Spring Boot Actuator

Last Updated : 23 Jul, 2025

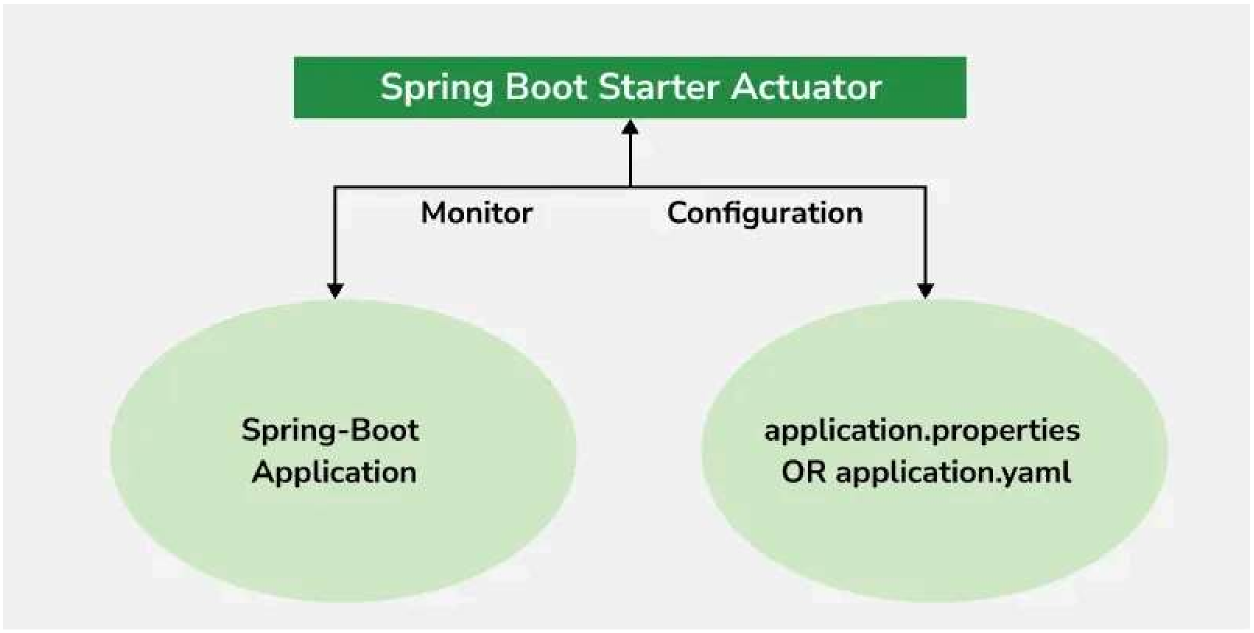
Developing and managing an application are the two most important aspects of the application's life cycle. It is very important to know what is going on beneath the application. Also, when we push the application into production, managing it gradually becomes critically important. Therefore, it is always recommended to monitor the application both during the development phase and at the production phase.

For the same use case, Spring Boot provides an **actuator dependency** that can be used **to monitor and manage your Spring Boot application**. **Spring Boot Actuator** is a key feature of Spring Boot, and with the help of this, developers can manage their application easily. By **/actuator** and **/actuator/health** endpoints, we can achieve the purpose of monitoring.

- With the help of Spring Boot, we can achieve the above objectives.

- We can use it to monitor and manage the application with the help of HTTP endpoints or with JMX.

The image below demonstrates the **structure of the Spring Boot Starter Actuator:**


Spring Boot Starter Actuator
Monitor
Configuration
Spring-Boot Application
application.properties OR application.yaml

Advantages of Actuator Application

- It increases customer satisfaction.
- It reduces downtime.
- It boosts productivity.
- It improves Cybersecurity Management.
- It increases the conversion rate.

1. Configuration for Actuator

In order to use hibernate validators, these configurations are necessary in your Spring Boot project.

1.1 Adding Actuator Dependency

To use the "Actuator", add the following dependency in your application's project settings file.

For Maven (pom.xml):

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

For Gradle (build.gradle):

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

1.2 Configuring Actuator in application.properties

Actuator provides several configuration options to customize its behavior. Below are some common configurations:

- We can also change the default endpoint by adding the following in the **application.properties** file.

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

- 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
```

- List down all IDs that we want to include which are separated by a comma.

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

- Include only metrics and info IDs and will exclude all others ('health' too).

To add/include all ID information about our application, we can do it in the application.properties file by simply adding the following

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

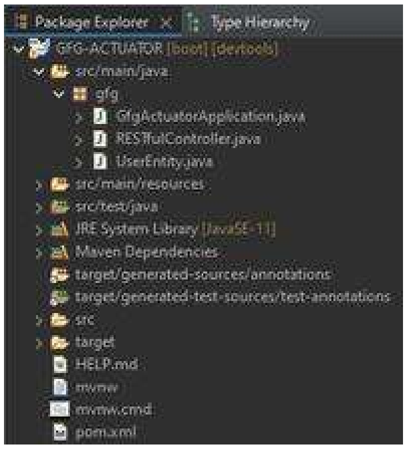
- Excluding IDs/Endpoints

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

Project Folder Structure:

The below image demonstrates the picture of how your project must look:



```
GFG-ACTUATOR [boot] [devtools]
  src/main/java
    gfg
      GfgActuatorApplication.java
      RESTfulController.java
      UserEntity.java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

2. Implementing the Project

2.1 Entity Class

UserEntity.java (Entity class representing the model data) is explained below:

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

Implementation:

```java
package gfg;

import lombok.Data;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Component;

@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";
}
```

2.2 Controller

RESTfulController.java (A REST API controller) for defining APIs and testing the program.

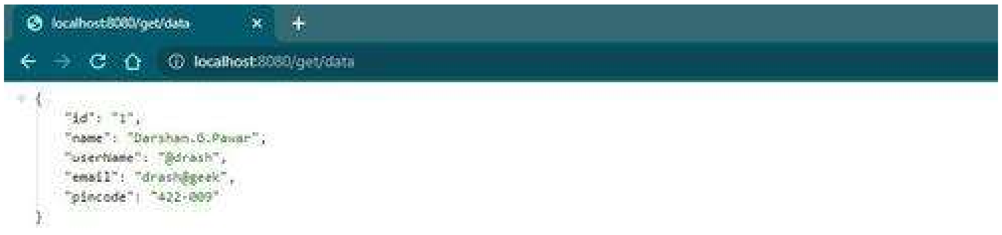
This controller's get() method uses the UserEntity bean to return JSON response. UserEntiy bean is outsourced through '*@Autowired*' annotation which was registered in Spring's application context.

```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;
    }
}
```

3. Testing Actuator APIs

3.1 Controller APIs

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



```
{
    "id": "1",
    "name": "Darshan.G.Pawar",
    "userName": "@drash",
    "email": "drash@geek",
    "pincode": "422-009"
}
```

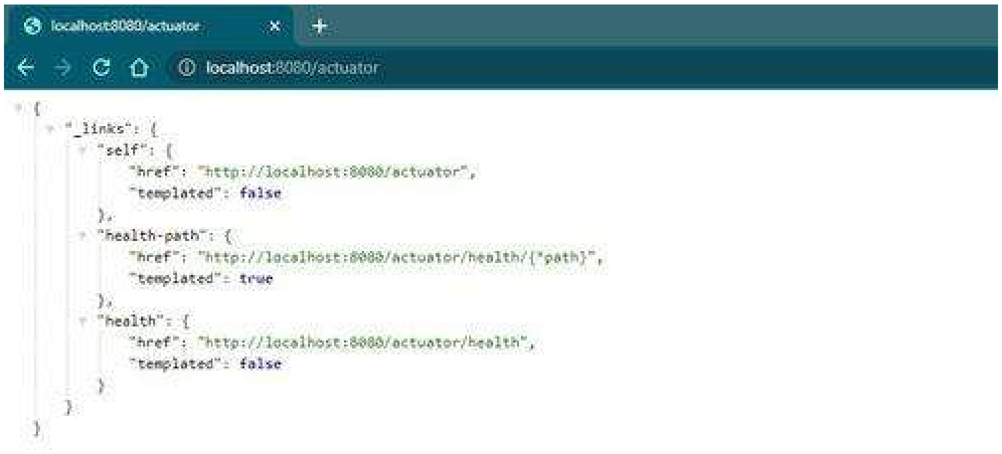
3.2 Working with Spring Boot Actuator APIs

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

3.2.1 /actuator

It's simple just hit the default endpoint '/actuator', ensure that your Application is running.

Example:

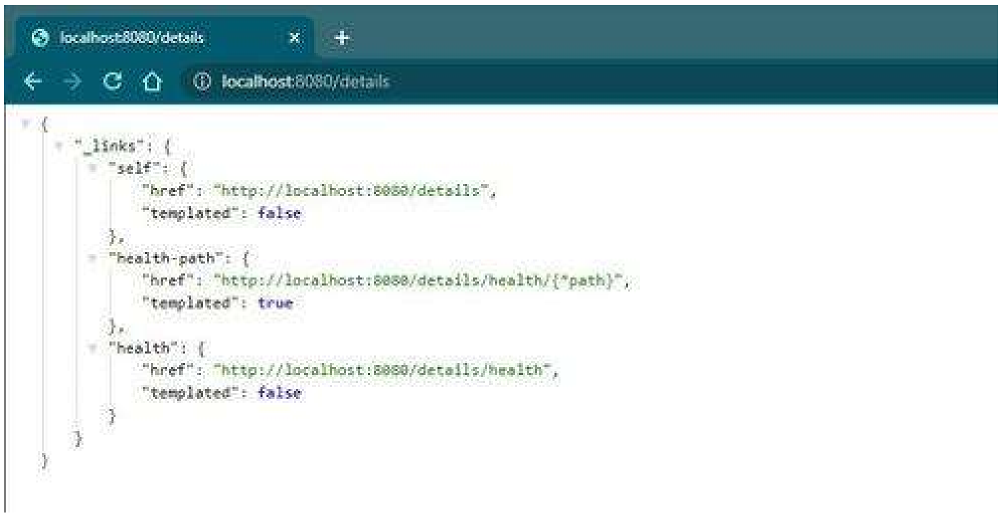


localhost:8080/actuator

```
{
    "_links": {
        "self": {
            "href": "http://localhost:8080/actuator",
            "templated": false
        },
        "health-path": {
            "href": "http://localhost:8080/actuator/health/{*path}",
            "templated": true
        },
        "health": {
            "href": "http://localhost:8080/actuator/health",
            "templated": false
        }
    }
}
```

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

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

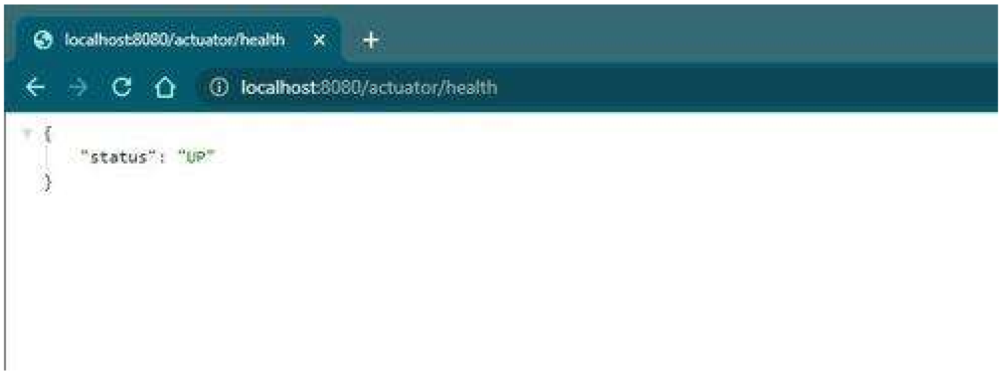


localhost:8080/details

```
{
    "_links": {
        "self": {
            "href": "http://localhost:8080/details",
            "templated": false
        },
        "health-path": {
            "href": "http://localhost:8080/details/health/{*path}",
            "templated": true
        },
        "health": {
            "href": "http://localhost:8080/details/health",
            "templated": false
        }
    }
}
```

3.2.2 /actuator/health

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


localhost:8080/actuator/health
localhost:8080/actuator/health

```
{
  "status": "UP"
}
```

'UP' means the application's health is good.

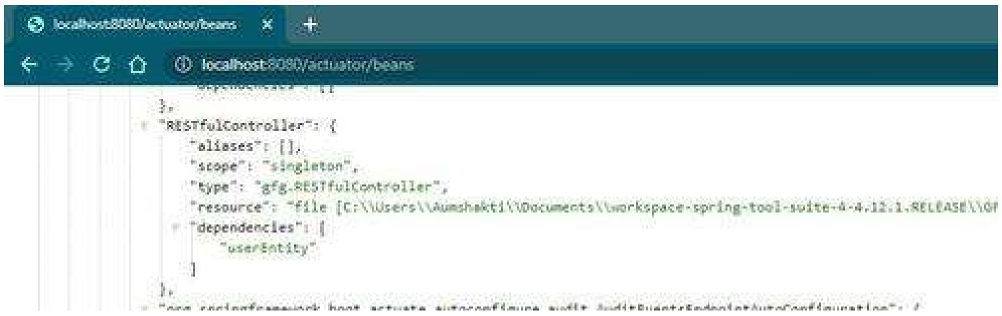
Commonly Used Actuator Endpoints

There are a total of 25 IDs out of which the commonly used are listed out below:

EndpointID	Description
beans	Displays a complete list of all the Spring beans in your application.
caches	Exposes available caches.
conditions	Shows conditions evaluated on configuration and auto-configuration classes.
health	Shows application health information.

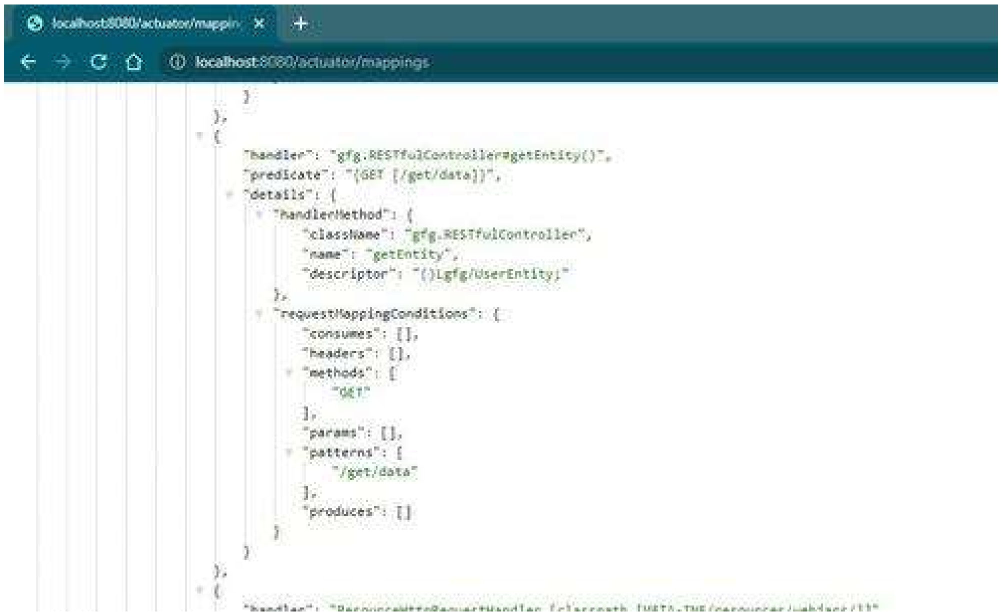
EndpointID	Description
httptrace	Displays HTTP trace information (last 100 requests).
loggers	Shows and modifies the configuration of loggers in the application.
mappings	Displays a collated list of all @RequestMapping paths.
sessions	Retrieves and deletes user sessions (requires Spring Session).
threaddump	Performs a thread dump.

3.2.3 /actuator/beans



```
    },
  "RESTfulController": {
      "aliases": [],
      "scope": "singleton",
      "type": "gfg.RESTfulController",
      "resource": "file [C:\\Users\\Aumshakti\\Documents\\workspace-spring-tool-suite-4-4.12.1.RELEASE\\GF
      "dependencies": [
          "userEntity"
      ]
    },
```

3.2.4 /actuator/mappings


localhost:8080/actuator/mappings
localhost:8080/actuator/mappings

```
        }
    },
    {
        "handler": "gfg.RESTfulController#getEntity()",
        "predicate": "{GET [/get/data]}",
        "details": {
            "handlerMethod": {
                "className": "gfg.RESTfulController",
                "name": "getEntity",
                "descriptor": "()Lgfg/UserEntity;"
            },
            "requestMappingConditions": {
                "consumes": [],
                "headers": [],
                "methods": [
                    "GET"
                ],
                "params": [],
                "patterns": [
                    "/get/data"
                ],
                "produces": []
            }
        }
    },
    {
```

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

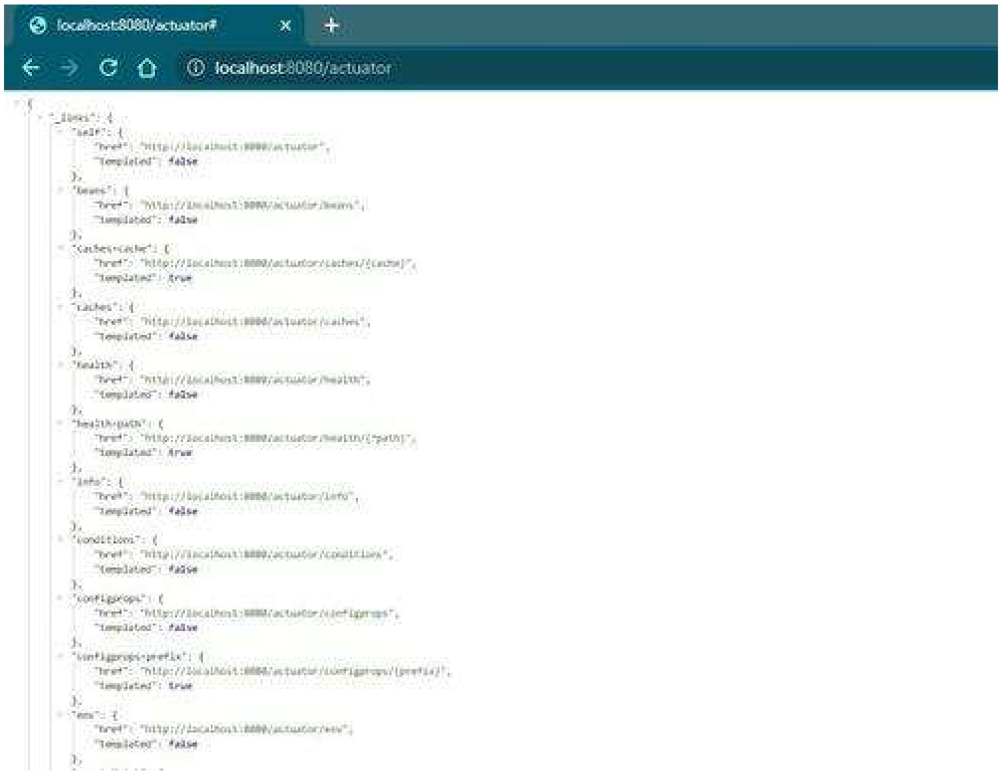
OR, we can just list down all IDs that you want to include which are separated by a comma.

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 our application, we can do it in the application.properties file by simply adding the following

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

Output:



```
{
  "_links": {
    "self": {
      "href": "http://localhost:8080/actuator",
      "templated": false
    },
    "beans": {
      "href": "http://localhost:8080/actuator/beans",
      "templated": false
    },
    "caches-cache": {
      "href": "http://localhost:8080/actuator/caches/{cache}",
      "templated": true
    },
    "caches": {
      "href": "http://localhost:8080/actuator/caches",
      "templated": false
    },
    "health": {
      "href": "http://localhost:8080/actuator/health",
      "templated": false
    },
    "health-path": {
      "href": "http://localhost:8080/actuator/health/{*path}",
      "templated": true
    },
    "info": {
      "href": "http://localhost:8080/actuator/info",
      "templated": false
    },
    "conditions": {
      "href": "http://localhost:8080/actuator/conditions",
      "templated": false
    },
    "configprops": {
      "href": "http://localhost:8080/actuator/configprops",
      "templated": false
    },
    "configprops-prefix": {
      "href": "http://localhost:8080/actuator/configprops/{prefix}",
      "templated": true
    },
    "env": {
      "href": "http://localhost:8080/actuator/env",
      "templated": false
    },
```

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.*