Spring Boot Actuator

Spring Boot ships with a module called actuator which enables things like metrics and statistics about the application.

For example, we can collect logs, view metrics, perform thread dumps, show environment variables, understand garbage collection, and show what beans are configured in the BeanFactory.

We can expose this information via HTTP, JMX, or we can even log in directly to the process via SSH.

Enable actuator

To start using the existing actuators in Boot – we'll just need to add the spring-boot-actuator dependency to the pom:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
  <version>1.4.2.RELEASE</version>
</dependency>
```

Customizing the management server port

Exposing management endpoints using the default HTTP port is a sensible choice for cloud based deployments.

However, we can expose a separate management HTTP port.

The management.port property can be used to change the HTTP port.

management.port=8081

Security

Most of the times, the details exposed via the endpoints are sensitive and requires authorized user to view the details. We can add spring security to secure your endpoints.

Just add the dependency, when the spring security files are available in the classpath, it will be automatically configured.

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

application.properties

```
management.security.enabled=true
security.basic.enabled=true
security.user.name=admin
security.user.password=admin
```

Actuator Endpoint

Provides a hypermedia-based “discovery page” for the other endpoints.
Requires Spring HATEOAS to be on the classpath.

```
<dependency>
      <groupId>org.springframework.hateoas</groupId>
      <artifactId>spring-hateoas</artifactId>
      <version>0.23.0.RELEASE</version>
   </dependency>
```

Actuator Endpoints

Here are some of the most common endpoints Boot provides out of the box:

/health – Shows application health information (a simple ‘status’ when accessed over an unauthenticated connection or full message details when authenticated). It is not sensitive by default.

/info – Displays arbitrary application info. Not sensitive by default.

/metrics – Shows ‘metrics’ information for the current application. It is also sensitive by default.

/trace – Displays trace information (by default the last few HTTP requests). We can find the full list of existing endpoints over on the official docs.

Metrics Endpoint

The metrics endpoint is one of the more important endpoints as it gathers and publishes information about OS, JVM and Application level metrics; out of the box, we get things like:

memory, heap, processors, threads, classes loaded, classes unloaded, thread pools along with some HTTP metrics as well.

Customizing Endpoints

Each endpoint can be customized with properties using the following format:
endpoints.[endpoint name].[property to customize]

Three properties are available:

id – by which this endpoint will be accessed over HTTP
enabled – if true then it can be accessed otherwise not
sensitive – if true then need the authorization to show crucial information over HTTP.

For example, add the following properties will customize the /beans endpoint:

endpoints.beans.id=springbeans
endpoints.beans.sensitive=false
endpoints.beans.enabled=true

Custom Health Endpoint

```
@Component
public class CustomHealthCheck implements HealthIndicator {
        public Health health() {
            int errorCode = 0;
            if (errorCode != 1) {
                return Health.down().withDetail("Error Code",
errorCode).build();
            }
            return Health.up().build();
        }

    }
```