

Microservices

Spring Boot

Course Microservice



Service URL

Name

☒ Use default location

Location

Type: Packaging:

Java Version: Language:

Group

Artifact

Version

Description

Package

Working sets

☐ Add project to working sets

Working sets:


Spring Boot Version:



















Available:


Selected:

- ▶ AI
- ▶ Developer Tools
- ▶ Google Cloud
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL
- ▶ Observability
- ▶ Ops
- ▶ SQL
- ▶ Security
- ▶ Spring Cloud

- X Spring Data JPA
- X MySQL Driver
- X Spring Web

▼  course-service [boot]

- ▼  src/main/java
 - >  com.rit
 - ▼  com.rit.controller
 - >  CourseController.java
 - ▼  com.rit.entity
 - >  Course.java
 - ▼  com.rit.repository
 - >  CourseRepository.java
 - ▼  com.rit.request
 - >  CourseRequest.java
 - ▼  com.rit.response
 - >  CourseResponse.java
 - ▼  com.rit.service
 - >  CourseService.java
- ▼  src/main/resources
 -  static
 -  templates
 -  application.properties

 application.properties X

```
1 spring.application.name=course-service
2
3 server.port=8082
4
5 spring.datasource.url=jdbc:mysql://localhost:3306/microservicesdb
6 spring.datasource.username=root
7 spring.datasource.password=root
8
9 spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11
12
```

```
3+ import jakarta.persistence.Entity;
7 @Entity
8 public class Course {
9
10+     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private Long courseId;
13     private String courseName;
14     private Double courseFees;
15
16     //Getter & Setter
17+     public Long getCourseId() {
20+     public void setCourseId(Long courseId) {
23+     public String getCourseName() {
26+     public void setCourseName(String courseName) {
29+     public Double getCourseFees() {
32+     public void setCourseFees(Double courseFees) {
35 }
--
```

```
1 package com.rit.request;
2
3 public class CourseRequest {
4
5     private String courseName;
6     private Double courseFees;
7
8     // Getter & Setter
9     public String getCourseName() {
10
11
12     public void setCourseName(String courseName) {
13
14
15     public Double getCourseFees() {
16
17
18     public void setCourseFees(Double courseFees) {
19
20
21 }
22
23
```

```
5 public class CourseResponse {
6     private Long courseId;
7     private String courseName;
8     private Double courseFees;
9     //Constructor to Convert Course Entity to Response
10    public CourseResponse(Course course) {
11        super();
12        this.courseId = course.getCourseId();
13        this.courseName = course.getCourseName();
14        this.courseFees = course.getCourseFees();
15    }
16    //Getter & Setter
17    public Long getCourseId() {..
20    public void setCourseId(Long courseId) {..
23    public String getCourseName() {..
26    public void setCourseName(String courseName) {..
29    public Double getCourseFees() {..
32    public void setCourseFees(Double courseFees) {..
35 }
```

CourseRepository.java ×

```
1 package com.rit.repository;
2
3 import org.springframework.data.jpa.repository.JpaRepository;
4
5
6
7
8 @Repository
9 public interface CourseRepository extends JpaRepository<Course, Long>{
10
11 }
12
```



```
15 @RestController
16 @RequestMapping("/api/course")
17 public class CourseController {
18
19     @Autowired
20     private CourseService courseService;
21
22     @PostMapping("/create")
23     public CourseResponse createCourse(@RequestBody CourseRequest courseRequest) {
24         return courseService.createCourse(courseRequest);
25     }
26
27     @GetMapping("/getById/{id}")
28     public CourseResponse getById(@PathVariable long id) {
29         return courseService.getById(id);
30     }
31 }
32
```

```
14 @Service
15 public class CourseService {
16
17     Logger logger = LoggerFactory.getLogger(CourseService.class);
18
19     @Autowired
20     private CourseRepository courseRepository;
21
22     public CourseResponse createCourse(CourseRequest courseRequest) {
23         Course course = new Course();
24         course.setCourseName(courseRequest.getCourseName());
25         course.setCourseFees(courseRequest.getCourseFees());
26         courseRepository.save(course);
27         return new CourseResponse(course);
28     }
29
30     public CourseResponse getById(long id) {
31         logger.info("In getById : "+id);
32         Course course = courseRepository.findById(id).get();
33         return new CourseResponse(course);
34     }
35 }
```



http://localhost:8082/api/course/create



Save

POST



http://localhost:8082/api/course/create

Send

Body



ooo

raw



JSON



Beautify

```
1 {  
2   .... "courseName" : "Java Full Stack",  
3   .... "courseFees" : 25000.00  
4 }
```

Body



200 OK

291 ms

230 B

Save Response

Pretty



JSON



```
1 {  
2   "courseId": 1,  
3   "courseName": "Java Full Stack",  
4   "courseFees": 25000.0  
5 }
```



http://localhost:8082/api/course/getById/3

GET



http://localhost:8082/api/course/getById/3

Params

Authorization

Headers (8)

Body ●

Pre-request Script

Tests

Settings

Body

Cookies

Headers (5)

Test Results



200 OK

41 ms

232 B

Pretty

Raw

Preview

Visualize

JSON



1

```
{
```

2

```
  "courseId": 3,
```

3

```
  "courseName": "Python Full Stack",
```

4

```
  "courseFees": 27000.0
```

5

```
}
```

Student Microservice

Service URL

Name

☒ Use default location

Location

Type: Packaging:

Java Version: Language:

Group

Artifact

Version

Description

Package

Working sets

☐ Add project to working sets

Working sets:

Spring Boot Version:

Frequently Used:

☒ MySQL Driver

☒ Spring Data JPA

☒ Spring Web

Available:

Selected:

- ▶ AI
- ▶ Developer Tools
- ▶ Google Cloud
- ▶ I/O
- ▶ Messaging
- ▶ Microsoft Azure
- ▶ NoSQL

- X Spring Data JPA
- X MySQL Driver
- X Spring Web

- > course-service [boot]
- ▼ student-service [boot]
 - ▼ src/main/java
 - ▼ com.rit
 - > StudentServiceApplication.java
 - ▼ com.rit.controller
 - > StudentController.java
 - ▼ com.rit.entity
 - > Student.java
 - ▼ com.rit.repository
 - > StudentRepository.java
 - ▼ com.rit.request
 - > StudentRequest.java
 - ▼ com.rit.response
 - > StudentResponse.java
 - ▼ com.rit.service
 - > StudentService.java
 - ▼ src/main/resources
 - static
 - templates
 - application.properties
 - > src/test/java
 - > JRE System Library [JavaSE-17]

application.properties

```
1 spring.application.name=student-service
2
3 server.port=8080
4
5 spring.datasource.url=jdbc:mysql://localhost:3306/microservicesdb
6 spring.datasource.username=root
7 spring.datasource.password=root
8
9 spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11
```

```
5 import jakarta.persistence.Entity;
7 @Entity
8 public class Student {
9
10     @Id
11     @GeneratedValue(strategy = GenerationType.IDENTITY)
12     private Long studentId;
13     private String firstName;
14     private String lastName;
15     private String email;
16
17     //Getter & Setter
18     public Long getStudentId() {}
21     public void setStudentId(Long studentId) {}
24     public String getFirstName() {}
27     public void setFirstName(String firstName) {}
30     public String getLastName() {}
33     public void setLastName(String lastName) {}
36     public String getEmail() {}
39     public void setEmail(String email) {}
42 }
```


StudentRequest.java ×

```
1 package com.rit.request;
2
3 public class StudentRequest {
4
5     private String firstName;
6     private String lastName;
7     private String email;
8
9     // Getter & Setter
10    public String getFirstName() {
13    public void setFirstName(String firstName) {
16    public String getLastName() {
19    public void setLastName(String lastName) {
22    public String getEmail() {
25    public void setEmail(String email) {
28 }
29
```

```
5 public class StudentResponse {
6     private Long studentId;
7     private String firstName;
8     private String lastName;
9     private String email;
10
11     //Constructor to Convert Student Entity to Response
12     public StudentResponse(Student student) {
13         super();
14         this.studentId = student.getId();
15         this.firstName = student.getFirstName();
16         this.lastName = student.getLastName();
17         this.email = student.getEmail();
18     }
19     //Getter & Setter
20     public Long getId() {
21
22     }
23     public void setId(Long studentId) {
24
25     }
26     public String getFirstName() {
27
28     }
29     public void setFirstName(String firstName) {
30
31     }
32     public String getLastName() {
33
34     }
35     public void setLastName(String lastName) {
36
37     }
38     public String getEmail() {
39
40     }
41     public void setEmail(String email) {
42
43     }
44 }
```

StudentRepository.java X

```
1 package com.rit.repository;
2
3+ import org.springframework.data.jpa.repository.JpaRepository;
4
5
6
7
8 @Repository
9 public interface StudentRepository extends JpaRepository<Student, Long>{
10
11 }
12
```

```
15 @RestController
16 @RequestMapping("/api/student")
17 public class StudentController {
18
19     @Autowired
20     private StudentService studentService;
21
22     @PostMapping("/create")
23     public StudentResponse createStudent(@RequestBody StudentRequest studentRequest) {
24         return studentService.createStudent(studentRequest);
25     }
26
27     @GetMapping("/getById/{id}")
28     public StudentResponse getById(@PathVariable long id) {
29         return studentService.getById(id);
30     }
31 }
32
```

```
14 @Service
15 public class StudentService {
16
17     Logger logger = LoggerFactory.getLogger(StudentService.class);
18
19     @Autowired
20     private StudentRepository studentRepository;
21
22     public StudentResponse createStudent(StudentRequest studentRequest) {
23         Student student = new Student();
24         student.setFirstName(studentRequest.getFirstName());
25         student.setLastName(studentRequest.getLastName());
26         student.setEmail(studentRequest.getEmail());
27         studentRepository.save(student);
28         return new StudentResponse(student);
29     }
30     public StudentResponse getById(long id) {
31         logger.info("In getById : "+id);
32         Student student = studentRepository.findById(id).get();
33         return new StudentResponse(student);
34     }
35 }
```



http://localhost:8080/api/student/create



POST



http://localhost:8080/api/student/create

Send

Body



raw



JSON



Beautify

```
1 {  
2   "firstName": "Manoj",  
3   "lastName": "kumar",  
4   "email": "manoj@gmail.com"  
5 }
```

Body



200 OK

190 ms

244 B

Save Response

Pretty



JSON



```
1 {  
2   "studentId": 1,  
3   "firstName": "Manoj",  
4   "lastName": "kumar",  
5   "email": "manoj@gmail.com"  
6 }
```



http://localhost:8080/api/student/getById/2

GET



http://localhost:8080/api/student/getById/2



Body

Cookies

Headers (5)

Test Results



200 OK

45 ms

242 B

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "studentId": 2,  
3   "firstName": "Siva",  
4   "lastName": "kumar",  
5   "email": "siva@gmail.com"  
6 }
```

Get Course along with Student

Using WebClient

WebClient

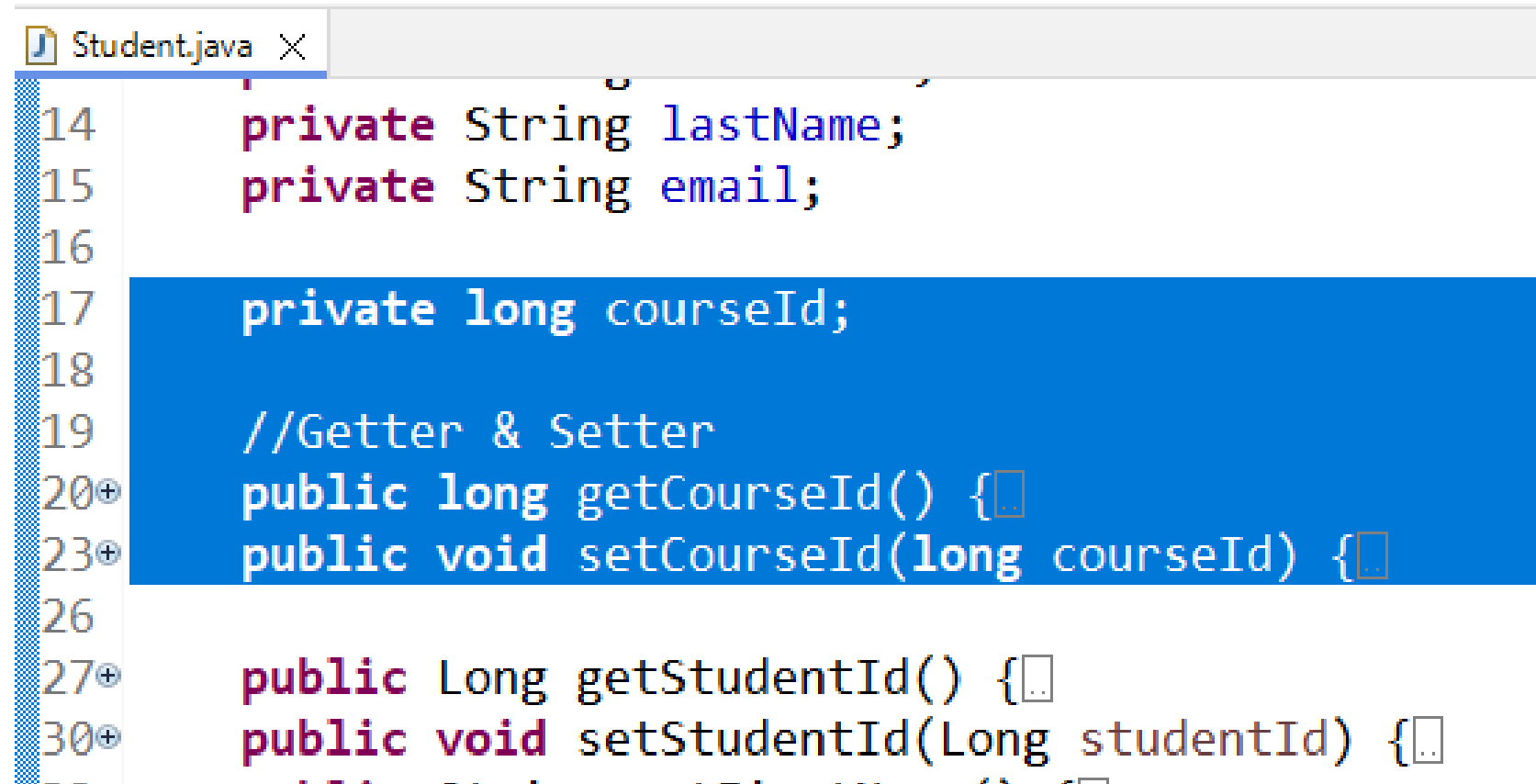
WebClient is a reactive, non-blocking web client introduced in Spring 5 as part of the Spring WebFlux module.

It is designed to replace the classic RestTemplate for making HTTP requests in a more modern and efficient way.

WebClient also supports synchronous and Asynchronous operations, making it versatile for different use cases

```
WebClient client = WebClient.builder()
    .baseUrl("http://localhost:8080")
    .defaultHeader(HttpHeaders.CONTENT_TYPE, MediaType.APPLICATION_JSON_VALUE)
    .build();
```

Student: Add CourseId

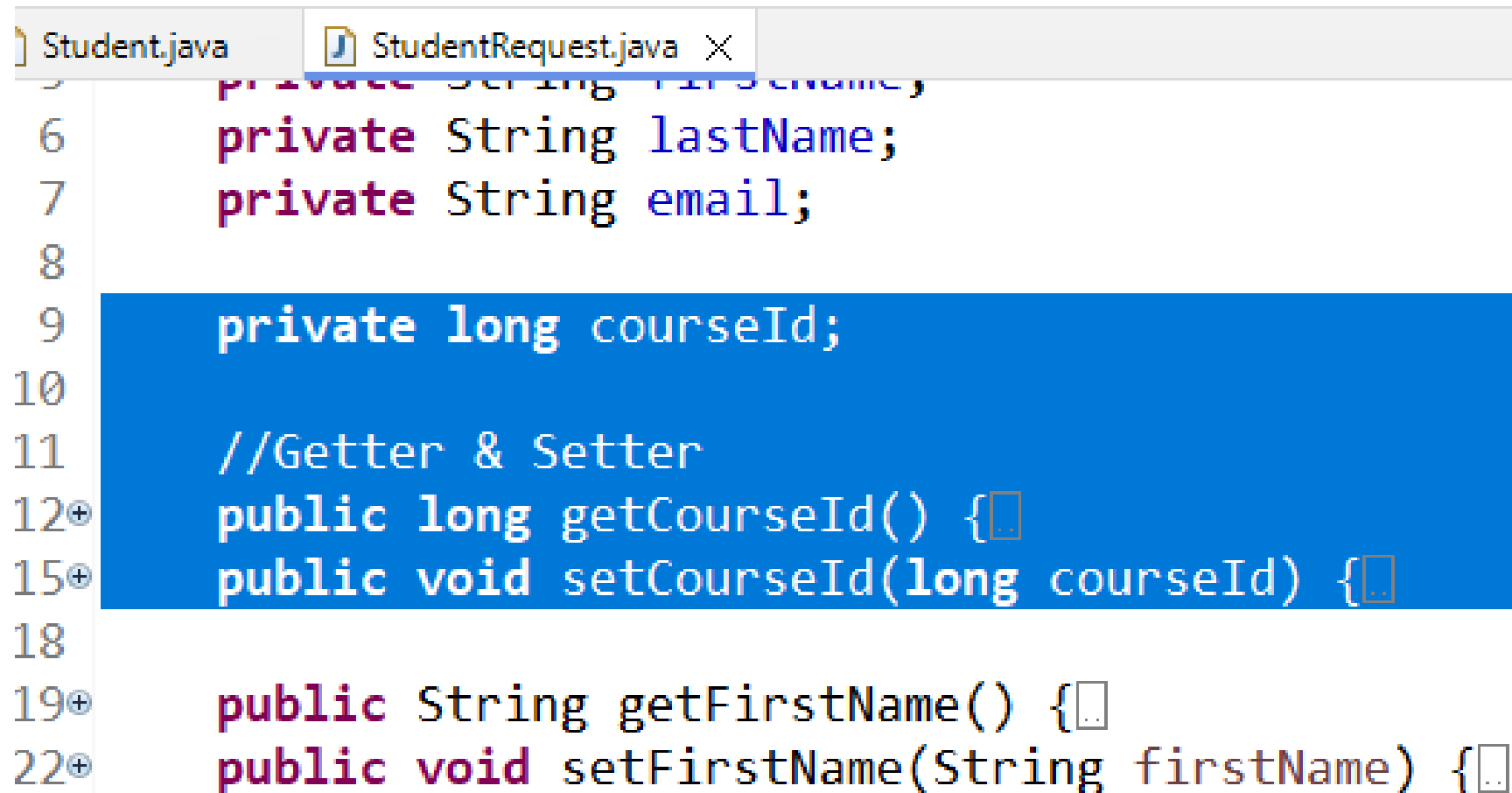


The screenshot shows a code editor window titled "Student.java". The code is as follows:

```
14     private String lastName;  
15     private String email;  
16  
17     private long courseId;  
18  
19     //Getter & Setter  
20     public long getCourseId() {..  
23     public void setCourseId(long courseId) {..  
26  
27     public Long getStudentId() {..  
30     public void setStudentId(Long studentId) {..  
--
```

Lines 17 through 23 are highlighted in blue. The line numbers on the left are 14, 15, 16, 17, 18, 19, 20+, 23+, 26, 27+, and 30+.

StudentRequest: Add CourseId



```
Student.java StudentRequest.java X
5 private String firstName;
6 private String lastName;
7 private String email;
8
9 private long courseId;
10
11 //Getter & Setter
12+ public long getCourseId() {
15+ public void setCourseId(long courseId) {
18
19+ public String getFirstName() {
22+ public void setFirstName(String firstName) {
```

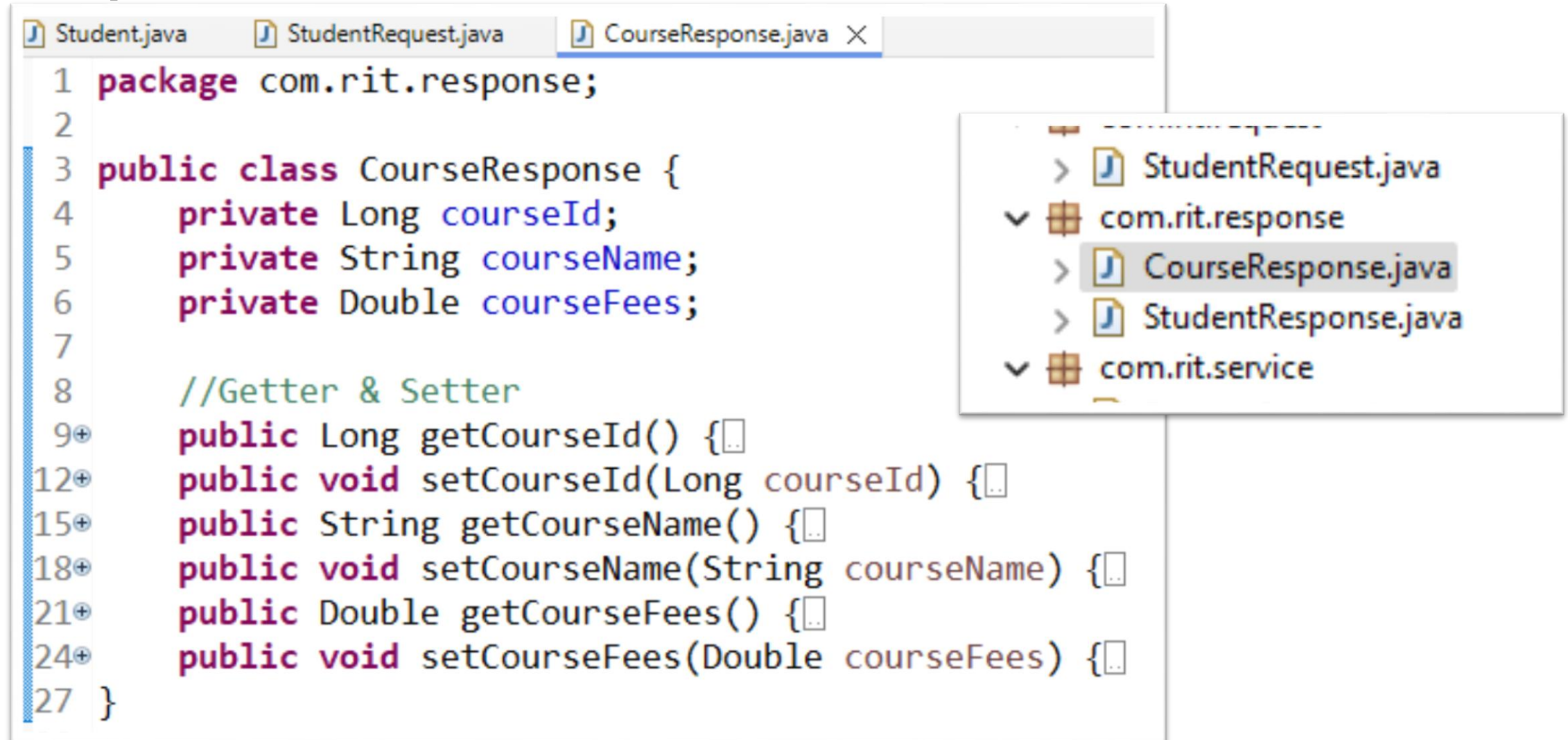
Update it in Service class

StudentService.java ×

```
27  
28 public StudentResponse createStudent(StudentRequest studentRequest) {  
29     Student student = new Student();  
30     student.setFirstName(studentRequest.getFirstName());  
31     student.setLastName(studentRequest.getLastName());  
32     student.setEmail(studentRequest.getEmail());  
33     student.setCourseId(studentRequest.getCourseId());  
34     studentRepository.save(student);  
}
```

Create a CourseResponse Class in response

```
Student.java StudentRequest.java CourseResponse.java X
1 package com.rit.response;
2
3 public class CourseResponse {
4     private Long courseId;
5     private String courseName;
6     private Double courseFees;
7
8     //Getter & Setter
9     public Long getCourseId() {..
12    public void setCourseId(Long courseId) {..
15    public String getCourseName() {..
18    public void setCourseName(String courseName) {..
21    public Double getCourseFees() {..
24    public void setCourseFees(Double courseFees) {..
27 }
```



The image shows a screenshot of an IDE with three tabs: Student.java, StudentRequest.java, and CourseResponse.java. The CourseResponse.java tab is active, displaying the following code:

```
1 package com.rit.response;
2
3 public class CourseResponse {
4     private Long courseId;
5     private String courseName;
6     private Double courseFees;
7
8     //Getter & Setter
9     public Long getCourseId() {..
10
11
12    public void setCourseId(Long courseId) {..
13
14
15    public String getCourseName() {..
16
17
18    public void setCourseName(String courseName) {..
19
20
21    public Double getCourseFees() {..
22
23
24    public void setCourseFees(Double courseFees) {..
25
26
27 }
```

On the right side, there is a project structure view showing the following hierarchy:

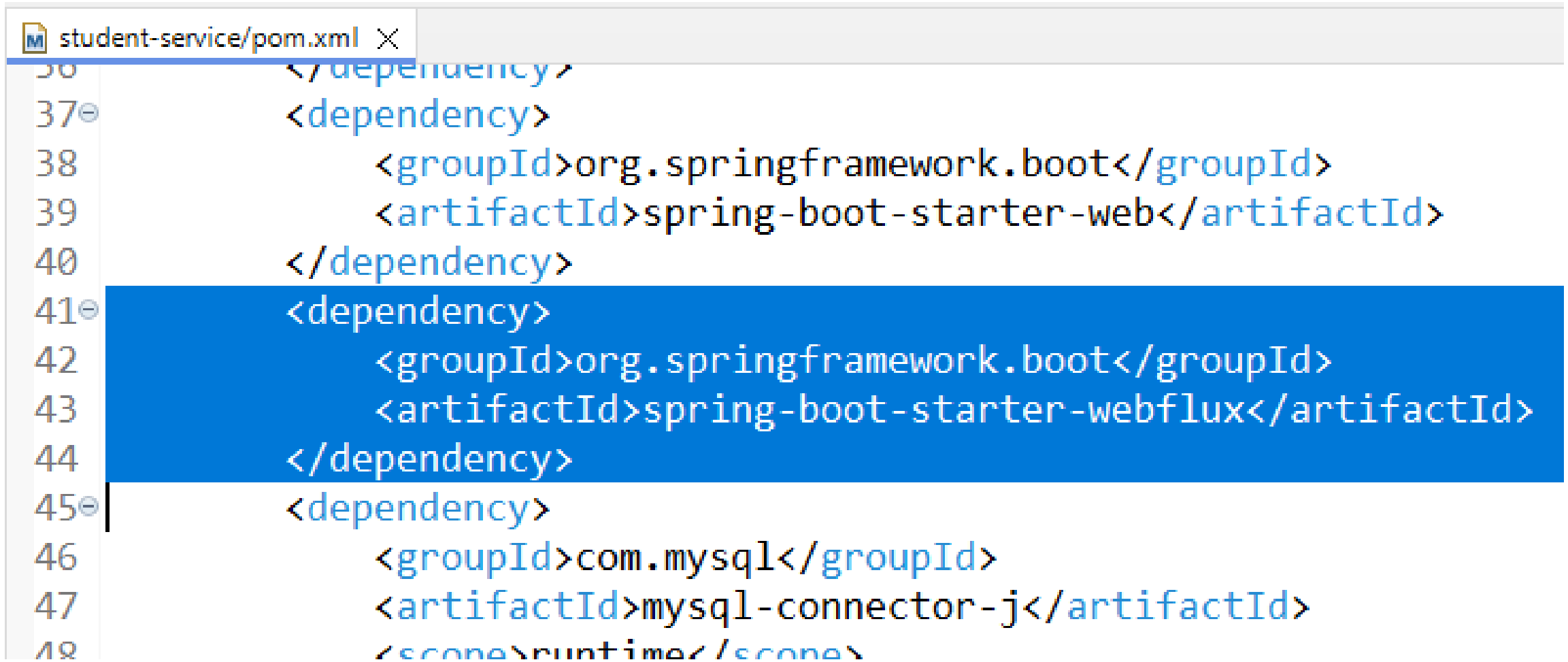
- com.rit.response
 - StudentRequest.java
 - CourseResponse.java (selected)
 - StudentResponse.java
- com.rit.service

Add CourseResponse in StudentResponse

StudentResponse.java

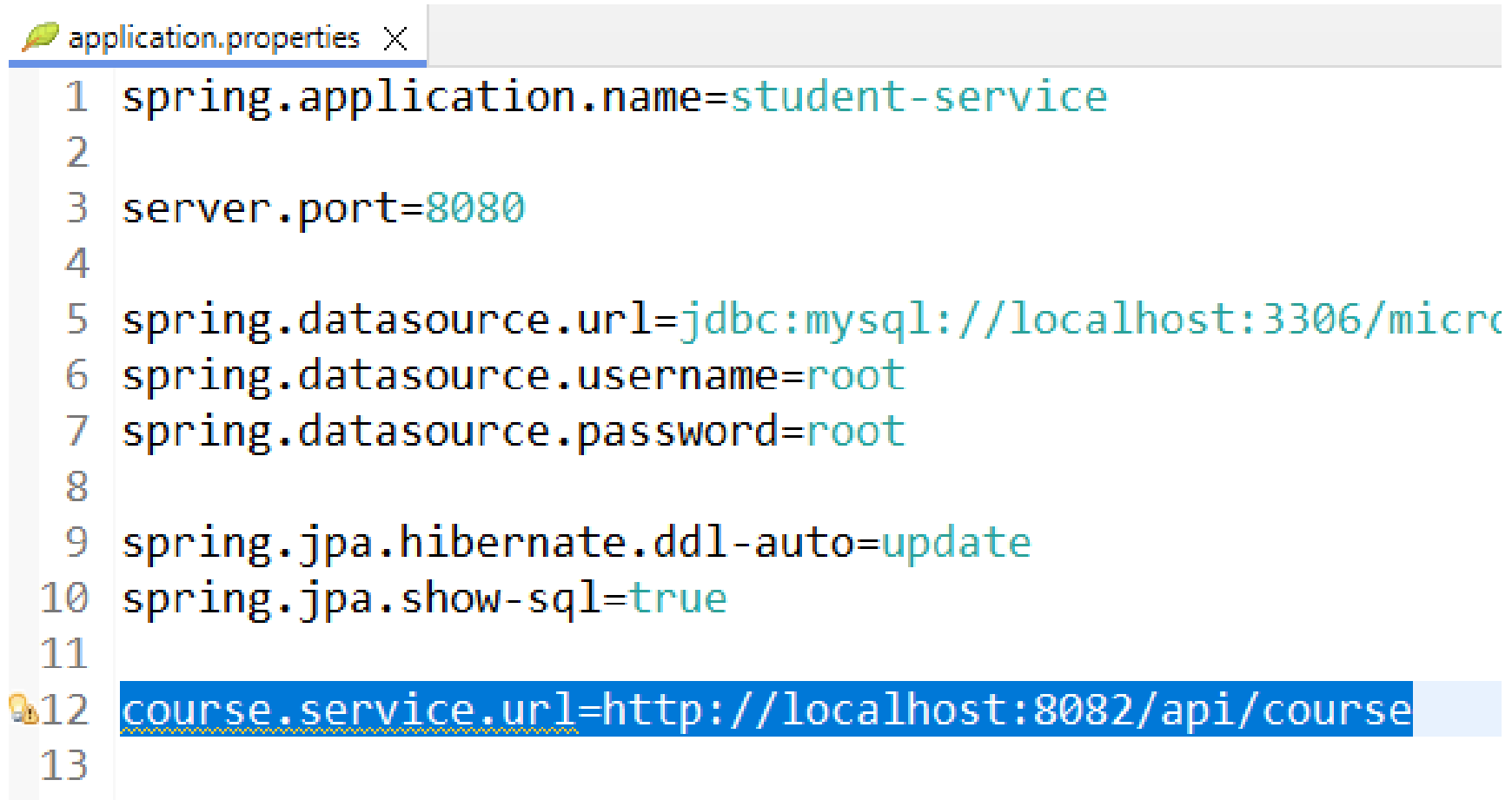
```
5 public class StudentResponse {  
6     private Long studentId;  
7     private String firstName;  
8     private String lastName;  
9     private String email;  
10  
11     private CourseResponse course;  
12  
13     //Getter & Setter  
14+ public CourseResponse getCourse() {  
17+ public void setCourse(CourseResponse course) {
```

Add Webflux dependency (for WebClient)



```
student-service/pom.xml X
36 </dependency>
37 <dependency>
38     <groupId>org.springframework.boot</groupId>
39     <artifactId>spring-boot-starter-web</artifactId>
40 </dependency>
41 <dependency>
42     <groupId>org.springframework.boot</groupId>
43     <artifactId>spring-boot-starter-webflux</artifactId>
44 </dependency>
45 <dependency>
46     <groupId>com.mysql</groupId>
47     <artifactId>mysql-connector-j</artifactId>
48 </dependency>
```

Add course API URL in properties file



The screenshot shows a code editor window titled 'application.properties' with a close button. The file contains the following properties:

```
1 spring.application.name=student-service
2
3 server.port=8080
4
5 spring.datasource.url=jdbc:mysql://localhost:3306/micro
6 spring.datasource.username=root
7 spring.datasource.password=root
8
9 spring.jpa.hibernate.ddl-auto=update
10 spring.jpa.show-sql=true
11
12 course.service.url=http://localhost:8082/api/course
13
```

The line 'course.service.url=http://localhost:8082/api/course' is highlighted in blue. A yellow warning icon is visible to the left of line 12.

Add url and WebClient Bean in Main class

```
StudentServiceApplication.java ×
9 @SpringBootApplication
10 public class StudentServiceApplication {
11
12     @Value("${course.service.url}")
13     private String courseServiceUrl;
14
15     public static void main(String[] args) {
16         SpringApplication.run(StudentServiceApplication.class, args);
17     }
18
19     @Bean
20     public WebClient webClient() {
21         WebClient webClient = WebClient.builder()
22             .baseUrl(courseServiceUrl).build();
23         return webClient;
24     }
25 }
```

In Student Service

- Autowire WebClient
- Create a method to get CourseResponse
- Update StudentReponse in getByld method
- Update StudentReponse in createStudent method

Add WebClient and CourseResponse Method

```
StudentService.java x
20  Logger logger = LoggerFactory.getLogger(StudentService.class);
21
22  @Autowired
23  private StudentRepository studentRepository;
24
25  @Autowired
26  private WebClient webClient;
27
28  public StudentResponse createStudent(StudentRequest studentRequest) {}
29
30  public StudentResponse getById(long id) {}
31
32  public CourseResponse getCourseById(long courseId) {
33      Mono<CourseResponse> courseResponse = webClient.get()
34          .uri("/getById/"+courseId).retrieve()
35          .bodyToMono(CourseResponse.class);
36      return courseResponse.block();
37  }
38 }
```

Update in create & getById methods

```
StudentService.java X
31      student.setLastName(studentRequest.getLastName());
32      student.setEmail(studentRequest.getEmail());
33      student.setCourseId(studentRequest.getCourseId());
34      studentRepository.save(student);
35
36      StudentResponse studentResponse = new StudentResponse(student);
37      studentResponse.setCourse(getCourseById(student.getCourseId()));
38      return studentResponse;
39  }
40
41  public StudentResponse getById(long id) {
42      logger.info("In getById : "+id);
43      Student student = studentRepository.findById(id).get();
44
45      StudentResponse studentResponse = new StudentResponse(student);
46      studentResponse.setCourse(getCourseById(student.getCourseId()));
47      return studentResponse;
48  }
49
50  public CourseResponse getCourseById(long courseId) {
```



http://localhost:8080/api/student/create

Save

POST

http://localhost:8080/api/student/create

Send

Body

...

raw

JSON

Beautify

```
1 {
2   ...."firstName"::"Bruce",
3   ...."lastName"::"Lee",
4   ...."email"::"bruce@gmail.com",
5   ...."courseId"::1
6 }
```

Body



200 OK

938 ms

318 B

Save Response

Pretty

JSON



```
1 {
2   "studentId": 5,
3   "firstName": "Bruce",
4   "lastName": "Lee",
5   "email": "bruce@gmail.com",
6   "course": {
7     "courseId": 1,
8     "courseName": "Java Full Stack",
9     "courseFees": 25000.0
10  }
11 }
```