Spring Boot Actuator (Production-ready features)

By Ramesh Fadatare (Java Guides)

Spring Boot Actuator

1. Spring Boot Actuator module provides production-ready features such as monitoring, metrics and health checks.

2. The Spring Boot Actuator enables you to monitor the application using HTTP endpoints and JMX.

3. Spring Boot Provides a spring-boot-starter-actuator library to auto-configure Actuator.

Commonly used Actuator endpoints

1. Actuator **/info** endpoint
2. Actuator **/health** endpoint
3. Actuator **/beans** endpoint
4. Actuator **/conditions** endpoint
5. Actuator **/mappings** endpoint
6. Actuator **/configprops** endpoint
7. Actuator **/matrics** endpoint
8. Actuator **/env** endpoint
9. Actuator **/threaddump** endpoint
10. Actuator **/loggers** endpoint
11. Actuator **/shutdown** endpoint

The /info Endpoint

If you added any information about the application in application.properties then we can view it using **/info** endpoint:

http://localhost:8080/actuator/info

The /health Endpoint

The **/health** endpoint shows the health of the application, including the disk space, databases and more.

http://localhost:8080/actuator/health

The /beans Endpoint

The **/beans** endpoint shows all the beans registered in your application, including the beans you explicitly configured and those auto configured by Spring Boot.

http://localhost:8080/actuator/beans

The /conditions Endpoint

The **/conditions** endpoint shows the auto configuration report, categorised into pasitiveMatches and negativeMatches

http://localhost:8080/actuator/conditions

The /mappings Endpoint

The **/mappings** endpoint shows all the **@RequestMapping** paths declared in the application.

This is very helpful for checking which request path will be handled by which controller method.

http://localhost:8080/actuator/mappings

The /configprops Endpoint

The **/configprops** endpoint offers all the configuration properties defined by @ConfigurationProperties bean, including your configuration properties defined in the application.properties or YAML files.

http://localhost:8080/actuator/configprops

The /metrics Endpoint

The **/metrics** endpoint shows various metrics about the current application such as how much memory it is using, how much memory is free, the size of the heap used, the number of threads used, and so on.

http://localhost:8080/actuator/metrics

The /env Endpoint

The **/env** endpoint exposes all the properties from the Spring's **ConfigurableEnvironment** interface, such as a list of active profiles, application properties, system environment variables and so on.

http://localhost:8080/actuator/env

The /threaddump Endpoint

Using **/threaddumb** endpoint, you can view your application's thread dumb with running threads details and JVM stack trace.

http://localhost:8080/actuator/threaddump

The /loggers Endpoint

The /loggers endpoint allows you to view and configure the log levels of your application at runtime.

http://localhost:8080/actuator/loggers

You can view the logging level of the specific logger:

http://localhost:8080/actuator/loggers/{name}

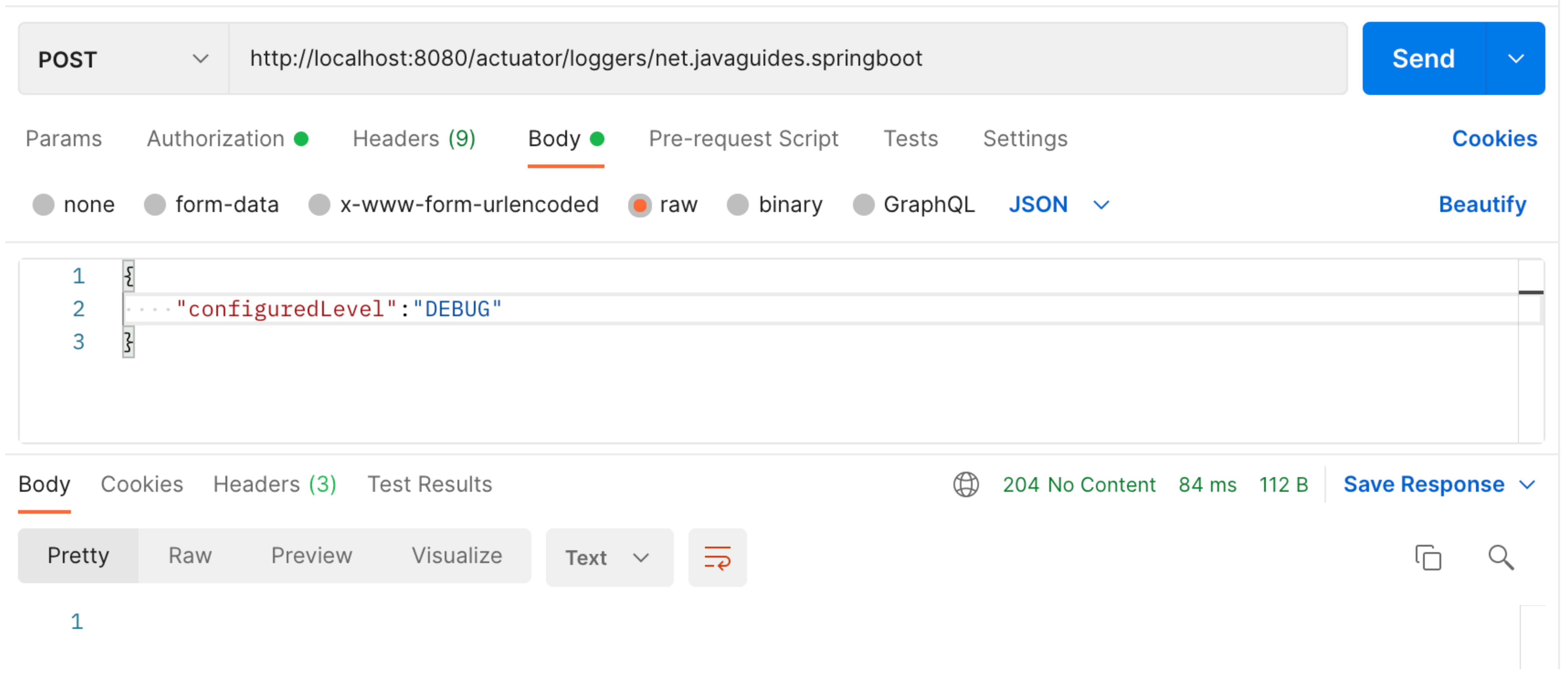
Ex:

http://localhost:8080/actuator/loggers/net.javaguides.springboot

The /loggers Endpoint

You can update the logging level of the logger at a runtime by sending a POST request to URL: http://localhost:8080/actuator/loggers/{name}

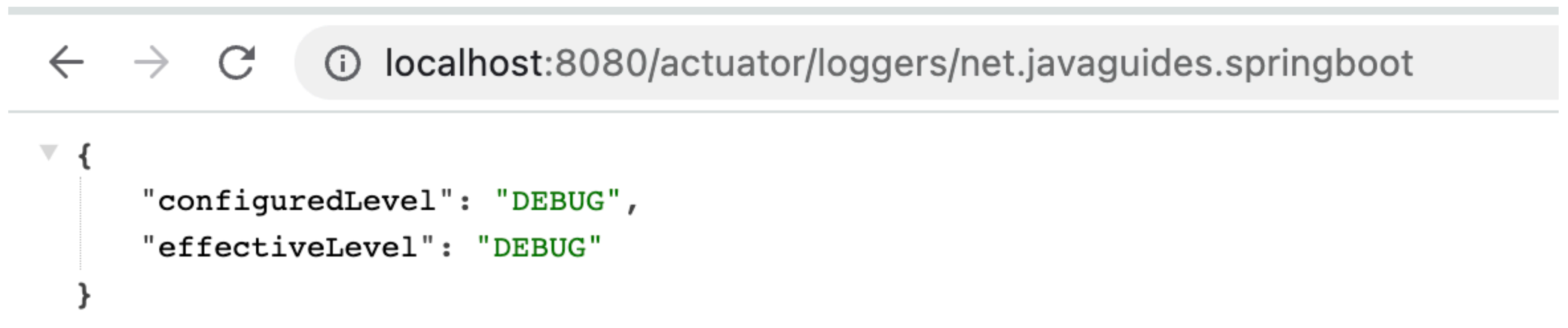
Ex: http://localhost:8080/actuator/loggers/net.javaguides.springboot



http://localhost:8080/actuator/loggers/net.javaguides.springboot

```
{
    "configuredLevel":"DEBUG"
}
```

204 No Content 84 ms 112 B



localhost:8080/actuator/loggers/net.javaguides.springboot

```
{
    "configuredLevel": "DEBUG",
    "effectiveLevel": "DEBUG"
}
```

The /shutdown Endpoint

The /shutdown endpoint can be used to gracefully shut down the application.

This endpoint not enabled by default. You can enable this endpoint by adding this property in application.properties file:

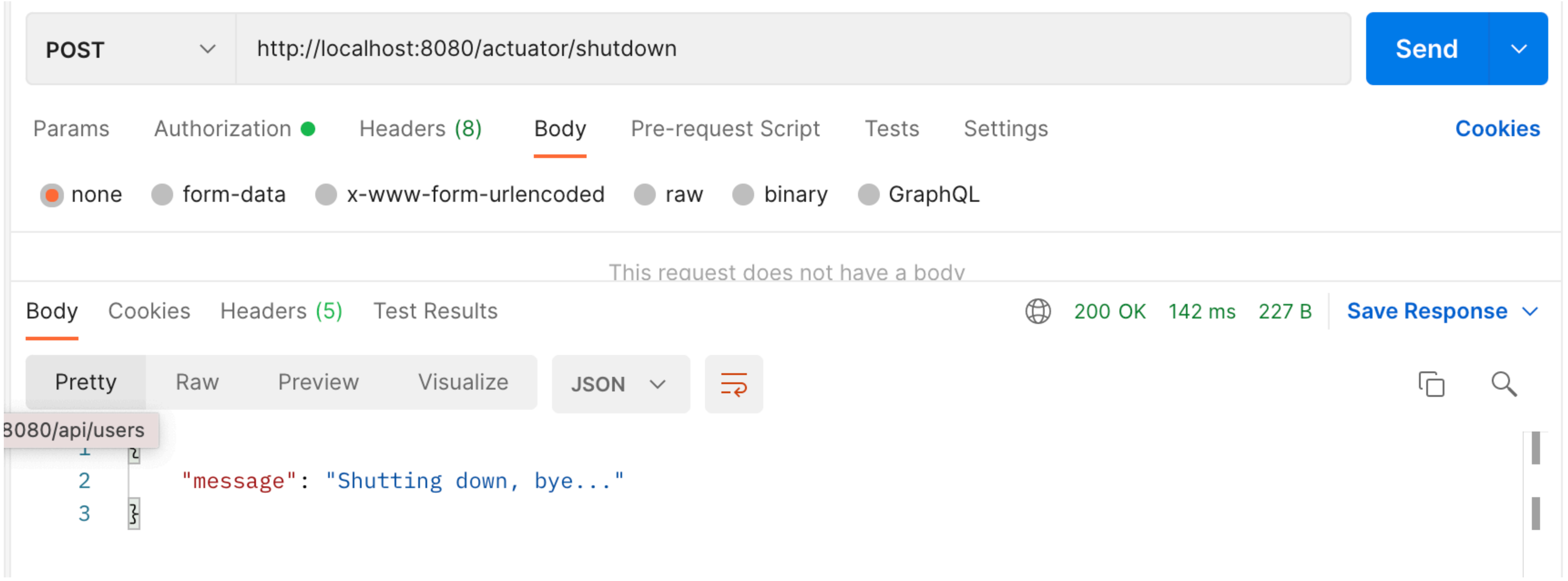
Management.endpoint.shutdown.enabled=true

After adding this property, we need to send the HTTP POST request to below endpoint:

http://localhost:8080/actuator/shutdown

The /shutdown Endpoint

Ex:



```
http://localhost:8080/actuator/shutdown
```

This request does not have a body

```
{
    "message": "Shutting down, bye..."
}
```

The /shutdown Endpoint

1. This endpoint not enabled by default. You can enable this endpoint by adding this property in application.properties file:

Management.endpoint.shutdown.enabled=true

2. After adding this property, we need to send the HTTP POST request to below endpoint:

http://localhost:8080/actuator/shutdown

3. Watch the console log for spring boot application shutdown