

# 使用Spring Boot Admin实现运维监控平台

官方地址: <https://github.com/codecentric/spring-boot-admin>

我们知道, 使用Actuator可以收集应用系统的健康状态、内存、线程、堆栈、配置等信息, 比较全面地监控了Spring Boot应用的整个生命周期。但是还有一个问题: 如何呈现这些采集到的应用监控数据、性能数据呢? 在这样的背景下, 就诞生了另一个开源软件Spring Boot Admin。下面就来介绍什么是Spring Boot Admin以及如何使用Spring Boot Admin搭建完整的运维监控平台。

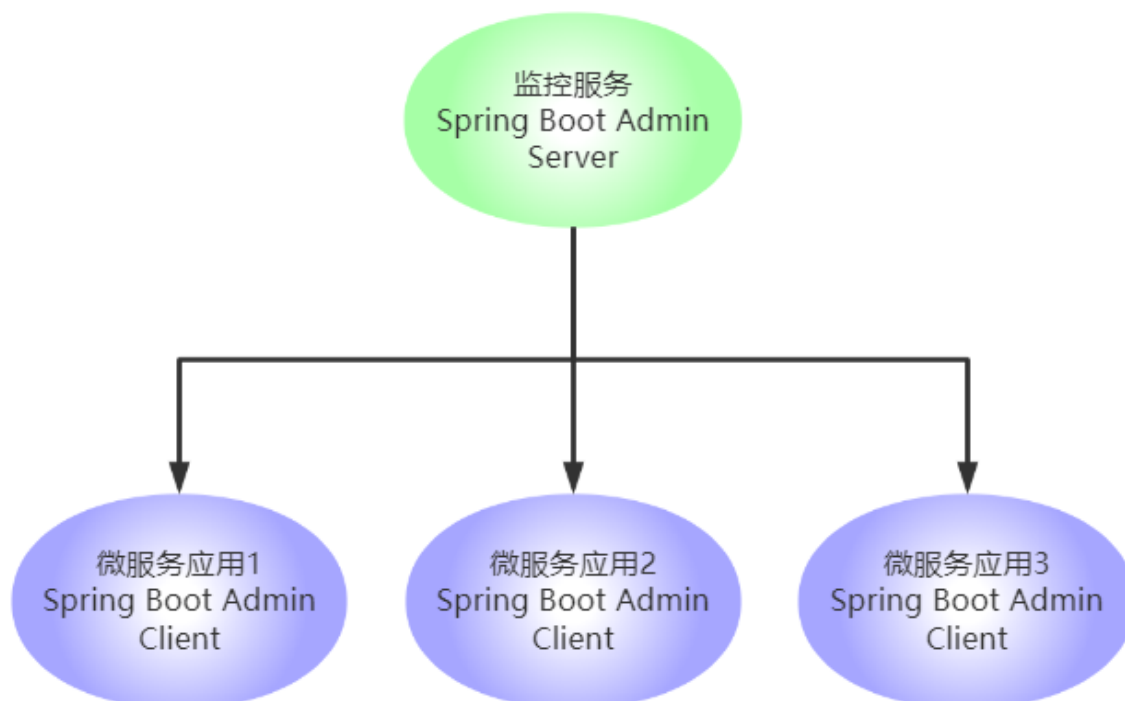
## 1 关于Spring Boot Admin

Spring Boot Admin是一个管理和监控Spring Boot应用程序的开源项目, 在对单一应用服务监控的同时也提供了集群监控方案, 支持通过eureka、consul、zookeeper等注册中心的方式实现多服务监控与管理。Spring Boot Admin UI部分使用Vue JS将数据展示在前端。

Spring Boot Admin分为服务端 (spring-boot-admin-server) 和客户端 (spring-boot-admin-client) 两个组件:

- spring-boot-admin-server: 通过采集actuator端点数据显示在spring-boot-admin-ui上, 已知的端点几乎都有进行采集。
- spring-boot-admin-client: 是对Actuator的封装, 提供应用系统的性能监控数据。此外, 还可以通过spring-boot-admin动态切换日志级别、导出日志、导出heapdump、监控各项性能指标等。

Spring Boot Admin服务器端负责收集各个客户的数据。各台客户端配置服务器地址, 启动后注册到服务器。服务器不停地请求客户端的信息 (通过Actuator接口)。具体架构如下图所示。



上图为Spring Boot Admin的整体架构, 在每个Spring Boot应用程序上增加Spring Boot Admin Client 组件。这样每个Spring Boot应用即Admin客户端, Admin服务端通过请求Admin客户端的接口收集所有的Spring Boot应用信息并进行数据呈现, 从而实现Spring Boot应用监控。

## 2 搭建运维监控平台

## 2.1 搭建服务端

### 导入依赖

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5          https://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7      <groupId>com.zeroone.star</groupId>
8      <artifactId>project-admin-server</artifactId>
9      <version>1.0.0-SNAPSHOT</version>
10     <properties>
11         <java.version>1.8</java.version>
12         <spring-boot.version>2.3.12.RELEASE</spring-boot.version>
13         <spring-boot-admin.version>2.3.1</spring-boot-admin.version>
14     </properties>
15     <dependencies>
16         <!-- web -->
17         <dependency>
18             <groupId>org.springframework.boot</groupId>
19             <artifactId>spring-boot-starter-web</artifactId>
20         </dependency>
21         <!-- spring boot admin server -->
22         <dependency>
23             <groupId>de.codecentric</groupId>
24             <artifactId>spring-boot-admin-starter-server</artifactId>
25         </dependency>
26     </dependencies>
27     <dependencyManagement>
28         <dependencies>
29             <!-- spring boot -->
30             <dependency>
31                 <groupId>org.springframework.boot</groupId>
32                 <artifactId>spring-boot-dependencies</artifactId>
33                 <version>${spring-boot.version}</version>
34                 <type>pom</type>
35                 <scope>import</scope>
36             </dependency>
37             <!-- spring boot admin server -->
38             <dependency>
39                 <groupId>de.codecentric</groupId>
40                 <artifactId>spring-boot-admin-starter-server</artifactId>
41                 <version>${spring-boot-admin.version}</version>
42             </dependency>
43         </dependencies>
44     </dependencyManagement>
45     <build>
46         <plugins>
47             <plugin>
48                 <groupId>org.springframework.boot</groupId>
49                 <artifactId>spring-boot-maven-plugin</artifactId>
50                 <version>${spring-boot.version}</version>
51                 <executions>
```

```

51         <execution>
52             <id>repackage</id>
53             <goals>
54                 <goal>repackage</goal>
55             </goals>
56         </execution>
57     </executions>
58 </plugin>
59 </plugins>
60 </build>
61 </project>

```

### 修改项目配置文件

```

1  spring:
2      application:
3          name: PROJECT-ADMIN-SERVER
4  server:
5      port: 8080

```

### 修改程序入口

```

1  /**
2   * @Description 程序入口
3   * @Author 阿伟学长
4   * @Copy &copy;01星球
5   * @Address 01星球总部
6   */
7  @EnableAdminServer
8  @SpringBootApplication
9  public class Application {
10     public static void main(String[] args) {
11         SpringApplication.run(Application.class, args);
12     }
13 }

```

启动服务访问测试，访问地址：<http://localhost:8080>，浏览器显示效果如下图所示：



暂无应用注册。

从Admin服务端的启动界面可以看到，Applications页面会展示应用数量、实例数量和状态3个信息。这里由于没有启动客户端，因此显示出“暂无应用注册”的信息。

## 2.2 搭建客户端

### 导入依赖

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5         http://maven.apache.org/xsd/maven-4.0.0.xsd">
6     <modelVersion>4.0.0</modelVersion>
7     <groupId>com.zeroone.star</groupId>
8     <artifactId>project-admin-client</artifactId>
9     <version>1.0.0-SNAPSHOT</version>
10    <properties>
11        <java.version>1.8</java.version>
12        <spring-boot.version>2.3.12.RELEASE</spring-boot.version>
13        <spring-boot-admin.version>2.3.1</spring-boot-admin.version>
14    </properties>
15    <dependencies>
16        <!-- web -->
17        <dependency>
18            <groupId>org.springframework.boot</groupId>
19            <artifactId>spring-boot-starter-web</artifactId>
20        </dependency>
21        <!-- spring boot admin client -->
22        <dependency>
23            <groupId>de.codecentric</groupId>
24            <artifactId>spring-boot-admin-starter-client</artifactId>
25        </dependency>
26    </dependencies>
27    <dependencyManagement>
28        <dependencies>
29            <!-- spring boot -->
30            <dependency>
31                <groupId>org.springframework.boot</groupId>
32                <artifactId>spring-boot-dependencies</artifactId>
33                <version>${spring-boot.version}</version>
34                <type>pom</type>
35                <scope>import</scope>
36            </dependency>
37            <!-- spring boot admin client -->
38            <dependency>
39                <groupId>de.codecentric</groupId>
40                <artifactId>spring-boot-admin-starter-client</artifactId>
41                <version>${spring-boot-admin.version}</version>
42            </dependency>
43        </dependencies>
44    </dependencyManagement>
45    <build>
46        <plugins>
47            <plugin>
48                <groupId>org.springframework.boot</groupId>
49                <artifactId>spring-boot-maven-plugin</artifactId>
50                <version>${spring-boot.version}</version>
51                <executions>
```

```

51         <execution>
52             <id>repackage</id>
53             <goals>
54                 <goal>repackage</goal>
55             </goals>
56         </execution>
57     </executions>
58 </plugin>
59 </plugins>
60 </build>
61 </project>

```

修改项目配置文件

```

1  spring:
2      application:
3          name: PROJECT-ADMIN-CLIENT
4      boot:
5          admin:
6              client:
7                  # 配置Admin服务器的地址
8                  url: http://localhost:8080
9                  instance:
10                     # 通过IP注册
11                     prefer-ip: true
12      server:
13          port: 8081
14
15      # 打开客户端Actuator的监控
16      management:
17          endpoints:
18              web:
19                  exposure:
20                      include: "*"

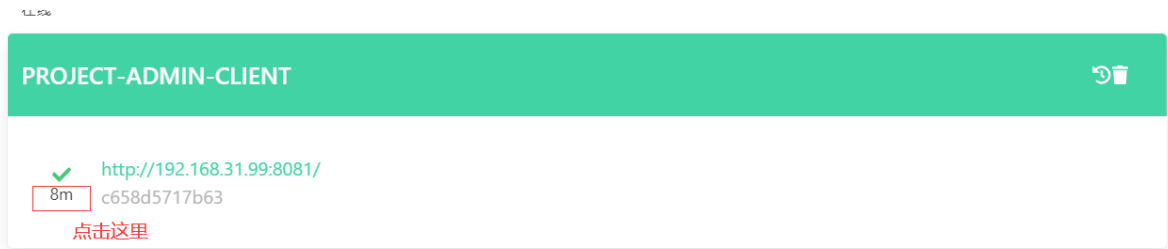
```

启动服务，然后关系监控服务端页面，可以看到如下图所示的效果

The screenshot shows the Spring Boot Admin web interface. At the top, there's a header with the Spring Boot Admin logo and navigation links: 应用墙 (Applications Wall), 应用 (Applications), 日志报表 (Log Reports), 关于我们 (About Us), and zh-CN (Language). Below the header, there are three summary statistics: 应用数 (Number of Applications) is 1, 实例数 (Number of Instances) is 1, and 实例状态 (Instance Status) is 全部在线 (All Online). A search bar is present below these statistics. Under the '在线' (Online) filter, a single application is listed: PROJECT-ADMIN-CLIENT, with a status of 4m and a URL of http://192.168.31.99:8081/.

客户端启动之后，Admin服务器界面的应用数量会增加。

页面会展示被监控的应用列表，单击应用名称会进入此应用的详细监控信息页面。



可以看到下面所示的效果



这个页面会实时显示应用的运行监控信息，包括Actuator所有的端点数据信息。

Spring Boot Admin以图形化的形式展示了应用的各项信息，这些信息大多来自于Spring Boot Actuator提供的接口。利用图形化的形式很容易看到应用的各项参数变化，甚至有些页面还可以进行一些配置操作，比如改变打印日志的级别等。

## 3 告警提醒功能

Spring Boot Admin提供了强大的提醒功能，能够在发生服务状态变更的时候发出告警。支持的Email等提醒功能，同时也支持自定义告警提醒。

### 3.1 邮件提醒

在服务端添加邮件组件

```
1 <dependency>
2   <groupId>org.springframework.boot</groupId>
3   <artifactId>spring-boot-starter-mail</artifactId>
4 </dependency>
```

修改配置文件

```
1 spring:
2   application:
3     name: PROJECT-ADMIN-SERVER
4   # 邮件服务器配置
5   mail:
6     # 邮箱站点配置，这是使用QQ邮箱的smtp协议
```

```
7      host: smtp.qq.com
8      port: 465
9      # 登录邮箱账号
10     username: xxxxx@qq.com
11     # 授权码
12     password: xxxxxxxx
13     properties:
14         mail:
15             smtp:
16                 auth: true
17                 ssl:
18                     enable: true
19     boot:
20         admin:
21             notify:
22                 mail:
23                     # 开启邮件通知
24                     enabled: true
25                     # 发件人邮箱地址和邮箱服务器用户名保持一致
26                     from: xxxxx@qq.com
27                     # 接收通知的收件人
28                     to: toxxxxx@qq.com
29     server:
30         port: 8080
```

如何获取授权码，下面以QQ邮箱为例，在邮箱设置中找到SMTP服务，开启它。

参考链接：<https://service.mail.qq.com/cgi-bin/help?subtype=1&&id=10000&&no=1001607>



开启服务

×

1 身份验证

————

2 生成授权码

生成授权码完成，可以在第三方客户端访问QQ邮箱时使用

1 复制授权码

授权码备注：

服务器状态邮箱服务

2 填写备注


你可拥有多个授权码，所以无需记住该授权码，也不要告诉其他人。

测试验证，我们再次启动服务端和客户端，然后停止客户端，模拟应用宕机的情况。这样Spring Boot Admin 就会发送告警邮件提醒。

邮箱中收到邮件，效果如下图所示

## PROJECT-ADMIN-CLIENT (c658d5717b63) is OFFLINE ☆

  发给 我

 邮件可翻译为中文 [立即翻译](#)

## PROJECT-ADMIN-CLIENT (c658d5717b63) is OFFLINE

Instance c658d5717b63 changed status from UP to OFFLINE

### Status Details

exception  
io.netty.channel.AbstractChannel\$AnnotatedConnectException  
message  
Connection refused: no further information: /192.168.31.99:8081

### Registration

Service Url <http://192.168.31.99:8081/>  
Health Url <http://192.168.31.99:8081/actuator/health>  
Management Url<http://192.168.31.99:8081/actuator>



## 3.2 自定义告警

除了邮件提醒之外，通常我们还需要其他的提醒方式，比如：短信，日志等。我们可以通过自定义的方式实现自定义的消息告警方式。

Spring Boot Admin 实现自定义告警提醒也非常简单，只要实现Notifier接口即可。具体实现方式：继承AbstractEventNotifier或AbstractStatusChangeNotifier这两个类。然后重写doNotify中实现具体的业务逻辑。

下面通过示例演示自定义告警提醒功能：

首先，创建LoggingNotifier类，实现告警提醒功能，示例代码如下：

```
1  /**
2   * @Description 自定义一个日志告警示例
3   * @Author 阿伟学长
4   * @Copy &copy;01星球
5   * @Address 01星球总部
6   */
7  @Component
8  public class LoggingNotifier extends AbstractEventNotifier {
9      private static final Logger LOGGER =
10         LoggerFactory.getLogger(LoggingNotifier.class);
11     protected LoggingNotifier(InstanceRepository repository) {
12         super(repository);
13     }
14     @Override
15     protected Mono<Void> doNotify(InstanceEvent event, Instance instance) {
16         return Mono.fromRunnable(() -> {
17             if (event instanceof InstanceStatusChangedEvent) {
18                 LOGGER.info("Instance {} ({{}}) is {}",
19                     instance.getRegistration().getName(),
20                     event.getInstance(),
21                     ((InstanceStatusChangedEvent)
22                     event).getStatusInfo().getStatus());
23             } else {
24                 LOGGER.info("Instance {} ({{}}) {}",
25                     instance.getRegistration().getName(),
26                     event.getInstance(),
27                     event.getType());
28             }
29         });
30     }
31 }
```

然后，再次运行服务端和客户端。启动成功之后，再关掉客户端，模拟应用宕机的情况。

```
c.z.s.p.components.LoggingNotifier : Instance PROJECT-ADMIN-CLIENT (c658d5717b63) is OFFLINE
```

我们看到服务端后台日志显示，服务端已经收到了客户端状态改变的告警消息。客户端状态已经变为OFFLINE。

## 4 使用注册中心

因为nacos自动帮我们整合了与admin的关联工作,只需要注册进nacos,并且与服务端保持在同一命名空间和分组下即可。

## 4.1 服务端调整

添加nacos依赖

```
1  <!-- ..... -->
2  <properties>
3      <java.version>1.8</java.version>
4      <spring-boot.version>2.3.12.RELEASE</spring-boot.version>
5      <spring-boot-admin.version>2.3.1</spring-boot-admin.version>
6
7      <spring.cloud.alibaba.version>2.2.8.RELEASE</spring.cloud.alibaba.version>
8  </properties>
9  <dependencies>
10     <!-- ..... -->
11     <!-- spring cloud alibaba -->
12     <dependency>
13         <groupId>com.alibaba.cloud</groupId>
14         <artifactId>spring-cloud-starter-alibaba-nacos-
15         discovery</artifactId>
16     </dependency>
17 </dependencies>
18 <dependencyManagement>
19     <dependencies>
20         <!-- ..... -->
21         <!-- spring cloud alibaba -->
22         <dependency>
23             <groupId>com.alibaba.cloud</groupId>
24             <artifactId>spring-cloud-alibaba-dependencies</artifactId>
25             <version>${spring.cloud.alibaba.version}</version>
26             <type>pom</type>
27             <scope>import</scope>
28         </dependency>
29     </dependencies>
30 </dependencyManagement>
31 <!-- ..... -->
```

修改配置文件

```
1  spring:
2      # .....
3      # nacos配置
4      cloud:
5          nacos:
6              discovery:
7                  server-addr: 192.168.220.128:8848
8                  namespace: 4833404f-4b82-462e-889a-3c508160c6b4
9
10 management:
11     endpoints:
12         web:
13             exposure:
```

```
14         include: '*'
15     endpoint:
16         health:
17             show-details: always
```

修改程序入口

```
1  @EnableAdminServer
2  @EnableDiscoveryClient
3  @SpringBootApplication
4  public class Application {
5      public static void main(String[] args) {
6          SpringApplication.run(Application.class, args);
7      }
8  }
```

## 4.2 客户端调整

添加nacos相关依赖，参考4.1服务端调整

修改配置文件

```
1  server:
2      port: 8081
3  spring:
4      application:
5          name: PROJECT-ADMIN-CLIENT
6      # boot:
7      #   admin:
8      #       client:
9      #           # 配置Admin服务器的地址
10         url: http://localhost:8080
11         instance:
12             # 通过IP注册
13             prefer-ip: true
14     # nacos配置
15     cloud:
16         nacos:
17             discovery:
18                 server-addr: 192.168.220.128:8848
19                 namespace: 4833404f-4b82-462e-889a-3c508160c6b4
20
21     # 打开客户端Actuator的监控
22     management:
23         endpoints:
24             web:
25                 exposure:
26                     include: "*"

```

修改程序入口

```

1  @EnableDiscoveryClient
2  @SpringBootApplication
3  public class Application {
4      public static void main(String[] args) {
5          SpringApplication.run(Application.class, args);
6      }
7  }

```

启动服务端和客服端测试一下


**Spring Boot Admin**
应用墙 应用 日志报表 关于我们 zh-CN

应用数	实例数	实例状态
2	2	全部在线

在线

<div>  PROJECT-ADMIN-CLIENT  42s <a href="http://192.168.125.1:8081">http://192.168.125.1:8081</a> </div>	服务端和客服端都会列举出来
<div>  PROJECT-ADMIN-SERVER  10m <a href="http://192.168.125.1:8080">http://192.168.125.1:8080</a> </div>	

## 4.3 服务端认证

下面为服务器加入安全证

添加依赖

```

1  <!-- security -->
2  <dependency>
3      <groupId>org.springframework.boot</groupId>
4      <artifactId>spring-boot-starter-security</artifactId>
5  </dependency>

```

修改配置

```

1  spring:
2      # .....
3      # nacos配置
4      cloud:
5          nacos:
6              discovery:
7                  server-addr: 192.168.220.128:8848
8                  namespace: 4833404f-4b82-462e-889a-3c508160c6b4
9                  metadata:
10                     user.name: admin
11                     user.password: admin123
12      # 认证账号配置
13      security:
14          user:
15              name: admin
16              password: admin123

```

## 编写security配置

```

1  /**
2   * @Description Security配置
3   * @Author 阿伟学长
4   * @Copy &copy;01星球
5   * @Address 01星球总部
6   */
7  @Configuration
8  public class SecuritySecureConfig extends WebSecurityConfigurerAdapter {
9      private final String adminContextPath;
10
11     public SecuritySecureConfig(AdminServerProperties adminServerProperties)
12     {
13         this.adminContextPath = adminServerProperties.getContextPath();
14     }
15
16     @Override
17     protected void configure(HttpSecurity http) throws Exception {
18         // 登录成功处理类
19         SavedRequestAwareAuthenticationSuccessHandler successHandler =
20             new SavedRequestAwareAuthenticationSuccessHandler();
21         successHandler.setTargetUrlParameter("redirectTo");
22         successHandler.setDefaultTargetUrl(adminContextPath + "/");
23
24         http.authorizeRequests()
25             //静态文件允许访问
26             .antMatchers(adminContextPath + "/assets/**").permitAll()
27             //登录页面允许访问
28             .antMatchers(adminContextPath + "/login", "/css/**",
29                 "/js/**", "/image/**").permitAll()
30             //其他所有请求需要登录
31             .anyRequest().authenticated()
32             .and()
33             //登录页面配置，用于替换security默认页面
34             .formLogin().loginPage(adminContextPath +
35                 "/login").successHandler(successHandler)
36             .and()
37             //登出页面配置，用于替换security默认页面
38             .logout().logoutUrl(adminContextPath + "/logout")
39             .and()
40             .httpBasic()
41             .and()
42             .csrf()
43
44             .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
45             .ignoringAntMatchers(
46                 "/instances",
47                 "/actuator/**"
48             );
49     }
50 }

```

启动服务后，就会出现登录界面



The image shows the Spring Boot Admin login interface. At the top is a green hexagonal logo with a white heartbeat line. Below the logo is the text "Spring Boot Admin" in green. There are two input fields: the first is labeled "用户名" (Username) and the second is labeled "密码" (Password). Below the password field is a checkbox labeled "记住用户" (Remember user). At the bottom is a large green button labeled "登录" (Login).

输入账号和密码就可以进入到应用页面