Spring Boot Actuator is a module within the Spring Boot framework that provides production-ready features for monitoring and managing a Spring Boot application. It exposes various operational insights and management capabilities through built-in endpoints, primarily accessible via HTTP or JMX.

Overview:

Actuator simplifies the process of understanding and interacting with a running Spring Boot application in a production environment. It offers a set of pre-defined endpoints that provide information about the application's health, metrics, environment, configuration, and more, without requiring extensive custom development.

Capabilities:

Spring Boot Actuator offers a range of capabilities through its various endpoints:

* **Health Checks:**

The /health endpoint provides a summary of the application's health status, including details about database connectivity, disk space, and other integrated components.

* **Metrics:**

The /metrics endpoint exposes various application metrics, such as CPU usage, memory consumption, thread counts, and custom metrics defined within the application.

* **Environment Information:**

The /env endpoint reveals the Spring Environment properties, including active profiles, configuration properties, and system environment variables.

* **Configuration Details:**

Endpoints like /beans, /conditions, and /configprops provide insights into the application's Spring beans, auto-configuration conditions, and configuration properties, respectively.

* **Auditing and Tracing:**

Actuator can be configured to expose audit events and provide tracing information for requests, aiding in debugging and performance analysis.

* **Log Management:**

The /loggers endpoint allows querying and dynamic modification of logging levels for different loggers within the application.

* **Scheduled Tasks:**

The /scheduledtasks endpoint lists details about all scheduled tasks configured in the application.

* **Thread Dumps and Heap Dumps:**

Endpoints like /threaddump and /heapdump enable the generation of thread dumps and heap dumps for detailed analysis of application performance and memory usage.

* **Application Shutdown:**

The /shutdown endpoint, though disabled by default for security reasons, allows for a graceful shutdown of the application.

* **Custom Endpoints:**

Actuator provides the flexibility to create custom endpoints to expose application-specific information or management operations.

Enabling and Exposure:

To use Spring Boot Actuator, the spring-boot-starter-actuator dependency must be added to the project. While most endpoints are enabled by default, their exposure over HTTP or JMX can be configured using properties like management.endpoints.web.exposure.include or management.endpoints.jmx.exposure.include. Security measures, such as requiring authentication, are crucial for production environments.

eg

To see it, go to <http://localhost:8082/actuator>

**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.

You can [control access](https://docs.spring.io/spring-boot/reference/actuator/endpoints.html#actuator.endpoints.controlling-access) to each individual endpoint and [expose them (make them remotely accessible) over HTTP or JMX](https://docs.spring.io/spring-boot/reference/actuator/endpoints.html#actuator.endpoints.exposing). An endpoint is considered to be available when access to it is permitted and it is exposed. The built-in endpoints are auto-configured only when they are available. Most applications choose exposure over HTTP, where the ID of the endpoint and a prefix of /actuator is mapped to a URL. For example, by default, the health endpoint is mapped to /actuator/health.

Reference site

<https://docs.spring.io/spring-boot/reference/actuator/endpoints.html>