Spring Boot Actuator

Production-Ready Features in Spring Applications

Outline

- Getting Started & Endpoints
- Health Indicators
- HTTP Access
- Metrics

Spring Boot Actuator

Overview

- Enable Out-of-the-box Production-Ready Features in Spring Apps
- Run-time Inspection of Configuration Details
  - Beans, Environment, Auto-Configuration, Config Properties
- Monitor Application Status & Behavior
  - Health Checks, Metrics
- Simple Instrumentation
  - Graceful Shutdown
- Several Endpoint/Access-Point Types
  - HTTP
  - JMX
  - Remote Shell
- Security Aware

Spring Boot Actuator

Enabling w/ Dependency

Import Maven Dependency

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

Actuator HTTP Endpoints

App Configuration

Auto-Config Candidates:

```
GET /autoconfig
```

Spring Managed Beans:

```
GET /beans
```

Environment:

```
GET /env
```

@ConfigurationProperties:

```
GET /configprops
```

Actuator HTTP Endpoints

App Info

Health Info:

```
GET /health
```

General App Info:

```
GET /info
```

Log file:

```
GET /logfile
```

property: ***`logging.file`*** / ***`logging.path`***

Metrics:

```
GET /metrics
```

Actuator HTTP Endpoints

MVC & Web Apps

MVC @RequestMapping HTTP Endpoints:

```
GET /mappings
```

Trace (HTTP Requests):

```
GET /trace
```

Actuator HTTP Endpoints

Other

Gracefully shutdown:

```
POST /shutdown
```

Database migrations:

```
GET /flyway
GET /liquibase
```

Actuator Endpoints

Enabling/Disabling

Enabling/Disabling Individual Endpoints:

application.properties

```
endpoints.shutdown.enabled=true
endpoints.mappings.enabled=false
endpoints.trace.enabled=false
```

Global Enabling/Disabling Endpoints:

```
endpoints.enabled=false
endpoints.info.enabled=true
```

Enabling/Disabling Endpoints HTTP Access:

```
management.port=-1
```

Enabling/Disabling Endpoints JMX Access:

```
endpoints.jmx.enabled=false
```

Actuator Endpoints

Configuration

Changing Security Requirements (Sensitivity)

```
endpoints.shutdown.sensitive=true
endpoints.mappings.sensitive=false
endpoints.trace.sensitive=false
```

Renaming Endpoints

```
endpoints.beans.id=springbeans
endpoints.trace.id=httprequests
endpoints.trace.logfile=log
```

Outline

- Getting Started & Endpoints
- Health Indicators
- HTTP Access
- Metrics

Health Endpoint

Built-in Health Indicators

Health Indicator	Performed Checks
`DiskSpaceHealthIndicator`	Checks for low Disk space
`DataSourceHealthIndicator`	Check DataSource connection
`ElasticsearchHealthIndicator`	Checks ElasticSearch cluster is up
`JmsHealthIndicator`	Checks that a JMS broker is up
`MailHealthIndicator`	Checks that a mail server is up
`MongoHealthIndicator`	Checks that a Mongo database is up
`RabbitHealthIndicator`	Checks that a Rabbit server is up
`RedisHealthIndicator`	Checks that a Redis server is up
`SolrHealthIndicator`	Checks that a Solr server is up

Health Endpoint

Built-in Indicators Example

Unsecured Access

```
curl localhost:8080/health
```

```
{ "status" : "UP" }
```

Disk Health

```
curl -u user:pa$s localhost:8080/health
```

```
{
    "status": "UP",
    "diskSpace": {
        "status": "UP",
        "total": 733921406976,
        "free": 268554510336,
        "threshold": 10485760
    }
}
```

Health Endpoint

Built-in Indicators Example

Rabbit Health - **DOWN**

```
curl -u user:pa$s localhost:8080/health
```

```
{
    "status": "DOWN",
    { ... },
    "rabbit": {
        "status": "DOWN",
        "error": "org.springframework.amqp.AmqpConnectException:
        java.net.ConnectException: Connection refused: connect"
    }
}
```

Health Endpoint

Built-in Indicators Example

Rabbit Health - **UP**

```
rabbitmq-server
```

```
              RabbitMQ 3.4.1. Copyright (C) 2007-2014 GoPivotal, Inc.
  ##  ##      Licensed under the MPL.  See http://www.rabbitmq.com/
  ##  ##
  ##########  Logs: C:/Users/MyUser/AppData/Roaming/RabbitMQ/log/rabbit@mypc.log
  ######  ##        C:/Users/MyUser/AppData/Roaming/RabbitMQ/log/rabbit@mypc.log
  ##########
              Starting broker... completed with 12 plugins.
```

```
curl -u user:pa$s localhost:8080/health
```

```
{
    "status": "UP",
    { ... },
    "rabbit": {
        "status": "UP",
        "version": "3.4.1"
    }
}
```

Health Endpoint

Custom Health Indicator

Spring Managed Component Implementing **`HealthIndicator`**

```
import org.springframework.boot.actuate.health.HealthIndicator;
```

```
@Component
public class MyServiceHealth implements HealthIndicator {

    @Autowired
    private MyService service;


    @Override
    public Health health() {
        int errorCode = service.checkStatus();
        return (errorCode != 0) ?
            Health.down().withDetail("Error Code", errorCode).build() :
            Health.up().build();
    }
}
```

Outline

- Getting Started & Endpoints
- Health Indicators
- HTTP Access
- Metrics

HTTP Access

Securing Sensitive Endpoints

Importing Spring Security Dependency

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

```
Using default security password: c654eed6-b877-4494-8fb2-8e9fb21f00c4
```

Setting Required Authentication Credentials & Authorization Roles

```
security.user.name=admin
security.user.password=admin$ecret
management.security.role=ADMIN
```

Disable All Security Checks for Actuator Endpoints (behind firewall)

```
management.security.enabled=false
```

HTTP Access

Securing Endpoints Example

Failed Authentication

```
curl  localhost:8080/beans
```

```
{"timestamp":1444654289352,"status":401,"error":"Unauthorized","message":
"Full authentication is required to access this resource","path":"/beans"}
```

CURL -u user:pass /beans

```
curl -u user:c654eed6-b877-4494-8fb2-8e9fb21f00c4 localhost:8080/beans
```

```
[{"context":"application","parent":null,"beans":[{"bean":
"boot0xActuatorSolutionApplication", "scope":"singleton",
"type":"boot.Boot0xActuatorSolutionApplication$$...",
"resource":"null","dependencies":[]}, … ]
```

```
security.user.name=admin
security.user.password=admin$ecret
```

```
curl -u admin:admin$secret localhost:8080/beans
```

Outline

- Getting Started & Endpoints
- Health Indicators
- HTTP Access
- Metrics

Boot Actuator Metrics

Overview

- Capture & Expose Scalar Metrics on System/App State & History
  - Counters - integer “event” count (inc&dec operations)
  - Gauge - numeric current state measure
  - Custom - Arbitrary metrics information (interface PublicMetrics)
- Many Built-in Metrics
  - System Metrics - Processor & Threads, Memory
  - DataSource Metrics
  - Cache Metrics
  - Web Request Metrics - Request counts grouped by Status Code
  - Web Session Metrics - Open/Max Session Counts
- Metrics Exportable to External DB/Service
  - Redis, Open TSDB, Statsd, JMX, Dropwizard

Built-in Metrics

System Metrics

Processor, App Instance, Threads, ClassLoader

`processors`	Processor count
`uptime`	System uptime (milliseconds)
`instance.uptime`	ApplicationContext uptime (milliseconds)
`systemload.average`	Average system load
`threads`	Thread count
`thread.peak`	
`thread.daemon`	Background (Daemon) Thread count
`classes`	Count of loaded Classes
`classes.loaded`	Current count of loaded Classes
`classes.unloaded`	Count of unloaded Classes

Built-in Metrics

System Metrics

Memory, Heap, Garbage Collection

`mem`	Total system memory (KB)
`mem.free`	Free memory (KB)
`heap`	Heap Size (KB)
`heap.committed`	
`heap.init`	
`heap.used`	
`gc.xxx.count`	Garbage collection information
`gc.xxx.time`	

Boot Actuator Metrics

System Metrics Example

CURL /metrics

```
curl  localhost:8080/metrics
```


```
{
    "mem": 331776,
    "mem.free": 251566,
    "processors": 4,
    "instance.uptime": 153727,
    "uptime": 189397,
    "systemload.average": -1,
    "heap.committed": 331776,
    "heap.init": 90112,
    "heap.used": 80209,
    "heap": 1269248,
    "threads.peak": 24,
    "threads.daemon": 20,
    "threads": 24,
    "classes": 8358,
    "classes.loaded": 8359,
    "classes.unloaded": 1,
    "gc.ps_scavenge.count": 11,
    "gc.ps_scavenge.time": 497,
    "gc.ps_marksweep.count": 2,
    "gc.ps_marksweep.time": 624,
    "httpsessions.max": -1,
    "httpsessions.active": 0,
    "gauge.response.health": 12418,
    "gauge.response.info": 17,
    "counter.status.200.info": 1,
    "counter.status.503.health": 1
}
```

Built-in Metrics

DataSource Metrics

General Naming Pattern

`datasource.xxx.active`	Active connections count
`datasource.xxx.usage`	Connection pool current usage (%)

DataSource Bean Name & Qualifiers - Mapping

`datasource.primary.active` `datasource.primary.usage`	Qualifier Annotation: @Primary
`datasource.products.active` `datasource.products.usage`	Bean name: productsDataSource
`datasource.imageStore.active` `datasource.imageStore.usage`	Bean name: imageStore

Actuator Metrics

Recording Custom Metrics

Using CounterService & GaugeService API

```
import org.springframework.boot.actuate.metrics.CounterService;
import org.springframework.boot.actuate.metrics.GaugeService;
```

```
@Service
public class MyLoginService {
    @Autowired
    private final CounterService counterService;

    @Autowired
    private final GaugeService gaugeService;

    public void userLogin(String username) {
        this.counterService.increment("services.myapp.login.count");
        this.gauceService.submit("services.myapp.login.time." + username,
            System.currentTimeMillis());
    }

    public void userLogout(String username) {
        this.counterService.decrement("services.myapp.login.count");
        this.gauceService.submit("services.myapp.login.exit." + username,
            System.currentTimeMillis());
   }
}
```