SPRING BOOT ADMIN 接入文档

SPRING BOOT 2.x

springboot 版本:

```
        <spring-boot.version>2.1.7.RELEASE</spring-boot.version>
```

springcloud 版本:

```
        <spring-cloud.version>Greenwich.RELEASE</spring-cloud.version>
```

springboot admin版本:

```
    <spring-boot-admin.version>2.1.2</spring-boot-admin.version>
```

前提

服务接入同一个服务注册中心(EUREKA),父工程统一指定 `springboot` 和 `springcloud` 版本 以及 `Springboot Admin` 版本

```
<dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-dependencies</artifactId>
                <version>${spring-boot.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
             <dependency>
                <groupId>de.codecentric</groupId>
                <artifactId>spring-boot-admin-starter-server</artifactId>
```

```
                <version>2.1.2</version>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

Spring Boot Admin Server 部署接入

POM配置

```
<dependencies>

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

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>

        <dependency>
            <groupId>de.codecentric</groupId>
            <artifactId>spring-boot-admin-starter-server</artifactId>
        </dependency>

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

        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-commons</artifactId>
        </dependency>

        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <scope>compile</scope>
        </dependency>

</dependencies>
```

配置文件

```yaml
server:
  port: 8081

eureka:
  client:
    service-url:
      defaultZone: http://127.0.0.1:7001/eureka
  instance:
    instance-id: ${spring.cloud.client.ip-address}:${server.port}
    lease-expiration-duration-in-seconds: 10
    lease-renewal-interval-in-seconds: 10
    prefer-ip-address: true
    metadata-map:
      user.name: ${spring.security.user.name}
      user.password: ${spring.security.user.password}
management:
  health:
    # 指定磁盘告警
    diskspace:
      threshold: '10GB'
  endpoint:
    health:
      show-details: always
  endpoints:
    web:
      exposure:
        include: "*"
        exclude:
'env,sessions,heapdump,flyway,loggers,logfile,beans,caches,sessions,scheduledtasks'
spring:
  application:
    name: springboot-admin-server
  boot:
    admin:
      ding-talk-token: "xx"
      probed-endpoints:
      instance:
        # 指定监控的服务
        register-pattern: "springboot-admin*"
      instance-auth:
        default-user-name: "admin"
        default-user-password: "test"
  # 接入spring security
  security:
    user:
      name: "admin"
```

```
      password: "test"
  profiles:
    active: dev
```

- 配置文件中需指定:
  - eureka地址: `eureka.client.service-url.defaultZone`
  - 指定的监控服务: `spring.boot.admin.instance.register-pattern`
  - Security 配置: `spring.security.user.name`, `spring.security.user.password`
  - 钉钉机器人token: `spring.booot.admin.ding-talk-token`

代码

1. 添加`@EnableAdminServer`注解
2. 添加配置文件
   1. `InstanceDiscoveryConfig.java`

```java
import de.codecentric.boot.admin.server.cloud.discovery.EurekaServiceInstanceConverter;
import de.codecentric.boot.admin.server.cloud.discovery.InstanceDiscoveryListener;
import de.codecentric.boot.admin.server.domain.entities.InstanceRepository;
import de.codecentric.boot.admin.server.services.InstanceRegistry;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.cloud.client.discovery.DiscoveryClient;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

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

@Configuration
public class InstanceDiscoveryConfig {

    @Value("${spring.boot.admin.instance.register-pattern}")
    private String serviceRegisterPattern;

    @Bean
    public EurekaServiceInstanceConverter serviceInstanceConverter() {
        return new EurekaServiceInstanceConverter();
```

```
    }

    @Bean
    public InstanceDiscoveryListener instanceDiscoveryListener(EurekaServiceInstanceConverter serviceInstanceConverter,
                                                               DiscoveryClient discoveryClient,
                                                               InstanceRegistry registry,
                                                               InstanceRepository repository) {
        InstanceDiscoveryListener listener = new InstanceDiscoveryListener(discoveryClient, registry, repository);
        listener.setConverter(serviceInstanceConverter);

        String[] split = serviceRegisterPattern.split(",");
        Set<String> services = new HashSet<>(Arrays.asList(split));
        listener.setServices(services);
        return listener;
    }
}
```

2. `SecurityConfig`

```
import de.codecentric.boot.admin.server.config.AdminServerProperties;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuccessHandler;
import org.springframework.security.web.csrf.CookieCsrfTokenRepository;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    private final String adminContextPath;

```

```
    public SecurityConfig(AdminServerProperties adminServerProperties) {
        this.adminContextPath = adminServerProperties.getContextPath();
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        SavedRequestAwareAuthenticationSuccessHandler successHandler = new SavedRequestAwareAuthenticationSuccessHandler();
        successHandler.setTargetUrlParameter("redirectTo");
        successHandler.setDefaultTargetUrl(adminContextPath + "/");

        http.authorizeRequests()
                .antMatchers(adminContextPath + "/assets/**").permitAll()
                .antMatchers(adminContextPath + "/login").permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage( adminContextPath + "/login")
                .successHandler(successHandler)
                .and()
                .logout().logoutUrl(adminContextPath + "/logout").and()
                .httpBasic()
                .and()
                .csrf()

.csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
                .ignoringAntMatchers(
                        adminContextPath + "/instances",
                        adminContextPath + "/actuator/**"
                );
    }
}

```

3. `RestTemplateConfig`

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.Primary;
import org.springframework.http.client.ClientHttpRequestFactory;
```

```
import org.springframework.http.client.SimpleClientHttpRequestFactory;
import org.springframework.web.client.RestTemplate;

@Configuration
public class RestTemplateConfig {

    @Primary
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate(simpleClientHttpRequestFactory());
    }

    @Bean
    public ClientHttpRequestFactory simpleClientHttpRequestFactory() {
        SimpleClientHttpRequestFactory factory = new SimpleClientHttpRequestFactory();
        factory.setReadTimeout(3000);
        factory.setConnectTimeout(3000);
        return factory;
    }
}
```

3. 添加告警
    1. 添加对应的告警MODEL
        1. `TextMessageContent`

```
import lombok.Data;

@Data
public class TextMessageContent {
    private String content;
}
```

        2. `DingTalkTextMessage`

```java
import lombok.Data;

@Data
public class DingTalkTextMessage<T>{

    private String msgtype = "text";

    private T text;

    private boolean isAtAll = true;

}
```

2. 添加告警监听`DingTalkNotify`

```java

import de.codecentric.boot.admin.server.domain.entities.Instance;
import de.codecentric.boot.admin.server.domain.entities.InstanceRepository;
import de.codecentric.boot.admin.server.domain.events.InstanceEvent;
import de.codecentric.boot.admin.server.domain.events.InstanceStatusChangedEvent;
import de.codecentric.boot.admin.server.notify.AbstractEventNotifier;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang.time.DateFormatUtils;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;
import reactor.core.publisher.Mono;

import java.util.Date;

@Slf4j
@Service
public class DingTalkNotify extends AbstractEventNotifier {

    /**
     * 消息模板
     */
```

```
27      private static final String template = "monitor:\n时间: %s \n环境: %s \n服务名:%s(%s) \n状态:%s(%s) \n服务ip:%s";
28
29      @Value("${spring.boot.admin.ding-talk-token}")
30      private String dingTalkToken;
31
32      @Value("${spring.profiles.active}")
33      private String evn;
34
35      private RestTemplate restTemplate;
36
37      public DingTalkNotify(InstanceRepository repository, RestTemplate restTemplate) {
38          super(repository);
39          this.restTemplate = restTemplate;
40      }
41
42      @Override
43      protected Mono<Void> doNotify(InstanceEvent event, Instance instance) {
44          return Mono.fromRunnable(() -> {
45              if (event instanceof InstanceStatusChangedEvent) {
46                  log.info("Instance {} ({}) is {}", instance.getRegistration().getName(), event.getInstance(),
47                          ((InstanceStatusChangedEvent) event).getStatusInfo().getStatus());
48
49
50                  String status = ((InstanceStatusChangedEvent) event).getStatusInfo().getStatus();
51                  String messageText = null;
52                  String dateTime = DateFormatUtils.ISO_DATETIME_FORMAT.format(new Date());
53                  switch (status) {
54                      // 健康检查没通过
55                      case "DOWN":
56                          log.info("发送 健康检查没通过 的通知! ");
57                          messageText = String.format(template,dateTime, evn, instance.getRegistration().getName(), event.getInstance(), ((InstanceStatusChangedEvent) event).getStatusInfo().getStatus(), "健康检查没通过", instance.getRegistration().getServiceUrl());
58                          this.sendMessage(messageText);
59                          break;
60                      // 服务离线
61                      case "OFFLINE":
62                          log.info("发送 服务离线 的通知! ");
```

```
                        messageText =
String.format(template,dateTime, evn,
instance.getRegistration().getName(), event.getInstance(),
((InstanceStatusChangedEvent) event).getStatusInfo().getStatus(),
"服务离线", instance.getRegistration().getServiceUrl());
                        this.sendMessage(messageText);
                        break;
                    //服务上线
                    case "UP":
                        log.info("发送 服务上线 的通知! ");
                        messageText =
String.format(template,dateTime, evn,
instance.getRegistration().getName(), event.getInstance(),
((InstanceStatusChangedEvent) event).getStatusInfo().getStatus(),
"服务上线", instance.getRegistration().getServiceUrl());
                        this.sendMessage(messageText);
                        break;
                    // 服务未知异常
                    case "UNKNOWN":
                        log.info("发送 服务未知异常 的通知! ");
                        messageText =
String.format(template,dateTime, evn,
instance.getRegistration().getName(), event.getInstance(),
((InstanceStatusChangedEvent) event).getStatusInfo().getStatus(),
"服务未知异常", instance.getRegistration().getServiceUrl());
                        this.sendMessage(messageText);
                        break;
                    default:
                        break;
                }
            } else {
                log.info("Instance {} ({}) {}",
instance.getRegistration().getName(), event.getInstance(),
                        event.getType());
            }
        });
    }

    /**
     * 发送消息
     *
     * @param messageText
     */
    private void sendMessage(String messageText) {
        HttpHeaders headers = new HttpHeaders();
        headers.setContentType(MediaType.APPLICATION_JSON_UTF8);

        DingTalkTextMessage<TextMessageContent>
dingTalkTextMessage = new DingTalkTextMessage<>();
```

```
        TextMessageContent textMessageContent = new TextMessageContent();
        textMessageContent.setContent(messageText);
        dingTalkTextMessage.setText(textMessageContent);

        HttpEntity<DingTalkTextMessage> request = new HttpEntity<>(dingTalkTextMessage, headers);
        String url = "https://oapi.dingtalk.com/robot/send?access_token=" + dingTalkToken;
        ResponseEntity<String> stringResponseEntity = restTemplate.postForEntity(url, request, String.class);

        log.info("返回:{}", stringResponseEntity.getBody());

    }
}
```

钉钉群创建并添加机器人

1. 创建项目告警群. 并在群管理-> 智能群助手中添加自定义机器人
2. 添加自定义机器人时, 保留token , 并在安全设置中选用自定义关键词(告警信息中需包含该关键词)->`monitor`

Spring Boot Admin Client 接入

POM配置

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

       <dependency>
           <groupId>org.springframework.boot</groupId>
           <artifactId>spring-boot-actuator-autoconfigure</artifactId>
       </dependency>

       <dependency>
           <groupId>org.springframework.cloud</groupId>
           <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
       </dependency>

        <dependency>
           <groupId>org.springframework.cloud</groupId>
           <artifactId>spring-cloud-commons</artifactId>
       </dependency>
```

```
21
22          <dependency>
23              <groupId>org.springframework.boot</groupId>
24              <artifactId>spring-boot-starter-web</artifactId>
25          </dependency>
26      </dependencies>
```

配置文件

```
1  eureka:
2    client:
3      service-url:
4        defaultZone: http://127.0.0.1:7001/eureka
5    instance:
6      instance-id: ${spring.cloud.client.ip-address}:${server.port}
7      lease-expiration-duration-in-seconds: 10
8      lease-renewal-interval-in-seconds: 10
9      prefer-ip-address: true
10     metadata-map:
11       user.name: ${spring.security.user.name}
12       user.password: ${spring.security.user.password}
13 management:
14   health:
15     # 指定磁盘告警
16     diskspace:
17       threshold: '10GB'
18   endpoint:
19     health:
20       show-details: always
21   endpoints:
22     web:
23       exposure:
24         include: "*"
25         exclude:
   'env,sessions,heapdump,flyway,loggers,logfile,beans,caches,sessions,schedu
   ledtasks'
26
27 spring:
28   application:
29     name: springboot-admin-client
30   # 接入spring security
31   security:
32     user:
33       name: "admin"
34       password: "test"
35 server:
36   port: 8082
37
```

- 配置文件中需指定:
  - eureka地址: `eureka.client.service-url.defaultZone`
  - 指定的监控服务: `spring.boot.admin.instance.register-pattern`
  - Security 配置: `spring.security.user.name`, `spring.security.user.password`

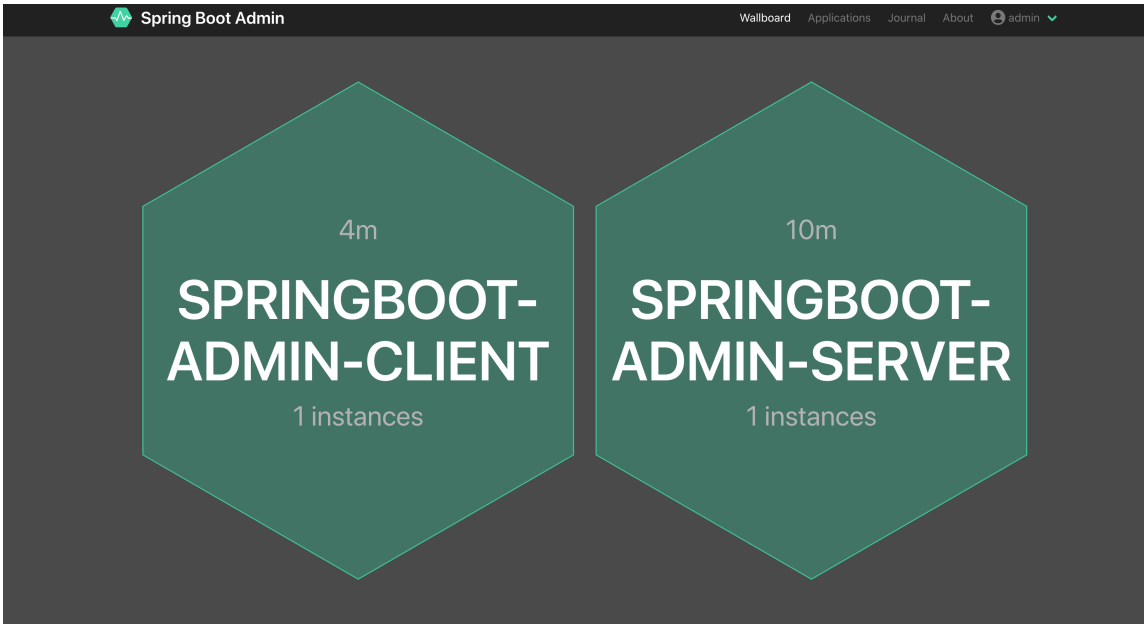
Security 配置

1. `SecurityConfig`

```
import org.springframework.boot.actuate.autoconfigure.security.servlet.EndpointRequest;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.requestMatcher(EndpointRequest.toAnyEndpoint());
        http.httpBasic();
    }
}

```

启动后效果

- 访问server

- 

- 告警效果
  - 

代码demo 仓库

http://code.corp.rs.com/H.chen.chen/springboot-admin-demo