**Introduction to Spring Boot Framework**

Use Cases Manual

Sandbox Link [Spring Boot](https://share.percipio.com/cd/J676mja_Z)

**Creating a REST Service with Spring Boot**

This guide will help you create a simple REST service using Spring Boot.

**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?

**Tools you will need**

* Maven 3.0+ is your build tool
* Your favorite IDE. We use Eclipse.
* JDK 1.8+
* POSTMAN

What is REST?

* REST stands for REpresentational State Transfer.
* REST specifies a set of architectural constraints.

The five important constraints for RESTful Web Service are

1. Client - Server : There should be a service producer and a service consumer.
2. The interface (URL) is uniform and exposing resources.
3. The service is stateless.
4. The service results should be Cacheable. HTTP cache, for example.
5. Service should assume a Layered architecture. Client should not assume direct connection to server - it might be getting info from a middle layer - cache.

Using appropriate Request Methods

Always use HTTP Methods. Best practices with respect to each HTTP method is described below:

* GET : Should not update anything. Should be idempotent (same result in multiple calls). Possible Return Codes 200 (OK) + 404 (NOT FOUND) +400 (BAD REQUEST)
* POST : Should create new resource. Ideally return JSON with link to newly created resource. Same return codes as get possible. In addition : Return code 201 (CREATED) is possible.
* PUT : Update a known resource. ex: update client details. Possible Return Codes : 200(OK)
* DELETE : Used to delete a resource.

Bootstrapping REST Services with Spring Initializr

Creating a REST service with Spring Initializr

Spring Initializr <http://start.spring.io/> is great tool to bootstrap your Spring Boot projects.

Launch Spring Initializr and choose the following

Choose com.optum.springboot as Group

Choose student-services as Artifact

Choose following dependencies

Web

Actuator

DevTools

Click Generate Project.

Import the project into Eclipse. **File -> Import -> Existing Maven Project**.

If you want to understand all the files that are part of this project, you can go here.

**A few details:**

**StudentController.java** - Rest controller exposing all the 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.

**StudentControllerIT.java** - Integration Tests for the Rest Services.

**StudentControllerTest.java** - Unit Tests for the Rest Services.

**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.

Implementing 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

**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.

{

"id": "Course1",

"name": "Spring",

"description": "10 Steps",

"steps": [

"Learn Maven",

"Import Project",

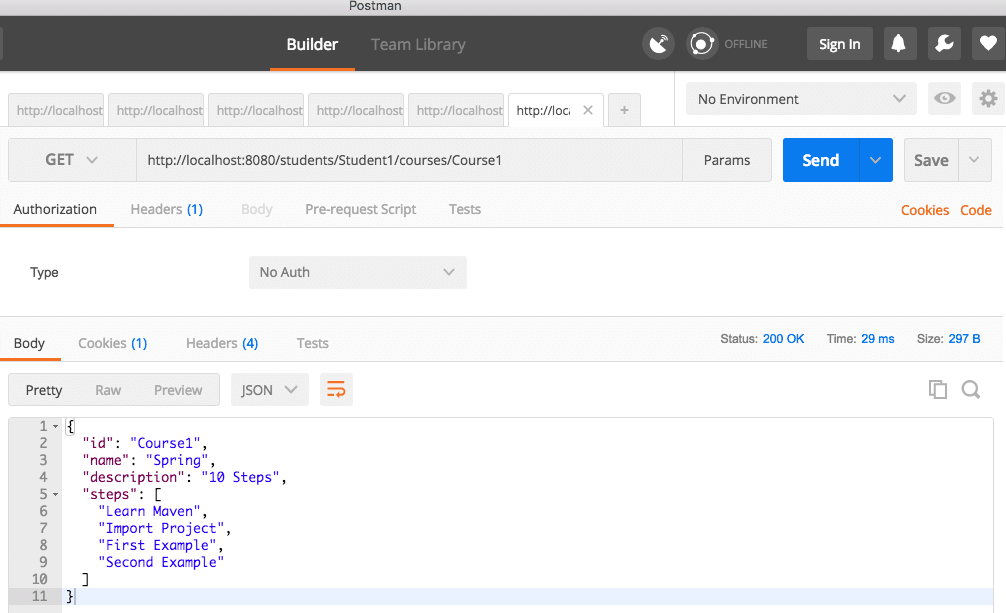
"First Example",

"Second Example"

]

}

Below picture shows how we can execute this Get Service from Postman - my favorite tool to run rest services.



**Executing a POST Rest Service**

Example Request is shown below. It contains all the details to register a course to a student.

{

"name": "Microservices",

"description": "10 Steps",

"steps": [

"Learn How to Break Things Up",

"Automate the hell out of everything",

"Have fun"

]

}

Note : Make sure you go to the Body tab and select raw. Select JSON from the dropdown. Copy above request into body.

The URL we use is <http://localhost:8080/students/Student1/courses>.

