33 Spring Boot Admin 介绍

更新时间：2019-07-30 09:30:37

> 没有智慧的头脑，就像没有蜡烛的灯笼。
>
> ——托尔斯泰

在前面的文章中，我们介绍了 Spring Boot Actuator 的使用，并且结合 Prometheus 和 Grafana 进行了数据展示，其中Spring Boot Actuator 提供了对单个 Spring Boot 的监控，监控信息包含：应用状态、内存、线程、堆栈等等，比较全面的监控了 Spring Boot 应用的整个生命周期。

不过它也有一些不足之处：

- 所有的监控都需要调用固定的接口来查看，如果全面查看应用状态需要调用很多接口，并且接口返回的 Json 信息不方便运营人员理解；
- 如果 Spring Boot 应用集群非常大，每个应用都需要调用不同的接口来查看监控信息，操作非常繁琐低效。

我们之前通过 Prometheus + Grafana 将信息聚合到一起集中显示，今天则要和大家来说说另一个展示工具 Spring Boot Admin 。

Spring Boot Admin 介绍

Spring Boot Admin 是一个管理和监控 Spring Boot 应用程序的开源软件:

- 项目 GitHub 地址

在 Spring Boot Admin 体系中，每个应用都是一个客户端，通过 HTTP 或者使用 Eureka 注册到 Admin Server 中进行展示，Spring Boot Admin UI 部分使用 VueJs 将数据展示在前端，是一套目前非常流行的技术架构！Spring Boot Admin 主要提供了如下数据展示功能：

- 展示客户端的健康状况

- 展示应用的一些详细信息，例如 JVM 运行状况、内存使用情况、数据库使用情况以及缓存情况等
- 展示项目构建信息
- 查看项目运行日志
- 查看 Spring Boot 配置信息
- 查看 JVM 和环境信息
- 支持 Spring Cloud 的 postable /env- 和 /refresh-endpoint
- 容易的日志级别管理
- 可以方便的与 JMX-beans 进行交互
- 查看线程堆栈
- 查看 HTTP 追踪信息
- 查看 auditevents、http-endpoints、定时任务、数据库迁移
- 下载 headump
- 邮件报警
- 等等...

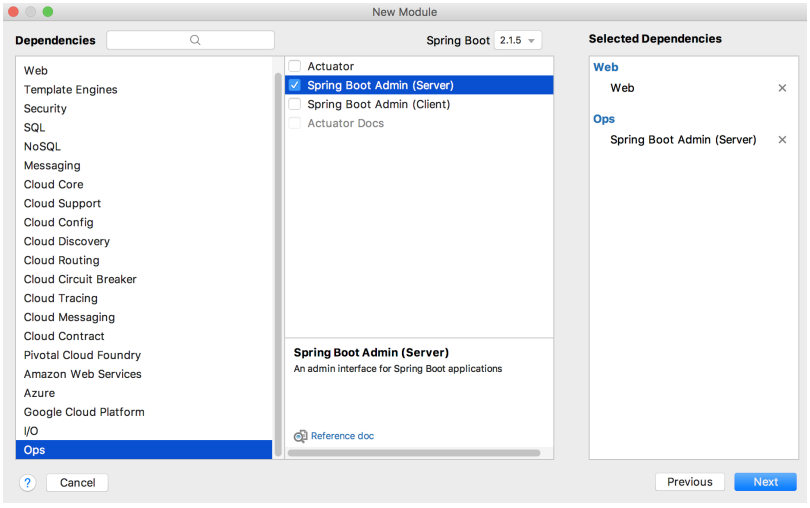
快速开始

接下来我们通过一个简单的案例来看看 Spring Boot Admin 的基本使用。

首先我们创建一个名为 monitor 的空的 Maven 父工程，这一步和前面的微服务基本一致，比较简单，我就不再赘述。

创建 Admin Server

在 monitor 中创建一个名为 adminserver 的 Spring Boot 工程，创建时需要添加 Web 和 Admin Server 依赖，如下：



创建成功之后，项目的依赖如下：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>de.codecentric</groupId>
        <artifactId>spring-boot-admin-starter-server</artifactId>
    </dependency>
</dependencies>
```

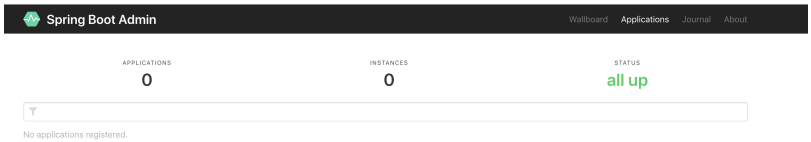
项目创建成功之后，在启动类上添加 `@EnableAdminServer` 注解，表示开启 AdminServer：

```
@SpringBootApplication
@EnableAdminServer
public class AdminserverApplication {

    public static void main(String[] args) {
        SpringApplication.run(AdminserverApplication.class, args);
    }

}
```

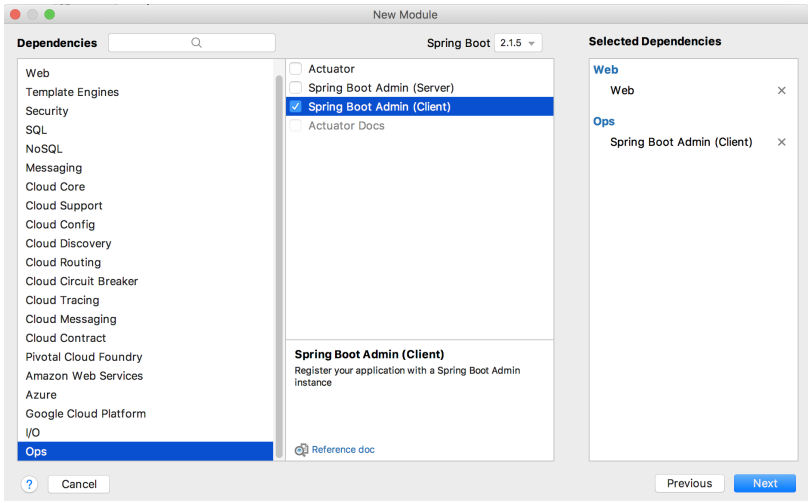
配置完成之后，就可以启动项目了。项目启动成功之后，浏览器中输入 http://localhost:8080 ，我们就可以看到项目的启动页面，如下：



此时因为并没有客户端注册到 Admin Server ，所以实例数为 0 。

创建 Client

接下来，我们创建一个 adminclient ，注册到 adminserver 上，adminclient 就是我们微服务开发中一个一个的服务，这里我们暂时先不引入微服务那一套，就先来一个单机版的服务。因此，创建 adminclient ，我们添加两个依赖，一个 Web，另一个 Admin Client ，如下：



项目创建成功后的 pom.xml 文件如下：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>de.codecentric</groupId>
        <artifactId>spring-boot-admin-starter-client</artifactId>
    </dependency>
</dependencies>
```

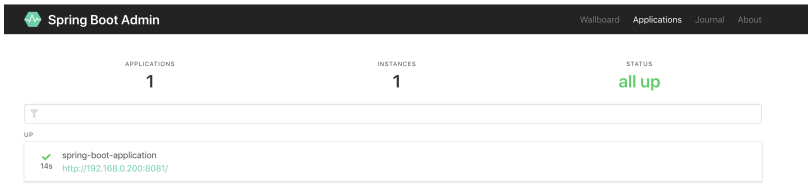
然后我们在 application.properties 中添加如下一些配置信息：

```
server.port=8081
spring.boot.admin.client.url=http://localhost:8080
```

这里就配置两项：

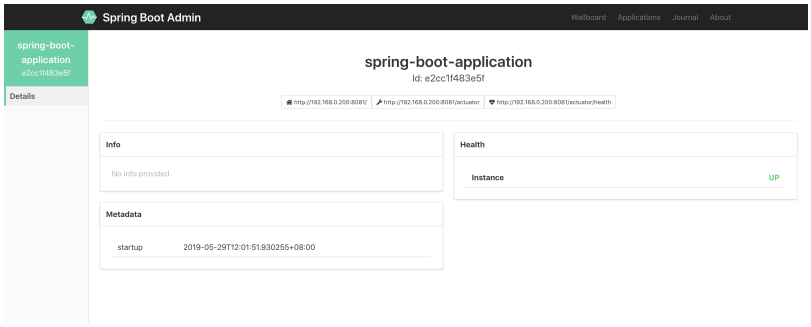
- 第一个端口号，这个不需多说；
- 第二个配置表示配置 admin server 的地址，当 admin client 启动后，会自动注册到 admin server 上去。

配置完成后，这个时候就可以启动 adminclient 了。启动成功之后，我们再回到 adminserver 的管理页面，不用刷新， Admin Server 会自动监测到服务上线、下线等事件，如下：



可以看到注册上来的客户端信息已经显示出来了。

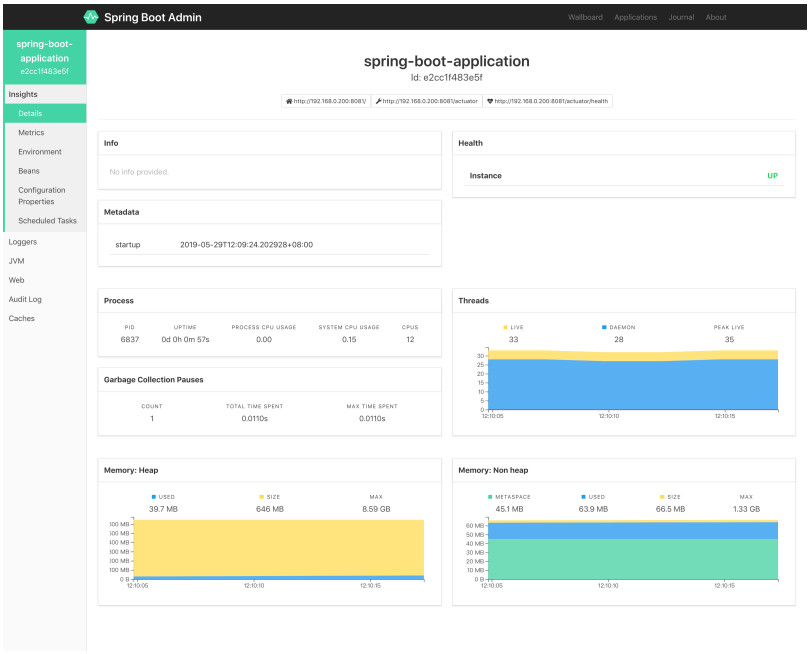
点进到服务里边之后，如下：



我们可以看到，只展示出来了 Info 和 Helath ，这是因为默认有一些端点没有暴露，此时我们需要回到 adminclient 中，再添加一行暴露所有 endpoint 的配置，如下：

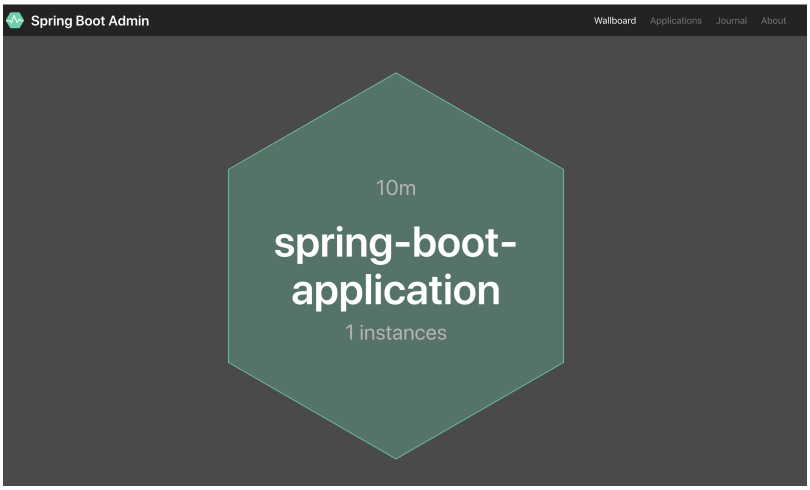
```
management.endpoints.web.exposure.include=*
```

配置完成之后，重启项目，再来看这个页面，此时效果如下：

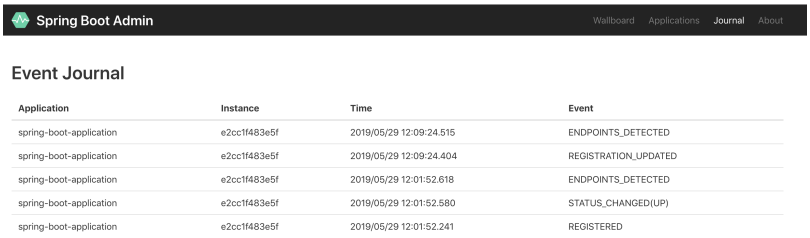


点击左侧的菜单，可以查看详细信息。

点击右上角的 Wallboard 可以看到目前注册上来的所有服务，当然当前只有一个：



点击 Journal 可以看到项目的一些事件信息，如下：



小结

本文主要和大家聊了一下 Spring Boot Admin 的基本功能和基本用法，对服务监控的基本支持。可以说，这个东西还是非常方便的，只是本文我们还没有引入微服务，注册的配置都要挨个配置，略微有一些麻烦，下一篇文章和大家再来详细说说 Spring Boot Admin 结合微服务的用法，以及相关的报警机制。

32 Zipkin 实践
34 Spring Boot Admin 实践