Hi Team,  
[#REST](https://www.facebook.com/hashtag/rest?source=feed_text&epa=HASHTAG) Service [#with](https://www.facebook.com/hashtag/with?source=feed_text&epa=HASHTAG) [#Spring](https://www.facebook.com/hashtag/spring?source=feed_text&epa=HASHTAG) Boot [#Real\_Time](https://www.facebook.com/hashtag/real_time?source=feed_text&epa=HASHTAG)  
#Rest Services in this Guide  
In this guide, we will create three services using proper URIs and HTTP methods:  
@GetMapping("/students/{studentId}/courses"): You can ask the courses a specific student has registered for using request method Get and example uri /students/Student1/courses.  
@GetMapping("/students/{studentId}/courses/{courseId}"): You can ask a specific course for a specific student using request method Get and example uri /students/Student1/courses/Course1.  
@PostMapping("/students/{studentId}/courses") : You can register a student for a course by sending a POST request to URI /students/Student1/courses

[#What](https://www.facebook.com/hashtag/what?source=feed_text&epa=HASHTAG) is REST?  
REST stands for REpresentational State Transfer. REST specifies a set of architectural constraints. Any service which satisfies these constraints is called RESTful Service.

The five important constraints for RESTful Web Service are

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

[#Richardson](https://www.facebook.com/hashtag/richardson?source=feed_text&epa=HASHTAG) Maturity Model  
Richardson Maturity Model is used to identify the maturity level of a Restful Web Service. Following are the different levels and their characteristics:

Level 0 : Expose SOAP web services in REST style. Expose action based services (http://server/getPosts, http://server/deletePosts, http://server/doThis, http://server/doThat etc) using REST.  
Level 1 : Expose Resources with proper URI’s (using nouns). Ex: http://server/accounts, http://server/accounts/10. However, HTTP Methods are not used.  
Level 2 : Resources use proper URI’s + HTTP Methods. For example, to update an account, you do a PUT to . The create an account, you do a POST to . Uri’s look like posts/1/comments/5 and accounts/1/friends/1.  
Level 3 : HATEOAS (Hypermedia as the engine of application state). You will tell not only about the information being requested but also about the next possible actions that the service consumer can do. When requesting information about a facebook user, a REST service can return user details along with information about how to get his recent posts, how to get his recent comments and how to retrieve his friend’s list.

[#Using](https://www.facebook.com/hashtag/using?source=feed_text&epa=HASHTAG) 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.

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.  
Bootstrapping REST Services with Spring Initializr  
Creating a REST service with Spring Initializr is a cake walk. We will use Spring Web MVC as our web framework.

Spring Initializr [http://start.spring.io/](https://l.facebook.com/l.php?u=http%3A%2F%2Fstart.spring.io%2F%3Ffbclid%3DIwAR0tfZU0XauKAeR1eFNVAeJg55jteobd_DBEZmVBA3IcjGQP4XHHc_6wSV4&h=AT0qAh0mN3DT7w87SapjTZ6iZ05Zvs6euETPVraJpQjohav11Tf397k5gL5b1EA9vbC5TlHBwKtDbZF4uw2Y6nm-RL2eh_svhRClval3IK3o8lk18su8cGQAQTVrbqH7HvaBB7L_alwl8B84kOhzmX_-G6TMgzlxmPo) is great tool to bootstrap your Spring Boot projects.  
As shown in the image above, following steps have to be done

Launch Spring Initializr and choose the following  
Choose com.in28minutes.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.  
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  
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/in28minutes/springboot/model/Course.java  
src/main/java/com/in28minutes/springboot/model/Student.java  
src/main/java/com/in28minutes/springboot/service/StudentService.java  
Adding Couple of GET Rest Services  
The Rest Service StudentController exposes 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.in28minutes.springboot.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.in28minutes.springboot.model.Course;  
import com.in28minutes.springboot.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.

{  
"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. Image

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();  
}

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

Complete Code Example  
pom.xml  
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="[http://maven.apache.org/POM/4.0.0](https://l.facebook.com/l.php?u=http%3A%2F%2Fmaven.apache.org%2FPOM%2F4.0.0%3Ffbclid%3DIwAR08q6WCN4xhj3S-JNLQ0aFFU6FhdnIWk_2Gl3qAUQbkFKaBkcKBMwyCx9g&h=AT2vOordcUZeuKWCueuajkbeLWFf6wo_RPDCzuCaUOdmy5UO3zMtxYcb9fnwraGfuYeA75gVJg4ReuK1h7dMneU3J6-9KtMxnRA0FEVXj6RCI6Vj2acO6mr-yQx01mnRXNrZc7EjjTxr5zfsc2bPZCGkKiugl4aZl-M)" xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance](https://l.facebook.com/l.php?u=https%3A%2F%2Fwww.w3.org%2F2001%2FXMLSchema-instance%3Ffbclid%3DIwAR1nprzju2Yrvx1vGWsgybE_nHyGzCB0ob3WUrTBKW7MWDmpDOrjujGNiFI&h=AT2GwmfoW3MudV9SKGKdbAdn0It9lN7PI4xVFVCI4EiASbIAf-4mnLi5-2T14flIhOCR5IV3KstzgOkv4EV8rWAhYaDqHuyUpapq1VjNyX2Phnd1fZ6C1kjSVFWcrk-ojRR4Iv_3rSElT5gpcwZPCCKZc6BAiKMFK80)"  
xsi:schemaLocation="[http://maven.apache.org/POM/4.0.0](https://l.facebook.com/l.php?u=http%3A%2F%2Fmaven.apache.org%2FPOM%2F4.0.0%3Ffbclid%3DIwAR3cuJemW-apilwa7sbA1Sembxl0boSLokvCpB4aIdY9qLuG6gIddr4H824&h=AT0IIeE3OJjQssh4DAm6PaiOo0d3q5pEouvtv_dcoUvw-pWEVDEodlTZm4AWKMm3IMhfLs4KQehfmN9uzkKHnOhYVece5ACJd9yhiUsNbqM5O1Ro3lcOl7v0DeIPOgNp0dBTUkh_M40po9dHG3N-jDVjxu89z0MHKc8)[http://maven.apache.org/xsd/maven-4.0.0.xsd](https://l.facebook.com/l.php?u=http%3A%2F%2Fmaven.apache.org%2Fxsd%2Fmaven-4.0.0.xsd%3Ffbclid%3DIwAR25MHkFB1tD8jATnUxCMZHbo_694Nd-K8QRlNR669xL3yXhTqGjt8rNiXE&h=AT0--0Zt2HyomCFNyfY8ylsbn2o2b6-O4x5-iNF8ses_4Mu-5GzzDrNCtBp8RMFLMwvqMw3KhzU07haFwYXxEBv-I3P_gY7dX62MnlfcrcM6cdTcSgGiyWwG4xWl5tqBVXfEviumnDv4ylFBxLAFJoLJhtmp3YvAkRU)">  
<modelVersion>4.0.0</modelVersion>

<groupId>com.in28minutes.springboot</groupId>  
<artifactId>student-services</artifactId>  
<version>0.0.1-SNAPSHOT</version>  
<packaging>jar</packaging>

<name>student-services</name>  
<description>Demo project for Spring Boot</description>

<parent>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-parent</artifactId>  
<version>1.4.4.RELEASE</version>  
<relativePath/> <!-- lookup parent from repository -->  
</parent>

<properties>  
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
<java.version>1.8</java.version>  
</properties>

<dependencies>  
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>  
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-web</artifactId>  
</dependency>

<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-devtools</artifactId>  
<scope>runtime</scope>  
</dependency>  
<dependency>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-starter-test</artifactId>  
<scope>test</scope>  
</dependency>  
</dependencies>

<build>  
<plugins>  
<plugin>  
<groupId>org.springframework.boot</groupId>  
<artifactId>spring-boot-maven-plugin</artifactId>  
</plugin>  
</plugins>  
</build>

</project>  
src/main/java/com/in28minutes/springboot/controller/StudentController.java  
package com.in28minutes.springboot.controller;

import java.net.URI;  
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.in28minutes.springboot.model.Course;  
import com.in28minutes.springboot.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);  
}

@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();  
}

}  
src/main/java/com/in28minutes/springboot/model/Course.java  
package com.in28minutes.springboot.model;

import java.util.List;

public class Course {  
private String id;  
private String name;  
private String description;  
private List<String> steps;

// Needed by Caused by: com.fasterxml.jackson.databind.JsonMappingException:  
// Can not construct instance of com.in28minutes.springboot.model.Course:  
// no suitable constructor found, can not deserialize from Object value  
// (missing default constructor or creator, or perhaps need to add/enable  
// type information?)  
public Course() {

}

public Course(String id, String name, String description, List<String> steps) {  
super();  
this.id = id;  
this.name = name;  
this.description = description;  
this.steps = steps;  
}

public String getId() {  
return id;  
}

public void setId(String id) {  
this.id = id;  
}

public String getDescription() {  
return description;  
}

public String getName() {  
return name;  
}

public List<String> getSteps() {  
return steps;  
}

@Override  
public String toString() {  
return String.format(  
"Course [id=%s, name=%s, description=%s, steps=%s]", id, name,  
description, steps);  
}

@Override  
public int hashCode() {  
final int prime = 31;  
int result = 1;  
result = prime \* result + ((id == null) ? 0 : id.hashCode());  
return result;  
}

@Override  
public boolean equals(Object obj) {  
if (this == obj)  
return true;  
if (obj == null)  
return false;  
if (getClass() != obj.getClass())  
return false;  
Course other = (Course) obj;  
if (id == null) {  
if (other.id != null)  
return false;  
} else if (!id.equals(other.id))  
return false;  
return true;  
}

}  
src/main/java/com/in28minutes/springboot/model/Student.java  
package com.in28minutes.springboot.model;

import java.util.List;

public class Student {  
private String id;  
private String name;  
private String description;  
private List<Course> courses;

public Student(String id, String name, String description,  
List<Course> courses) {  
super();  
this.id = id;  
this.name = name;  
this.description = description;  
this.courses = courses;  
}

public String getId() {  
return id;  
}

public void setId(String id) {  
this.id = id;  
}

public String getName() {  
return name;  
}

public void setName(String name) {  
this.name = name;  
}

public String getDescription() {  
return description;  
}

public void setDescription(String description) {  
this.description = description;  
}

public List<Course> getCourses() {  
return courses;  
}

public void setCourses(List<Course> courses) {  
this.courses = courses;  
}

@Override  
public String toString() {  
return String.format(  
"Student [id=%s, name=%s, description=%s, courses=%s]", id,  
name, description, courses);  
}  
}  
src/main/java/com/in28minutes/springboot/service/StudentService.java  
package com.in28minutes.springboot.service;

import java.math.BigInteger;  
import java.security.SecureRandom;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.List;

import org.springframework.stereotype.Component;

import com.in28minutes.springboot.model.Course;  
import com.in28minutes.springboot.model.Student;

@Component  
public class StudentService {

private static List<Student> students = new ArrayList<>();

static {  
//Initialize Data  
Course course1 = new Course("Course1", "Spring", "10 Steps", Arrays  
.asList("Learn Maven", "Import Project", "First Example",  
"Second Example"));  
Course course2 = new Course("Course2", "Spring MVC", "10 Examples",  
Arrays.asList("Learn Maven", "Import Project", "First Example",  
"Second Example"));  
Course course3 = new Course("Course3", "Spring Boot", "6K Students",  
Arrays.asList("Learn Maven", "Learn Spring",  
"Learn Spring MVC", "First Example", "Second Example"));  
Course course4 = new Course("Course4", "Maven",  
"Most popular maven course on internet!", Arrays.asList(  
"Pom.xml", "Build Life Cycle", "Parent POM",  
"Importing into Eclipse"));

Student ranga = new Student("Student1", "Ranga Karanam",  
"Hiker, Programmer and Architect", new ArrayList<>(Arrays  
.asList(course1, course2, course3, course4)));

Student satish = new Student("Student2", "Satish T",  
"Hiker, Programmer and Architect", new ArrayList<>(Arrays  
.asList(course1, course2, course3, course4)));

students.add(ranga);  
students.add(satish);  
}

public List<Student> retrieveAllStudents() {  
return students;  
}

public Student retrieveStudent(String studentId) {  
for (Student student : students) {  
if (student.getId().equals(studentId)) {  
return student;  
}  
}  
return null;  
}

public List<Course> retrieveCourses(String studentId) {  
Student student = retrieveStudent(studentId);

if (student == null) {  
return null;  
}

return student.getCourses();  
}

public Course retrieveCourse(String studentId, String courseId) {  
Student student = retrieveStudent(studentId);

if (student == null) {  
return null;  
}

for (Course course : student.getCourses()) {  
if (course.getId().equals(courseId)) {  
return course;  
}  
}

return null;  
}

private SecureRandom random = new SecureRandom();

public Course addCourse(String studentId, Course course) {  
Student student = retrieveStudent(studentId);

if (student == null) {  
return null;  
}

String randomId = new BigInteger(130, random).toString(32);  
course.setId(randomId);

student.getCourses().add(course);

return course;  
}  
}  
src/main/java/com/in28minutes/springboot/StudentServicesApplication.java  
package com.in28minutes.springboot;

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

@SpringBootApplication  
public class StudentServicesApplication {

public static void main(String[] args) {  
SpringApplication.run(StudentServicesApplication.class, args);  
}  
}