## FlightService Actuator

Actuator -- endpoints (info, health, metrics, ….

POM / Starter

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

## Simple Metrics Registry

Metrics

<dependency>

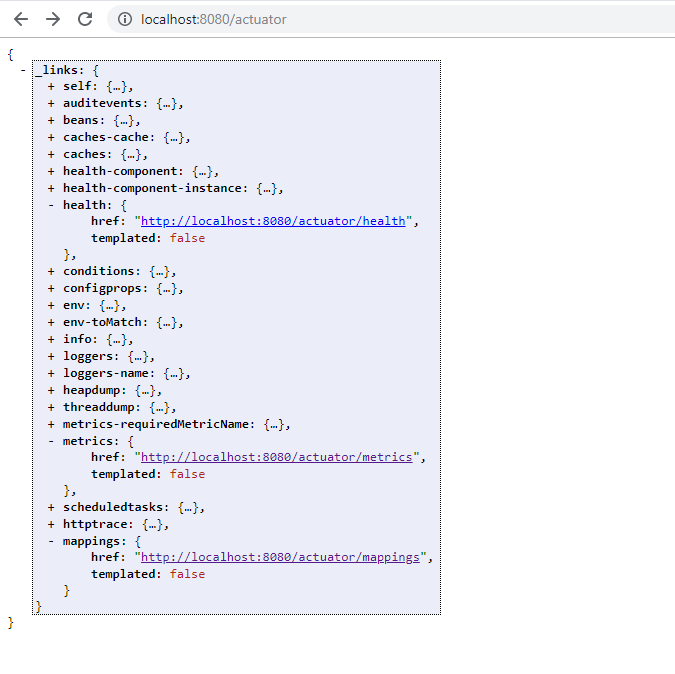
<groupId>io.micrometer</groupId>

<artifactId>micrometer-core</artifactId>

</dependency>

Application Properties

management.endpoints.web.exposure.include=\* --- or specific list (info, ….)



Per default, Spring instantiates a SimpleMeterRegistry Bean.

Metrics Types:

* Counter, only incremental
* Gauge, used for range/periods
* Timers
* DistributionSummary, tracks the sample distribution of events, e.g. response sizes for requests

Define FlightServiceCounter component for counting Route Requests:

@Component

**public** **class** FlightServiceCounter {

**private** MeterRegistry meterRegistry;

**private** Counter routeRequestCount;

**public** FlightServiceCounter(MeterRegistry meterRegistry) {

**this**.meterRegistry = meterRegistry;

**this**.initializeCounters();

}

**void** initializeCounters () {

routeRequestCount = Counter

.*builder*("routeRequests")

.description("counts number of routes query requests")

.tags("clients", "performance")

.register(**this**.meterRegistry);

}

**public** **double** incrementRouteRequestCountCounter() {

routeRequestCount.increment();

**return** routeRequestCount.count();

}

}

Count number of Route requests in class RouteQueryResolver:

RootQueryResolver:

@Autowired

FlightServiceCounter fsCounter;

…

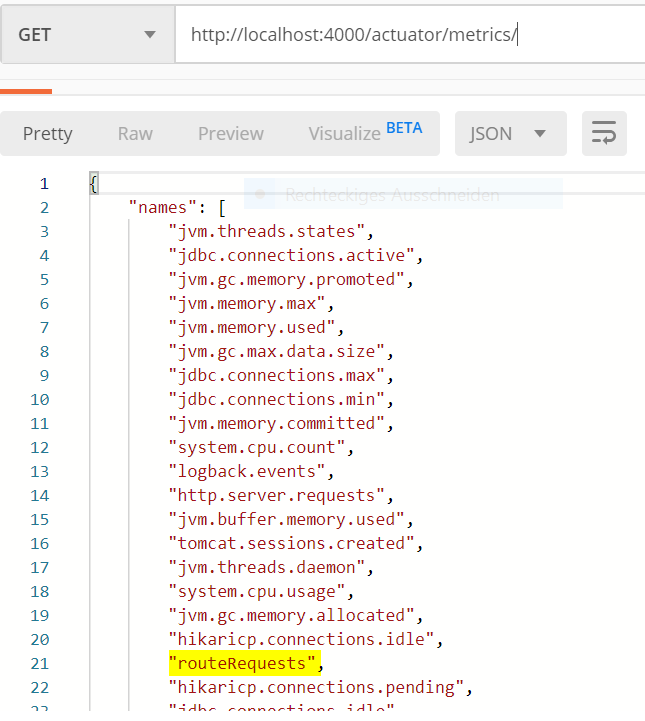
**public** List<Route> routes() {

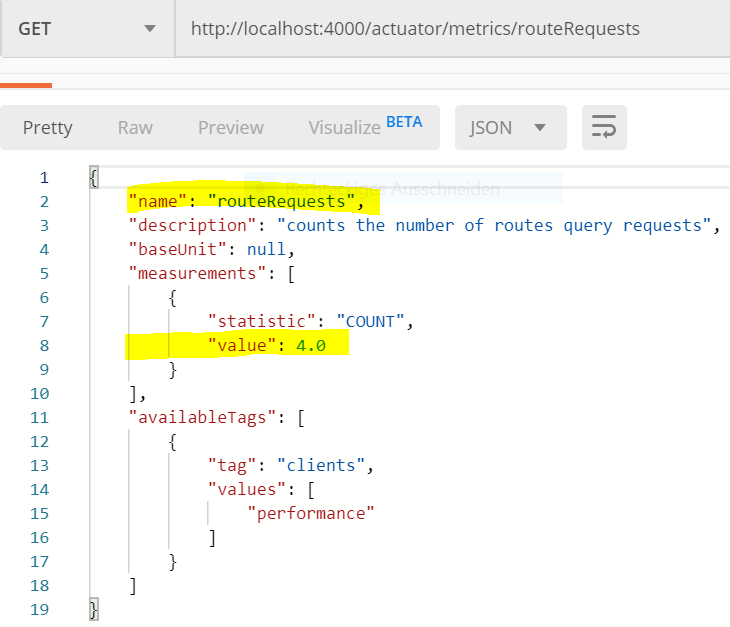
fsCounter.incrementRouteRequestCountCounter();

…

}

List all metrics by name:





## Prometheus

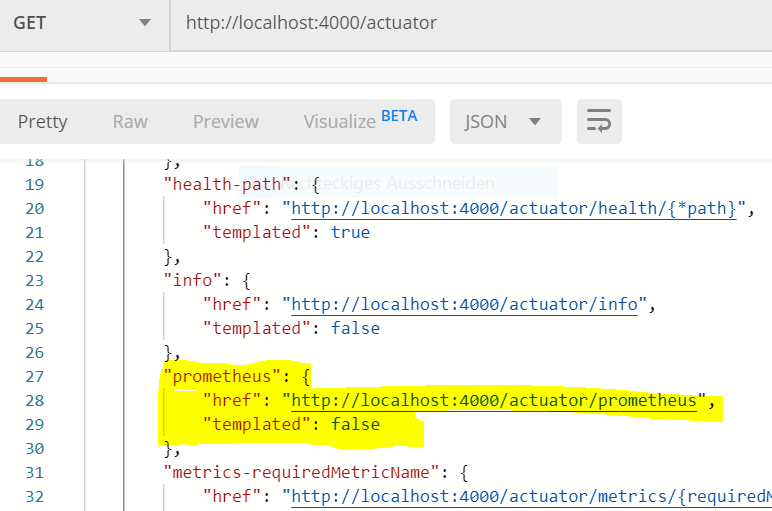
Starter:

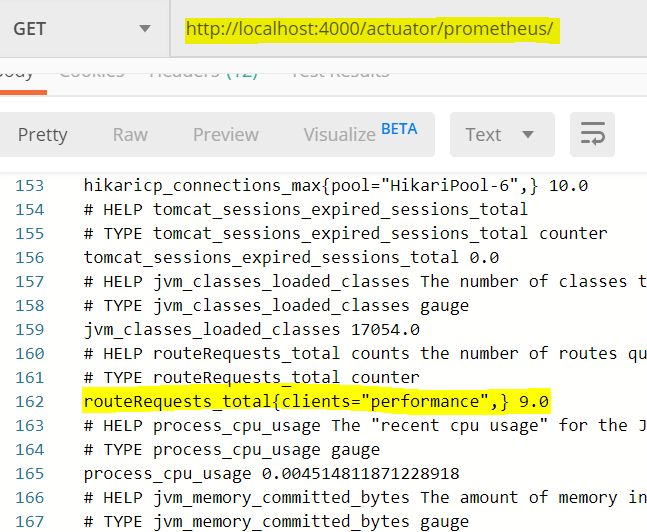
<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-registry-prometheus</artifactId>

</dependency>





## Prometheus Configuration File: (prometheus.yml)

# A scrape configuration scraping a Node Exporter and the Prometheus server

# itself.

global:

scrape\_interval: 10s

scrape\_configs:

- job\_name: 'spring\_micrometer'

metrics\_path: '/actuator/prometheus'

scrape\_interval: 5s

static\_configs:

- targets: ['10.0.75.1:4000'] -- ip address of machine running flight service application

## Prometheus Docker Container:

Run container using prometheus.yml file.

Prometheus Dashport endpoint: 9090

docker run -d -p 9090:9090 -v ~/prometheus.yml:/etc/prometheus/prometheus.yml prom/Prometheus

### Prometheus Dashboard

