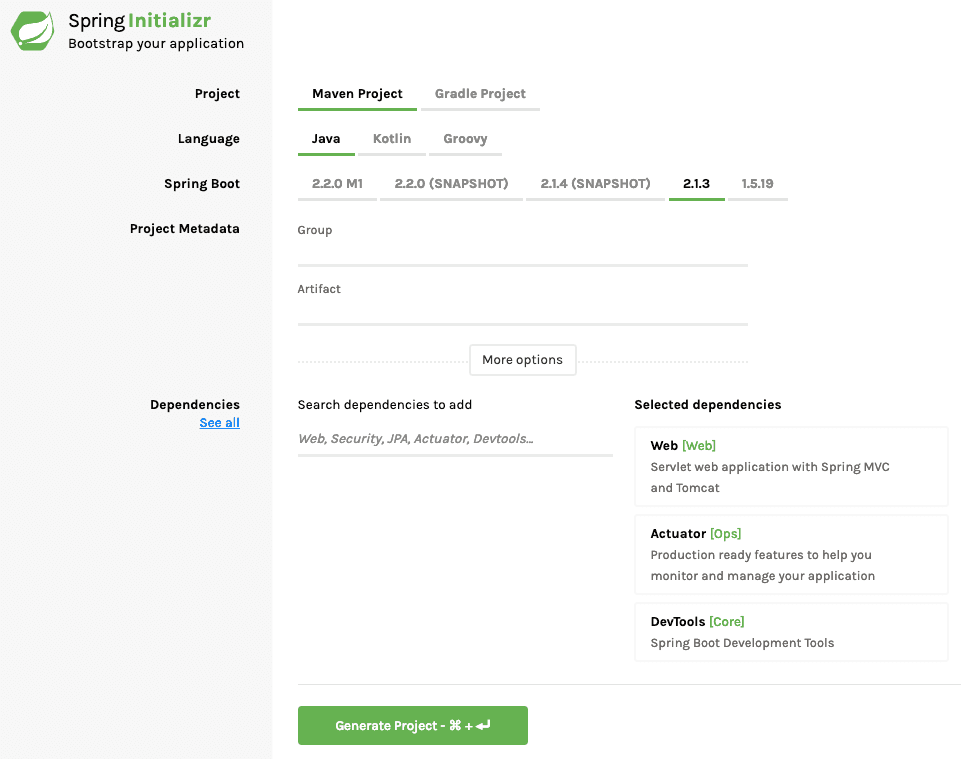
# **Hands-On Exercise 01: Creating a REST Service with Spring Boot**

## **Let’s Get started**

Let’s get started with the exercise by Creating Bootstrapping REST Services with Spring Initializr.

Go to [**http://start.spring.io/**](https://start.spring.io/), then set the properties as following:



## **Launch Spring Initializr and choose the following**

* + Choose com.exmaple as Group
  + Choose student-services as Artifact
  + Choose following dependencies
    - Web
    - Actuator
    - DevTools

Click **Generate Project**.

Now Import the project into Eclipse/IntelliJ. File -> Import -> Existing Maven Project.

Let’s Implement Business Service for your Application. All applications need data. Instead of talking to a real database, we will use an ArrayList - kind of an in-memory data store.

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. We have StudentService exposing methods to

* public List<Student> retrieveAllStudents() - Retrieve details for all students
* public Student retrieveStudent(String studentId) - Retrieve a specific student details
* public List<Course> retrieveCourses(String studentId) - Retrieve all courses a student is registered for
* public Course retrieveCourse(String studentId, String courseId) - Retrieve details of a specific course a student is registered for
* public Course addCourse(String studentId, Course course) - Add a course to an existing student

Refer to these files at the bottom of the article for exact implementation of the Service StudentService and the model classes Course and Student.

* src/main/java/com/exmaple/springboot/model/Course.java
* src/main/java/com/exmaple/springboot/model/Student.java
* src/main/java/com/exmaple/springboot/service/StudentService.java

## **Adding Couple of GET Rest Services**

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("/students/{studentId}/courses"): Exposing a Get Service with studentId as a path variable
* @GetMapping("/students/{studentId}/courses/{courseId}"): Exposing a Get Service for retrieving specific course of a student.
* @PathVariable String studentId: Value of studentId from the uri will be mapped to this parameter.

|  |
| --- |
| package com.example.controller;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.web.bind.annotation.GetMapping;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RestController;  import com.example.model.Course;  import com.example.service.StudentService;  @RestController  public class StudentController {  @Autowired  private StudentService studentService;  @GetMapping("/students/{studentId}/courses")  public List<Course> retrieveCoursesForStudent(@PathVariable String studentId) {  return studentService.retrieveCourses(studentId);  }    @GetMapping("/students/{studentId}/courses/{courseId}")  public Course retrieveDetailsForCourse(@PathVariable String studentId,  @PathVariable String courseId) {  return studentService.retrieveCourse(studentId, courseId);  }  } |

## **Executing the Get Service Using Postman**

We will fire a request to http://localhost:8080/students/Student1/courses/Course1 to test the service. Response is as shown below.



## **Adding a POST Rest Service**

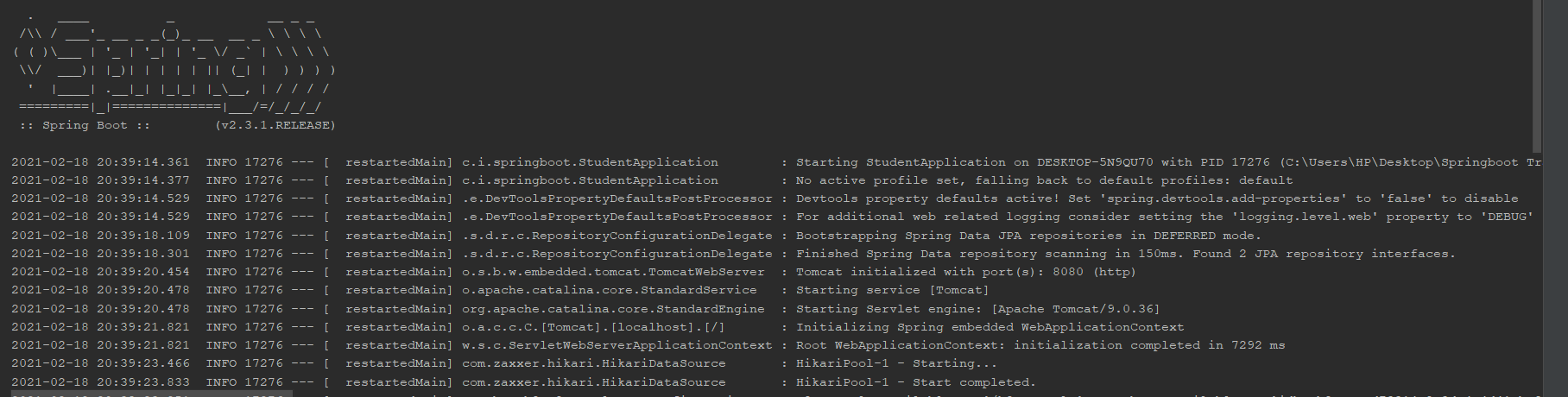
A POST Service should return a status of created (201) when the resource creation is successful.

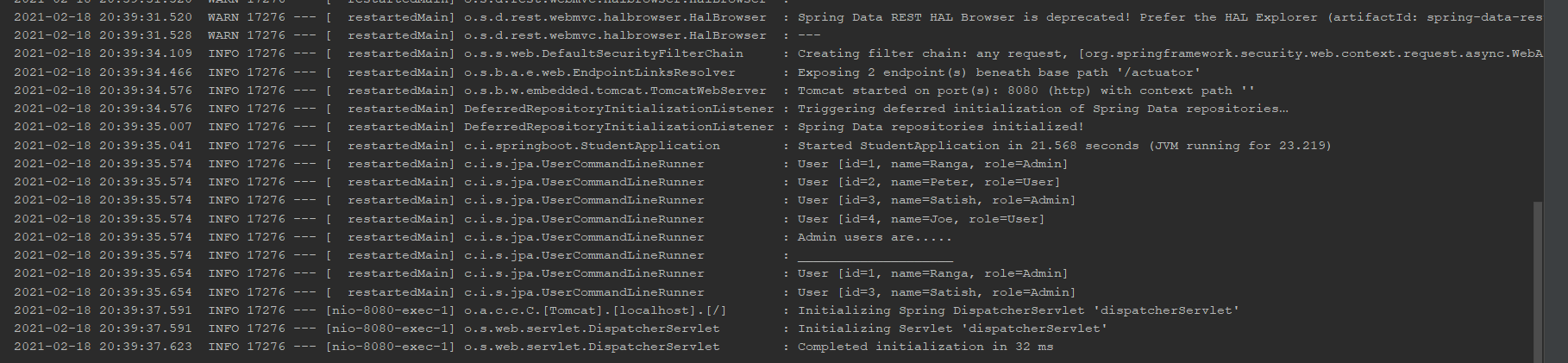
@PostMapping("/students/{studentId}/courses"): Mapping a url for the POST Request @RequestBody Course newCourse: Using Binding to bind the body of the request to Course object. ResponseEntity.created(location).build(): Return a status of created. Also return the location of created resource as a Response Header.

|  |
| --- |
| @PostMapping("/students/{studentId}/courses")  public ResponseEntity<Void> registerStudentForCourse(  @PathVariable String studentId, @RequestBody Course newCourse) {  Course course = studentService.addCourse(studentId, newCourse);  if (course == null)  return ResponseEntity.noContent().build();  URI location = ServletUriComponentsBuilder.fromCurrentRequest().path(  "/{id}").buildAndExpand(course.getId()).toUri();  return ResponseEntity.created(location).build();} |

## **Now Run the code in your IDE**

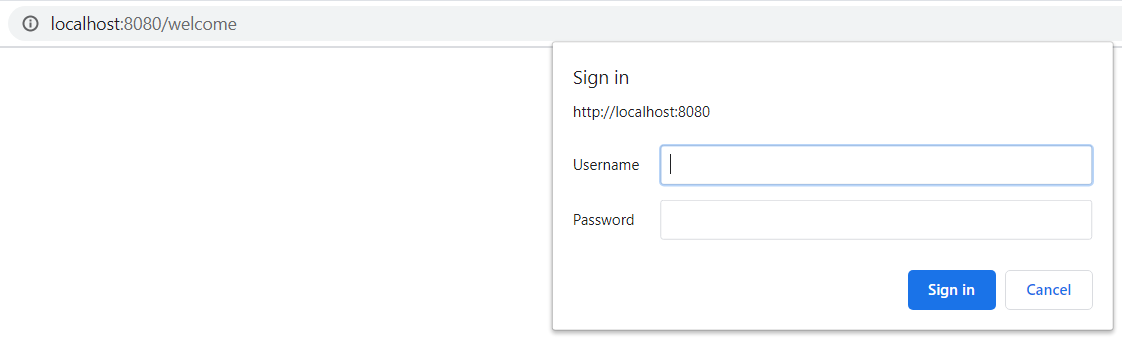
Click on the run button present on top of page to run the code. You would receive such output.



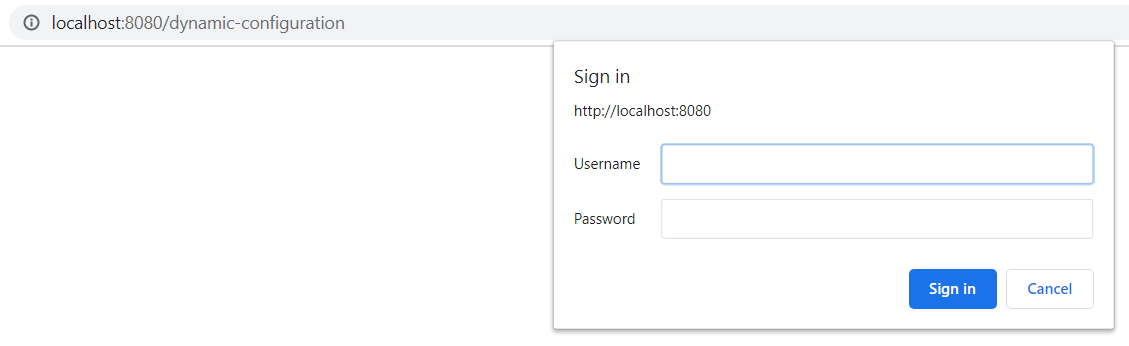


## **Executing a POST Rest Service**

Example Request is shown below. It contains all the details to register a course to a student. Type localhost:8080/welcome or localhost:8080/dynamic-configuration. A login screen would appear, login with correct credentials.



Note: Use Username as admin and password as admin



Note: Use Username as admin and password as admin

**Voila!!** We have successfully completed this exercise.