Iamneo | Rest API with Spring Boot and JPA

Objective
To build a complete REST API backend

Day 1 | Class Exercise Lab 1 | Introduction to REST API

In this Lab, we will create three services using proper URIs and HTTP methods:

1. *@GetMapping("/"):*
   Return a string message "Welcome to REST API"
   API Endpoint uri – /welcome
2. *@GetMapping("/student"):*
   Retrieve all the students
   API Endpoint uri - /students.
3. *@GetMapping("/students/{studentId}"):*
   Retrieve specific student
   API Endpoint uri - /students/{studentId}.

What will you learn?

You will learn
- What is a REST Service?
- How to bootstrap a Rest Service application with Spring Initializr?
- How to create a Get REST Service for retrieving the courses that a student registered for?
- How to create a Post REST Service for registering a course for student?
- How to execute Rest Services from Postman?

User Story #1 | Tools and Setup

You should be able to install the following tools in the system
* Maven 3.0+ is your build tool
* Your favourite IDE. We use Eclipse.
* JDK 17
* Postman or Swagger

User Story #2 | Bootstrap a Rest Service application with Spring Initializr

You should be able to bootstrap the REST Services with Spring Initializr. To do this, click [https://start.spring.io/]. Spring Initializr [http://start.spring.io/] is great tool to bootstrap your Spring Boot projects.

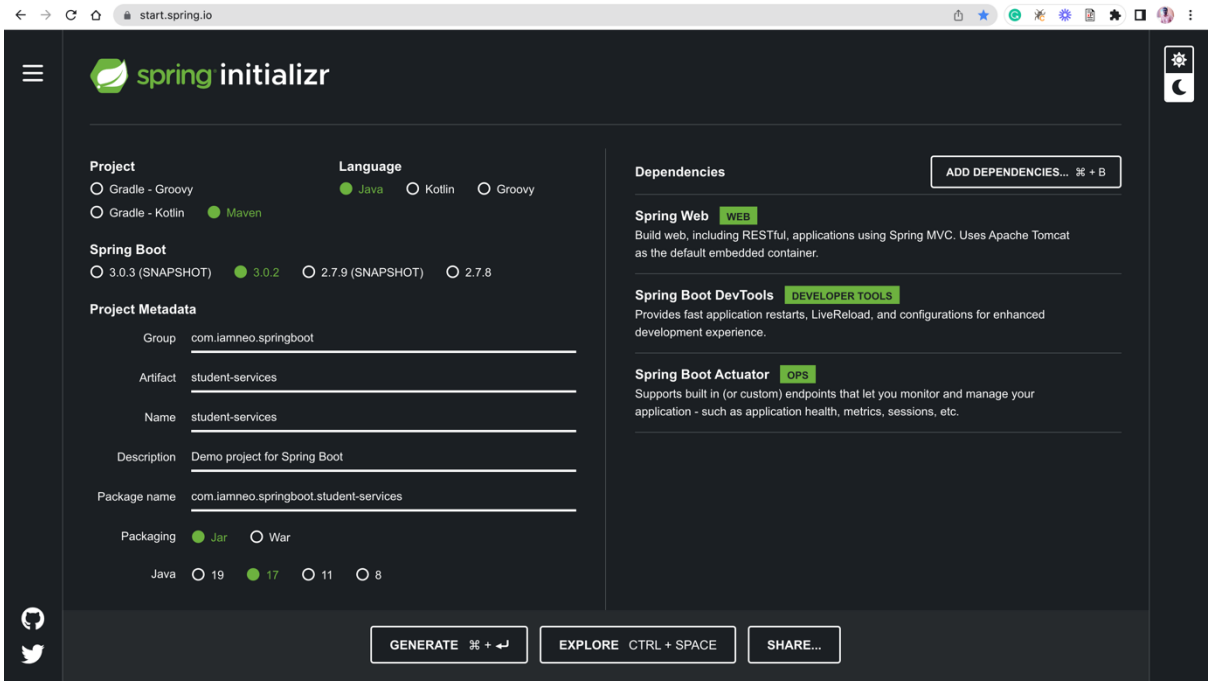
* Launch Spring Initializr and choose the following
* Choose com.iamneo.springboot as Group

* Choose student-services as Artifact
* Choose the following dependencies

- Web
- Actuator
- DevTools

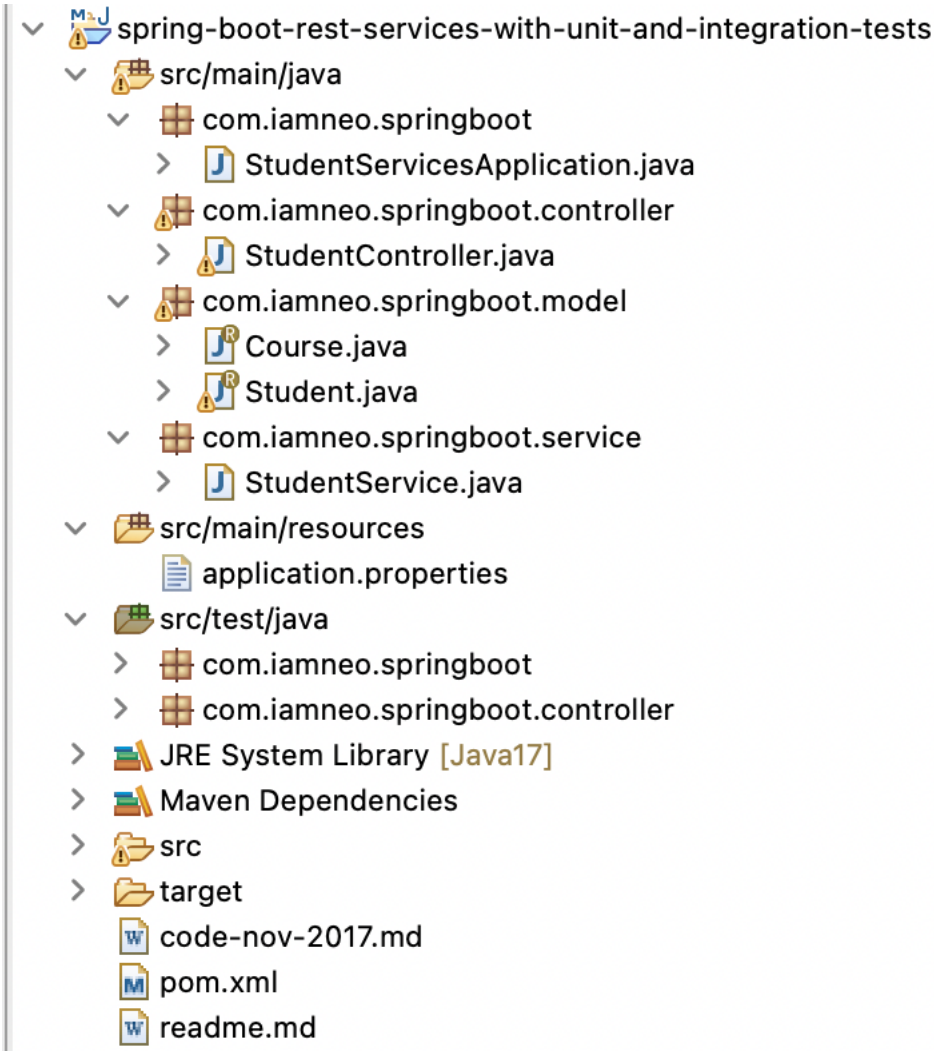
* Check if maven and Jdk 17 are selected.
* Click Generate Project.
* Import the project into Eclipse. File -> Import -> Existing Maven Project.

Output Screenshot:



User Story #3 | Create the project structure

Reference Project Structure



A few details:
StudentController.java - Rest controller exposing all three service methods discussed above.

Course.java, Student.java, StudentService.java - Business Logic for the application. StudentService exposes a couple of methods we would consume from our Rest Controller.

StudentServicesApplication.java - Launcher for the Spring Boot Application. To run the application, just launch this file as Java Application.

pom.xml - Contains all the dependencies needed to build this project. We will use Spring Boot Starter Web.

Implement the Model Classes

A student can take multiple courses. A course has an id, name, description and a list of steps you need to complete to finish the course. A student has an id, name, description and a list of courses he/she is currently registered for.

User Story #4 | Create a Model Class Student.java

Create a record class called Student.java inside com.iamneo.springboot.model and define the attributes

Student.java

```
public record Student{String id,
                      String name,
                      String description,
                      List<Course> courses
}
```

Implement the Business Layer

User Story #5 | Create a class called StudentService.java and implement the following methods

- public String getMessage() – Retrieve a string message "*Welcome to REST API"*
- public List<Student> retrieveAllStudents() - Retrieve details for all students
- public Student retrieveStudent(String studentId) - Retrieve a specific student details

Implement the REST Controller

Adding a couple of GET Rest Services

User Story #6 | Implement the StudentController.java

The Rest Service StudentController exposes a couple of get services.

- @Autowired private StudentService studentService : We are using Spring Autowiring to wire the student service into the StudentController.
- @GetMapping("/welcome "): Exposing a Get Service
- @GetMapping("/students "): Exposing a Get Service
- @GetMapping("/students/{studentId}"): Exposing a Get Service for retrieving specific student.

User Story #7 | Executing the Http Get Operation Using Postman

We will access a request to http://localhost:8080/students/welcome to test the service.