SPRING BOOT APPLICATION WITH ACTUATOR AND PROMETHEUS INTEGRATION

Sandra Kumi

REVIEWER Mr. Thomas Darko

Spring Boot Application with Actuator and Prometheus Integration

1. **Overview**

   This documentation provides an overview of the Spring Boot application that integrates Spring Actuator for monitoring and Prometheus for external metric collection. The application includes custom endpoints, secured access (optional), and integration with Prometheus for monitoring and alerting purposes.

2. Spring Actuator Features

Spring Boot Actuator provides production-ready features such as monitoring, metrics, and health checks. It exposes various HTTP endpoints to monitor and interact with the application.

2.1 Enabled Actuator Endpoints

Actuator includes a variety of endpoints out of the box. Some common endpoints are:

- /actuator/health: Provides application health status.
- /actuator/info: Displays arbitrary application information.
- /actuator/metrics: Displays various application metrics.
- /actuator/env: Shows environment properties.

These endpoints are customizable and can be enabled or disabled via the application.properties or application.yml file.

Example configuration in application.properties:

```
management.endpoints.web.exposure.include=health,info,metrics,env

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

2.2 Custom Actuator Endpoints

In this project, a custom Actuator endpoint is created to provide additional information about the application's state.

```java
import org.springframework.boot.actuate.endpoint.annotation.Endpoint;

import org.springframework.boot.actuate.endpoint.annotation.ReadOperation;

import org.springframework.stereotype.Component;


@RestController

@Endpoint(id = "customInfo")

public class CustomActuatorEndpoint {

   @ReadOperation

   public String getCustomInfo() {

      return "Application is running smoothly!";

   }

}
```

This custom endpoint is available at /actuator/customInfo.

2.3 Securing Actuator Endpoints (Optional)

For security, sensitive Actuator endpoints can be secured to prevent unauthorized access. This is especially important in production environments.

```properties
management.endpoints.web.exposure.include=health,info,metrics,env,customInfo

management.endpoint.health.show-details=always


# Security configuration

spring.security.user.name=admin

spring.security.user.password=password
```

```
management.endpoint.health.roles=ADMIN
```

3. Prometheus Integration

Prometheus is an open-source monitoring system often used in conjunction with Actuator to gather metrics from a Spring Boot application.

3.1 Dependency

To integrate Prometheus with Spring Boot, add the following dependency to your pom.xml file:

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

3.2 Configuration

Once the Prometheus dependency is added, metrics are automatically exposed at the /actuator/prometheus endpoint.

To ensure Prometheus metrics are available, include the following configuration:

```
management.endpoints.web.exposure.include=Prometheus
```

3.3 Setting Up Prometheus

Prometheus collects metrics by scraping the /actuator/prometheus endpoint. The following is a sample Prometheus configuration for scraping metrics from a Spring Boot application:

```
scrape_configs:
  - job_name: 'spring-actuator'
  metrics_path: '/actuator/prometheus'

  static_configs:

    - targets: ['localhost:8080']
```

4. Best Practices for Using Spring Actuator and Prometheus

4.1 Actuator Best Practices

- Limit exposed endpoints: Only expose endpoints necessary for monitoring and troubleshooting.
- Secure sensitive endpoints: Use Spring Security to restrict access to endpoints like /actuator/env and /actuator/metrics.
- Customize health checks: Implement custom health indicators to monitor the health of critical components like databases, external services, etc.

4.2 Prometheus Best Practices

- Scrape interval: Configure the appropriate scrape interval in Prometheus to balance load and responsiveness.
- Alerting: Define alert rules in Prometheus to trigger notifications when certain conditions are met (e.g., high memory usage, slow response times).
- Data retention: Configure Prometheus to retain only the necessary amount of historical data to prevent large storage consumption.

5. Conclusion

This Spring Boot application demonstrates the power of Spring Actuator for monitoring and provides a foundation for integrating external monitoring tools like Prometheus. Actuator endpoints offer real-time insights into the health, metrics, and performance of the application, while Prometheus enables long-term data storage and alerting.

With these tools, you can ensure that your Spring Boot application is both observable and responsive to changes in production.

---

6. References

- Spring Boot Actuator Documentation
- Prometheus Documentation