Micrometer concepts

作者：devcui
原文链接：https://ld246.com/article/1618368544872
来源网站：链滴

1.什么是 Micrometer

- 针对JVM Application的指标库工具
- SPI 服务商应用接口

2.支持的服务商

- AppOptics
- Atlas
- Azure Monitor
- Cloudwatch
- Datadog
- Datadog StatsD
- Dynatrace
- Elastic
- Humio
- Influx
- KairosDB
- New Relic
- Prometheus
- SignalFx
- Sysdig StatsD
- Telegraf StatsD
- Wavefront

3.注册 Registry

- Meter是测量你的应用程序指标的一个集合
- Meter在Micrometer中被MeterRegistry创建并保存
- 每个受支持的监控服务商都有一套对应的MeterRegistry实现
- SimpleMeterRegistry可以将数据保存在内存中,不会将数据导出到任何地方
- 假如没有首选监控系统，可以从SimpleMeterRegistry开始
- MeterRegistry registry = new SimpleMeterRegistry();
- SimpleMeterRegistry可以在Springboot中直接注入

3.1 Composite registries

- 提供了CompositeMeterRegistry，可以添加多个注册表
- 通过CompositeMeterRegistry可以将指标推送到多个监控系统

原文链接：Micrometer concepts

```
// 声明一个复合注册表
CompositeMeterRegistry composite = new CompositeMeterRegistry();
// 从复合注册表声明一个名为counter的计数器
Counter compositeCounter = composite.counter("counter");
// 计数器增1
compositeCounter.increment();
// 简单注册表
SimpleMeterRegistry simple = new SimpleMeterRegistry();
// 将简单注册表加入复合注册表中
composite.add(simple); (2)
// 简单注册表和复合注册表的计数器都会累加1
compositeCounter.increment();
```

3.2 Global registry

- 提供一个全局静态的注册表以及一组用于基于此注册表生成仪表的静态生成器

```
class MyComponent {
    Counter featureCounter = Metrics.counter("feature", "region", "test");

    void feature() {
        featureCounter.increment();
    }

    void feature2(String type) {
        Metrics.counter("feature.2", "type", type).increment();
    }
}

class MyApplication {
    void start() {
        // wire your monitoring system to global static state
        Metrics.addRegistry(new SimpleMeterRegistry());
    }
}
```

4.Meters

- Micrometer 提供了仪表集合
- Timer
- Counter
- Gauge
- DistributionSummary
- LongTaskTimer
- FunctionCounter
- FUnctionTimer
- TimeGauge
- 不同的仪表类型导致不同数量的时间序列指标

原文链接：Micrometer concepts

5.Naming meters

- 用 . 分割最小单词
- 比如在代码中`registry.timer("http.server.requests")
- 传入prometheus就会变为 http_server_requests_duration_seconds

5.1 Tag naming

```
registry.counter("database.calls", "db", "users")
registry.counter("http.requests", "uri", "/api/users")
```

可以从tag推测出测量的指标数据表示的内容为推荐命名方法

5.2 Common tags

- 可以在注册表级别定义公共标签
- registry.config().commonTags("stack","prod","region","us-east-l")
- registry.config().commonTags(Arrays.asList(Tag.of("stack", "prod"), Tag.of("region", "us-east 1")))
- 如果在Spring环境中，请通过添加MeterRegistryCustomizer bean添加通用标签，以确保在自动置仪表绑定程序之前应用了通用标签。

5.3 Tag values

- value必须非空

6. Meter filters

- 过滤功能
- Deny
- Transform
- Configure

6.1 Deny/accept meters

```
new MeterFilter(){
    @Override
    public MeterFilterReply accept(Meter.Id id){
        if(id.getName().contains("test")){
            return MeterFilterReply.DENY;
        }
        return MeterFilterReply.NEUTRAL;
    }
}
```

- MeterFilterReply.DENY 请勿注册该仪表

原文链接：Micrometer concepts

- MeterFilterReply.NEUTRAL 如果没有其他仪表过滤器返回DENY，则仪表的注册将照常进行
- MeterFIlterReply.ACCEPT 如果过滤器返回“接受”，仪表将立即注册，而不会询问任何其他过滤的接受方法

6.11 Convenience methods

MeterFIlter 为拒绝/接受类型过滤器提供了几个方便的静态生成器

- accept()
- accept(Predicate<Meter.id>)
- acceptNameStartsWith(String)
- deny()
- deny(Predicate<Meter.id>)
- denyNameStartsWith(String)
- maximumAllowableMetrics(int) 达到一定数量后拒绝
- maximumAllowableTags(String meterNamePrefix,String tagKey,int maximumTagValues,MetrFilter) 匹配tag
- denyUnless(Predicate<Meter.Id>)

6.12 Chaining deny/accept meters

如下，该注册表中仅存名称包含http的指标

```
registry.config()
    .meterFilter(MeterFilter.acceptNameStartsWith("http"))
    .meterFilter(MeterFilter.deny());
```

6.2 Transforming metrics

该过滤器有条件地向仪表添加以名称“ test”开头的名称前缀和附加标签。

```
new MeterFilter() {
    @Override
    public Meter.Id map(Meter.Id id) {
      if(id.getName().startsWith("test")) {
        return id.withName("extra." + id.getName()).withTag("extra.tag", "value");
      }
      return id;
    }
}
```

- commonTags(Iterable<Tag>)
- ignoreTags(String...)
- replaceTagValues(String tagKey,Function<String,String>replcaement,String...exceptions)
- renameTag(String meterNamePrefix,String fromTagKey,String toTagKey)

原文链接：Micrometer concepts

6.3 Configuring distribution statistics

● 除了可以通过过滤器配置的计数，总计和最大值的基本知识外，Timer和DistributionSummary还含一组可选的分配统计信息。

● 这些分布统计信息包括预先计算的百分位数，SLA和直方图。

● 通常，您应该仅使用要配置的部分创建一个新的DistributionStatisticConfig，然后将其与输入配合并。

● 这使您可以下拉到注册表提供的分发统计信息的默认值，并将多个过滤器链接在一起，每个过滤器配置了分发统计信息的某些部分

```
new MeterFilter() {
    @Override
    public DistributionStatisticConfig configure(Meter.Id id, DistributionStatisticConfig config) {
        if (id.getName().startsWith(prefix)) {
            return DistributionStatisticConfig.builder()
                    .publishPercentiles(0.9, 0.95)
                    .build()
                    .merge(config);
        }
        return config;
    }
};
```

● maxExpected(Duration/long)

● minExpected(Duration/long)

7. Rate aggregation

pass

8. Counters

计数器报告一个指标，一个计数。计数器界面允许您以固定数量递增，该数量必须为正。

```
Counter counter = Counter
    .builder("counter")
    .baseUnit("beans") // optional
    .description("a description of what this counter does") // optional
    .tags("region", "test") // optional
    .register(registry);
```

8.1 Function-tracking counters

```
MyCounterState state = ...;

FunctionCounter counter = FunctionCounter
    .builder("counter", state, state -> state.count())
    .baseUnit("beans") // optional
    .description("a description of what this counter does") // optional
    .tags("region", "test") // optional
```

```
    .register(registry);
```

9. Guages

```
List<String> list = registry.gauge("listGauge", Collections.emptyList(), new ArrayList<>(), List::s
ze);
List<String> list2 = registry.gaugeCollectionSize("listSize2", Tags.empty(), new ArrayList<>());
Map<String, Integer> map = registry.gaugeMapSize("mapGauge", Tags.empty(), new HashM
p<>());
```

9.1 Manually incrementing/decrementing a Guage

```
// maintain a reference to myGauge
AtomicInteger myGauge = registry.gauge("numberGauge", new AtomicInteger(0));

// ... elsewhere you can update the value it holds using the object reference
myGauge.set(27);
myGauge.set(11);
```

9.2 Gauge fluent builder

```
Gauge gauge = Gauge
    .builder("gauge", myObj, myObj::gaugeValue)
    .description("a description of what this gauge does") // optional
    .tags("region", "test") // optional
    .register(registry);
```

9.3 Why is my guage reporting NaN or disapperaring

pass

9.4 Multi-gauge

```
// SELECT count(*) from job group by status WHERE job = 'dirty'
MultiGauge statuses = MultiGauge.builder("statuses")
    .tag("job", "dirty")
    .description("The number of widgets in various statuses")
    .baseUnit("widgets")
    .register(registry);

...

// run this periodically whenever you re-run your query
statuses.register(
    resultSet.stream()
        .map(result -> Row.of(Tags.of("status", result.getAsString("status")), result.getAsInt("coun
")))));
```

10. Timmers

- 计时器用于测量短时延以及此类事件的频率。

● Timer的所有实现至少将总时间和事件计数报告为单独的时间序列。

● 尽管可以将Timers用于其他用例，但请注意不支持负值，并且记录更长的持续时间可能会导致总时溢出Long.MAX_VALUE纳秒（292.3年）。

```
Timer timer = Timer
   .builder("my.timer")
   .description("a description of what this timer does") // optional
   .tags("region", "test") // optional
   .register(registry);
```

10.1 Recording blocks of code

● The Timer interface exposes several convenience overloads for recording timings inline

```
timer.record(() -> dontCareAboutReturnValue());
timer.recordCallable(() -> returnValue());

Runnable r = timer.wrap(() -> dontCareAboutReturnValue()); (1)
Callable c = timer.wrap(() -> returnValue());
```

10.2 Storing start state in Timer.Sample

● 您也可以将启动状态存储在一个示例实例中，以后可以将其停止。

● 该示例根据注册表的时钟记录开始时间。开始采样后，执行要计时的代码，并通过对采样调用stop Timer）来完成操作。

```
Timer.Sample sample = Timer.start(registry);

// do stuff
Response response = ...

sample.stop(registry.timer("my.timer", "response", response.status()));
```

10.3 The @Timed annotation

● 核心模块包含@Timed批注，框架可使用该批注为特定类型的方法（例如为Web请求端点提供服务方法）或通常为所有方法添加计时支持。

```
@Service
public class ExampleService {

  @Timed
  public void sync() {
    // @Timed will record the execution time of this method,
    // from the start and until it exits normally or exceptionally.
    ...
  }

  @Async
  @Timed
  public CompletableFuture<?> async() {
```

```
    // @Timed will record the execution time of this method,
    // from the start and until the returned CompletableFuture
    // completes normally or exceptionally.
    return CompletableFuture.supplyAsync(...);
  }

}
```

10.4 Function-tracking timers

- 提供了一种更不常用的计时器模式，该模式可跟踪两个单调递增的函数

```
IMap<?, ?> cache = ...

FunctionTimer.builder("cache.gets.latency", cache,
        c -> c.getLocalMapStats().getGetOperationCount(),
        c -> c.getLocalMapStats().getTotalGetLatency(),
        TimeUnit.NANOSECONDS)
    .tags("name", cache.getName())
    .description("Cache gets")
    .register(registry);
```

10.5 Pause detection

```
registry.config().pauseDetector(new ClockDriftPauseDetector(sleepInterval, pauseThreshold));
registry.config().pauseDetector(new NoPauseDetector());
```

10.6 Memory footprint estimation

pass

11. Distribution summaries

```
DistributionSummary summary = DistributionSummary
    .builder("response.size")
    .description("a description of what this summary does") // optional
    .baseUnit("bytes") // optional (1)
    .tags("region", "test") // optional
    .scale(100) // optional (2)
    .register(registry);
```

11.1 Scaling and histograms

```
DistributionSummary.builder("my.ratio")
   .scale(100)
   .sla(70, 80, 90)
   .register(registry)
```

11.2 Memory footprint estimation

pass

12. Long task timers

```
@Timed(value = "aws.scrape", longTask = true)
@Scheduled(fixedDelay = 360000)
void scrapeResources() {
    // find instances, volumes, auto-scaling groups, etc...
}

LongTaskTimer scrapeTimer = registry.more().longTaskTimer("scrape");
void scrapeResources() {
    scrapeTimer.record(() => {
        // find instances, volumes, auto-scaling groups, etc...
    });
}

LongTaskTimer longTaskTimer = LongTaskTimer
    .builder("long.task.timer")
    .description("a description of what this timer does") // optional
    .tags("region", "test") // optional
    .register(registry);
```

13. Histograms and percentiles

```
Timer.builder("my.timer")
  .publishPercentiles(0.5, 0.95) // median and 95th percentile
  .publishPercentileHistogram()
  .sla(Duration.ofMillis(100))
  .minimumExpectedValue(Duration.ofMillis(1))
  .maximumExpectedValue(Duration.ofSeconds(10))
```

基本的概念和例子过了一遍，现在写一下DEMO

首先引入依赖

```
 <dependency>
        <groupId>io.micrometer</groupId>
        <artifactId>micrometer-registry-prometheus</artifactId>
        <version>1.6.5</version>
    </dependency>
    <dependency>
        <groupId>io.micrometer</groupId>
        <artifactId>micrometer-core</artifactId>
        <version>1.6.5</version>
    </dependency>
```

然后搞一个配置类把指标加上commonTag

```
package com.yunzainfo.cloud.staruniverse.config;

import io.micrometer.core.instrument.MeterRegistry;
import org.springframework.beans.factory.annotation.Value;
```

```
import org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointPropertie
;
import org.springframework.boot.actuate.autoconfigure.metrics.MeterRegistryCustomizer;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;

import java.util.HashSet;
import java.util.Set;


@Configuration
public class MetricConfig {
    // 配置所有的指标前缀为
    // contextPath 去掉斜杠
    // 可以少配置一个applicationName
    @Bean
    MeterRegistryCustomizer<MeterRegistry> configurer(@Value("${server.servlet.context-pat
}") String contextPath) {
        return registry -> registry.config().commonTags("application", contextPath.replaceAll("/",
""));
    }
    // 这里主要是不想在改properties了，直接返回一个BEAN
    @Bean
    @Primary
    WebEndpointProperties asd() {
        WebEndpointProperties properties = new WebEndpointProperties();
        Set<String> includes = new HashSet<>();
        includes.add("*");
        properties.getExposure().setInclude(includes);
        return properties;
    }
}
```

好了，现在说一下go和java的exporter的区别

- 用go写的exporter是放出了一个url，Prometheus在时间间隔内访问我的exporter_url来scrape数据
- 用java就是　定时 查询指标，然后放在内存里，micrometer 集成了Springboot ，所以不用自定url/接口来返回数据进行刮取，他会把内存中的数据通过url放出去给prometheus收集

实现

```
package com.yunzainfo.cloud.staruniverse.service.impl;

import com.alibaba.fastjson.JSONObject;
import com.yunzainfo.cloud.staruniverse.dto.PersonDto;
import com.yunzainfo.cloud.staruniverse.service.MetricsTaskService;
import io.micrometer.core.instrument.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.context.event.ApplicationReadyEvent;
import org.springframework.context.event.EventListener;
import org.springframework.stereotype.Service;
import reactor.core.publisher.Flux;
```

```
import java.time.Duration;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;

@Service
public class MetricsTaskServiceImpl implements MetricsTaskService {

    @Autowired
    MeterRegistry meterRegistry;
    PersonDto personDto = new PersonDto();
    MultiGauge g;

    @EventListener(ApplicationReadyEvent.class)
    public void mixMetrics() {
        this.registerMetrics();
        Flux.interval(Duration.ofSeconds(5)).map(this::pushToHTTP).subscribe();
    }


    public void registerMetrics() {
        Counter.builder("simulation.people.request")
                .description("这是个计数器，只能加不能减")
                .tag("sex", "男")
                .tag("age", "30-50")
                .register(this.meterRegistry);


        // 注册可变动值需要保留引用
        personDto.setAge(Math.random());
        Gauge.builder("simulation.people.ages", personDto, personDto)
                .description("这是一个可以变动的数值")
                .register(this.meterRegistry);

        // 多个可变动的值
        g = MultiGauge.builder("simulation.people.statuses")
                .tag("job", "dirty")
                .description("The number of widgets in various statuses")
                .baseUnit("widgets")
                .register(this.meterRegistry);

    }

    public void setPeopleNumber() {
        // SELECT COUNT(*) FROM HTTP_REQUESTS WHERE SEX ='男' AND age >= 30 AND age
<=50
        meterRegistry.get("simulation.people.request").counter().increment(1);
    }

    public void setPeopleAge() {
        personDto.setAge(Math.random());
```

```
    }

    public void setStatus() {
        Set<PersonDto> hashSet = new HashSet<>();
        PersonDto p1 = new PersonDto();
        p1.setName("啊哈哈");
        p1.setAge(Math.random());
        p1.setStatus("健康");
        PersonDto p2 = new PersonDto();
        p2.setName("哇咔咔");
        p2.setAge(Math.random());
        p2.setStatus("虚弱");
        PersonDto p3 = new PersonDto();
        p3.setName("咦嘻嘻");
        p3.setAge(Math.random());
        p3.setStatus("亚健康");
        hashSet.add(p1);
        hashSet.add(p2);
        hashSet.add(p3);
        g.register(hashSet.stream().map(result -> MultiGauge.Row.of(Tags.of("status", result.getS
atus()), result.getAge())).collect(Collectors.toList()), true);
    }

    public int pushToHTTP(Long l) {
        setPeopleNumber();
        setPeopleAge();
        setStatus();
        return 0;
    }

}
```

效果

```
tomcat_cache_access_total{application="star-universe",} 0.0
# HELP simulation_people_ages 这是一个可以变动的数值
# TYPE simulation_people_ages gauge
simulation_people_ages{application="star-universe",} 0.7189995941088779
# TYPE jdbc_connections_max gauge
jdbc_connections_max{application="star-universe",name="dataSource",} 5.0
# HELP simulation_people_request_total 这是个计数器，只能加不能减
# TYPE simulation_people_request_total counter
simulation_people_request_total{age="30-50",application="star-universe",sex="男",} 4.0
# HELP process_files_max_files The maximum file descriptor count
process_files_open_files{application="star-universe",} 361.0
# HELP simulation_people_statuses_widgets The number of widgets in various statuses
# TYPE simulation_people_statuses_widgets gauge
simulation_people_statuses_widgets{application="star-universe",job="dirty",status="虚弱",} 0.5565179917489952
simulation_people_statuses_widgets{application="star-universe",job="dirty",status="健康",} 0.02159359691618523
simulation_people_statuses_widgets{application="star-universe",job="dirty",status="亚健康",} 0.13441605875895168
# HELP hikaricp_connections Total connections
# TYPE hikaricp_connections gauge
```

结束，休息□□hotsprings!!!

原文链接：Micrometer concepts