

IN 28 MINUTES



Spring Boot and Swagger - Documenting RESTful Services

This guide will help you use Swagger with Spring Boot to document your RESTful services. We will learn how to expose automated swagger documentation from your application. We will also add documentation to the REST API with swagger annotations.

You will learn

- What is the need for documenting your RESTful services?
- How do you document RESTful web services?
- Why Swagger?
- How can you use Swagger UI?
- How do you automate generation of Swagger Documentation from RESTful Web Services?
- How do you add custom information to Swagger Documentation generated from RESTful Web Services?
- What is Swagger-UI?

Recommended REST API Course



Deploy Spring Boot Microservices to AWS - ECS & AWS Fargate

NEW 89 lectures • 8 hours • All Levels

Learn Amazon Web Services (AWS) and AWS ECS deploying Docker based Spring Boot Microservices to AWS Fargate | By **in28Minutes** Official

★★★★★ 4.4
(6 ratings)



Deploy Java Spring Boot Apps to AWS with Elastic Beanstalk

BESTSELLER 66 lectures • 6.5 hours • Beginner

Take your first steps towards Amazon Web Services - AWS. Deploy Java Spring Boot REST APIs & Full Stack Apps to AWS. | By **in28Minutes** Official

★★★★★ 4.4
(110 ratings)



Go Full Stack with Spring Boot and React

127 lectures • 11.5 hours • Beginner

Build Your First Full Stack Application with React and Spring Boot. Become a Full Stack Web Developer Now! | By **in28Minutes** Official

★★★★★ 4.4
(346 ratings)



Go Full Stack with Spring Boot and Angular

118 lectures • 11 hours • All Levels

Build Your First Full Stack Application with Angular and Spring Boot. Become a Full Stack Web Developer Now! | By **in28Minutes** Official

★★★★★ 4.4
(1,570 ratings)



Master Microservices with Spring Boot and Spring Cloud

BESTSELLER 127 lectures • 11 hours • All Levels

An awesome journey from Restful Web Services to Microservices with Java, Spring Boot and Spring Cloud | By **in28Minutes** Official

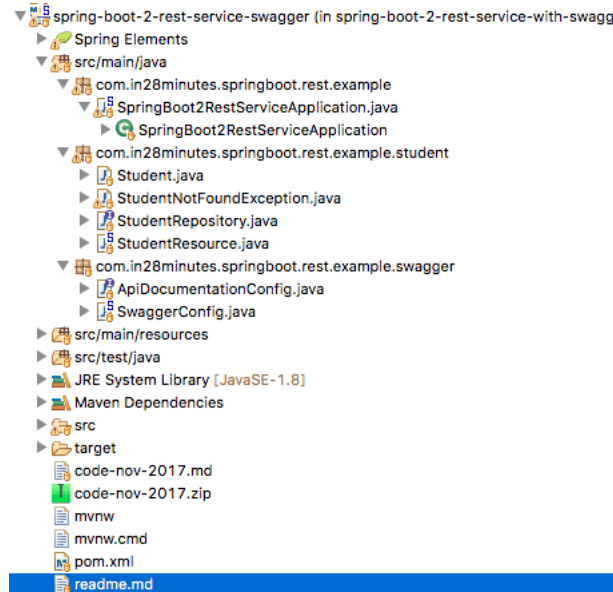
★★★★★ 4.4
(10,517 ratings)

10 Step Reference Courses

- [Spring Framework for Beginners in 10 Steps](#)
- [Spring Boot for Beginners in 10 Steps](#)
- [Spring MVC in 10 Steps](#)
- [JPA and Hibernate in 10 Steps](#)
- [Eclipse Tutorial for Beginners in 5 Steps](#)
- [Maven Tutorial for Beginners in 5 Steps](#)
- [JUnit Tutorial for Beginners in 5 Steps](#)
- [Mockito Tutorial for Beginners in 5 Steps](#)
- [Complete in28Minutes Course Guide](#)

Project Code Structure

Following screenshot shows the structure of the project we will create.



A few details:

- SpringBoot2RestServiceApplication.java – The Spring Boot Application class generated with Spring Initializer. This class acts as the launching point for application.
- pom.xml – Contains all the dependencies needed to build this project. We will use Spring Boot Starter AOP.
- Student.java – Student JPA Entity
- StudentRepository.java – Student JPA Repository. This is created using Spring Data JpaRepository.
- StudentResource.java – Spring Rest Controller exposing all services on the student resource.
- data.sql – Initial data for the student table. Spring Boot would execute this script after the tables are created from the entities.
- ApiDocumentationConfig.java – Meta Information about the API that will included in the documentation.
- SwaggerConfig.java – Contains the Swagger Configuration for generating documentation

Tools you will need

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

Complete Maven Project With Code Examples

Our Github repository has all the code examples – <https://github.com/in28minutes/spring-boot-examples/tree/master/spring-boot-2-rest-service-swagger>

Why do we need to document your RESTful API?

The most important design principle for RESTful Services is

Think about consumer of the API.

How does the consumer know

- What is format of the request?
- What content types your API supports?
- What is the structure of the response?
- Do you use HATEOAS?
- How to test your API?
- What kind of security mechanism you use?

REST does not specify a documentation standard or a contract like SOAP (WSDL). REST gives you the flexibility to choose your documentation format and approach. But that does not mean “No documentation”

■ *It's a misconception that REST means No documentation. You need to document your API.*

How do you document your RESTful API?

One option is to maintain documentation manually. But that gets outdated quickly.

Other option is to generate documentation from code. And that's the approach we would discuss in this guide.

There are multiple approaches to documenting your RESTful API

- WADL
- RESTDocs
- Swagger or OpenDocs

Swagger has picked up momentum in the last couple of years and is now the most popular REST API documentation standard. We will use Swagger in this guide.

Bootstrapping a Project with a REST Resource

In the previous article in the series – <http://www.springboottutorial.com/spring-boot-crud-rest-service-with-jpa-hibernate>, we set up a simple restful service with a resource exposing CRUD methods.

■ *We will use the same example to generate Swagger Documentation.*

Generating Swagger Documentation with Spring Boot

We would need to add a couple of dependencies related to Swagger and configure a Docket to generate Swagger Documentation. We will also use Swagger UI to have a visual representation of the Documentation and execute Test Requests.

Adding Swagger Dependencies

Let's add a couple of dependencies to our Swagger Project pom.xml.

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.4.0</version>
</dependency>

<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.4.0</version>
</dependency>
```

Adding Swagger Spring Configuration Docket

Let's now add the Spring configuration needed to generate Swagger Documentation.

/src/main/java/com/in28minutes/springboot/rest/example/swagger/SwaggerConfig.java

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    public static final Contact DEFAULT_CONTACT = new Contact(
        "Ranga Karanam", "http://www.in28minutes.com", "in28minutes@gmail.com");

    public static final ApiInfo DEFAULT_API_INFO = new ApiInfo(
        "Awesome API Title", "Awesome API Description", "1.0",
        "urn:tos", DEFAULT_CONTACT,
        "Apache 2.0", "http://www.apache.org/licenses/LICENSE-2.0");

    private static final Set<String> DEFAULT_PRODUCES_AND_CONSUMES =
        new HashSet<String>(Arrays.asList("application/json",
            "application/xml"));

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(DEFAULT_API_INFO)
            .produces(DEFAULT_PRODUCES_AND_CONSUMES)
            .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
    }
}
```

Notes

- @Configuration – This file contains Spring configuration.

- `@EnableSwagger2` - Annotation to Enable Swagger Documentation on the API
- `public static final Contact DEFAULT_CONTACT` - Has the contact information of the API. This will be exposed as part of the Swagger Documentation.
- `public static final ApiInfo DEFAULT_API_INFO` - Meta information about the API - Description, Licensing etc. This will be exposed as part of the Swagger Documentation.
- `private static final Set<String> DEFAULT_PRODUCES_AND_CONSUMES` - What content types does your API support?
- `public Docket api() {` - Docket to decide what kind of APIs you would want to document. In this example, we are documenting all APIs. You can filter out APIs you do not want to document with Swagger.

Exposing meta API information using @SwaggerDefinition

You can also expose meta API information using `@SwaggerDefinition` as shown below. The information in the class is self explanatory.

```
@SwaggerDefinition(
    info = @Info(
        description = "Awesome Resources",
        version = "V12.0.12",
        title = "Awesome Resource API",
        contact = @Contact(
            name = "Ranga Karanam",
            email = "ranga@in28minutes.com",
            url = "http://www.in28minutes.com"
        ),
        license = @License(
            name = "Apache 2.0",
            url = "http://www.apache.org/licenses/LICENSE-2.0"
        )
    ),
    consumes = {"application/json", "application/xml"},
    produces = {"application/json", "application/xml"},
    schemes = {SwaggerDefinition.Scheme.HTTP, SwaggerDefinition.Scheme.HTTPS},
    externalDocs = @ExternalDocs(value = "Read This For Sure", url = "http://in28minutes.com")
)
public interface ApiDocumentationConfig {
}
```

Generated Swagger Documentation

When you restart the application, you are all set to view the documentation that is generated.

Go to URL `http://localhost:8080/v2/api-docs`

At the top of the documentation is the Meta Information of the API

```
{
  "swagger": "2.0",
  "info": {
    "description": "Awesome API Description",
    "version": "1.0",
    "title": "Awesome API Title",
    "termsOfService": "urn:tos",
    "contact": {
      "name": "Ranga Karanam",
      "url": "http://www.in28minutes.com",
      "email": "in28minutes@gmail.com"
    },
    "license": {
      "name": "Apache 2.0",
      "url": "http://www.apache.org/licenses/LICENSE-2.0"
    }
  },
  "host": "localhost:8080",
  "basePath": "/",
  "tags": [
    {
      "name": "web-mvc-endpoint-handler-mapping",
      "description": "Web Mvc Endpoint Handler Mapping"
    },
    {
      "name": "student-resource",
      "description": "Student Resource"
    },
    {
      "name": "operation-handler",
      "description": "Operation Handler"
    },
    {
      "name": "basic-error-controller",
      "description": "Basic Error Controller"
    }
  ],
  "consumes": [
    "application/xml",
    "application/json"
  ],
  "produces": [
    "application/xml",
    "application/json"
  ]
}
```

```
    "application/json"
  ],

```

The paths contain the details of the resources being exposed

- You can see the different request methods, a summary of each method and all details about each request and response

```
"paths": {
  "/students": {
    "get": {
      "tags": [
        "student-resource"
      ],
      "summary": "retrieveAllStudents",
      "operationId": "retrieveAllStudentsUsingGET",
      "consumes": [
        "application/xml",
        "application/json"
      ],
      "produces": [
        "application/xml",
        "application/json"
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "array",
            "items": {
              "$ref": "#/definitions/Student"
            }
          }
        },
        "401": {
          "description": "Unauthorized"
        },
        "403": {
          "description": "Forbidden"
        },
        "404": {
          "description": "Not Found"
        }
      }
    },
    "post": {
      "tags": [
        "student-resource"
      ],
      "summary": "createStudent",
      "operationId": "createStudentUsingPOST",
      "consumes": [
        "application/xml",
        "application/json"
      ],
      "produces": [
        "application/xml",
        "application/json"
      ],
      "parameters": [
        {
          "in": "body",
          "name": "student",
          "description": "student",
          "required": true,
          "schema": {
            "$ref": "#/definitions/Student"
          }
        }
      ],
      "responses": {
        "200": {
          "description": "OK",
          "schema": {
            "type": "object"
          }
        },
        "201": {
          "description": "Created"
        },
        "401": {
          "description": "Unauthorized"
        },
        "403": {
          "description": "Forbidden"
        },
        "404": {
          "description": "Not Found"
        }
      }
    }
  },
  "/students/{id}": {
    "get": {
      "tags": [
        "student-resource"
      ],

```

```

"summary": "Find student by id",
"description": "Also returns a link to retrieve all students with rel - all-students",
"operationId": "retrieveStudentUsingGET",
"consumes": [
  "application/xml",
  "application/json"
],
"produces": [
  "application/xml",
  "application/json"
],
"parameters": [
  {
    "name": "id",
    "in": "path",
    "description": "id",
    "required": true,
    "type": "integer",
    "format": "int64"
  }
],
"responses": {
  "200": {
    "description": "OK",
    "schema": {
      "$ref": "#/definitions/Resource«Student»"
    }
  },
  "401": {
    "description": "Unauthorized"
  },
  "403": {
    "description": "Forbidden"
  },
  "404": {
    "description": "Not Found"
  }
}
},
"put": {
  "tags": [
    "student-resource"
  ],
  "summary": "updateStudent",
  "operationId": "updateStudentUsingPUT",
  "consumes": [
    "application/xml",
    "application/json"
  ],
  "produces": [
    "application/xml",
    "application/json"
  ],
  "parameters": [
    {
      "in": "body",
      "name": "student",
      "description": "student",
      "required": true,
      "schema": {
        "$ref": "#/definitions/Student"
      }
    },
    {
      "name": "id",
      "in": "path",
      "description": "id",
      "required": true,
      "type": "integer",
      "format": "int64"
    }
  ],
  "responses": {
    "200": {
      "description": "OK",
      "schema": {
        "type": "object"
      }
    },
    "201": {
      "description": "Created"
    },
    "401": {
      "description": "Unauthorized"
    },
    "403": {
      "description": "Forbidden"
    },
    "404": {
      "description": "Not Found"
    }
  }
},
"delete": {
  "tags": [
    "student-resource"
  ],
  "summary": "deleteStudent",

```

```

"operationId": "deleteStudentUsingDELETE",
"consumes": [
  "application/xml",
  "application/json"
],
"produces": [
  "application/xml",
  "application/json"
],
"parameters": [
  {
    "name": "id",
    "in": "path",
    "description": "id",
    "required": true,
    "type": "integer",
    "format": "int64"
  }
],
"responses": {
  "200": {
    "description": "OK"
  },
  "204": {
    "description": "No Content"
  },
  "401": {
    "description": "Unauthorized"
  },
  "403": {
    "description": "Forbidden"
  }
}
}
}
},

```

Definitions contain the detailed structure of the elements used in Request and Responses above.

```

"definitions": {
  "Resource«Student»": {
    "type": "object",
    "properties": {
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "links": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/Link"
        }
      },
      "name": {
        "type": "string",
        "description": "Name should have atleast 2 characters"
      },
      "passportNumber": {
        "type": "string"
      }
    }
  },
  "Map«string,Link»": {
    "type": "object",
    "additionalProperties": {
      "$ref": "#/definitions/Link"
    }
  },
  "Student": {
    "type": "object",
    "properties": {
      "id": {
        "type": "integer",
        "format": "int64"
      },
      "name": {
        "type": "string",
        "description": "Name should have atleast 2 characters"
      },
      "passportNumber": {
        "type": "string"
      }
    }
  },
  "description": "All details about the student. "
},
"Link": {
  "type": "object",
  "properties": {
    "href": {
      "type": "string"
    },
    "templated": {
      "type": "boolean"
    }
  }
}

```

```
}  
}  
}
```

Launching Swagger UI

You can also use the Swagger UI available at <http://localhost:8080/swagger-ui.html>.

Below screenshot shows the Home Page of Swagger UI. It shows a list of all the resources that are exposed.

swagger default (/v2/api-docs) api_key **Explore**

Awesome API Title

Awesome API Description

Created by Ranga Karanam
See more at <http://www.in28minutes.com>
[Contact the developer](#)
[Apache 2.0](#)

basic-error-controller : Basic Error Controller	Show/Hide	List Operations	Expand Operations
operation-handler : Operation Handler	Show/Hide	List Operations	Expand Operations
student-resource : Student Resource	Show/Hide	List Operations	Expand Operations
web-mvc-endpoint-handler-mapping : Web Mvc Endpoint Handler Mapping	Show/Hide	List Operations	Expand Operations

[BASE URL: / , API VERSION: 1.0]

Choosing the Student resource takes you to details of the resource. It shows all the request methods that can be used with a Resource.

student-resource : Student Resource Show/Hide List Operations Expand Operations

GET	/students	retrieveAllStudents
POST	/students	createStudent
DELETE	/students/{id}	deleteStudent
GET	/students/{id}	Find student by id
PUT	/students/{id}	updateStudent

You can also see the details for a Specific Request Method.

student-resource : Student Resource Show/Hide List Operations Expand Operations

GET /students retrieveAllStudents

Response Class (Status 200)
OK

Model | Model Schema

```
[  
  {  
    "id": 0,  
    "name": "string",  
    "passportNumber": "string"  
  }  
]
```

Response Content Type application/xml

Response Messages

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

You can use the 'Try it out' button to execute a request and see the response.

[Try it out!](#)
[Hide Response](#)

Curl

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/students'
```

Request URL

```
http://localhost:8080/students
```

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
[
  {
    "id": 10001,
    "name": "Ranga",
    "passportNumber": "E1234567"
  },
  {
    "id": 10002,
    "name": "Ravi",
    "passportNumber": "A1234568"
  }
]
```

Response Code

```
200
```

Response Headers

```
{
  "content-type": "application/json;charset=UTF-8",
  "date": "Fri, 08 Dec 2017 10:23:56 GMT",
  "transfer-encoding": "Identity"
}
```

Customizing Swagger Documentation with Annotations

You can add notes on the resource method to add more documentation

```
@GetMapping("/students/{id}")
@ApiOperation(value = "Find student by id",
  notes = "Also returns a link to retrieve all students with rel - all-students")
public Resource<Student> retrieveStudent(@PathVariable long id) {
```

Also supported is enhancing the documentation on the Request and Response Beans.

```
@Entity
@ApiModel(description="All details about the student. ")
public class Student {

  @ApiModelProperty(notes="Name should have atleast 2 characters")
  @Size(min=2, message="Name should have atleast 2 characters")
  private String name;
```

Congratulations! You are reading an article from a series of 50+ articles on Spring Boot and Microservices. We also have 20+ projects on our Github repository. For the complete series of 50+ articles and code examples, [click here](#).

Next Steps



Deploy Spring Boot Microservices to AWS - ECS & AWS Fargate

NEW 89 lectures • 8 hours • All Levels

Learn Amazon Web Services (AWS) and AWS ECS deploying Docker based Spring Boot Microservices to AWS Fargate | By **in28Minutes** Official

★★★★★ 4.4
(6 ratings)



Deploy Java Spring Boot Apps to AWS with Elastic Beanstalk

BESTSELLER 66 lectures • 6.5 hours • Beginner

Take your first steps towards Amazon Web Services - AWS. Deploy Java Spring Boot REST APIs & Full Stack Apps to AWS. | By **in28Minutes** Official

★★★★★ 4.4
(110 ratings)



Go Full Stack with Spring Boot and React

127 lectures • 11.5 hours • Beginner

Build Your First Full Stack Application with React and Spring Boot. Become a Full Stack Web Developer Now! | By **in28Minutes** Official

★★★★★ 4.4
(346 ratings)



Go Full Stack with Spring Boot and Angular

118 lectures • 11 hours • All Levels

Build Your First Full Stack Application with Angular and Spring Boot. Become a Full Stack Web Developer Now! | By **in28Minutes** Official

★★★★★ 4.4
(1,570 ratings)



Master Microservices with Spring Boot and Spring Cloud

BESTSELLER 127 lectures • 11 hours • All Levels

An awesome journey from Restful Web Services to Microservices with Java, Spring Boot and Spring Cloud | By **in28Minutes** Official

★★★★★ 4.4
(10,517 ratings)



Master Java Web Services and RESTful API with Spring Boot

BESTSELLER 105 lectures • 9.5 hours • All Levels

Build Amazing Java Web Services - RESTful & SOAP - using Spring & Spring Boot. Master REST APIs & SOAP Web Services Now! | By **in28Minutes** Official

★★★★★ 4.4
(3,582 ratings)



Spring Framework Interview Guide - 200+ Questions & Answers

73 lectures • 6 hours • All Levels

Get Ready for Your Spring Interview with Spring, Spring Boot, RESTful, SOAP Web Services and Spring MVC | By **in28Minutes** Official

★★★★★ 4.4
(784 ratings)

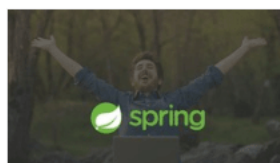


Learn Spring Boot in 100 Steps - Beginner to Expert

117 lectures • 13.5 hours • All Levels

Become an expert on Spring Boot developing a REST API and a Spring MVC Web application in 100 steps | By **in28Minutes** Official

★★★★★ 4.3
(1,175 ratings)



Spring Framework Master Class - Learn Spring the Modern Way!

BESTSELLER 134 lectures • 12 hours • All Levels

Learn the magic of Spring Framework in 100 Steps with Spring Boot, Spring JDBC, Spring AOP, Hibernate, JUnit & Mockito | By **in28Minutes** Official

★★★★★ 4.3
(11,910 ratings)

- Join 100,000 Learners and Become a Spring Boot Expert – [5 Awesome Courses on Microservices, API's, Web Services with Spring and Spring Boot. Start Learning Now](#)
- Learn Basics of Spring Boot – [Spring Boot vs Spring vs Spring MVC, Auto Configuration, Spring Boot Starter Projects, Spring Boot Starter Parent, Spring Boot Initializer](#)



Complete Code Example

/pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.in28minutes.springboot.rest.example</groupId>
  <artifactId>spring-boot-2-rest-service-swagger</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>spring-boot-2-rest-service</name>
  <description>Spring Boot 2 and REST - Example Project</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.0.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-data-jpa</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-hateoas</artifactId>
    </dependency>
    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger2</artifactId>
      <version>2.4.0</version>
    </dependency>
    <dependency>
      <groupId>io.springfox</groupId>
      <artifactId>springfox-swagger-ui</artifactId>
      <version>2.4.0</version>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>com.h2database</groupId>
      <artifactId>h2</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

```

    </dependency>
</dependencies>

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

<repositories>
  <repository>
    <id>spring-snapshots</id>
    <name>Spring Snapshots</name>
    <url>https://repo.spring.io/snapshot</url>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </repository>
  <repository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>https://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </repository>
</repositories>

<pluginRepositories>
  <pluginRepository>
    <id>spring-snapshots</id>
    <name>Spring Snapshots</name>
    <url>https://repo.spring.io/snapshot</url>
    <snapshots>
      <enabled>true</enabled>
    </snapshots>
  </pluginRepository>
  <pluginRepository>
    <id>spring-milestones</id>
    <name>Spring Milestones</name>
    <url>https://repo.spring.io/milestone</url>
    <snapshots>
      <enabled>false</enabled>
    </snapshots>
  </pluginRepository>
</pluginRepositories>

</project>

```

/src/main/java/com/in28minutes/springboot/rest/example/SpringBoot2RestServiceApplic

```

package com.in28minutes.springboot.rest.example;

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

@SpringBootApplication
public class SpringBoot2RestServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(SpringBoot2RestServiceApplication.class, args);
    }
}

```

/src/main/java/com/in28minutes/springboot/rest/example/student/Student.java

```

package com.in28minutes.springboot.rest.example.student;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.validation.constraints.Size;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

@Entity
@ApiModel(description="All details about the student. ")
public class Student {
    @Id
    @GeneratedValue
    private Long id;
}

```

```

@ApiModelProperty(notes="Name should have atleast 2 characters")
@Size(min=2, message="Name should have atleast 2 characters")
private String name;

private String passportNumber;

public Student() {
    super();
}

public Student(Long id, String name, String passportNumber) {
    super();
    this.id = id;
    this.name = name;
    this.passportNumber = passportNumber;
}

public Long getId() {
    return id;
}

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

public String getName() {
    return name;
}

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

public String getPassportNumber() {
    return passportNumber;
}

public void setPassportNumber(String passportNumber) {
    this.passportNumber = passportNumber;
}
}

```

/src/main/java/com/in28minutes/springboot/rest/example/student/StudentNotFoundException.java

```

package com.in28minutes.springboot.rest.example.student;

public class StudentNotFoundException extends RuntimeException {

    public StudentNotFoundException(String exception) {
        super(exception);
    }

}

```

/src/main/java/com/in28minutes/springboot/rest/example/student/StudentRepository.java

```

package com.in28minutes.springboot.rest.example.student;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface StudentRepository extends JpaRepository<Student, Long>{

}

```

/src/main/java/com/in28minutes/springboot/rest/example/student/StudentResource.java

```

package com.in28minutes.springboot.rest.example.student;

import static org.springframework.hateoas.mvc.ControllerLinkBuilder.linkTo;
import static org.springframework.hateoas.mvc.ControllerLinkBuilder.methodOn;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.hateoas.Resource;
import org.springframework.hateoas.mvc.ControllerLinkBuilder;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
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.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;

```

```

import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import io.swagger.annotations.ApiOperation;

@RestController
public class StudentResource {

    @Autowired
    private StudentRepository studentRepository;

    @GetMapping("/students")
    public List<Student> retrieveAllStudents() {
        return studentRepository.findAll();
    }

    @GetMapping("/students/{id}")
    @ApiOperation(value = "Find student by id",
        notes = "Also returns a link to retrieve all students with rel - all-students")
    public Resource<Student> retrieveStudent(@PathVariable long id) {
        Optional<Student> student = studentRepository.findById(id);

        if (!student.isPresent())
            throw new StudentNotFoundException("id-" + id);

        Resource<Student> resource = new Resource<Student>(student.get());

        ControllerLinkBuilder linkTo = linkTo(methodOn(this.getClass()).retrieveAllStudents());

        resource.add(linkTo.withRel("all-students"));

        return resource;
    }

    @DeleteMapping("/students/{id}")
    public void deleteStudent(@PathVariable long id) {
        studentRepository.deleteById(id);
    }

    @PostMapping("/students")
    public ResponseEntity<Object> createStudent(@RequestBody Student student) {
        Student savedStudent = studentRepository.save(student);

        URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}")
            .buildAndExpand(savedStudent.getId()).toUri();

        return ResponseEntity.created(location).build();
    }

    @PutMapping("/students/{id}")
    public ResponseEntity<Object> updateStudent(@RequestBody Student student, @PathVariable long id) {

        Optional<Student> studentOptional = studentRepository.findById(id);

        if (!studentOptional.isPresent())
            return ResponseEntity.notFound().build();

        student.setId(id);

        studentRepository.save(student);

        return ResponseEntity.noContent().build();
    }
}

```

/src/main/java/com/in28minutes/springboot/rest/example/swagger/ApiDocumentationCo

```

package com.in28minutes.springboot.rest.example.swagger;

import io.swagger.annotations.Contact;
import io.swagger.annotations.ExternalDocs;
import io.swagger.annotations.Info;
import io.swagger.annotations.License;
import io.swagger.annotations.SwaggerDefinition;

@SwaggerDefinition(
    info = @Info(
        description = "Awesome Resources",
        version = "V12.0.12",
        title = "Awesome Resource API",
        contact = @Contact(
            name = "Ranga Karanam",
            email = "ranga.karanam@in28minutes.com",
            url = "http://www.in28minutes.com"
        ),
        license = @License(
            name = "Apache 2.0",
            url = "http://www.apache.org/licenses/LICENSE-2.0"
        )
    ),
    consumes = {"application/json", "application/xml"},

```

```

        produces = {"application/json", "application/xml"},
        schemes = {SwaggerDefinition.Scheme.HTTP, SwaggerDefinition.Scheme.HTTPS},
        externalDocs = @ExternalDocs(value = "Read This For Sure", url = "http://in28minutes.com")
    )
    public interface ApiDocumentationConfig {
    }

```

/src/main/java/com/in28minutes/springboot/rest/example/swagger/SwaggerConfig.java

```

package com.in28minutes.springboot.rest.example.swagger;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableSwagger2;

import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    public static final Contact DEFAULT_CONTACT = new Contact(
        "Ranga Karanam", "http://www.in28minutes.com", "in28minutes@gmail.com");

    public static final ApiInfo DEFAULT_API_INFO = new ApiInfo(
        "Awesome API Title", "Awesome API Description", "1.0",
        "urn:tos", DEFAULT_CONTACT,
        "Apache 2.0", "http://www.apache.org/licenses/LICENSE-2.0");

    private static final Set<String> DEFAULT_PRODUCES_AND_CONSUMES =
        new HashSet<String>(Arrays.asList("application/json",
            "application/xml"));

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(DEFAULT_API_INFO)
            .produces(DEFAULT_PRODUCES_AND_CONSUMES)
            .consumes(DEFAULT_PRODUCES_AND_CONSUMES);
    }
}

```

/src/main/resources/application.properties

/src/main/resources/data.sql

```

insert into student
values(10001,'Ranga', 'E1234567');

insert into student
values(10002,'Ravi', 'A1234568');

```

/src/test/java/com/in28minutes/springboot/rest/example/SpringBoot2RestServiceApplica

```

package com.in28minutes.springboot.rest.example;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@RunWith(SpringRunner.class)
@SpringBootTest
public class SpringBoot2RestServiceApplicationTests {

    @Test
    public void contextLoads() {
    }
}

```



```
}  
}
```

Do you wish to contribute?

Reach us at (in 28 minutes) at (gmail) dot (com)

300,000+ Learners doing 24 Best Selling Courses

- ✓ Amazing Hands-on Step By Step Learning Experiences
- ✓ Real Project Experiences with Best Tools and Frameworks
- ✓ Awesome Troubleshooting Guides with 200+ FAQs
- ✓ Friendly Support in Questions and Answers section
- ✓ Free Udemy Certificate on Completion of Course
- ✓ 30 Day "No Questions Asked" Money Back Guarantee!

50,000+ Amazing Reviews on in28Minutes Courses

- ★★★★★ Excellent, fabulous. The way he has prepared the material and the way he teaches is really awesome. What an effort .. Thanks a million
- ★★★★★ A lot of preparation work has taken place from the teacher and this is visible throughout the course.
- ★★★★★ This guy is fantastic. Really. Wonderful teaching skills, and goes well out of his way to make sure that everything he is doing is fully understood.
- ★★★★★ He explains the concepts really well and also makes sure that there is not a single line of code you type without understanding what it really does.
- ★★★★★ Amazing course. Explained super difficult concepts (that I have spent hours on the internet finding a good explanation) in under 5 minutes.

Try a Free Course Now!




















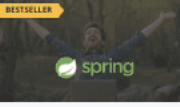
[Learn Spring Boot in 10 Steps](#)

Get 90% OFF on our Premium Courses

[Learn AWS, Spring Boot, Full stack \(React/Angular\), REST API, Microservices and more..](#)

Total students	Courses	Reviews
302,083	24	54,102

Courses you're teaching

 <p>Deploy Java Spring Boot Apps to AWS with Elastic... in28Minutes Official</p> <p>★★★★☆ 4.3 (18)</p> <p>₹12,800 ₹640</p>	 <p>Go Full Stack with Spring Boot and React in28Minutes Official</p> <p>★★★★☆ 4.5 (239)</p> <p>₹12,800 ₹1,920</p>	 <p>Go Full Stack with Spring Boot and Angular 7 in28Minutes Official</p> <p>★★★★☆ 4.5 (1,389)</p> <p>₹12,800 ₹1,920</p>	 <p>Your First Steps from Programmer to Softwar... in28Minutes Official</p> <p>★★★★☆ 4.4 (92)</p> <p>₹12,800 ₹780</p>
 <p>Python for Beginners - Go from Java to Python in 1... in28Minutes Official</p> <p>★★★★☆ 4.5 (185)</p> <p>₹12,800 ₹1,920</p>	 <p>Python Programming for Beginners - Learn in 100... in28Minutes Official</p> <p>★★★★☆ 4.3 (1,713)</p> <p>₹12,800 ₹1,920</p>	 <p>Master Automation Testing with Java and... in28Minutes Official</p> <p>★★★★☆ 4.3 (459)</p> <p>₹12,800 ₹1,280</p>	 <p>Master Java Unit Testing with Spring Boot &... in28Minutes Official</p> <p>★★★★☆ 4.4 (849)</p> <p>₹12,800 ₹1,920</p>
 <p>Spring Framework Interview Guide - 200+... in28Minutes Official</p> <p>★★★★☆ 4.3 (712)</p> <p>₹12,800 ₹1,920</p>	 <p>Master Java Web Services and RESTful API with... in28Minutes Official</p> <p>★★★★☆ 4.4 (3,405)</p> <p>₹12,800 ₹1,920</p>	 <p>Learn Spring Boot In 100 Steps - Beginner to Expert in28Minutes Official</p> <p>★★★★☆ 4.2 (1,715)</p> <p>₹12,800 ₹1,920</p>	 <p>Java Interview Guide : 200+ Interview Question... in28Minutes Official</p> <p>★★★★☆ 4.3 (2,078)</p> <p>₹12,800 ₹640</p>
 <p>Learn Unit Testing with Junit & Mockito in 30... in28Minutes Official</p> <p>★★★★☆ 4.2 (1,576)</p> <p>₹12,800 ₹640</p>	 <p>Java EE Made Easy - Patterns, Architecture a... in28Minutes Official</p> <p>★★★★☆ 4.3 (497)</p> <p>₹12,800 ₹640</p>	 <p>Spring MVC For Beginners : Build Java Web App in 2... in28Minutes Official</p> <p>★★★★☆ 4.4 (2,164)</p> <p>₹12,800 ₹640</p>	 <p>Eclipse Tutorial For Beginners : Learn Java I... in28Minutes Official</p> <p>★★★★☆ 4.2 (2,355)</p> <p>Free</p>
 <p>Java Servlets and JSP - Build Java EE(JEE) app in... in28Minutes Official</p> <p>★★★★☆ 4.4 (2,121)</p> <p>Free</p>	 <p>Maven Tutorial - Manage Java Dependencies in 20... in28Minutes Official</p> <p>★★★★☆ 4.1 (422)</p> <p>₹12,800 ₹570</p>	 <p>Learn Java Unit Testing with JUnit 5 in 20 Steps in28Minutes Official</p> <p>★★★★☆ 4.2 (5,789)</p> <p>₹12,800 ₹640</p>	 <p>Spring Framework Master Class - Learn Spring the... in28Minutes Official</p> <p>★★★★☆ 4.3 (11,641)</p> <p>₹12,800 ₹640</p>