



Spring Boot Actuator



pawardarshan461

Read

Discuss

Practice

The Spring Framework is the most used platform which was released in October 2002 for building effective and efficient web-based applications. On top of it, the Spring Boot framework was released in April 2014 to overcome the cumbersome effort of manual configuration. The main moto of Spring Boot was to achieve the Auto-Configuration feature. With the help of this and other features, we are able to create a stand-alone Spring web application. Developing and Managing an application are the two most important aspects of the application's life cycle. It is very crucial to know what's going on beneath the application. Also when we push the application on production, managing it gradually becomes critically important. Therefore, it is always recommended to monitor the application both while at the development phase and at the production phase.

Advantages of Monitoring/Managing the Application

1. It increases customer satisfaction.
2. It reduces downtime.
3. It boosts productivity.
4. It improves Cybersecurity Management.
5. It increases the conversion rate.

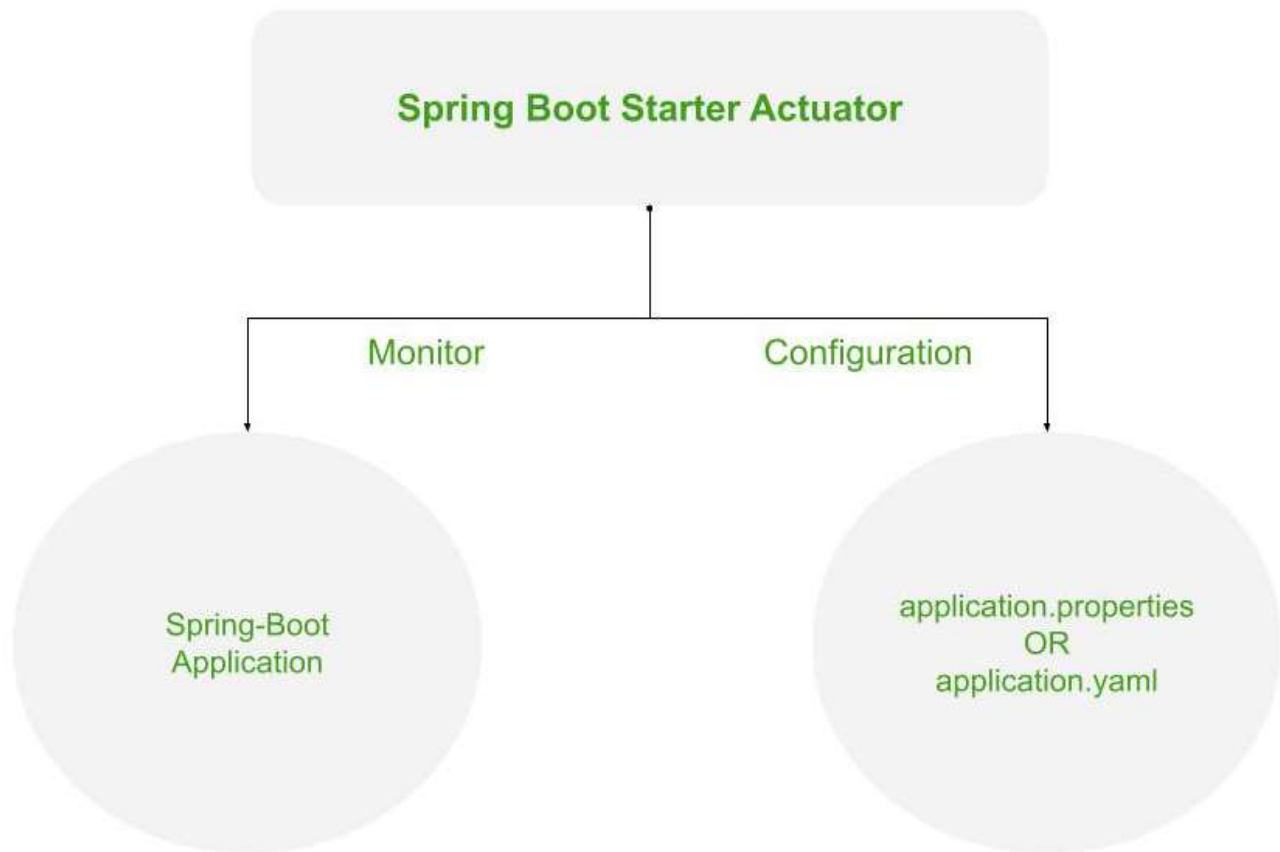
Spring Boot – Actuator

- With the help of Spring Boot, we can achieve the above objectives.
- Spring Boot's 'Actuator' dependency is used to monitor and manage the Spring web application.
- We can use it to monitor and manage the application with the help of HTTP endpoints or with the JMX.



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#).

Got It !



Working of the Spring's Actuator

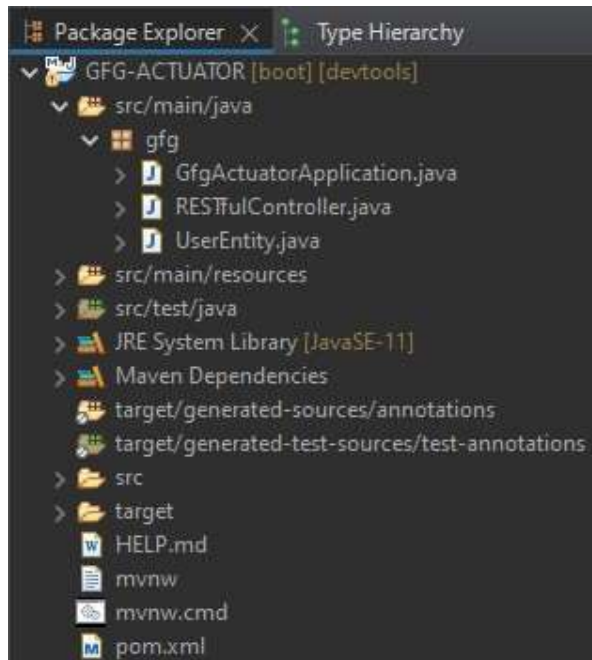
To use the 'Actuator' add the following dependency in your application's project build.

Maven -> pom.xml

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

Gradle -> build.gradle

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-actuator'
}
```



Project Structure – Maven

pom.xml (Configuration of the Web Application)

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
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.6.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>sia</groupId>
  <artifactId>GFG-ACTUATOR</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>GFG-ACTUATOR</name>
  <description>Spring Boot Starter Actuator</description>
  <properties>
    <java.version>11</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>
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

```

    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</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>
            <configuration>
                <excludes>
                    <exclude>
                        <groupId>org.projectlombok</groupId>
                        <artifactId>lombok</artifactId>
                    </exclude>
                </excludes>
            </configuration>
        </plugin>
    </plugins>
</build>
</project>

```

GfgActuatorApplication.java (Bootstrapping of the application)

```
package gfg;

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

@SpringBootApplication
public class GfgActuatorApplication {

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

UserEntity.java (Entity class representing the model data)

- This class acts as a simple java bean whose properties are returned as JSON response by the REST API's get() method.
- 'Lombok' library is used to generate GETTER/SETTER methods automatically at runtime using '@Data' annotation.
- '@RequiredArgsConstructor' annotation is used to generate a zero-argument constructor and if final or '@NonNull' fields are present, then respective arguments constructor is created.
- To add the 'Lombok' library in your application, add the following dependency in your application's project build.

Maven -> pom.xml

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

'@Component' annotation is used so that this bean automatically gets registered in Spring's application context.

Java

```
package gfg;

import lombok.Data;
import lombok.RequiredArgsConstructor;
```

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

```
@Component
@Data
@RequiredArgsConstructor
public class UserEntity {
    String id = "1";
    String name = "Darshan.G.Pawar";
    String userName = "@drash";
    String email = "drash@geek";
    String pincode = "422-009";
}
```

RESTfulController.java (A REST API controller)

This controller's get() method uses the UserEntity bean to return JSON response.

UserEntiy bean is outsourced through '@Autowired' annotation which was registered in Spring's application context.

Java

```
package gfg;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/get")
public class RESTfulController {

    @Autowired
    UserEntity entity;

    @GetMapping("/data") public UserEntity getEntity()
    {
        return entity;
    }
}
```

Output:

Here, the JSON Formatter Chrome extension is used to automatically parse the JSON body. Further, it will be required to work with 'Actuator'.



JSON response returned by REST API

Working with Spring Boot Actuator

To access the 'Actuator' services, you will have to use the HTTP endpoint as it becomes reliable to work with. The default endpoint is '/actuator'.

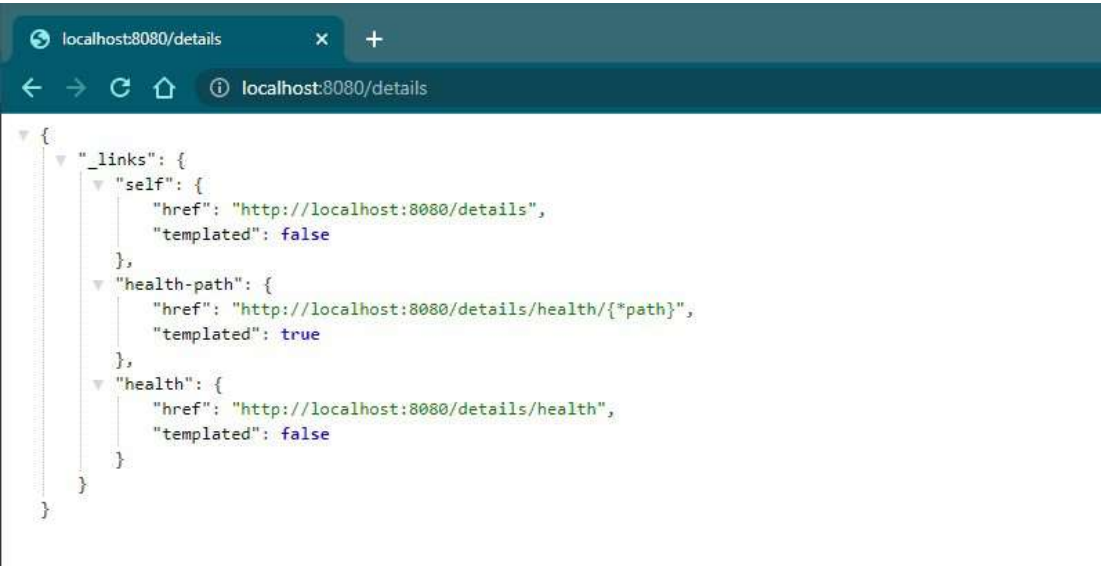
Example:



We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

You can also change the default endpoint by adding the following in the application.properties file.

```
management.endpoints.web.base-path=/details
```



You can click on these above links and see the respective information. Additionally, you can activate other Actuator IDs and use them after ‘/actuator’ to see more information. For example, ‘health’ ID is activated by default. Therefore you can click the link in the image or directly use ‘http://localhost:8080/actuator/health’.



The health of an application

‘UP’ means the application’s health is good. There are a total of 25 IDs out of which the commonly used are listed out here –

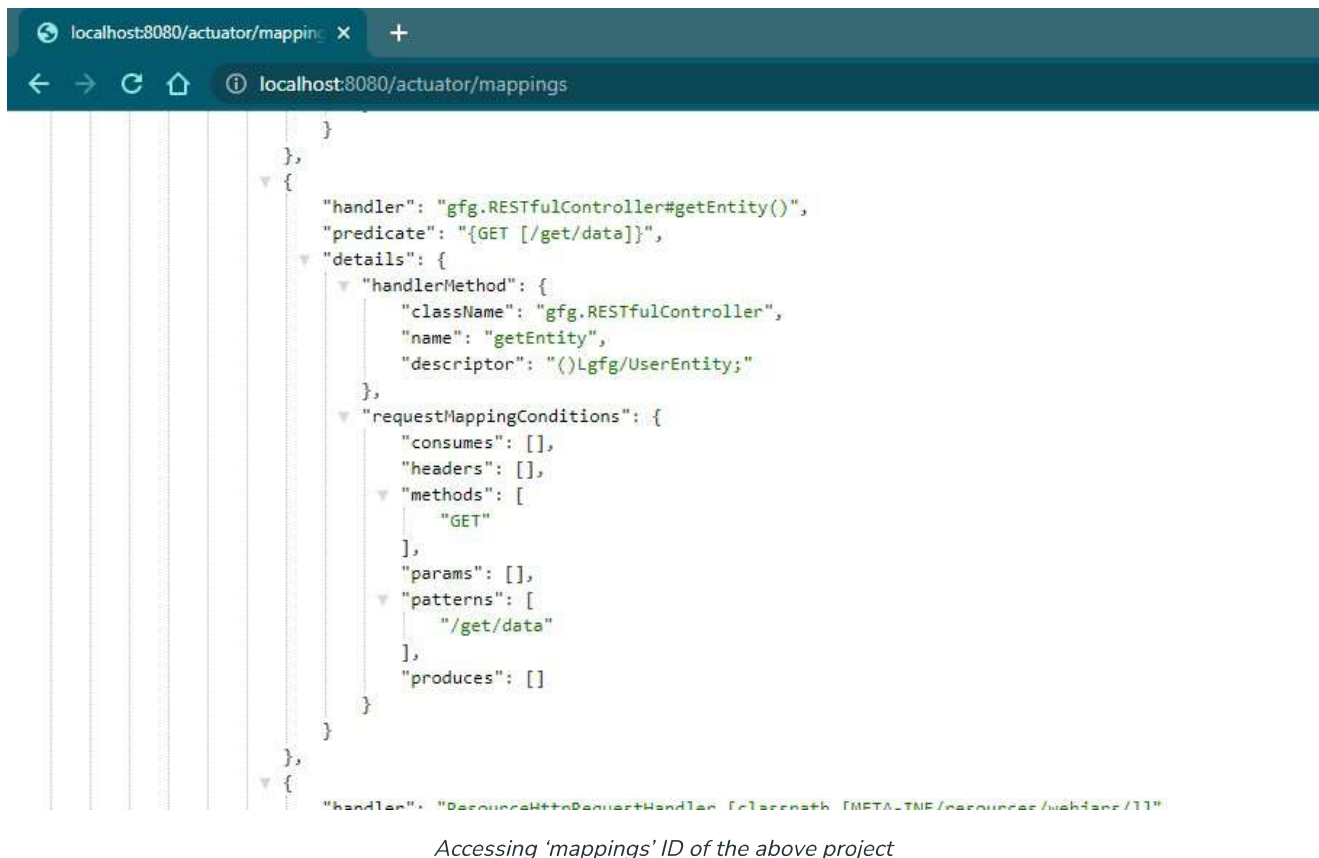
| ID | Description |
|----|-------------|
|----|-------------|

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

| | |
|---------------------|---|
| ID caches | Description Exposes available caches. |
| conditions | Shows the conditions that were evaluated on configuration and auto-configuration classes and the reasons why they did or did not match. |
| health | Shows application health information. |
| httptrace | Displays HTTP trace information (by default, the last 100 HTTP request-response exchanges). Requires an <code>HttpTraceRepository</code> bean. |
| loggers | Shows and modifies the configuration of loggers in the application. |
| mappings | Displays a collated list of all <code>@RequestMapping</code> paths. |
| sessions | Allows retrieval and deletion of user sessions from a Spring Session-backed session store. Requires a servlet-based web application that uses Spring Session. |
| threaddump | Performs a thread dump. |



Accessing 'beans' ID of the above project



Including IDs/Endpoints

By default, all IDs are set to false except for 'health'. To include an ID, use the following property in the application.properties file.

```
management.endpoint.<id>.enabled
```

Example -> `management.endpoint.metrics.enabled=true`

OR, you can just list down all IDs that you want to include which are separated by a comma.

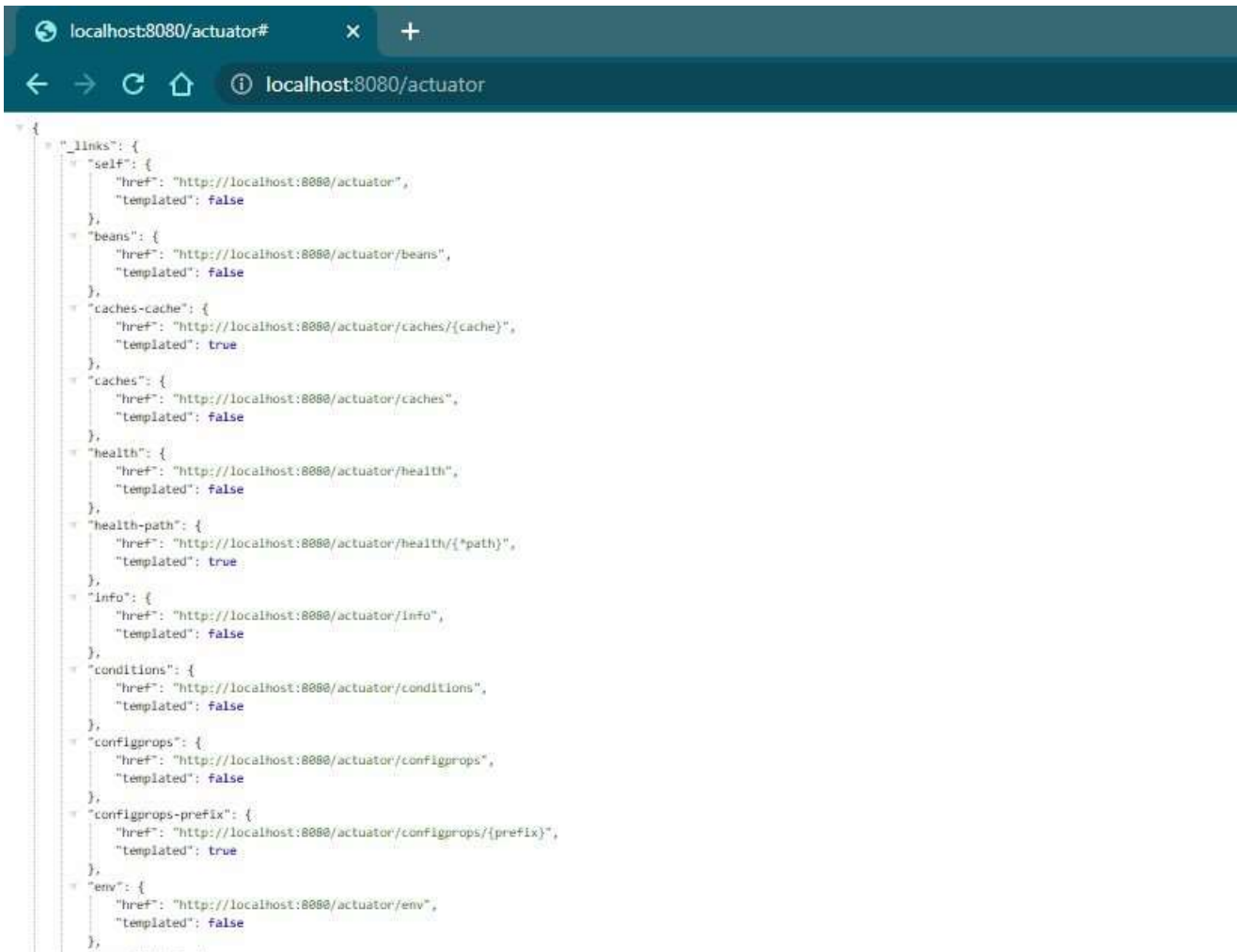
```
management.endpoints.web.exposure.include=metrics,info
```

This will include only metrics and info IDs and will exclude all others ('health' too). To add/include all ID information about your application, you can do it in the application.properties file by simply adding the following –

```
management.endpoints.web.exposure.include=*
```

Output:

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)



All the IDs or the Endpoint are now enabled

Excluding IDs/Endpoints

To exclude an ID or endpoint, use the following property and list out the respective IDs separated by a comma in the application.properties file.

```
management.endpoints.web.exposure.exclude
```

Example -> `management.endpoints.web.exposure.exclude=info`

Use '*' in place of IDs in property to exclude all the IDs or endpoints.

Notes:

1. Before setting the `management.endpoints.web.exposure.include`, ensure that the exposed actuators do not contain sensitive information.
2. They should be secured by placing them behind a firewall or are secured by something like Spring Security.

Last Updated : 09 Mar, 2022

62

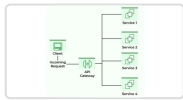
Similar Reads



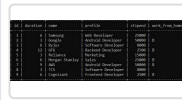
Difference Between Spring Boot Starter Web and Spring Boot Starter Tomcat



Spring Boot - Spring JDBC vs Spring Data JDBC



Java Spring Boot Microservices - Develop API Gateway Using Spring Cloud Gateway



Spring Boot | How to access database using Spring Data JPA



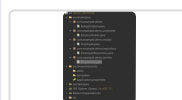
How to Create a Spring Boot Project in Spring Initializr and Run it in IntelliJ IDEA?



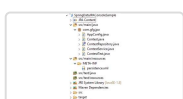
How to Create and Setup Spring Boot Project in Spring Tool Suite?



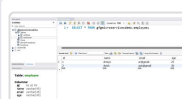
How to Run Your First Spring Boot Application in Spring Tool Suite?



Spring Boot - Spring Data JPA

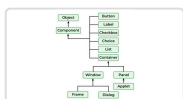


How to Make a Project Using Spring Boot, MySQL, Spring Data JPA, and Maven?



Java Spring Boot Microservices - Integration of Eureka, Feign & Spring Cloud Load Balancer

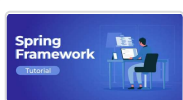
Related Tutorials



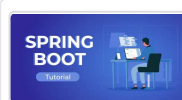
Java AWT Tutorial



Spring MVC Tutorial



Spring Tutorial



Spring Boot Tutorial



Java 8 Features - Complete Tutorial

[Previous](#)[Next](#)

Article Contributed By :



pawardarshan461

pawardarshan461

[Follow](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Current difficulty : [Easy](#)

[Easy](#)[Normal](#)[Medium](#)[Hard](#)[Expert](#)

Article Tags : [Geeks-Premier-League-2022](#), [Java-Spring-Boot](#), [Picked](#), [Geeks Premier League](#), [Java](#)

[Improve Article](#)[Report Issue](#)**GeeksforGeeks**

A-143, 9th Floor, Sovereign Corporate Tower, Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org



Company

[About Us](#)[Legal](#)[Careers](#)[In Media](#)[Contact Us](#)[Advertise with us](#)

Explore

[Job-A-Thon Hiring Challenge](#)[Hack-A-Thon](#)[GfG Weekly Contest](#)[Offline Classes \(Delhi/NCR\)](#)[DSA in JAVA/C++](#)[Master System Design](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial

DSA Concepts

Data Structures
Arrays
Strings
Linked List
Algorithms
Searching
Sorting
Mathematical
Dynamic Programming

DSA Roadmaps

DSA for Beginners
Basic DSA Coding Problems
Complete Roadmap To Learn DSA
DSA for FrontEnd Developers
DSA with JavaScript
Top 100 DSA Interview Problems
All Cheat Sheets
DSA Roadmap by Sandeep Jain

Web Development

HTML
CSS
JavaScript
Bootstrap
ReactJS
AngularJS
NodeJS
Express.js
Lodash

Computer Science

GATE CS Notes
Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

Python

Python Programming Examples
Django Tutorial
Python Projects
Python Tkinter
OpenCV Python Tutorial
Python Interview Question

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning Tutorial

DevOps

Git
AWS
Docker

[NumPy Tutorial](#)[GCP](#)[NLP Tutorial](#)[Deep Learning Tutorial](#)

Competitive Programming

[Top DSA for CP](#)[Top 50 Tree Problems](#)[Top 50 Graph Problems](#)[Top 50 Array Problems](#)[Top 50 String Problems](#)[Top 50 DP Problems](#)[Top 15 Websites for CP](#)

Interview Corner

[Company Wise Preparation](#)[Preparation for SDE](#)[Experienced Interviews](#)[Internship Interviews](#)[Competitive Programming](#)[Aptitude Preparation](#)

Commerce

[Accountancy](#)[Business Studies](#)[Economics](#)[Management](#)[Income Tax](#)[Finance](#)[Statistics for Economics](#)

SSC/ BANKING

[SSC CGL Syllabus](#)[SBI PO Syllabus](#)[SBI Clerk Syllabus](#)[IBPS PO Syllabus](#)[IBPS Clerk Syllabus](#)

System Design

[What is System Design](#)[Monolithic and Distributed SD](#)[Scalability in SD](#)[Databases in SD](#)[High Level Design or HLD](#)[Low Level Design or LLD](#)[Top SD Interview Questions](#)

GfG School

[CBSE Notes for Class 8](#)[CBSE Notes for Class 9](#)[CBSE Notes for Class 10](#)[CBSE Notes for Class 11](#)[CBSE Notes for Class 12](#)[English Grammar](#)

UPSC

[Polity Notes](#)[Geography Notes](#)[History Notes](#)[Science and Technology Notes](#)[Economics Notes](#)[Important Topics in Ethics](#)[UPSC Previous Year Papers](#)

Write & Earn

[Write an Article](#)[Improve an Article](#)[Pick Topics to Write](#)[Write Interview Experience](#)[Internships](#)

We use cookies to ensure you have the best browsing experience on our website. By using our site, you acknowledge that you have read and understood our [Cookie Policy](#) & [Privacy Policy](#)

SSC CGL Practice Papers

@geeksforgeeks , Some rights reserved