Spring Boot Actuator: Monitoring, Customization, and Security in Spring Boot Applications

Spring Boot Actuator provides a powerful set of tools to monitor and manage your Spring Boot application by exposing various system and application metrics, health information, and other details through HTTP endpoints. This guide will take you through the process of integrating Spring Actuator, exploring default and custom endpoints, securing them.

Overview

Key Objectives:

- **Enable and configure Actuator** in your Spring Boot application.
- **Explore default endpoints** for health, metrics, and environment data.
- **Create custom Actuator endpoints** to expose specific application metrics.
- **Secure sensitive Actuator endpoints** using authentication and authorization.

Step 1: Enabling Spring Actuator

Configure Actuator:

Once the dependency is added, configure Actuator in the `application.properties` file to expose specific endpoints.

```
# Enable all Actuator endpoints
management.endpoints.web.exposure.include=*

# Show health details always
management.endpoint.health.show-details=always
```

This configuration ensures that all Actuator endpoints are exposed and that detailed health information is available at `/actuator/health`.

Step 2: Exploring Default Actuator Endpoints

Once your application is running, you can access default Actuator endpoints to retrieve valuable information about your application's health, metrics, and more.

Common Endpoints:

- **Health Check**: /actuator/health
  - Shows the health status of the application (e.g., UP, DOWN).
- **Metrics**: /actuator/metrics
  - Displays various metrics such as memory usage, CPU load, request counts, and more.
- **Environment**: /actuator/env
  - Provides details about the application environment, including system properties and environment variables.

For example:

```
http://localhost:8084/actuator/metrics/jvm.memory.used
```

This endpoint will show you the amount of memory currently being used by the JVM.

Step 3: Creating Custom Actuator Endpoints

Spring Actuator allows you to create custom endpoints to expose specific application data, making it highly customizable.

Creating a Custom Endpoint:

Here's an example of how to create a custom endpoint that returns the count of active tasks:

```
import org.springframework.boot.actuate.endpoint.annotation.Endpoint;
import org.springframework.boot.actuate.endpoint.annotation.ReadOperation;
import org.springframework.stereotype.Component;

@Component
@Endpoint(id = "tasks")
public class TaskEndpoint {

   private final TaskService taskService;

   public TaskEndpoint(TaskService taskService) {
      this.taskService = taskService;
   }
```

```
    @ReadOperation
    public int getActiveTasks() {
        return taskService.getAllTasks().size();
    }
}
```

Accessing the Custom Endpoint:

Once the endpoint is created, you can access it via:

```
http://localhost:8084/actuator/tasks
```

This will return the count of active tasks in your application.

Step 4: Securing Actuator Endpoints

Since Actuator exposes sensitive information about your application, it's important to secure these endpoints using **Spring Security**.

Adding Spring Security:

Securing Actuator Endpoints:

Create a security configuration class to require authentication for accessing Actuator endpoints:

```
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapt
er;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
            .authorizeRequests()
            .antMatchers("/actuator/**").authenticated()  // Secure all Actuator endpoints
            .and()
            .httpBasic();  // Enable basic authentication
    }
}
```

Configure User Credentials:

In application.properties, you can set custom user credentials for basic authentication:

```
spring.security.user.name=admin
spring.security.user.password=secret
```

Now, all Actuator endpoints will be protected, and you will need to enter the username (`admin`) and password (`secret`) to access them.