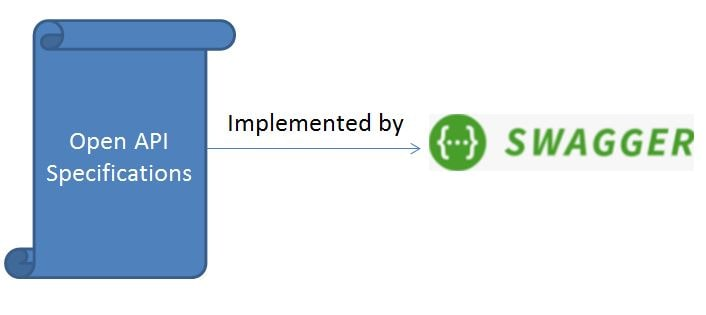
**Swagger**

Swagger is an open source project used to generate the REST API documents for RESTful web services. It provides a user interface to access our RESTful web services via the web browser.

### What is Swagger and Open API Specification

**OpenAPI Specification (formerly Swagger Specification)** is an API description format for REST APIs. An OpenAPI file allows you to describe your entire API.  
**Swagger** is a set of open-source tools built around the OpenAPI Specification that can help you design, build, document and consume REST APIs.



What is API?

[API stands for **Application Programming Interface.**](https://www.javatpoint.com/api-full-form)

It defines how two pieces of software talk to each other. There are several types of APIs, but the swagger specifically deals with the Web API.

Let's understand the working the Web API through an example. Suppose we opened the **Facebook** on our phone and made a request to the **Facebook** server. The request sent to the **Facebook** server is known as an **API request** and the **Facebook** server will send the response known as **API response**.

**Here, API definition works:**

* What requests are available
* What the response looks like for each request.

Swagger and Open API specification are mainly designed for the Rest API, where Rest is a type of web API. In Rest word, R stands for Representational, S stands for State, and T stands for Transfer.

**Here, API definition works:**

* What requests are available
* What the response looks like for each request.

Swagger and Open API specification are mainly designed for the Rest API, where Rest is a type of web API. In Rest word, R stands for Representational, S stands for State, and T stands for Transfer.

The API Definition is a file that describes all the things that we can do with an API. It contains all the requests that we can make to an API. It also describes what request to make and how would response look like for each request.

**Why create an API definition?**

There are several advantages of writing an API definition:

* It allows you to design the API before implementing it. The developers can review the API before writing the code for the API.
* It also helps in automated testing.
* It can automatically create a code in several languages.
* It can also be used to generate the documentation automatically.

**Anatomy of a Request**

There are five different parts to be found in the Http request:

1. Method: The method describes the action to be performed. The methods could be POST, PUT, DELETE, GET.
2. URL: It specifies the name on which the action is to be performed.
3. Query parameters
4. Headers: Headers are used to store the information about the request.
5. Body: Body contains the additional data.

**URL is broken down into several pieces:**

For example: the request URL is: https://api.example.com/v2/user

* Scheme: https
* Host: api.example.com
* Base path: /v2
* Path: user

**XML:**

**<name>**

**<firstname>** John **</firstname>**

**<lastname>** Malik **</lastname>**

**</name>**

JSON:

name: {

   "firstname": "John"

   "lastname": "Malik"

}

YAML

name:

  firstname: John

  lastname: Malik

Types

The types in YAML are determined from the context.

**For example:**

1. part\_no: A4786
2. description: Photoresistor
3. price: 1.47
4. quantity: 4

In the above scenario, **part\_no** will be treated as a string, **description** will also be treated as a string, **price** will be treated as a floating type, and **quantity** will be treated as an integer.

Swagger provides an editor for the Open API Specification files. To visit the swagger editor website, go to the following link:

<http://editor2.swagger.io>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version>

</dependency>

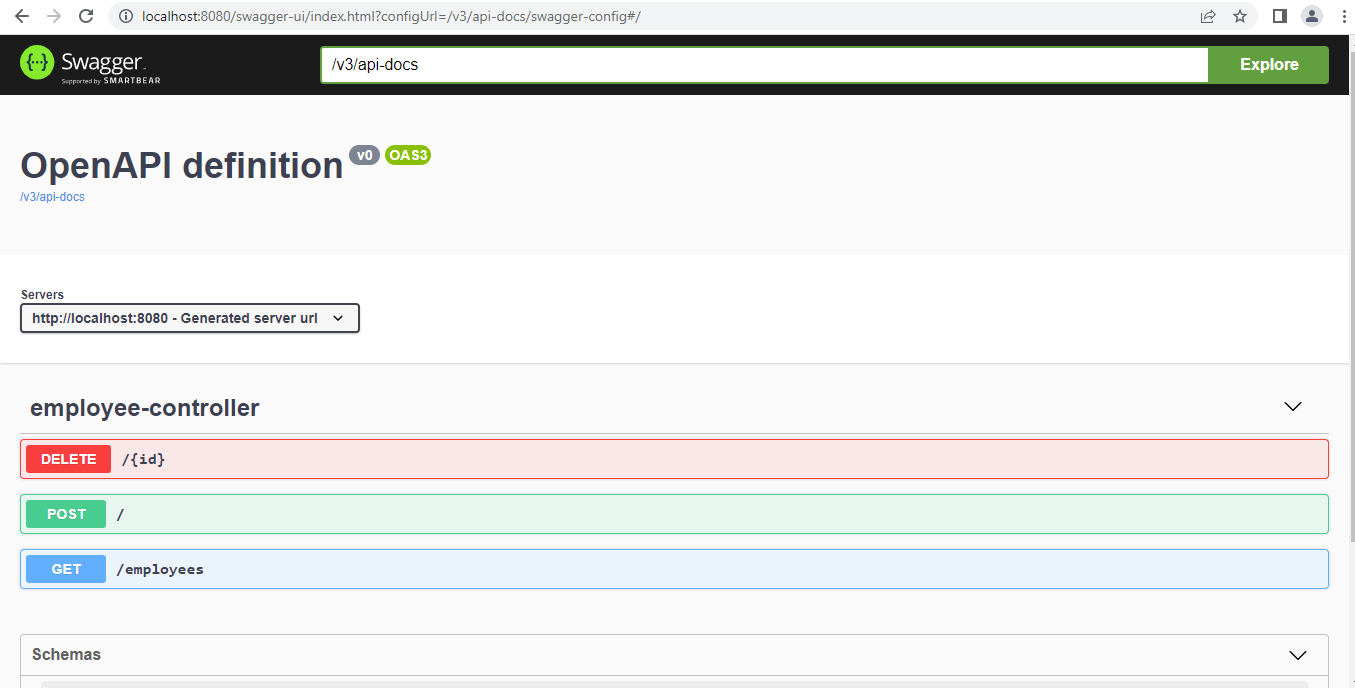
<dependency>

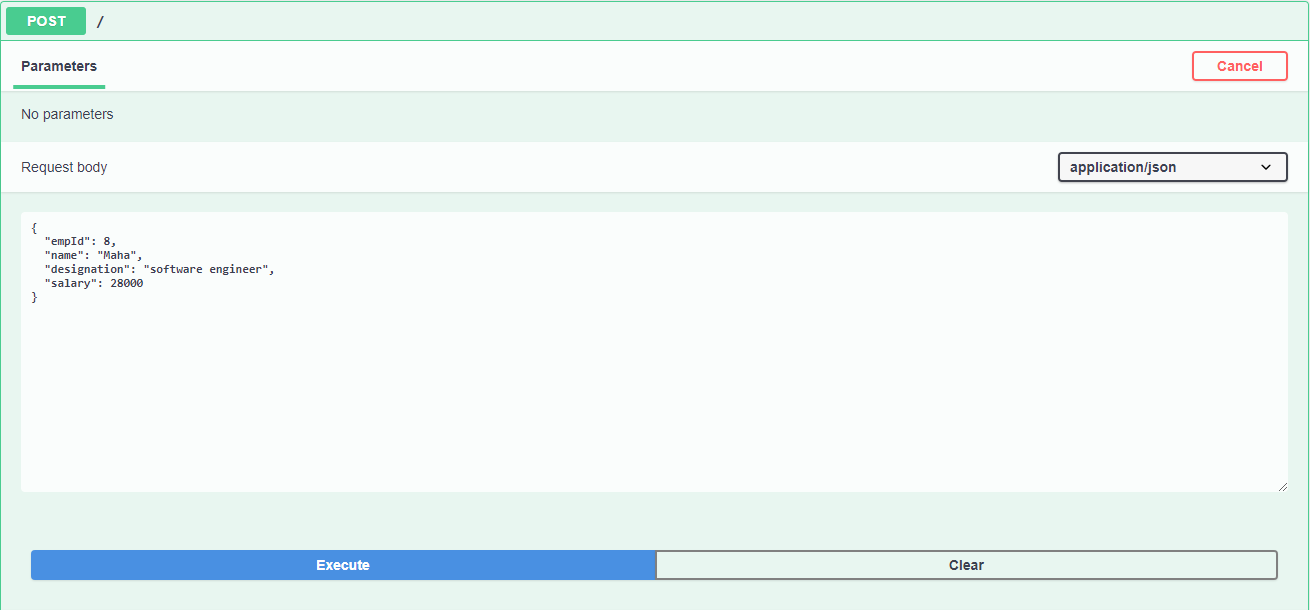
<groupId>io.springfox</groupId>

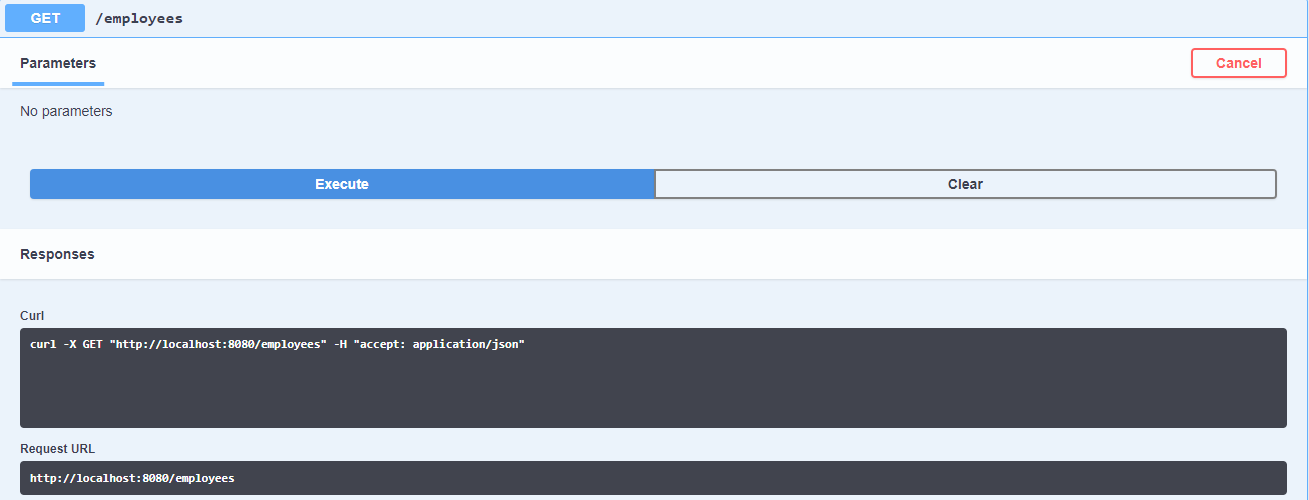
<artifactId>springfox-swagger-ui</artifactId>

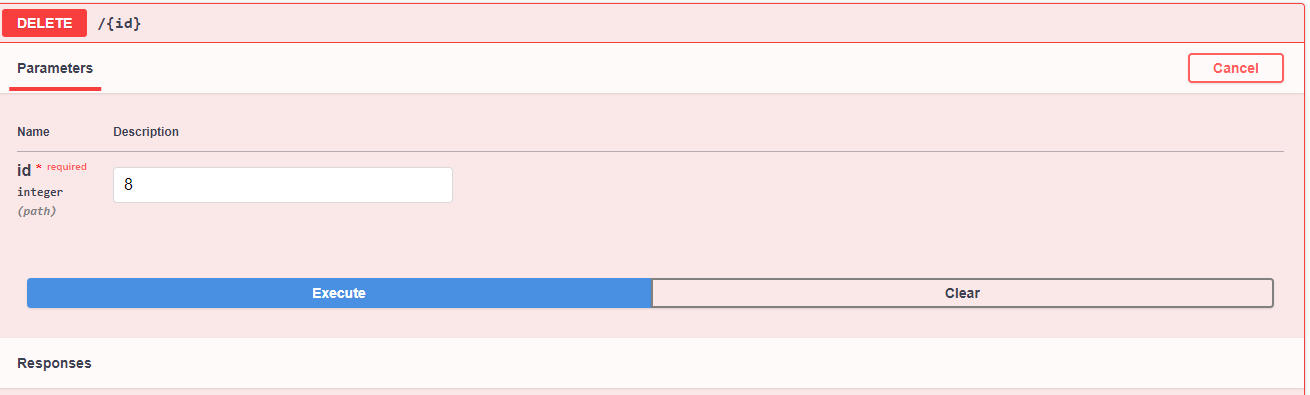
<version>2.7.0</version>

</dependency>









**Advantages:**

* Synchronizes the API documentation with the server and client at the same pace.
* Allows us to generate REST API documentation and interact with the REST API. The interaction with the REST API using the Swagger UI Framework gives clear insight into how the API responds to parameters.
* Provides responses in the format of JSON and XML.
* Implementations are available for various technologies, such as Scala, Java, and HTML5.

**Disadvantages:**

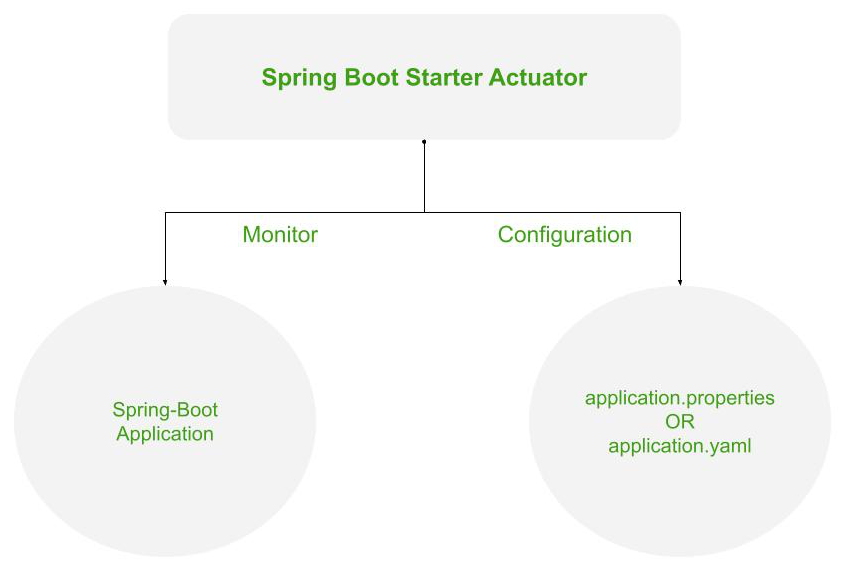
* Requires multiple specifications for some tools, including dev and QA.
* Doesn't allow for code reuse, includes, or extensions.
* Lacks strong developer tools.
* Requires schemas for all responses.

## **Spring Boot Actuator**

Spring Boot Actuator provides secured endpoints for monitoring and managing your Spring Boot application. By default, all actuator endpoints are secured

**Spring Boot Actuator** is a sub-project of the Spring Boot Framework..

It contains the actuator endpoints (the place where the resources live). We can use **HTTP** and **JMX** endpoints to manage and monitor the Spring Boot application.



### Spring Boot Actuator Features

There are **three** main features of Spring Boot Actuator:

* **Endpoints**
* **Metrics**
* **Audit**

**Endpoint:** The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints. **Endpoint:** The actuator endpoints allows us to monitor and interact with the application. Spring Boot provides a number of built-in endpoints.

**Metrics:** Spring Boot Actuator provides dimensional metrics by integrating with the**micrometer**. It provides vendor-neutral interfaces for **timers, gauges, counters, distribution summaries,** and **long task timers** with a dimensional data model.

**Audit:** Spring Boot provides a flexible audit framework that publishes events to an **AuditEventRepository.** It automatically publishes the authentication events if spring-security is in execution.

## **Enabling Spring Boot Actuator**

We can enable actuator by injecting the dependency **spring-boot-starter-actuator** in the pom.xml file.

**<dependency>**

**<groupId>**org.springframework.boot**</groupId>**

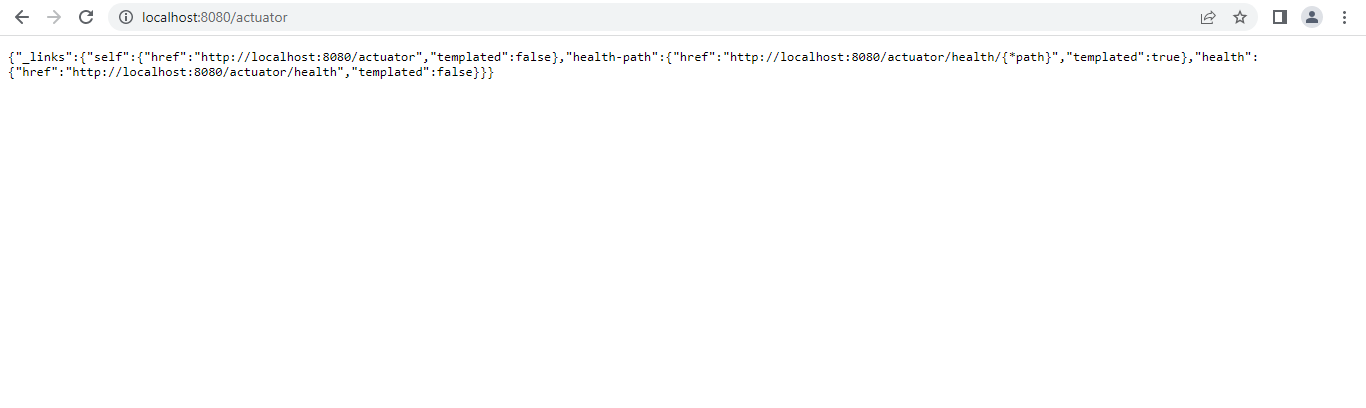
**<artifactId>**spring-boot-starter-actuator**</artifactId>**

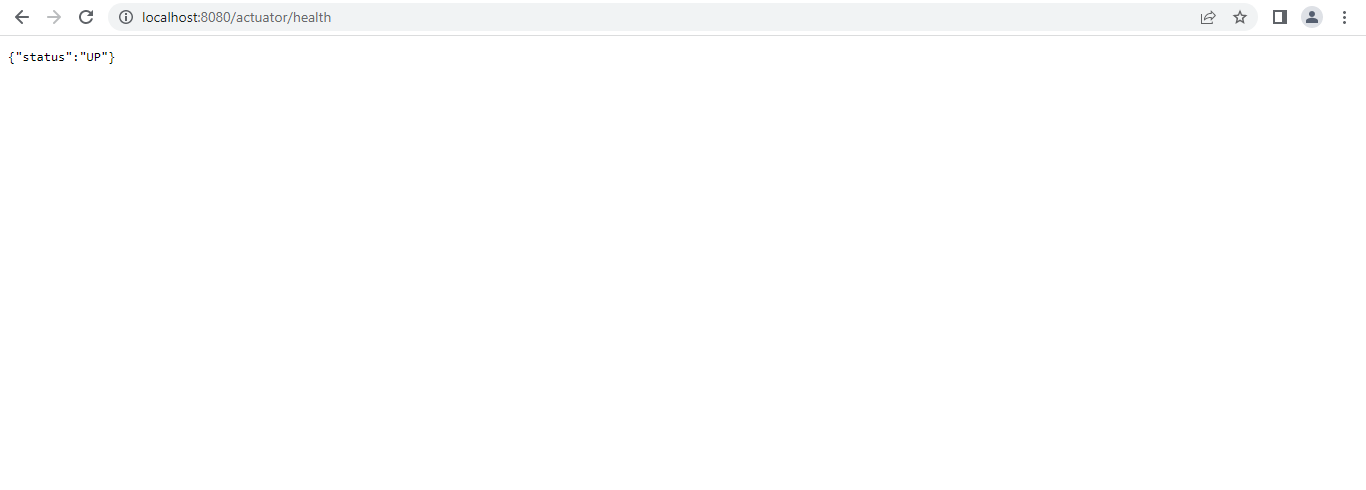
**<version>**2.2.2.RELEASE**</version>**

**</dependency>**

Open the **application. Properties** file and disable the security feature of the actuator by adding the following statement.

Let's invoke the **health** endpoint by invoking the URL http://localhost:8080/actuator/health. It denotes the status **UP**. It means the application is healthy and running without any interruption.





**Advantages:**

The main benefit of using this library is that we **get health and monitoring metrics from production-ready applications**.

* Provide Production Ready Feature. We have one module which provides us all the production-ready features. ...
* Very accurate control and positioning.
* Able to stop at any point of the stroke.
* Easy to set acceleration and deceleration.
* No external sensors.
* Low operating costs.

**DisAdvantages:**

* Non-explosion proof.
* Sensitive to vibration.
* More complex technology.

**DevTools:**

Spring Boot 1.3 provides another module called Spring Boot DevTools. DevTools stands for **Developer Tool.** The aim of the module is to try and improve the development time while working with the Spring Boot application. Spring Boot DevTools pick up the changes and restart the application.

1. **<dependency>**
2. **<groupId>**org.springframework.boot**</groupId>**
3. **<artifactId>**spring-boot-devtools**</artifactId>**
4. **<scope>**runtime**<scope** **>**
5. **</dependency>**

## **Spring Boot DevTools Features**

Spring Boot DevTools provides the following features:

* **Property Defaults**
* **Automatic Restart**
* **LiveReload**
* **Remote Debug Tunneling**
* **Remote Update and Restart**

**Property Defaults:** Spring Boot provides templating technology **Thymeleaf** that contains the property **spring.thymeleaf.cache.** It disables the caching and allows us to update pages without the need of restarting the application.

**Automatic Restart:** Auto-restart means reloading of Java classes and configure it at the server-side. After the server-side changes, it deployed dynamically, server restarts happen, and load the modified code.

* The classes that do not change (third-Jars) are loaded in the **base ClassLoader.**
* The classes that we are actively developing are loaded in the **restart ClassLoader.**

**LiveReload:** The Spring Boot DevTools module includes an embedded server called **LiveReload.** It allows the application to automatically trigger a browser refresh whenever we make changes in the resources. It is also known as **auto-refresh.**

We can also disable auto-reload in browser by excluding the above paths. For example:

1. spring.devtools.restart.exclude=public/\*\*, static/\*\*, templates/\*\*

We can see the other additional path by using the property **spring.devtools.restart.additional-paths.** For example:

1. spring.devtools.restart.additional-paths=/path-to-folder

If we want to exclude additional path and want to keep defaults then use the property **spring.devtools.restart.additional-exclude.** For example:

1. spring.devtools.restart.additional-exclude=styles/\*\*

## **Using a Trigger File**

Automatic restart sometimes can slow down development time due to frequent restarts. To remove this problem, we can use a **trigger file.**

spring.devtools.restart.trigger-file=c:/workspace-sts-3.9.9.RELEASE/restart-trigger.txt

Open the **SpringBootDevtoolsExampleApplication.java** and run it as Java Application.

After that, make any changes (edit or remove some file or code) in the application, and save that changes. As soon as we save the changes, the server restarts and pick up the changes.

**Advantages:**

* Easy Access on Desktop or Mobile. The second tab in the left corner is used to check your application view on mobile or desktop. ...
* Element Tab. ...
* Console. ...
* Network. ...
* Performance. ...

**Disadvantages:**

* No so easy documentation.
* Not very easy to use for a person who is not a developer.
* Reports can be improved.