 1
```

13. Creating servlet for ‘Find by Name of course’.

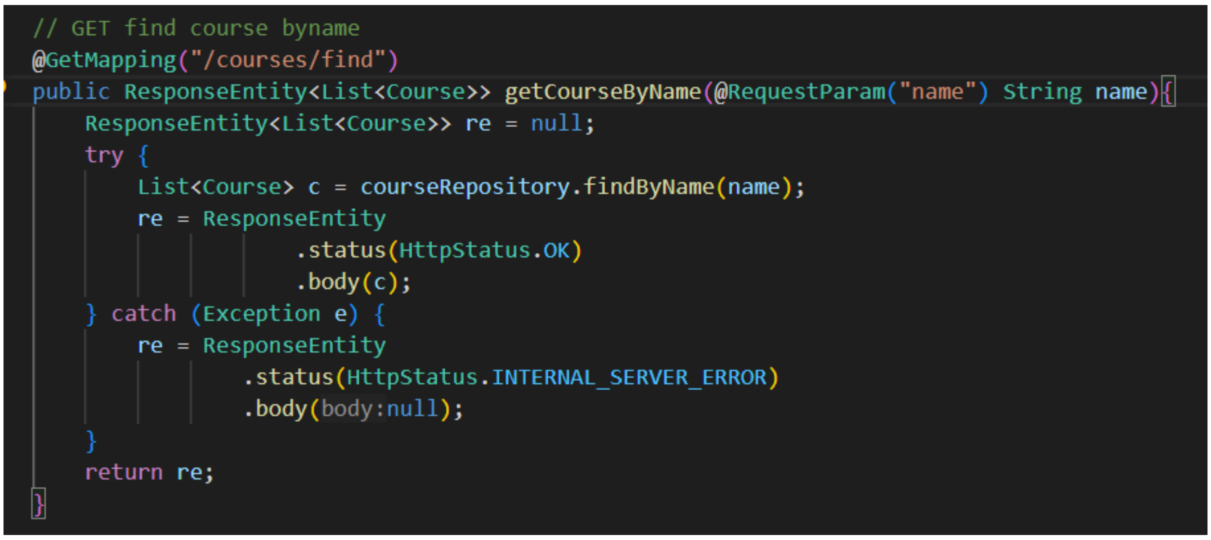
    Route -> ‘http://localhost:8080/api/v1/courses/find/?name=name.
    Method -> GET.
    Result -> Successful.
    Response Code -> 200 (OK).

    Created Method in ‘CourseRepository.java’ file.



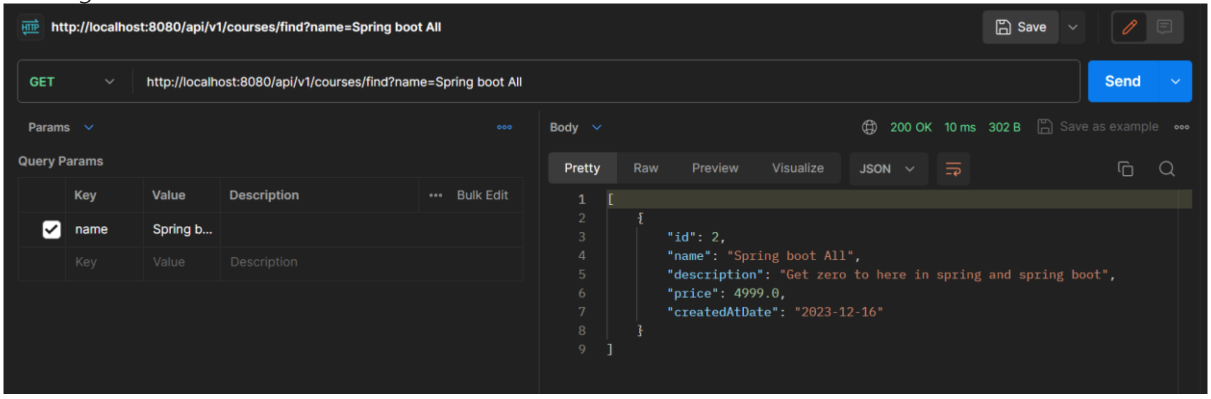
```java
public interface CourseRepository extends JpaRepository<Course, Integer> {

    List<Course> findByName(String name);

}
```

Java Servlet



```java
// GET find course byname
@GetMapping("/courses/find")
public ResponseEntity<List<Course>> getCourseByName(@RequestParam("name") String name){
    ResponseEntity<List<Course>> re = null;
    try {
        List<Course> c = courseRepository.findByName(name);
        re = ResponseEntity
                .status(HttpStatus.OK)
                .body(c);
    } catch (Exception e) {
        re = ResponseEntity
            .status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body(body:null);
    }
    return re;
}
```

Testing in Postman



http://localhost:8080/api/v1/courses/find?name=Spring boot All

GET http://localhost:8080/api/v1/courses/find?name=Spring boot All

Query Params

| | Key | Value | Description |
|---|---|---|---|
| ✓ | name | Spring b... | |
| | Key | Value | Description |

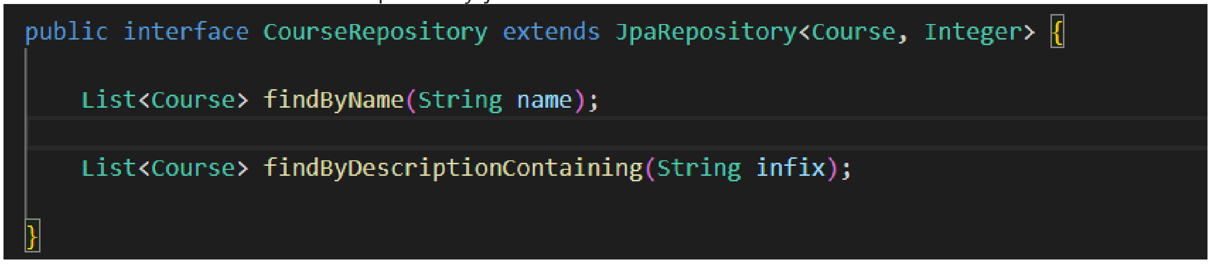
Body — 200 OK 10 ms 302 B

```json
[
    {
        "id": 2,
        "name": "Spring boot All",
        "description": "Get zero to here in spring and spring boot",
        "price": 4999.0,
        "createdAtDate": "2023-12-16"
    }
]
```

14. Created API for find Cources who contains given description.

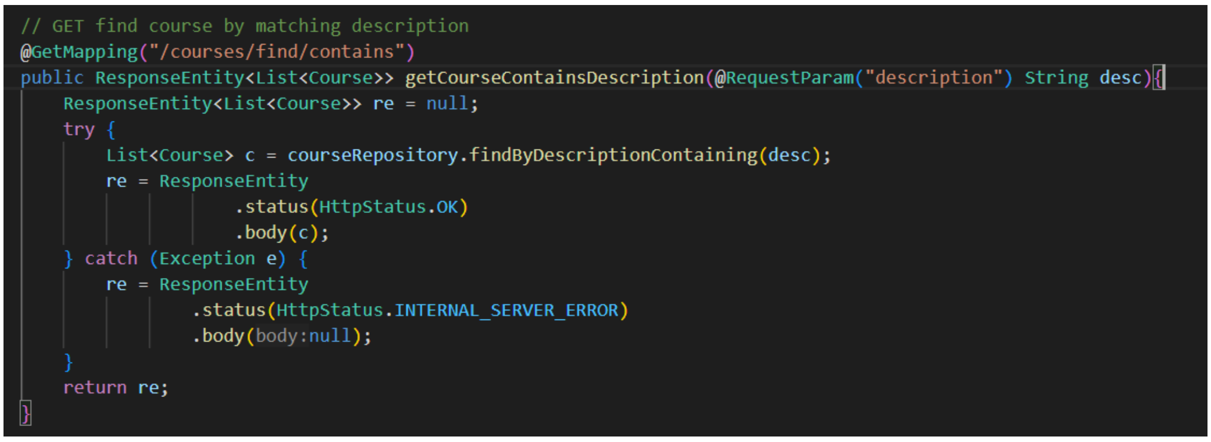
    Route -> 'http://localhost:8080/api/v1/courses/find/contains?description=desc.
    Method -> GET.
    Result -> Successful.
    Response Code -> 200 (OK).

    Created Method in 'CourseRepository.java' file.



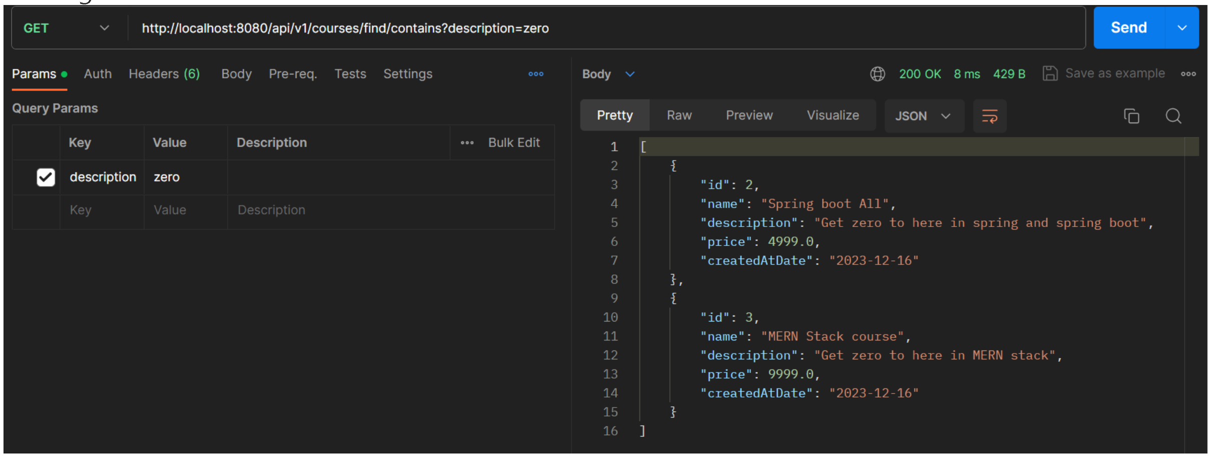
```java
public interface CourseRepository extends JpaRepository<Course, Integer> {

    List<Course> findByName(String name);

    List<Course> findByDescriptionContaining(String infix);

}
```

    Java Servlet



```java
// GET find course by matching description
@GetMapping("/courses/find/contains")
public ResponseEntity<List<Course>> getCourseContainsDescription(@RequestParam("description") String desc){
    ResponseEntity<List<Course>> re = null;
    try {
        List<Course> c = courseRepository.findByDescriptionContaining(desc);
        re = ResponseEntity
                .status(HttpStatus.OK)
                .body(c);
    } catch (Exception e) {
        re = ResponseEntity
            .status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body(body:null);
    }
    return re;
}
```

    Testing in Postman



GET http://localhost:8080/api/v1/courses/find/contains?description=zero

Query Params

| Key | Value | Description |
|---|---|---|
| ✓ description | zero | |

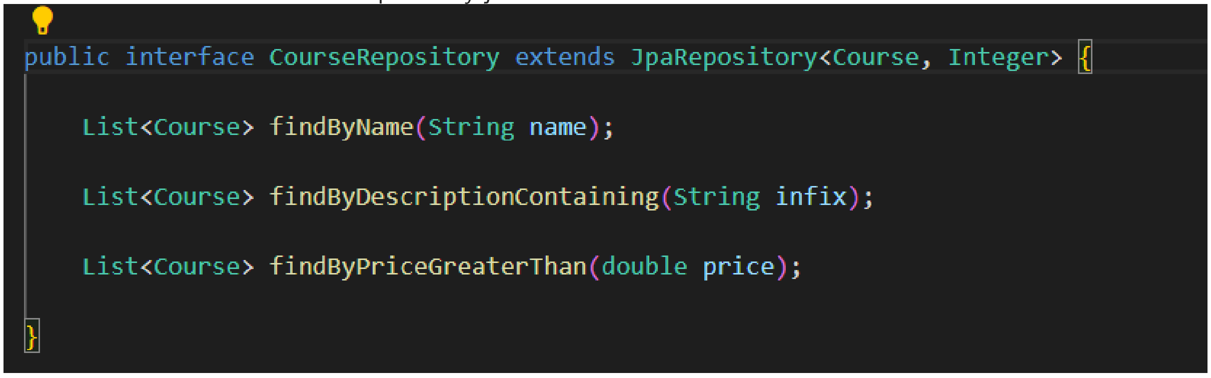
Body 200 OK 8 ms 429 B

```json
[
    {
        "id": 2,
        "name": "Spring boot All",
        "description": "Get zero to here in spring and spring boot",
        "price": 4999.0,
        "createdAtDate": "2023-12-16"
    },
    {
        "id": 3,
        "name": "MERN Stack course",
        "description": "Get zero to here in MERN stack",
        "price": 9999.0,
        "createdAtDate": "2023-12-16"
    }
]
```

15. Created API for find Cources who price is greater than given value.

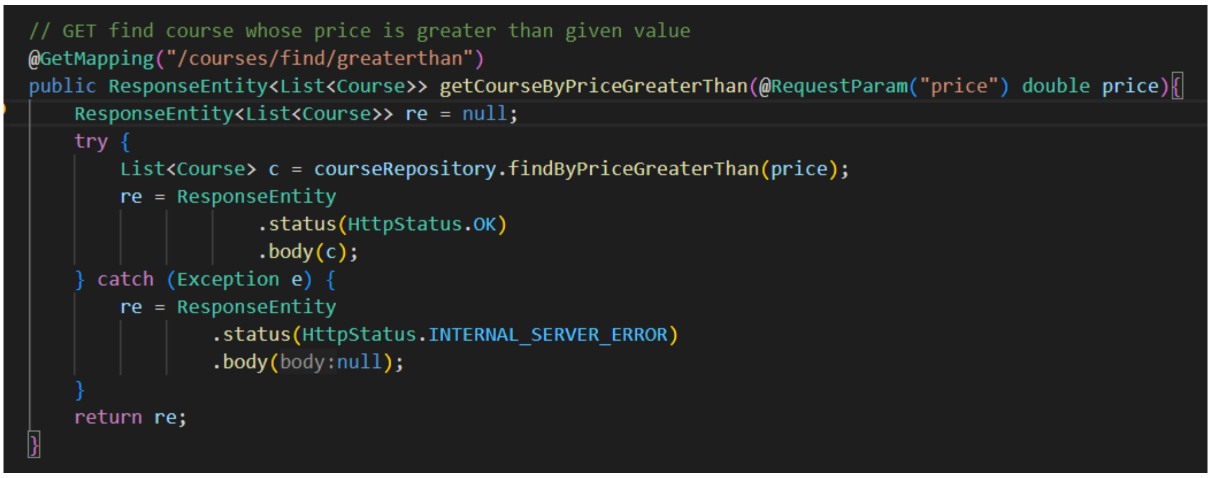
    Route -> 'http://localhost:8080/api/v1/courses/find/greaterthan?price=123.
    Method -> GET.
    Result -> Successful.
    Response Code -> 200 (OK).

    Created Method in 'CourseRepository.java' file.



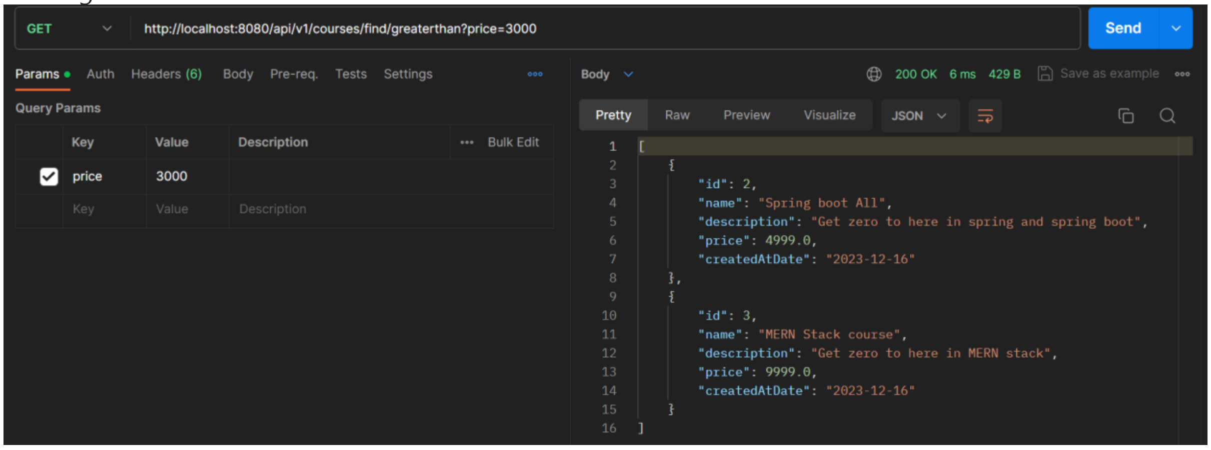
```java
public interface CourseRepository extends JpaRepository<Course, Integer> {

    List<Course> findByName(String name);

    List<Course> findByDescriptionContaining(String infix);

    List<Course> findByPriceGreaterThan(double price);

}
```

    Java Servlet



```java
// GET find course whose price is greater than given value
@GetMapping("/courses/find/greaterthan")
public ResponseEntity<List<Course>> getCourseByPriceGreaterThan(@RequestParam("price") double price){
    ResponseEntity<List<Course>> re = null;
    try {
        List<Course> c = courseRepository.findByPriceGreaterThan(price);
        re = ResponseEntity
                .status(HttpStatus.OK)
                .body(c);
    } catch (Exception e) {
        re = ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body(body:null);
    }
    return re;
}
```

    Testing in Postman



GET http://localhost:8080/api/v1/courses/find/greaterthan?price=3000

Query Params

| Key | Value | Description |
|---|---|---|
| price | 3000 | |

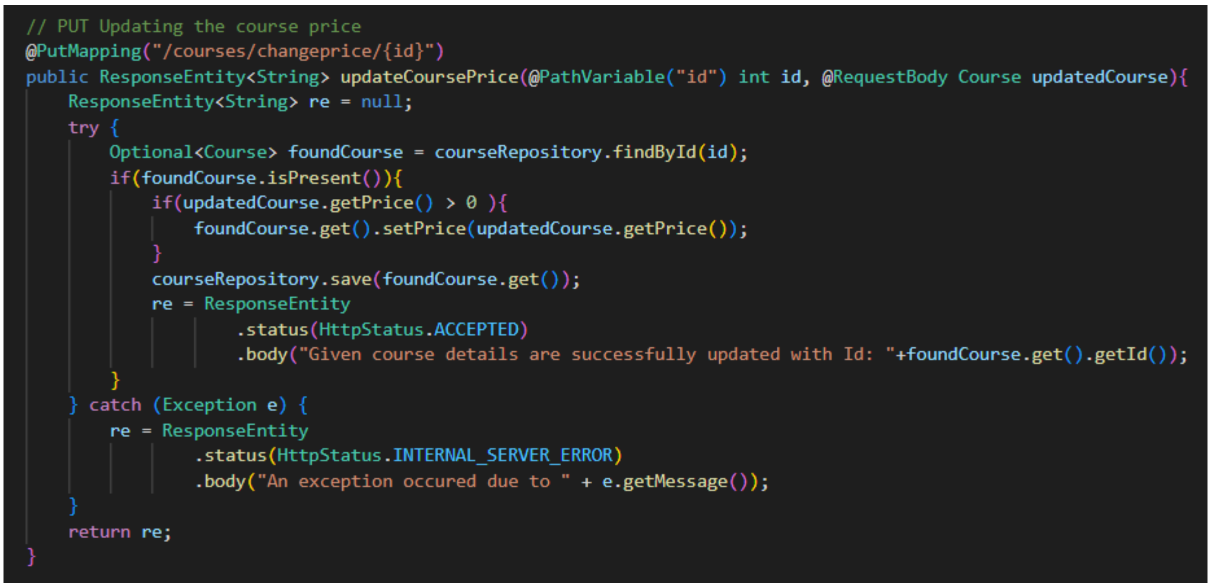
200 OK 6 ms 429 B

```json
[
    {
        "id": 2,
        "name": "Spring boot All",
        "description": "Get zero to here in spring and spring boot",
        "price": 4999.0,
        "createdAtDate": "2023-12-16"
    },
    {
        "id": 3,
        "name": "MERN Stack course",
        "description": "Get zero to here in MERN stack",
        "price": 9999.0,
        "createdAtDate": "2023-12-16"
    }
]
```

16. Created API for change course price by id.

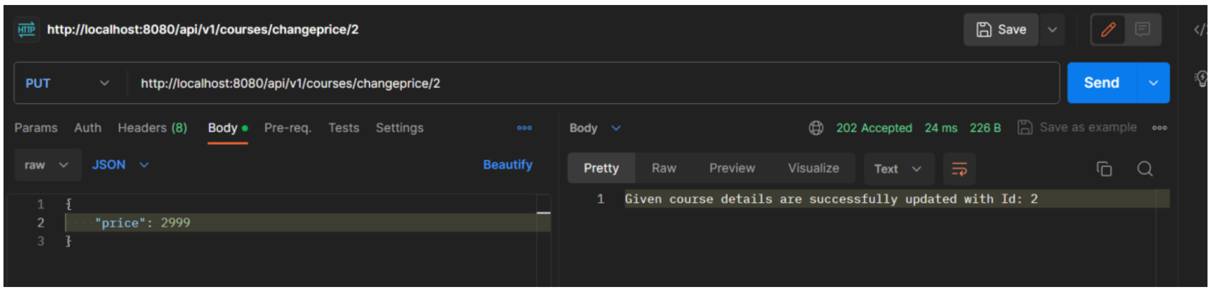
    Route -> ‘http://localhost:8080/api/v1/courses/updateprice/{id}.
    Method -> PUT.
    Result -> Successful.
    Response Code -> 202 (ACCEPTED).

Java Servlet



```java
// PUT Updating the course price
@PutMapping("/courses/changeprice/{id}")
public ResponseEntity<String> updateCoursePrice(@PathVariable("id") int id, @RequestBody Course updatedCourse){
    ResponseEntity<String> re = null;
    try {
        Optional<Course> foundCourse = courseRepository.findById(id);
        if(foundCourse.isPresent()){
            if(updatedCourse.getPrice() > 0 ){
                foundCourse.get().setPrice(updatedCourse.getPrice());
            }
            courseRepository.save(foundCourse.get());
            re = ResponseEntity
                    .status(HttpStatus.ACCEPTED)
                    .body("Given course details are successfully updated with Id: "+foundCourse.get().getId());
        }
    } catch (Exception e) {
        re = ResponseEntity
                .status(HttpStatus.INTERNAL_SERVER_ERROR)
                .body("An exception occured due to " + e.getMessage());
    }
    return re;
}
```

Testing in Postman



http://localhost:8080/api/v1/courses/changeprice/2

PUT http://localhost:8080/api/v1/courses/changeprice/2

```json
{
    "price": 2999
}
```

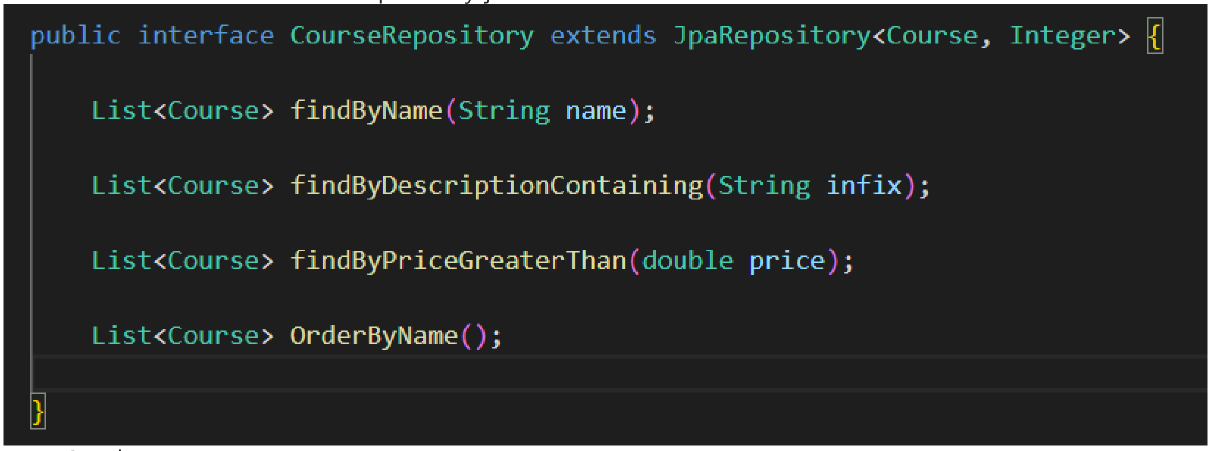
202 Accepted 24 ms 226 B

```
Given course details are successfully updated with Id: 2
```

17. Created API for finding courses order by name.

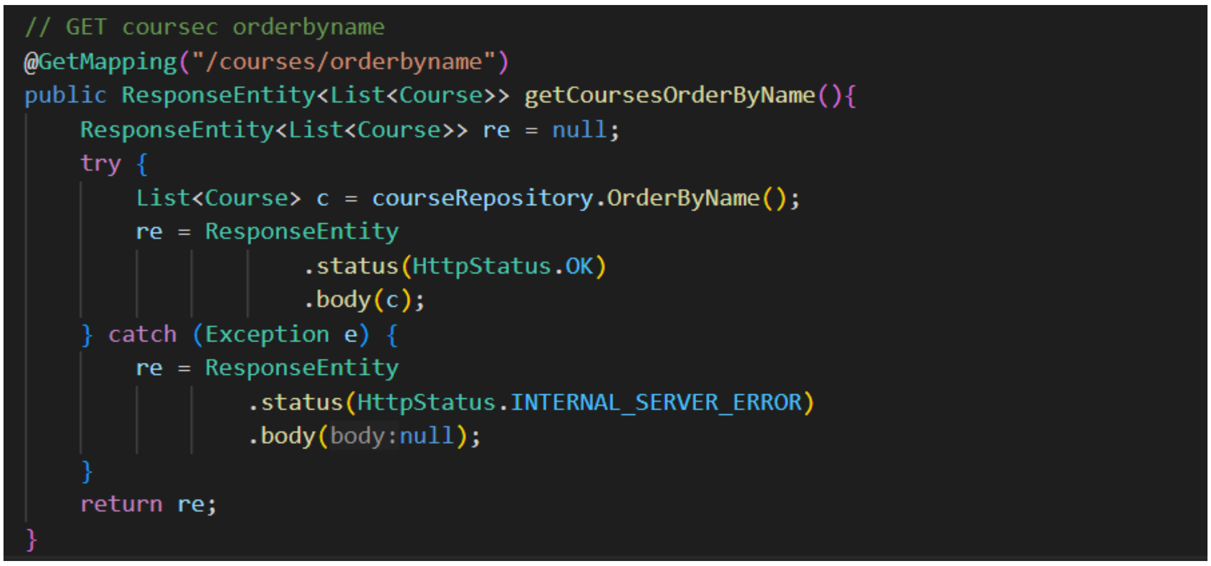
    Route -> 'http://localhost:8080/api/v1/courses/find/?date=123.
    Method -> GET.
    Result -> Successful.
    Response Code -> 200 (OK).

    Created Method in 'CourseRepository.java' file.



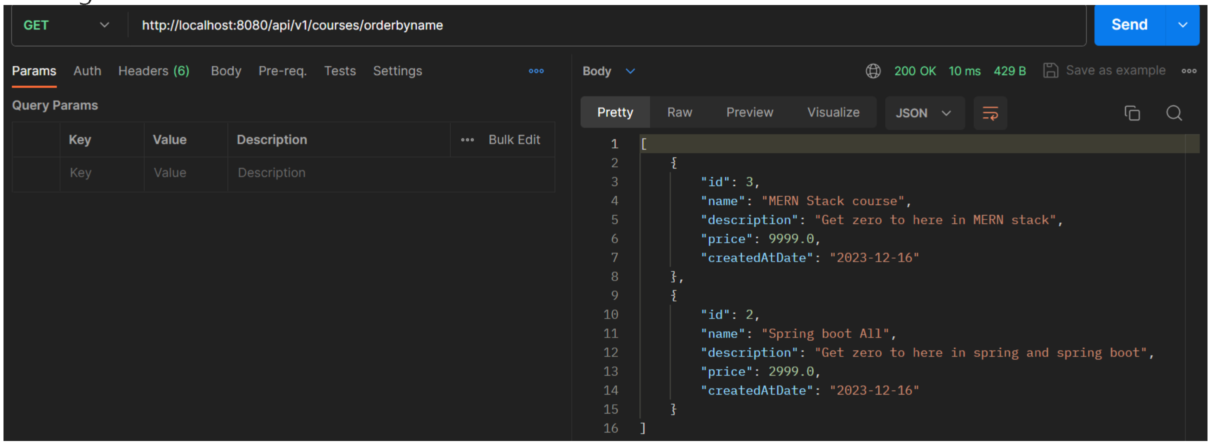
```java
public interface CourseRepository extends JpaRepository<Course, Integer> {

    List<Course> findByName(String name);

    List<Course> findByDescriptionContaining(String infix);

    List<Course> findByPriceGreaterThan(double price);

    List<Course> OrderByName();

}
```

Java Servlet



```java
// GET coursec orderbyname
@GetMapping("/courses/orderbyname")
public ResponseEntity<List<Course>> getCoursesOrderByName(){
    ResponseEntity<List<Course>> re = null;
    try {
        List<Course> c = courseRepository.OrderByName();
        re = ResponseEntity
                .status(HttpStatus.OK)
                .body(c);
    } catch (Exception e) {
        re = ResponseEntity
            .status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body(body:null);
    }
    return re;
}
```

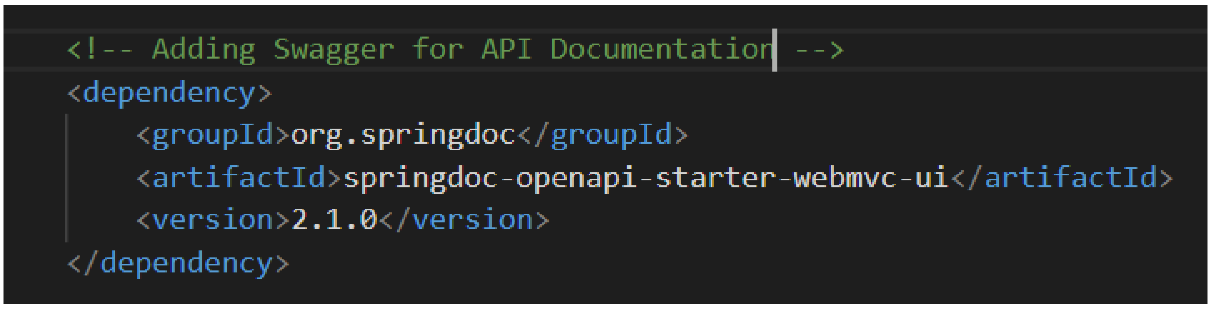
Testing in Postman



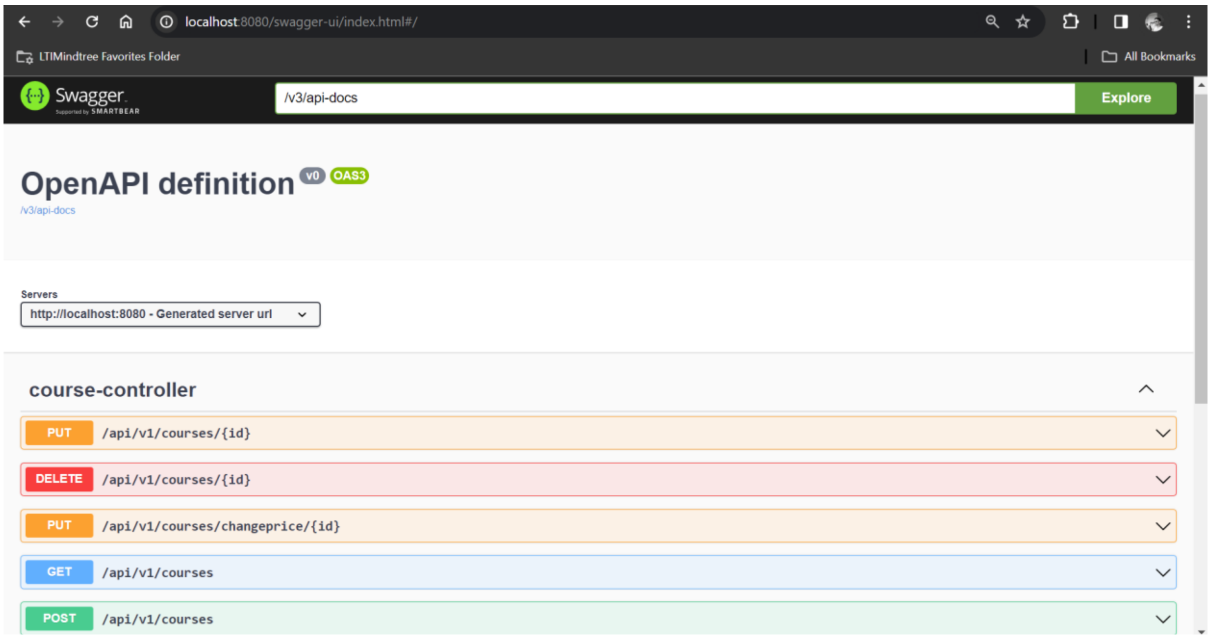
GET http://localhost:8080/api/v1/courses/orderbyname

```json
[
    {
        "id": 3,
        "name": "MERN Stack course",
        "description": "Get zero to here in MERN stack",
        "price": 9999.0,
        "createdAtDate": "2023-12-16"
    },
    {
        "id": 2,
        "name": "Spring boot All",
        "description": "Get zero to here in spring and spring boot",
        "price": 2999.0,
        "createdAtDate": "2023-12-16"
    }
]
```

18. Adding ‘Swagger’ dependency for API documentation.



```xml
<!-- Adding Swagger for API Documentation -->
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.1.0</version>
</dependency>
```


localhost:8080/swagger-ui/index.html#/
LTIMindtree Favorites Folder
All Bookmarks
Swagger
Supported by SMARTBEAR
/v3/api-docs
Explore

OpenAPI definition v0 OAS3

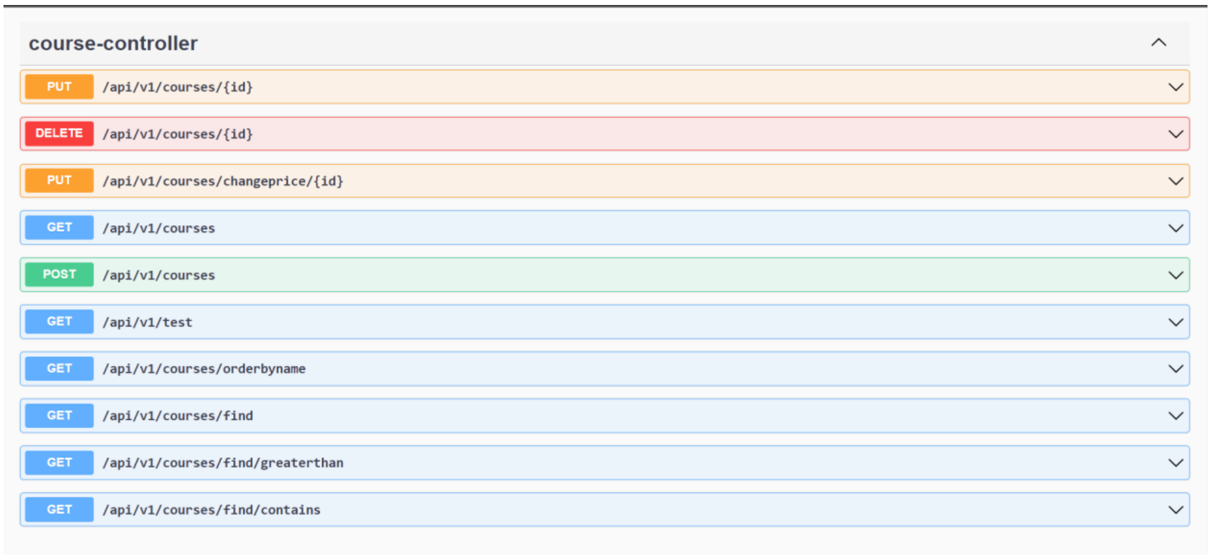
/v3/api-docs

Servers

http://localhost:8080 - Generated server url

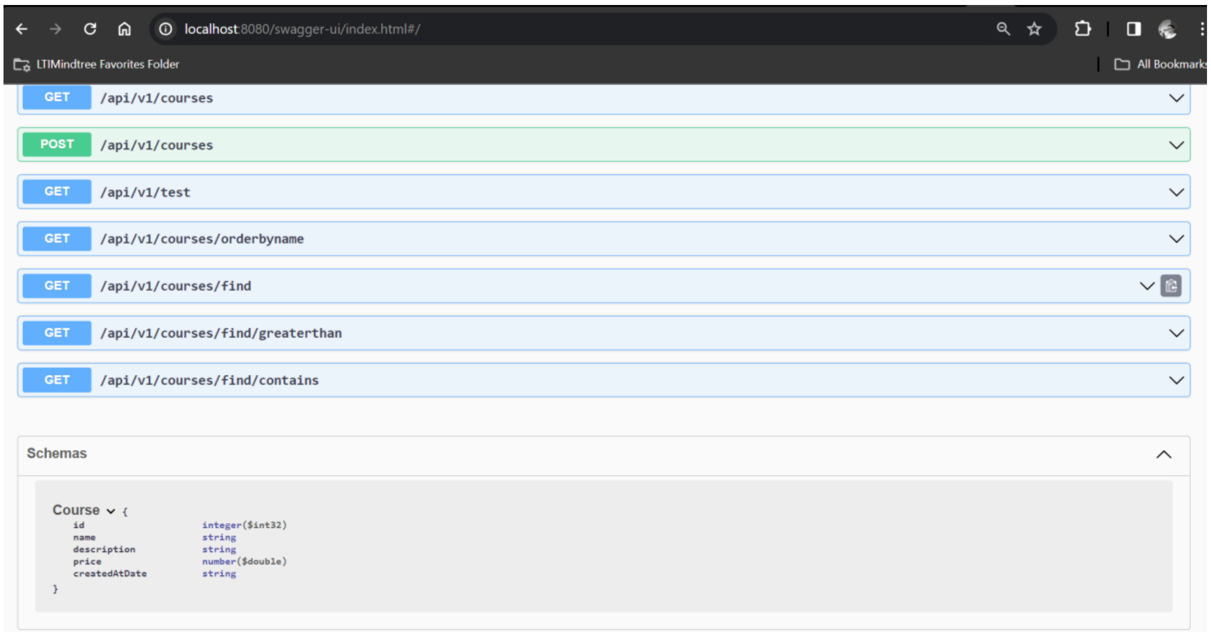
course-controller

```
PUT     /api/v1/courses/{id}
DELETE  /api/v1/courses/{id}
PUT     /api/v1/courses/changeprice/{id}
GET     /api/v1/courses
POST    /api/v1/courses
```



course-controller

```
PUT     /api/v1/courses/{id}
DELETE  /api/v1/courses/{id}
PUT     /api/v1/courses/changeprice/{id}
GET     /api/v1/courses
POST    /api/v1/courses
GET     /api/v1/test
GET     /api/v1/courses/orderbyname
GET     /api/v1/courses/find
GET     /api/v1/courses/find/greaterthan
GET     /api/v1/courses/find/contains
```



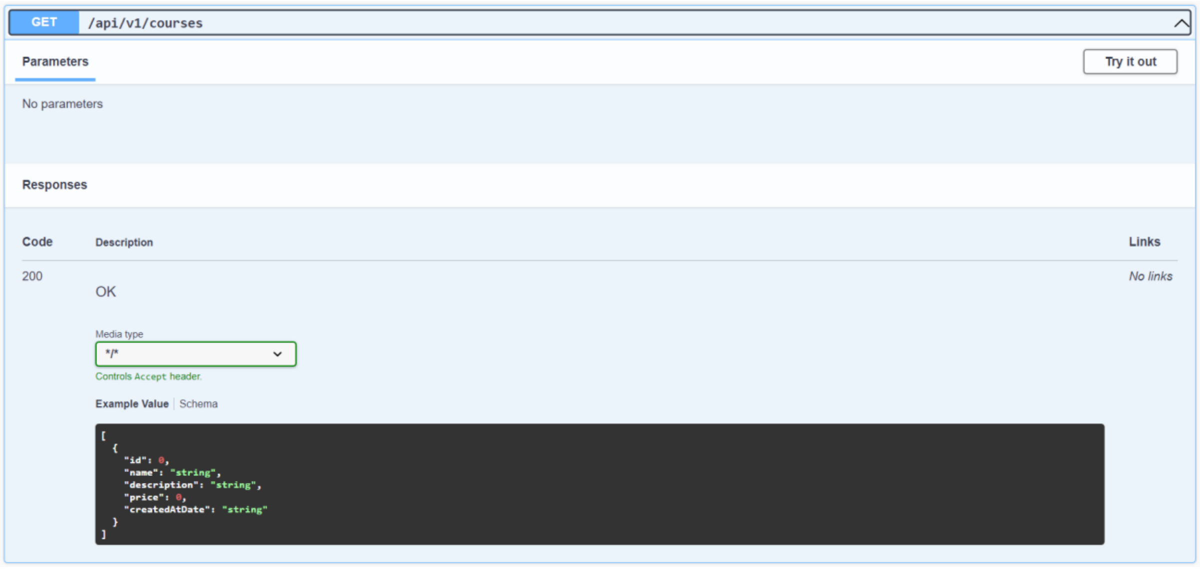
localhost:8080/swagger-ui/index.html#/

LTIMindtree Favorites Folder | All Bookmarks

| | |
|---|---|
| GET | /api/v1/courses |
| POST | /api/v1/courses |
| GET | /api/v1/test |
| GET | /api/v1/courses/orderbyname |
| GET | /api/v1/courses/find |
| GET | /api/v1/courses/find/greaterthan |
| GET | /api/v1/courses/find/contains |

Schemas

```
Course {
    id              integer($int32)
    name            string
    description     string
    price           number($double)
    createdAtDate   string
}
```



GET /api/v1/courses

Parameters | Try it out

No parameters

Responses

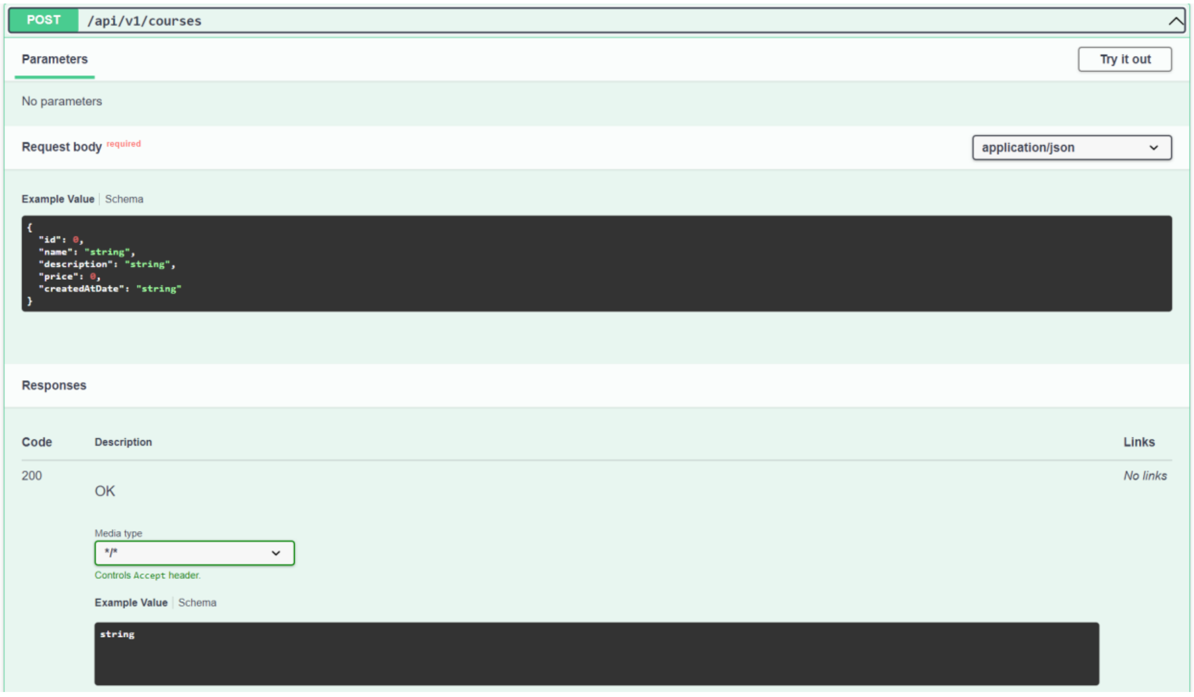
| Code | Description | Links |
|---|---|---|
| 200 | OK | *No links* |

Media type: */*

Controls Accept header.

Example Value | Schema

```
[
  {
    "id": 0,
    "name": "string",
    "description": "string",
    "price": 0,
    "createdAtDate": "string"
  }
]
```


POST /api/v1/courses

Parameters

No parameters

Request body required

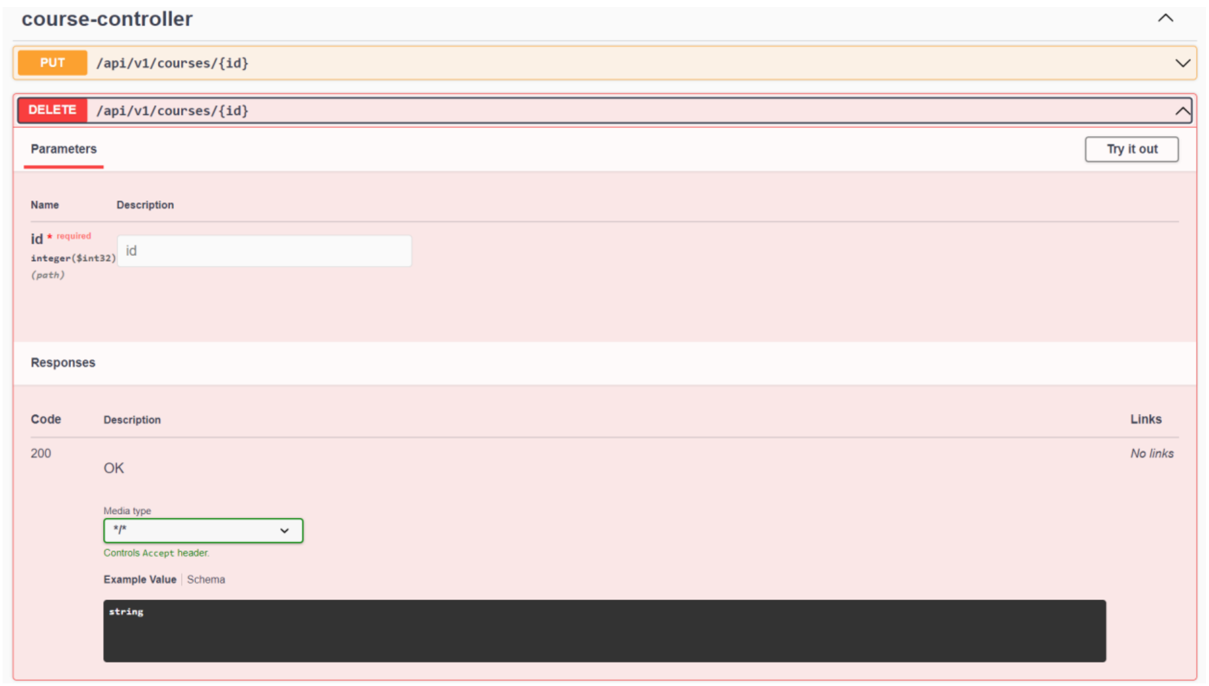
application/json

Example Value | Schema

```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "price": 0,
  "createdAtDate": "string"
}
```

Responses

| Code | Description | Links |
|---|---|---|
| 200 | OK<br>Media type<br>*/*<br>Controls Accept header.<br>Example Value \| Schema<br>`string` | *No links* |


course-controller

PUT /api/v1/courses/{id}

DELETE /api/v1/courses/{id}

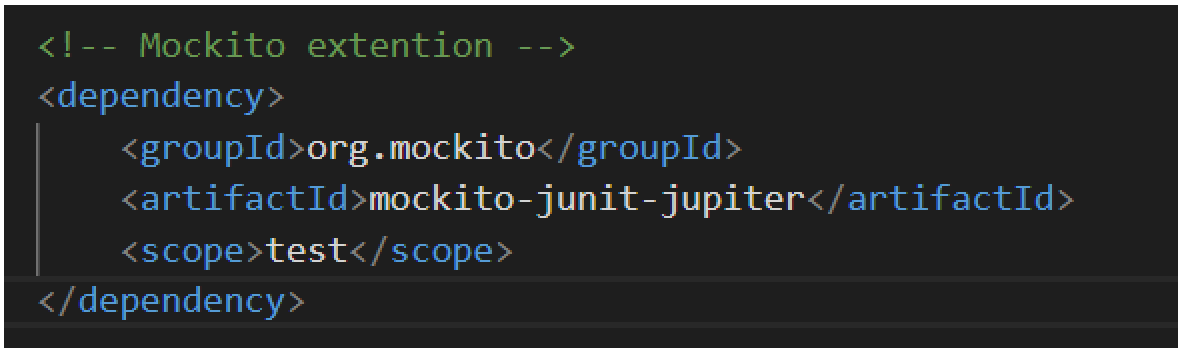
Parameters

| Name | Description |
|---|---|
| id * required<br>integer($int32)<br>*(path)* | id |

Responses

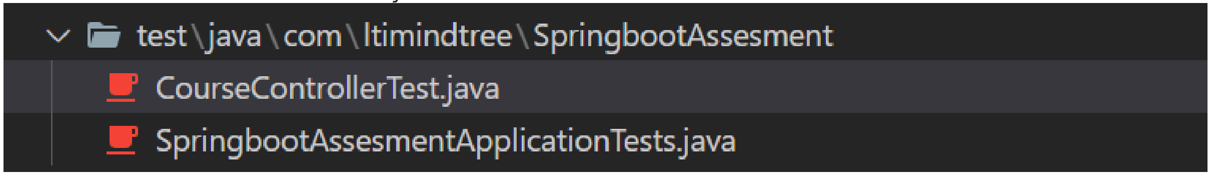
| Code | Description | Links |
|---|---|---|
| 200 | OK<br>Media type<br>*/*<br>Controls Accept header.<br>Example Value \| Schema<br>`string` | *No links* |

19. Adding ‘Mockito’ dependency for unit testing of API.



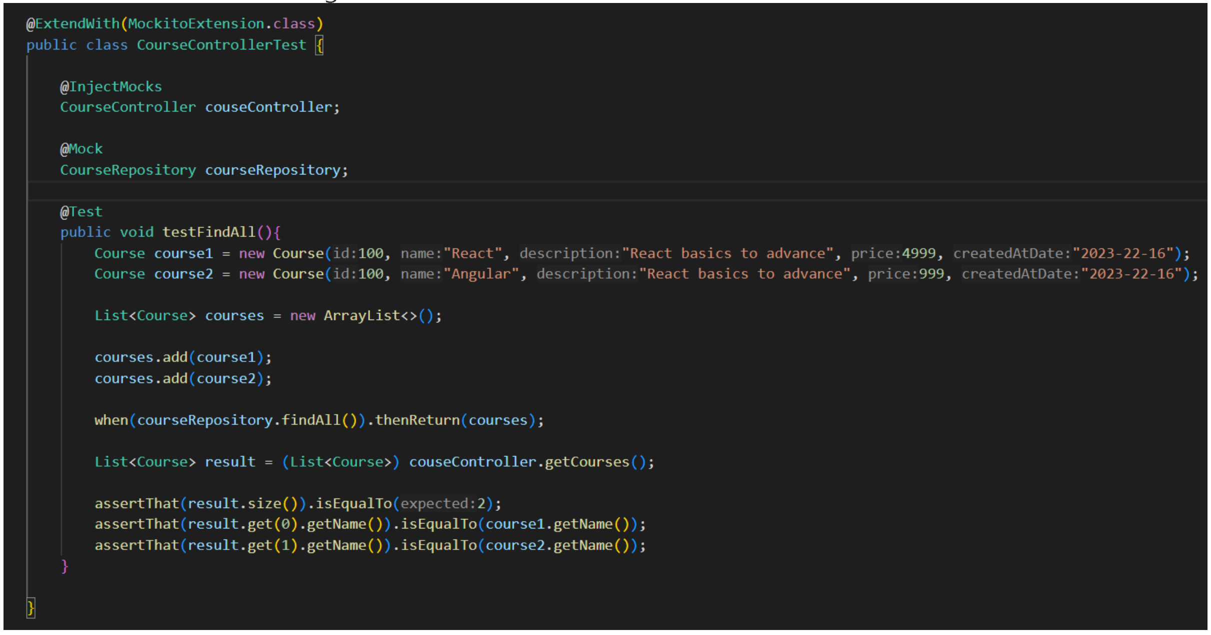
```xml
<!-- Mockito extention -->
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <scope>test</scope>
</dependency>
```

Created ‘CourceControllerTest.java’ file to create mock test and test API.



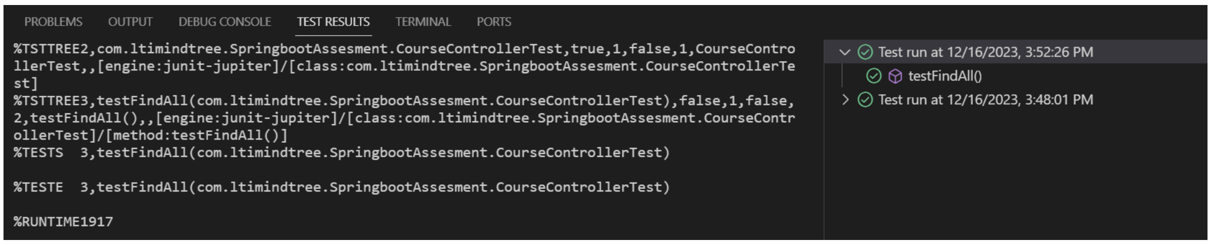
```
test\java\com\ltimindtree\SpringbootAssesment
    CourseControllerTest.java
    SpringbootAssesmentApplicationTests.java
```

Created test method for ‘getCourses’.



```java
@ExtendWith(MockitoExtension.class)
public class CourseControllerTest {

    @InjectMocks
    CourseController couseController;

    @Mock
    CourseRepository courseRepository;

    @Test
    public void testFindAll(){
        Course course1 = new Course(id:100, name:"React", description:"React basics to advance", price:4999, createdAtDate:"2023-22-16");
        Course course2 = new Course(id:100, name:"Angular", description:"React basics to advance", price:999, createdAtDate:"2023-22-16");

        List<Course> courses = new ArrayList<>();

        courses.add(course1);
        courses.add(course2);

        when(courseRepository.findAll()).thenReturn(courses);

        List<Course> result = (List<Course>) couseController.getCourses();

        assertThat(result.size()).isEqualTo(expected:2);
        assertThat(result.get(0).getName()).isEqualTo(course1.getName());
        assertThat(result.get(1).getName()).isEqualTo(course2.getName());
    }

}
```

After testing the test method test worked successfully.



```
%TSTTREE2,com.ltimindtree.SpringbootAssesment.CourseControllerTest,true,1,false,1,CourseContro
llerTest,,[engine:junit-jupiter]/[class:com.ltimindtree.SpringbootAssesment.CourseControllerTe
st]
%TSTTREE3,testFindAll(com.ltimindtree.SpringbootAssesment.CourseControllerTest),false,1,false,
2,testFindAll(),,[engine:junit-jupiter]/[class:com.ltimindtree.SpringbootAssesment.CourseContr
ollerTest]/[method:testFindAll()]
%TESTS  3,testFindAll(com.ltimindtree.SpringbootAssesment.CourseControllerTest)

%TESTE  3,testFindAll(com.ltimindtree.SpringbootAssesment.CourseControllerTest)

%RUNTIME1917
```

```
✓ Test run at 12/16/2023, 3:52:26 PM
    ✓ testFindAll()
✓ Test run at 12/16/2023, 3:48:01 PM
```