**Reference:** [**https://docs.spring.io/spring-boot/docs/current/reference/html/actuator.html**](https://docs.spring.io/spring-boot/docs/current/reference/html/actuator.html)

**Actuator**

* Actuator Overview and Capabilities
* Actuator Endpoints
* Enabling and disabling endpoints
* Creating Custom Actuators Endpoint

**Spring Boot Actuator:**

**Spring Boot Actuator** is a sub-project of the Spring Boot Framework. It includes several additional features that help us to monitor and manage the Spring Boot application. 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. If we want to get production-ready features in an application, we should use the S**pring Boot actuator.**

Spring Boot includes several additional features to help you monitor and manage your application when you push it to production. You can choose to manage and monitor your application by using HTTP endpoints or with JMX. Auditing, health, and metrics gathering can also be automatically applied to your application.

**1. Enabling Production-ready Features**

The [spring-boot-actuator](https://github.com/spring-projects/spring-boot/tree/v3.0.1/spring-boot-project/spring-boot-actuator) module provides all of Spring Boot’s production-ready features. The recommended way to enable the features is to add a dependency on the spring-boot-starter-actuator “Starter”.

Definition of Actuator

An actuator is a manufacturing term that refers to a mechanical device for moving or controlling something. Actuators can generate a large amount of motion from a small change.

To add the actuator to a Maven-based project, add the following ‘Starter’ dependency:

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

**2. Endpoints**

Actuator endpoints let you monitor and interact with your application. Spring Boot includes a number of built-in endpoints and lets you add your own. For example, the health endpoint provides basic application health information.

The following technology-agnostic endpoints are available:

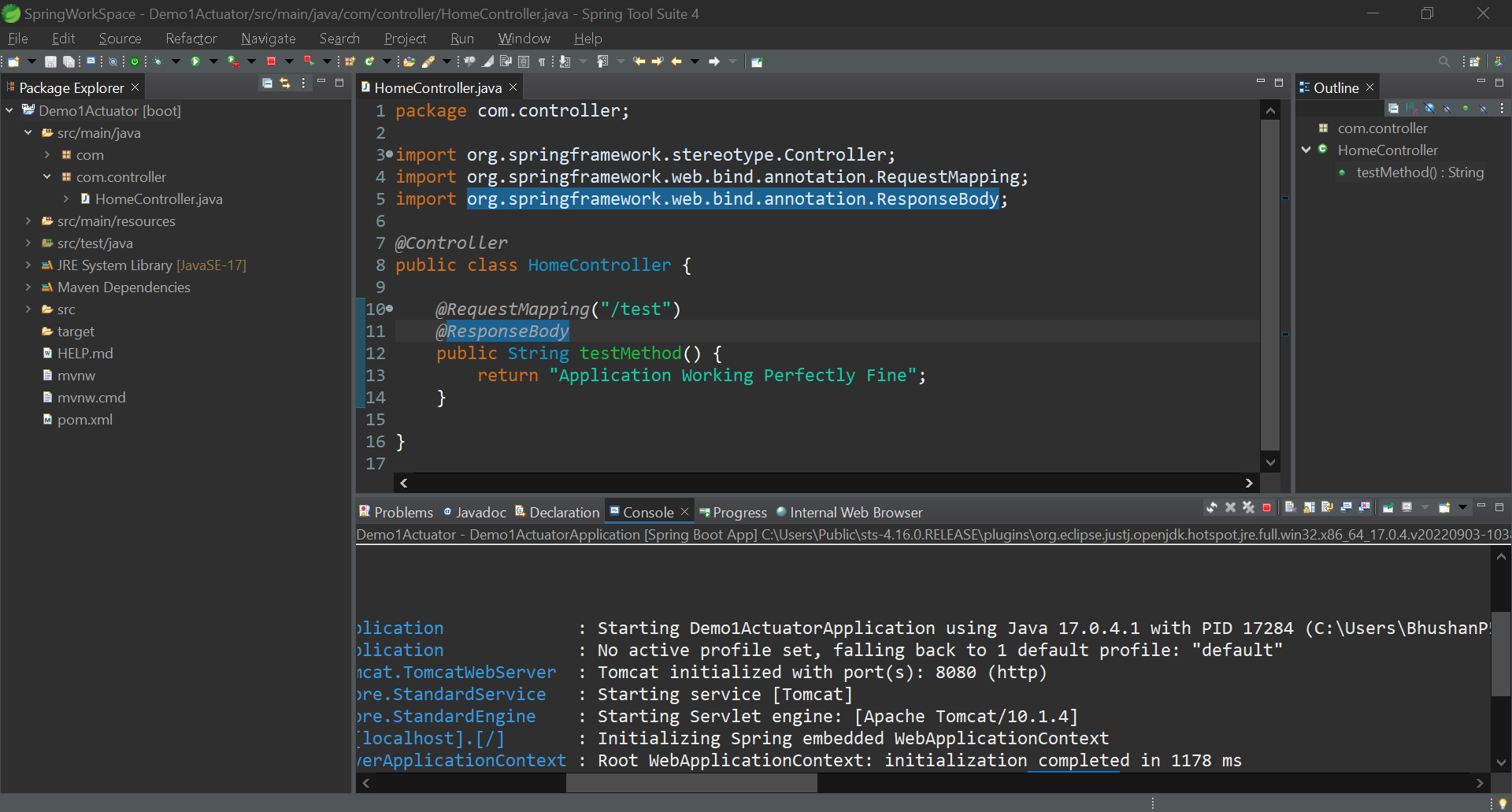
|  |  |
| --- | --- |
| ID | Description |
| auditevents | Exposes audit events information for the current application. Requires an AuditEventRepository bean. |
| beans | Displays a complete list of all the Spring beans in your application. |
| caches | Exposes available caches. |
| conditions | Shows the conditions that were evaluated on configuration and auto-configuration classes and the reasons why they did or did not match. |
| configprops | Displays a collated list of all @ConfigurationProperties. |
| env | Exposes properties from Spring’s ConfigurableEnvironment. |
| flyway | Shows any Flyway database migrations that have been applied. Requires one or more Flyway beans. |
| health | Shows application health information. |
| httpexchanges | Displays HTTP exchange information (by default, the last 100 HTTP request-response exchanges). Requires an HttpExchangeRepository bean. |
| info | Displays arbitrary application info. |
| integrationgraph | Shows the Spring Integration graph. Requires a dependency on spring-integration-core. |
| loggers | Shows and modifies the configuration of loggers in the application. |
| liquibase | Shows any Liquibase database migrations that have been applied. Requires one or more Liquibase beans. |
| metrics | Shows “metrics” information for the current application. |
| mappings | Displays a collated list of all @RequestMapping paths. |
| quartz | Shows information about Quartz Scheduler jobs. |
| scheduledtasks | Displays the scheduled tasks in your application. |
| 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. |
| shutdown | Lets the application be gracefully shutdown. Only works when using jar packaging. Disabled by default. |
| startup | Shows the [startup steps data](https://docs.spring.io/spring-boot/docs/current/reference/html/features.html" \l "features.spring-application.startup-tracking) collected by the ApplicationStartup. Requires the SpringApplication to be configured with a BufferingApplicationStartup. |
| threaddump | Performs a thread dump. |

If your application is a web application (Spring MVC, Spring WebFlux, or Jersey), you can use the following additional

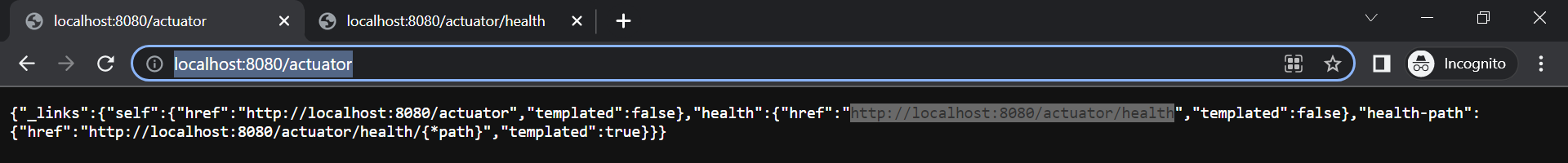
endpoints:

|  |  |
| --- | --- |
| ID | Description |
| heapdump | Returns a heap dump file. On a HotSpot JVM, an HPROF-format file is returned. On an OpenJ9 JVM, a PHD-format file is returned. |
| logfile | Returns the contents of the logfile (if the logging.file.name or the logging.file.path property has been set). Supports the use of the HTTP Range header to retrieve part of the log file’s content. |
| prometheus | Exposes metrics in a format that can be scraped by a Prometheus server. Requires a dependency on micrometer-registry-prometheus. |

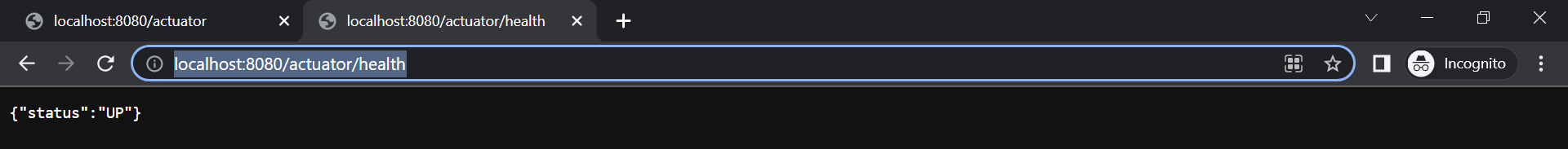
For Better understanding Create new Spring Starter Project with Spring Web and Spring Actuator as Starter Dependencies in it. Run Code and test it with /actuator.



<http://localhost:8080/actuator>



<http://localhost:8080/actuator/health>



To add more endpoints also to view details we can add some endpoints in **application.properties**

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

management.endpoint.health.show-details=always

**Que**: Let’s consider a scenario where you have included all endpoints and but later you want to disable endpoint of bean then how you can do it?

**Ans**: By adding a configuration line in application.properties

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

management.endpoint.health.show-details=always

**management.endpoint.beans.enabled=false**

**Que**: our base path is /actuator, can we change it?

**Ans:** By adding a configuration line in application.properties

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

management.endpoint.health.show-details=always

management.endpoint.beans.enabled=false

**management.endpoints.web.base-path=/admin**

**Que**: Can we create any custom

**Ans.**

**Step 1**. Create a class Ex. MyDatabaseHealthService

**Step 2.** Add @Controller annotation on top of class

**Step 3**. Implement this class with HealthIndicator

import org.springframework.boot.actuate.health.Health;

import org.springframework.boot.actuate.health.HealthIndicator;

import org.springframework.stereotype.Controller;

@Controller

public class MyDatabaseHealthService implements HealthIndicator {

@Override

public Health health() {

// TODO Auto-generated method stub

return null;

}

}

**Step 4**. Add given below code to customize health endpoint

package com.controller;

import org.springframework.boot.actuate.health.Health;

import org.springframework.boot.actuate.health.HealthIndicator;

import org.springframework.stereotype.Controller;

@Controller

public class MyDatabaseHealthService implements HealthIndicator {

public static final String DB\_SERVICE="Database Service";

public boolean isDbWorking() {

return false;

}

@Override

public Health health() {

if(isDbWorking()) {

return Health.up().withDetail(DB\_SERVICE, "Database Service is Running Fine").build();

}

else

return Health.down().withDetail(DB\_SERVICE, "Database Service Not Working").build();

}

}