Spring Boot Admin

1. 什么是Spring Boot Admin

`Spring Boot Admin` 是一个社区的项目用来监控和管理我们的`Spring Boot` 应用。应用注册可以通过 `Spring Boot Admin Client (Http)` 或者使用 `Spring Cloud` 注册中心去发现（例如`Eureka`，`Consul`)。

2. 配置 Spring Boot Admin Server

2.1 配置pom.xml

```
    <properties>
           <java.version>1.8</java.version>
           <spring-boot-admin.version>2.1.1</spring-boot-admin.version>
       </properties>

       <dependencies>
           <!-- 服务端：带UI界面 -->
           <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>

           <!-- 邮件提醒 -->
           <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-mail</artifactId>
           </dependency>

           <!-- 权限配置-->
           <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-security</artifactId>
           </dependency>

           <dependency>
               <groupId>org.springframework.boot</groupId>
               <artifactId>spring-boot-starter-test</artifactId>
               <scope>test</scope>
           </dependency>
       </dependencies>

       <dependencyManagement>
```

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

2.2 配置启动项

```
package com.rongshu.springbootadminserver;

import de.codecentric.boot.admin.server.config.EnableAdminServer;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

/**
 * 服务端配置
 * @author yuyang.zhang
 * @date 2019/1/1
 */
@SpringBootApplication
@EnableAdminServer
public class SpringBootAdminServerApplication {

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

}
```

> 添加 `@EnableAdminServer` 启动注解

2.3 安全配置

```
package com.rongshu.springbootadminserver.config;

import de.codecentric.boot.admin.server.config.AdminServerProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerA
dapter;
import
org.springframework.security.web.authentication.SavedRequestAwareAuthenticationSuccessH
andler;
import org.springframework.security.web.csrf.CookieCsrfTokenRepository;
```

```
/**
 * 安全权限配置
 *
 * @author <p>yuyang.zhang<p>
 * @date 2018-12-01 13:50
 * @since 1.0
 */
@Configuration
public class SecuritySecureConfig extends WebSecurityConfigurerAdapter {

    private final String adminContextPath;

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

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

        http.authorizeRequests()
                 // 授予公共访问所有静态资产和登录页面
                .antMatchers(adminContextPath + "/assets/**").permitAll()
                .antMatchers(adminContextPath + "/login").permitAll()
                // 其他必须经过身份验证的请求
                .anyRequest().authenticated()
                .and()
                // 配置登录和注销
                .formLogin().loginPage(adminContextPath +
"/login").successHandler(successHandler).and()
                .logout().logoutUrl(adminContextPath + "/logout").and()
                // 使http基本支持。这是spring的引导管理客户机所需的登记
                .httpBasic().and()
                // 使csrf保护使用cookie
                .csrf()
                // 禁用CRSF-Protection端点弹簧引导管理客户机使用登记
                .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())
                // 禁用CRSF-Protection致动器的端点
                .ignoringAntMatchers(
                        adminContextPath + "/instances",
                        adminContextPath + "/actuator/**"
                );
        // @formatter:on
    }
}
```

2.4 配置application.properties

```
# 配置启动端口
server.port=8080

# 邮件配置
spring.mail.host = smtp.163.com
spring.mail.username=m17612184394@163.com
spring.mail.password=pwd

# admin 发送配置
spring.boot.admin.notify.mail.from=m17612184394@163.com
spring.boot.admin.notify.mail.to=2544267857@qq.com

# security用户名密码
spring.security.user.name=admin
spring.security.user.password=123456
```

2.4 自定义提醒

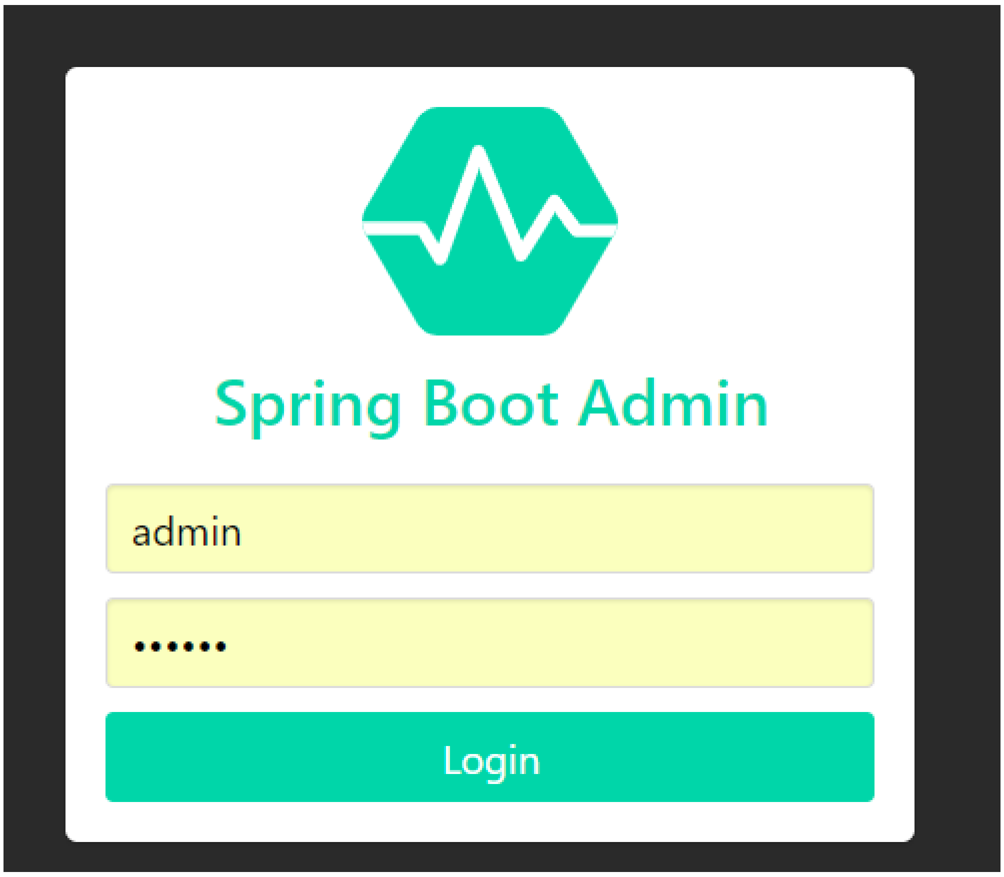
```
package com.rongshu.springbootadminserver.config;

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 de.codecentric.boot.admin.server.notify.LoggingNotifier;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import reactor.core.publisher.Mono;

/**
 * 自定义Notifier
 * @author <p>yuyang.zhang<p>
 * @date 2019-01-03 18:39
 * @since 1.0
 */
public class CustomNotifier  extends AbstractEventNotifier {

    private static final Logger LOGGER =
LoggerFactory.getLogger(LoggingNotifier.class);

    public CustomNotifier(InstanceRepository repository) {
        super(repository);
    }

    @Override
    protected Mono<Void> doNotify(InstanceEvent event, Instance instance) {
        return Mono.fromRunnable(() -> {
            if (event instanceof InstanceStatusChangedEvent) {
                LOGGER.info("Instance {} ({}) is {}",
instance.getRegistration().getName(), event.getInstance(),
                        ((InstanceStatusChangedEvent)
event).getStatusInfo().getStatus());
            } else {
                LOGGER.info("Instance {} ({}) {}",
instance.getRegistration().getName(), event.getInstance(),
                        event.getType());
            }
        });
    }
}
```

> 在#doNotify() 可以实现自己的逻辑。

2.5 启动项目

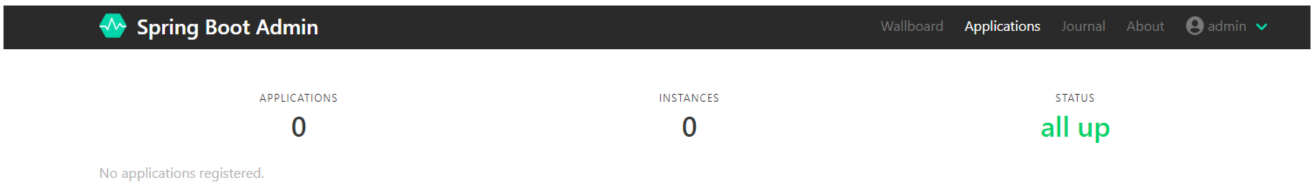
2.5.1 登录界面

http://localhost:8080/



> 输入 `application.properties` 配置的账号和密码

2.5.2 主页



APPLICATIONS
0

INSTANCES
0

STATUS
all up

No applications registered.

3. 注册 Client Applications

3.1 Spring Boot Admin Client

使用 `Spring Boot Admin Client` 进行注册

3.1.1 配置pom.xml

```xml
<properties>
    <java.version>1.8</java.version>
    <spring-boot-admin.version>2.1.1</spring-boot-admin.version>
</properties>

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

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

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

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

    <!-- JMX management -->
    <dependency>
        <groupId>org.jolokia</groupId>
        <artifactId>jolokia-core</artifactId>
    </dependency>

</dependencies>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>de.codecentric</groupId>
            <artifactId>spring-boot-admin-dependencies</artifactId>
            <version>${spring-boot-admin.version}</version>
```

```
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
```

3.1.2 安全配置

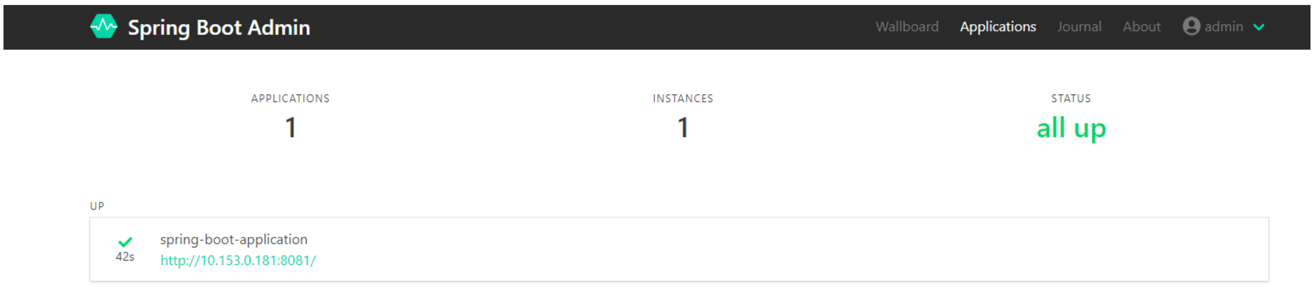
```
package com.rongshu.springbootadminclient.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerA
dapter;

/**
 * 安全权限配置
 * @author <p>yuyang.zhang<p>
 * @date 2018-12-01 13:50
 * @since 1.0
 */
@Configuration
public class SecurityPermitAllConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
                .anyRequest()
                .permitAll()
                .and()
                .csrf()
                .disable();
    }
}
```

3.1.3 配置application.properties

```
# 配置端口号
server.port=8081

# 客户端注册地址
spring.boot.admin.client.url=http://localhost:8080

# ip 地址,默认使用主机名
spring.boot.admin.client.instance.prefer-ip=true

# 配置用户名和密码
spring.boot.admin.client.username=admin
spring.boot.admin.client.password=123456

# 端点暴露,这个很重要,必须要
management.endpoints.web.exposure.include=*
```

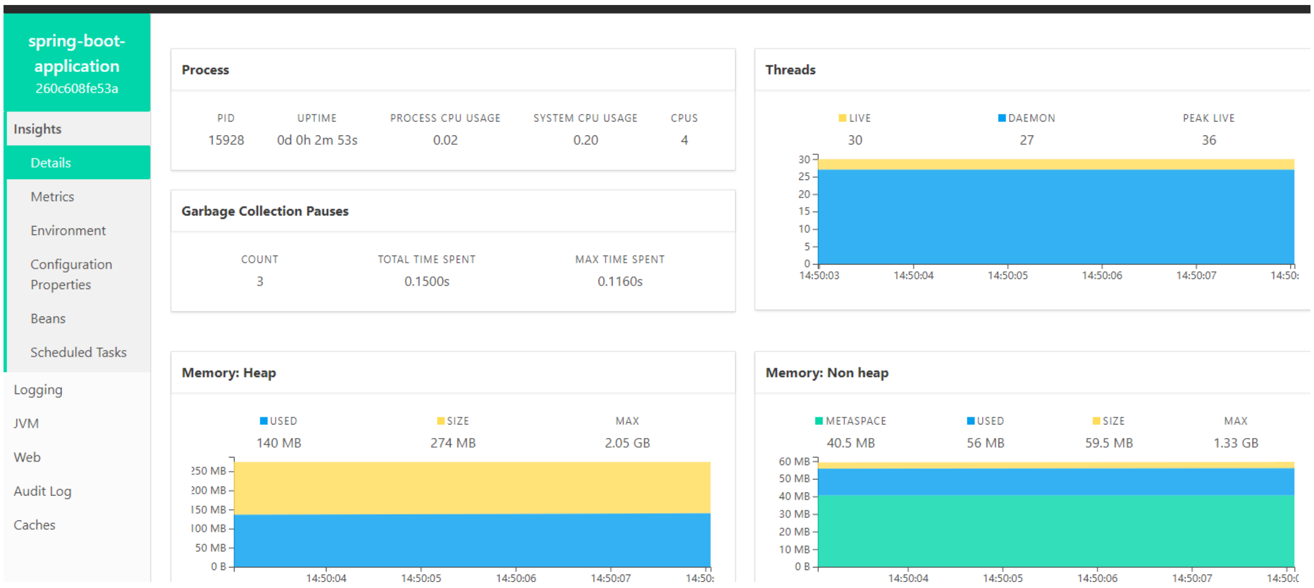
```
# 日志配置
logging.file=/var/log/sample-boot-application.log
logging.pattern.file=%clr(%d{yyyy-MM-dd HH:mm:ss.SSS}){faint} %clr(%5p) %clr(${PID})
{magenta} %clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:)
{faint} %m%n%wEx

# tags
#using the metadata
spring.boot.admin.client.instance.metadata.tags.environment=test
#using the info endpoint
info.tags.environment=test
```

3.1.4 启动项目

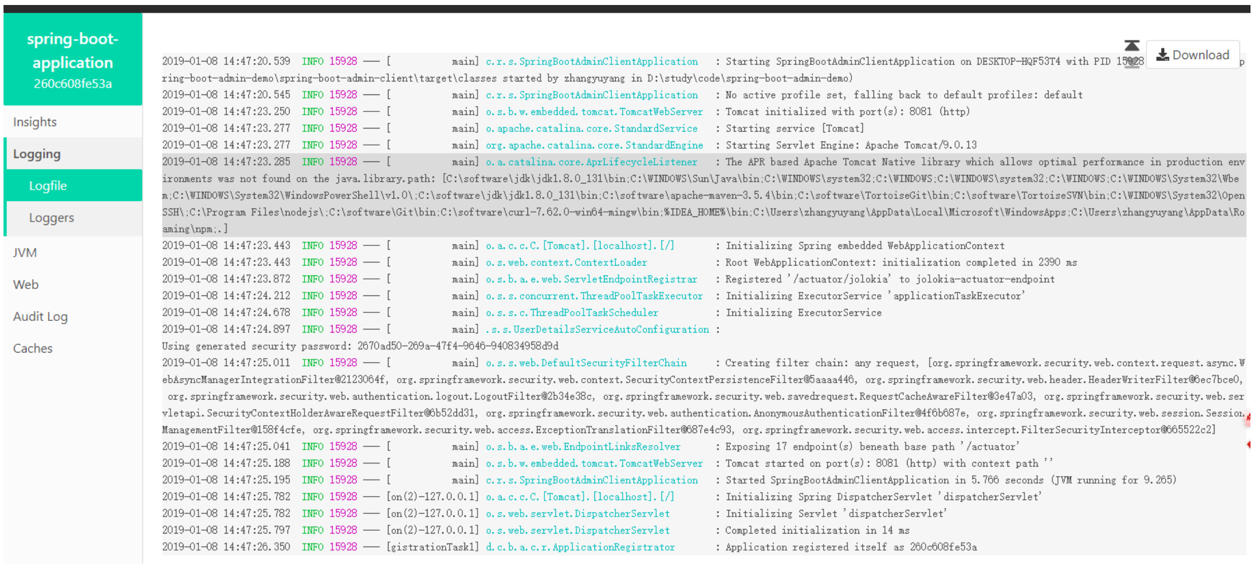


> Server 端 登录页面已经发生改变, 有一个应用已经成功注册。

- 主页展示



- 日志展示



```
2019-01-08 14:47:20.539  INFO 15928 --- [           main] c.r.s.SpringBootAdminClientApplication   : Starting SpringBootAdminClientApplication on DESKTOP-HQF53T4 with PID 15928 p
ring-boot-admin-demo\spring-boot-admin-client\target\classes started by zhangyuyang in D:\study\code\spring-boot-admin-demo)
2019-01-08 14:47:20.545  INFO 15928 --- [           main] c.r.s.SpringBootAdminClientApplication   : No active profile set, falling back to default profiles: default
2019-01-08 14:47:23.250  INFO 15928 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8081 (http)
2019-01-08 14:47:23.277  INFO 15928 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2019-01-08 14:47:23.277  INFO 15928 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet Engine: Apache Tomcat/9.0.13
2019-01-08 14:47:23.285  INFO 15928 --- [           main] o.a.catalina.core.AprLifecycleListener   : The APR based Apache Tomcat Native library which allows optimal performance in production env
ironments was not found on the java.library.path: [C:\software\jdk\jdk1.8.0_131\bin;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbe
m;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\software\jdk\jdk1.8.0_131\bin;C:\software\apache-maven-3.5.4\bin;C:\software\TortoiseGit\bin;C:\software\TortoiseSVN\bin;C:\WINDOWS\System32\Open
SSH\;C:\Program Files\nodejs\;C:\software\Git\bin;C:\software\curl-7.62.0-win64-mingw\bin;%IDEA_HOME%\bin;C:\Users\zhangyuyang\AppData\Local\Microsoft\WindowsApps;C:\Users\zhangyuyang\AppData\Ro
aming\npm;.]
2019-01-08 14:47:23.443  INFO 15928 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2019-01-08 14:47:23.443  INFO 15928 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2390 ms
2019-01-08 14:47:23.872  INFO 15928 --- [           main] o.s.b.a.e.web.ServletEndpointRegistrar   : Registered '/actuator/jolokia' to jolokia-actuator-endpoint
2019-01-08 14:47:24.212  INFO 15928 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2019-01-08 14:47:24.678  INFO 15928 --- [           main] o.s.s.c.ThreadPoolTaskScheduler          : Initializing ExecutorService
2019-01-08 14:47:24.897  INFO 15928 --- [           main] .s.s.UserDetailsServiceAutoConfiguration :
Using generated security password: 2670ad50-269a-47f4-9646-940834958d9d
2019-01-08 14:47:25.011  INFO 15928 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Creating filter chain: any request, [org.springframework.security.web.context.request.async.W
ebAsyncManagerIntegrationFilter@2123064f, org.springframework.security.web.context.SecurityContextPersistenceFilter@5aaaa446, org.springframework.security.web.header.HeaderWriterFilter@6ec7bce0,
 org.springframework.security.web.authentication.logout.LogoutFilter@2b34e38c, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@3e47a03, org.springframework.security.web.ser
vletapi.SecurityContextHolderAwareRequestFilter@6b52dd31, org.springframework.security.web.authentication.AnonymousAuthenticationFilter@4f6b687e, org.springframework.security.web.session.Session
ManagementFilter@158f4cfe, org.springframework.security.web.access.ExceptionTranslationFilter@687e4c93, org.springframework.security.web.access.intercept.FilterSecurityInterceptor@665522c2]
2019-01-08 14:47:25.041  INFO 15928 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 17 endpoint(s) beneath base path '/actuator'
2019-01-08 14:47:25.188  INFO 15928 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with context path ''
2019-01-08 14:47:25.195  INFO 15928 --- [           main] c.r.s.SpringBootAdminClientApplication   : Started SpringBootAdminClientApplication in 5.766 seconds (JVM running for 9.265)
2019-01-08 14:47:25.782  INFO 15928 --- [on(2)-127.0.0.1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2019-01-08 14:47:25.782  INFO 15928 --- [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2019-01-08 14:47:25.797  INFO 15928 --- [on(2)-127.0.0.1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 14 ms
2019-01-08 14:47:26.350  INFO 15928 --- [gistrationTask1] d.c.b.a.c.r.ApplicationRegistrator       : Application registered itself as 260c608fe53a
```

> 日志和数据都是实时监控,动态改变的。

3.2 Spring Cloud Discovery

3.2.1 Eureka Server端

3.2.1.1 配置pom.xml

```
    <parent>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-parent</artifactId>
          <version>2.0.7.RELEASE</version>
          <relativePath/> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.rongshu</groupId>
      <artifactId>spring-boot-eureka-server</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>spring-boot-eureka-server</name>
      <description>Demo project for Spring Boot</description>

      <properties>
          <java.version>1.8</java.version>
          <spring-cloud.version>Finchley.SR2</spring-cloud.version>
      </properties>

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

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

```xml
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

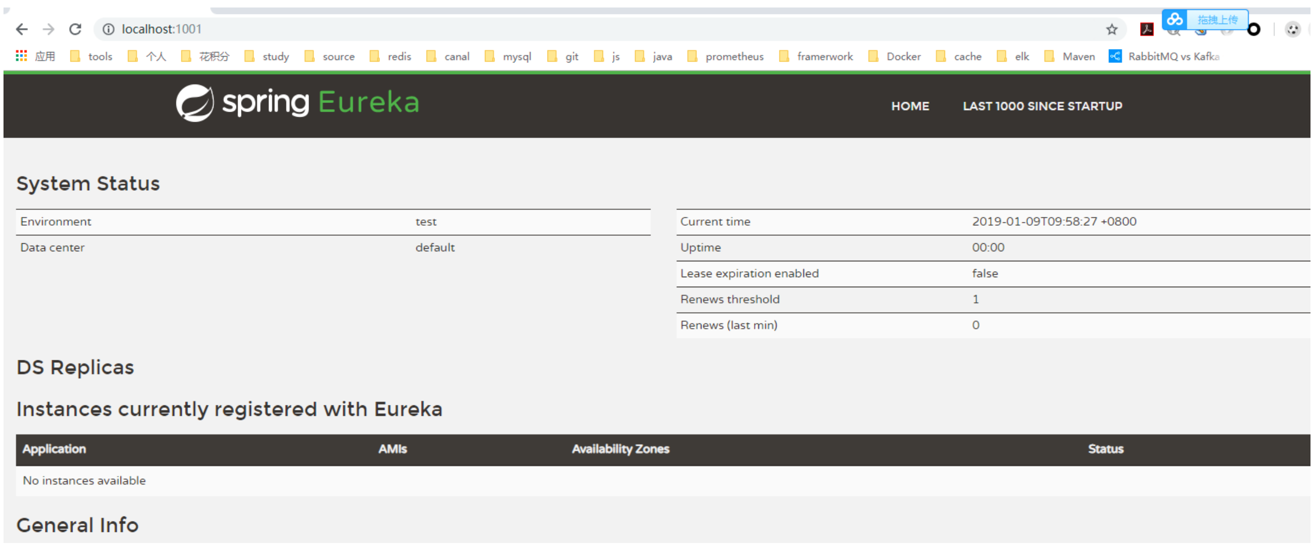
3.2.1.2 配置启动注解

```java
/**
 * Eureka 服务端启动类，添加{@link EnableEurekaServer}注解,启动服务
 * @author yuyang.zhang
 * @date 2019/1/9
 */
@EnableEurekaServer
@SpringBootApplication
public class SpringBootEurekaServerApplication {

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

}
```

3.2.1.3 配置 application.properties

```properties
# 应用名称
spring.application.name=eureka-server
# 端口
server.port=1001
# 主机名
eureka.instance.hostname=localhost
# 取消eureka自我注册功能
eureka.client.register-with-eureka=false
# 不需要检索服务
eureka.client.fetch-registry=false
```

3.2.1.4 启动项目



System Status

Environment	test
Data center	default

Current time	2019-01-09T09:58:27 +0800
Uptime	00:00
Lease expiration enabled	false
Renews threshold	1
Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

3.2.2 Eureka Client端

3.2.2.1 配置pom.xml

```
    <parent>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-parent</artifactId>
          <version>2.0.7.RELEASE</version>
          <relativePath/> <!-- lookup parent from repository -->
      </parent>
      <groupId>spring-boot-eureka-client</groupId>
      <artifactId>demo</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>demo</name>
      <description>Demo project for Spring Boot</description>

      <properties>
          <java.version>1.8</java.version>
          <spring-cloud.version>Finchley.SR2</spring-cloud.version>
      </properties>

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

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

          <dependency>
              <groupId>org.springframework.boot</groupId>
              <artifactId>spring-boot-starter-test</artifactId>
              <scope>test</scope>
```

```xml
        </dependency>

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

        <!-- JMX management -->
        <dependency>
            <groupId>org.jolokia</groupId>
            <artifactId>jolokia-core</artifactId>
        </dependency>
    </dependencies>

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

3.2.2.2 配置启动注解

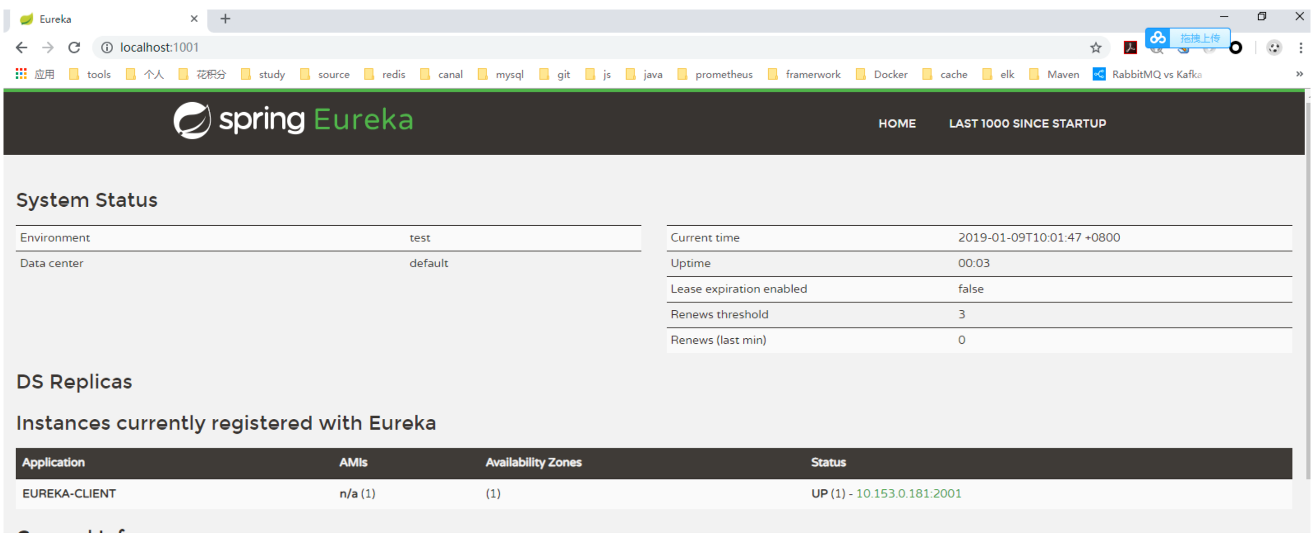
```java
/**
 * Eureka客户端启动类，配置{@link EnableDiscoveryClient}启动客户端
 * @author yuyang.zhang
 * @date 2019/1/9
 */
@EnableDiscoveryClient
@SpringBootApplication
public class DemoApplication {

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

}
```

3.2.2.3 配置application.properties

```
# 应用名称
spring.application.name=eureka-client
# 客户端端口
server.port=2001
# 注册地址
eureka.client.serviceUrl.defaultZone=http://localhost:1001/eureka/

#表示将ip注册到Eureka server
eureka.instance.prefer-ip-address=true

# 将Instance ID设置成IP:端口的形式
eureka.instance.instance-id=${spring.cloud.client.ip-address}:${server.port}

# 端点暴露,这个很重要,必须要
management.endpoints.web.exposure.include=*
```

> 在SpringBoot2.0 之后,不支持驼峰命名的方式 `spring.cloud.client.ip-address` 一定要这么写。

3.2.2.4 启动项目



System Status

Environment	test
Data center	default

Current time	2019-01-09T10:01:47 +0800
Uptime	00:03
Lease expiration enabled	false
Renews threshold	3
Renews (last min)	0

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLIENT	n/a (1)	(1)	UP (1) - 10.153.0.181:2001

> 查看Eureka发现有Eureka客户端注册上去了

3.2.3 Spring Boot Admin Server 端

3.2.3.1 配置pom.xml

```
    <parent>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-starter-parent</artifactId>
          <version>2.0.7.RELEASE</version>
          <relativePath/> <!-- lookup parent from repository -->
      </parent>
      <groupId>com.rongshu</groupId>
      <artifactId>spring-boot-admin-server</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <name>spring-boot-admin-server</name>
```

```xml
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
        <spring-boot-admin.version>2.0.4</spring-boot-admin.version>
        <spring-cloud.version>Finchley.SR2</spring-cloud.version>
    </properties>

    <dependencies>
        <!-- 服务端：带UI界面 -->
        <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>

        <!-- 邮件提醒 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-mail</artifactId>
        </dependency>

        <!-- 权限配置-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>

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

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

    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>de.codecentric</groupId>
                <artifactId>spring-boot-admin-dependencies</artifactId>
                <version>${spring-boot-admin.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>
            </dependencies>
        </dependencyManagement>

        <build>
            <plugins>
                <plugin>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-maven-plugin</artifactId>
                </plugin>
            </plugins>
        </build>
```

3.2.3.2 配置启动注解

```
/**
 * 服务端配置
 * @author yuyang.zhang
 * @date 2019/1/1
 */
@SpringBootApplication
@EnableAdminServer
@EnableDiscoveryClient
public class SpringBootAdminServerApplication {

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

}
```

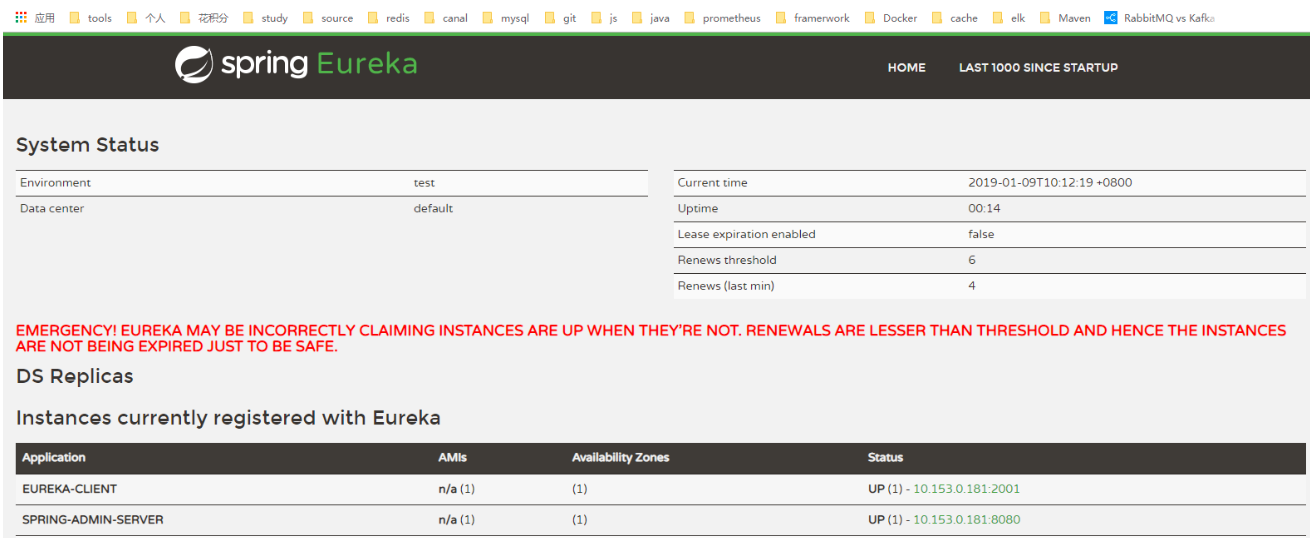
3.2.3.3 配置application.properties

```
# 配置启动端口
server.port=8080

# 邮件配置
spring.mail.host = smtp.163.com
spring.mail.username=m17612184394@163.com
spring.mail.password=Zyy15922313068

# admin 发送配置
spring.boot.admin.notify.mail.from=m17612184394@163.com
spring.boot.admin.notify.mail.to=2544267857@qq.com
```

```
# security用户名密码
spring.security.user.name=admin
spring.security.user.password=123456

# 应用名称
spring.application.name=spring-admin-server
# eureka注册地址
eureka.client.serviceUrl.defaultZone=http://localhost:1001/eureka/
# 将Instance ID设置成IP:端口的形式
eureka.instance.instance-id=${spring.cloud.client.ip-address}:${server.port}
#表示将ip注册到Eureka server
eureka.instance.prefer-ip-address=true
```

3.2.3.4 启动项目

http://localhost:1001/



System Status

Environment	test
Data center	default

Current time	2019-01-09T10:12:19 +0800
Uptime	00:14
Lease expiration enabled	false
Renews threshold	6
Renews (last min)	4

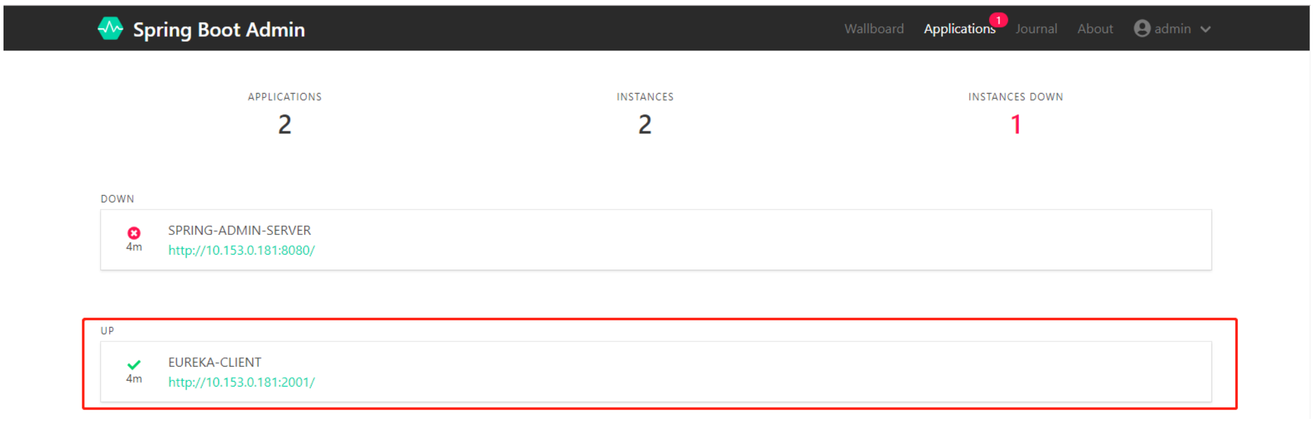
EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.

DS Replicas

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
EUREKA-CLIENT	n/a (1)	(1)	UP (1) - 10.153.0.181:2001
SPRING-ADMIN-SERVER	n/a (1)	(1)	UP (1) - 10.153.0.181:8080

http://localhost:8080/login



APPLICATIONS 2

INSTANCES 2

INSTANCES DOWN 1

DOWN

SPRING-ADMIN-SERVER 4m
http://10.153.0.181:8080/

UP

EUREKA-CLIENT 4m
http://10.153.0.181:2001/

> 登录后发现Eureka的客户端实例,已经可以被springBootAdmin监控了。

参考资料

https://codecentric.github.io/spring-boot-admin/2.0.4/

https://docs.spring.io/spring-boot/docs/2.0.7.RELEASE/reference/htmlsingle/

http://blog.didispace.com/spring-cloud-starter-dalston-1/