Names: Nadege Gaju

Spring Actuator Lab

What is Spring Actuator?

Spring Actuator is an extension of the Spring Boot framework, designed to help developers monitor and manage their applications with ease. It provides a range of ready-made features that expose details about the state and behavior of your application, including its health, metrics, and environment settings. This makes it especially useful in production environments, where monitoring and control are crucial.

Key Features of Spring Actuator

1. Health Checks

   - ✓ Health Endpoint: This endpoint provides a quick overview of your application's current health. For example, it can show whether critical services like the database are accessible, or whether disk space is running low.
   - ✓ Custom Health Indicators: You can add custom health checks based on specific components of your application. For instance, if you have a custom caching layer, you can create a health indicator that checks whether the cache is functioning as expected.

2. Metrics Collection

   - ✓ Metrics Endpoint: This feature gives you access to detailed performance metrics of your application, such as how many HTTP requests it's handling, memory usage, and JVM metrics like garbage collection activity. These metrics are invaluable for performance tuning.
   - ✓ Custom Metrics: Beyond the default metrics, Spring Actuator lets you define and expose your own application-specific metrics. For example, you could track how many users are logged in at any given time or how many transactions are processed per second.

3. Application Info

   - ✓ Info Endpoint: This provides metadata about the application itself, such as the version, build information, and any additional information you might want to expose. This can be useful for operations teams who need to know which version is currently deployed.

- ✓ Custom Info Contributors: If the default info isn't enough, you can add custom details to the /info endpoint. This can be tailored to your needs, for example, including team contact details or release notes.

4. Environment Properties

   - ✓ Environment Endpoint: This endpoint provides access to the environment properties and configuration values used by your application. It's handy for checking what settings the app is running with in production (e.g., active profiles, database URLs), without having to dig through configuration files.

5. Loggers

   - ✓ Loggers Endpoint: This gives you control over the logging levels while the application is running. If you need to troubleshoot a problem in production, you can adjust the logging level (e.g., from INFO to DEBUG) on the fly without restarting the app.

6. Thread Dump

   - ✓ Thread Dump Endpoint: This provides a snapshot of all active threads in the JVM. It's helpful for diagnosing performance issues, such as thread contention or deadlocks, by seeing what each thread is doing at a given time.

7. Custom Endpoints

   - ✓ Custom Endpoints: Spring Actuator allows you to create your own endpoints to expose specific information that is unique to your application. These custom endpoints can serve as an operational dashboard for monitoring specialized aspects of your system.

Best Practices

1. Securing Endpoints: Ensure that sensitive endpoints, such as those exposing environment variables or thread dumps, are secured. In production, these endpoints should require authentication to prevent unauthorized access, as they may expose critical internal information.

2. Using Custom Metrics: Defining your own application-specific metrics helps you understand your app's behavior in detail. For example, you can track user activity, transaction rates, or error counts, all of which help fine-tune performance and spot potential problems early.

3. Regular Monitoring: It's a good idea to regularly review your Actuator endpoints to ensure your application remains healthy. Setting up alerts on key metrics or health checks can help catch problems before they affect users.
4. Enable Endpoints in Production: In a production environment, enable Actuator's management endpoints to allow for real-time tracking of your app's health and performance. However, ensure that access to these endpoints is secure and only available to authorized personnel.
5. Document Custom Endpoints: If you create custom endpoints, make sure to document them clearly. This helps other team members understand what data these endpoints provide and how they can be used in production.
6. Pay Attention to Privacy and Security: Be mindful of the information exposed by Actuator, especially in production. Avoid exposing sensitive data (e.g., API keys, credentials) and use proper access control to prevent unauthorized access.

Example Use Cases

1. Monitoring Application Health: By using the /actuator/health endpoint, you can continuously monitor the key components of your application, like database connections or messaging queues, and set up alerts to notify you of issues before they escalate.
2. Collecting Metrics: Through the /actuator/metrics endpoint, you can collect various performance metrics over time and feed them into monitoring tools like Prometheus or Grafana. This helps you visualize trends and optimize performance.
3. Custom Endpoints: Let's say your application handles online orders. You could create a custom endpoint that tracks order success rates or average processing times, giving you an operational overview of your app's performance at any given moment.