Spring Boot Actuator

Help you monitor and manage your application when you push it to production.

Spring Boot Actuator

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.

Spring Boot Actuator

- Spring Boot includes a number of 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.

Maven Dependency

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

Endpoints

- Actuator endpoints let you monitor and interact with your application.
- By default the http actuators are not enable except **health** and **info**.
- JMX is inherently more secure than http
- Spring boot exposes more endpoints by default through JMX actuator
- Each individual endpoint can be enabled or disabled. This controls whether or not the endpoint is created and its bean exists in the application context.
- All actuators are now under **/actuator** base path from boot 2

Endpoints

- 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.
  - auditevents , beans , caches , conditions , conifgprops
  - env , flyway , health , httptrace , info , integrationgraph
  - loggers , liquibase , metrics , mappings
  - scheduledtasks , sessions , shutdown , threaddump

Controlling Endpoints

- Each individual endpoint can be enabled or disabled.
- This controls whether or not the endpoint is created and its bean exists in the application context.
- To be remotely accessible an endpoint also has to be exposed via JMX or HTTP.
- Most applications choose HTTP, where the ID of the endpoint along with a prefix of /actuator is mapped to a URL.
- For example, by default, the health endpoint is mapped to /actuator/health.

Controlling Endpoints

- Configuring actuator base path:
  - **mangement.endpoints.web.base-path=/admin**
- By Default all the endpoints are enabled except shutdown endpoint , but their expose rules are different.

- To change which endpoints are exposed, use the following technology-specific include and exclude properties:
  - **management.endpoints.jmx.exposure.exclude**
  - **management.endpoints.jmx.exposure.include**
  - **management.endpoints.web.exposure.exclude**
  - **management.endpoints.web.exposure.include**

Controlling Endpoints

- Exposing actuator endpoints over http :
  - **management.endpoints.web.exposure.include=env,beans,health,info**
  - **management.endpoints.web.exposure.include=***
  - **management.endpoints.web.exposure.exclude=beans**

Controlling Endpoints

- **Enabling Endpoints**
  - By default, all endpoints except for shutdown are enabled. To configure the enablement of an endpoint, use its **management.endpoint.<id>.enabled** property.
  - **management.endpoint.shutdown.enabled=true**
  - **shutting down application using shutdown endpoint**

    curl –X POST
    http://localhost:8080/actuator/shutdown
  - **management.endpoints.enabled-by-default=false**
  - **management.endpoint.info.enabled=true**

Controlling Endpoints

- Disabled endpoints are removed entirely from the application context.
- If you want to change only the technologies over which an endpoint is exposed, use the include and exclude properties instead.

- health endpoint by default just shows the status of the application as UP ,

- To view more health specific details configure the following commands.
  - **management.endpoint.health.show-details=always**

Custom Endpoints

Custom Endpoint with Spring Boot 2.x

- Spring Boot 2 provides an easy way to create custom endpoints. Spring Boot 2.x introduced @Endpoint annotation.

- Spring Boot automatically expose endpoints with @Endpoint, @WebEndpoint, or @WebEndpointExtension over HTTP using Jersey, Spring MVC, or Spring WebFlux.

- Spring Boot 2.x Actuator support CURD model, it supports read, writes and delete operation with the endpoints.

- The @Endpoint annotation can be used in combination with @ReadOperation,@WriteOperation and @DeleteOperation to develop endpoints.

Custom Endpoints

- Actuator Endpoints
  - Endpoint annotations
    - @Endpoint, @WebEndpoint, @JmxEndpoint
  - Tech independent operations
    - @ReadOperation, @WriteOperation, @DeleteOperation
  - Tech specific extensions
    - @EndpointWebExtension, @EndpointJmxExtension