

- [Start Here](#)

[Courses ▼▲](#)

[REST with Spring Boot](#)

The canonical reference for building a production grade API with Spring (**the price will increase by \$50 after the launch**)

[Learn Spring Security ▼▲](#)

THE unique Spring Security education if you're working with Java today

[Learn Spring Security Core](#)

[Focus on the Core of Spring Security 6](#)

[Learn Spring Security OAuth](#)

[Focus on the new OAuth2 stack in Spring Security 6](#)

[Learn Spring](#)

[From no experience to actually building stuff■](#)

[Learn Spring Data JPA](#)

[The full guide to persistence with Spring Data JPA](#)

[Guides ▼▲](#)

[Persistence](#)

[The Persistence with Spring guides](#)

[REST](#)

[The guides on building REST APIs with Spring](#)

[Security](#)

[The Spring Security guides](#)

[About ▼▲](#)

[Full Archive](#)

[The high level overview of all the articles on the site.](#)

[Baeldung Ebooks](#)

[Discover all of our eBooks](#)

[About Baeldung](#)

[About Baeldung.](#)

[Write for Baeldung](#)

[Become a writer on the site.](#)

How to Enable All Endpoints in Spring Boot Actuator

Last updated: May 11, 2024

- [Spring Boot](#)
- [Spring Boot Actuator](#)

Now that the new version of **REST With Spring** - "REST With Spring Boot" is finally out, the current price will be **available until the 22nd of June**, after which it will permanently increase by **50\$**

[>> GET ACCESS NOW](#)

Get started with Spring Boot and with core Spring, through the *Learn Spring* course:

[>> CHECK OUT THE COURSE](#)

1. Overview

In this tutorial, we're going to learn how to enable all the endpoints in the [Spring Boot Actuator](#). We'll start with the necessary Maven dependencies. From there, we'll look at how to control our endpoints via our properties files. We'll finish up with an overview of how to secure our endpoints.

There have been several [changes between Spring Boot 1.x and Spring Boot 2.x](#) in terms of how actuator endpoints are configured. We'll note these as they come up.

2. Setup

In order to use the actuator, we need to include the [spring-boot-starter-actuator](#) in our Maven configuration:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  <version>3.1.2</version>
</dependency>
```

Additionally, starting with Spring Boot 2.0, **we need to include the [web starter](#) if we want our endpoints exposed via HTTP**:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
  <version>3.1.2</version>
</dependency>
```

3. Enabling and Exposing Endpoints

Starting with Spring Boot 2, **we have to enable and expose our endpoints**. By default, all endpoints but `/shutdown` are enabled and only `/health` and `/info` are exposed. All endpoints are found at `/actuator` even if we've configured a different root context for our application.

That means that once we've added the appropriate starters to our Maven configuration, we can access the `/health` and `/info` endpoints at `http://localhost:8080/actuator/health` and `http://localhost:8080/actuator/info`.

Let's go to `http://localhost:8080/actuator` and view a list of available endpoints because the actuator endpoints are HATEOS enabled. We should see `/health` and `/info`.

```
{ "_links": { "self": { "href": "http://localhost:8080/actuator", "templated": false },
"health": { "href": "http://localhost:8080/actuator/health", "templated": false },
"info": { "href": "http://localhost:8080/actuator/info", "templated": false } }
```

3.1. Exposing All Endpoints

Now, let's expose all endpoints except `/shutdown` by modifying our `application.properties` file:

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

Once we've restarted our server and accessed the `/actuator` endpoint again we should see the other endpoints available with the exception of `/shutdown`:

```
{ "_links": { "self": { "href": "http://localhost:8080/actuator", "templated": false },
"beans": { "href": "http://localhost:8080/actuator/beans", "templated": false },
"caches": { "href": "http://localhost:8080/actuator/caches", "templated": false },
"health": { "href": "http://localhost:8080/actuator/health", "templated": false },
"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 },
"env": { "href": "http://localhost:8080/actuator/env", "templated": false },
"loggers": { "href": "http://localhost:8080/actuator/loggers", "templated": false },
"heapdump": { "href": "http://localhost:8080/actuator/heapdump", "templated": false },
"threaddump": { "href": "http://localhost:8080/actuator/threaddump", "templated": false },
"metrics": { "href": "http://localhost:8080/actuator/metrics", "templated": false },
"scheduledtasks": { "href": "http://localhost:8080/actuator/scheduledtasks", "templated": false },
"mappings": { "href": "http://localhost:8080/actuator/mappings", "templated": false } }
```

3.2. Exposing Specific Endpoints

Some endpoints can expose sensitive data, so let's learn how to be more fine-grained about which endpoints we expose.

The `management.endpoints.web.exposure.include` property can also take a comma-separated list of endpoints. So, let's only expose `/beans` and `/loggers`:

```
management.endpoints.web.exposure.include=beans, loggers
```

In addition to including certain endpoints with a property, we can also exclude endpoints. Let's expose all the endpoints except `/threaddump`:

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

Both the `include` and `exclude` properties take a list of endpoints. **The `exclude` property takes precedence over `include`.**

3.3. Enabling Specific Endpoints

Next, let's learn how we can get more fine-grained about which endpoints we have enabled.

First, we need to turn off the default that enables all the endpoints:

```
management.endpoints.enabled-by-default=false
```

Next, let's enable and expose only the `/health` endpoint:

```
management.endpoint.health.enabled=true
management.endpoints.web.exposure.include=health
```

With this configuration, we can access only the `/health` endpoint.

3.4. Enabling Shutdown

Because of its sensitive nature, **the `/shutdown` endpoint is disabled by default.**

Let's enable it now by adding a line to our `application.properties` file:

```
management.endpoint.shutdown.enabled=true
```

Now when we query the `/actuator` endpoint, we should see it listed. **The `/shutdown` endpoint only accepts `POST` requests**, so let's shut down our application gracefully:

```
curl -X POST http://localhost:8080/actuator/shutdown
```

4. Securing Endpoints

In a real-world application, we're most likely going to have security on our application. With that in mind, let's secure our actuator endpoints.

First, let's add security to our application by adding the [security starter Maven dependency](#):

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
  <version>2.5.1</version>
</dependency>
```

For the most basic security, that's all we have to do. Just by adding the security starter, we've automatically applied basic authentication to all exposed endpoints except `/info` and `/health`.

Now, let's customize our security to restrict the `/actuator` endpoints to an `ADMIN` role.

Let's start by excluding the default security configuration:

```
@SpringBootApplication(exclude = {
    SecurityAutoConfiguration.class,
    ManagementWebSecurityAutoConfiguration.class
})
```

Let's note the `ManagementWebSecurityAutoConfiguration.class` because this will let us apply our own security configuration to the `/actuator`.

Over in our configuration class, let's configure a couple of users and roles, so we have an `ADMIN` role to work with:

```
@Bean
public InMemoryUserDetailsManager userDetailsService() {
```

```

UserDetails user = User.withDefaultPasswordEncoder()
    .username("user")
    .password("password")
    .roles("USER")
    .build();

UserDetails admin = User.withDefaultPasswordEncoder()
    .username("admin")
    .password("password")
    .roles("USER", "ADMIN")
    .build();

return new InMemoryUserDetailsManager(user, admin);
}

```

SpringBoot provides us with a convenient request matcher to use for our actuator endpoints.

Let's use it to lockdown our */actuator* to only the *ADMIN* role:

```

http.authorizeHttpRequests(authz -> {
    authz.requestMatchers(mvc.pattern("/actuator/**"))
        .hasRole("ADMIN")
        .anyRequest()
        .authenticated();
});

```

5. Conclusion

In this tutorial, we learned how Spring Boot configures the actuator by default. After that, we customized which endpoints were enabled, disabled, and exposed in our *application.properties* file. Because Spring Boot configures the */shutdown* endpoint differently by default, we learned how to enable it separately.

After learning the basics, we then learned how to configure actuator security.

As always, the example code is available [over on GitHub](#).

Now that the new version of **REST With Spring** - "REST With Spring Boot" is finally out, the current price will be **available until the 22nd of June**, after which it will permanently increase by **50\$**

[>> GET ACCESS NOW](#)

Get started with Spring Boot and with core Spring, through the *Learn Spring* course:

[>> CHECK OUT THE COURSE](#)

Learning to build your API
with **Spring**?

[Download the E-book](#)

Comments are open for 30 days after publishing a post. For any issues past this date, use the Contact form on the site.

Courses

- [All Courses](#)
- [All Bulk Courses](#)
- [All Bulk Team Courses](#)
- [The Courses Platform](#)

Series

- [Java "Back to Basics" Tutorial](#)
- [Jackson JSON Tutorial](#)
- [Apache HttpClient Tutorial](#)
- [REST with Spring Tutorial](#)
- [Spring Persistence Tutorial](#)
- [Security with Spring](#)
- [Spring Reactive Tutorials](#)
- [Java Array](#)

About

- [About Baeldung](#)
- [The Full Archive](#)

- [Editors](#)
- [Jobs](#)
- [Our Partners](#)
- [Partner with Baeldung](#)
- [Terms of Service](#)
- [Privacy Policy](#)
- [Company Info](#)
- [Contact](#)