(/)

Una guía para Spring Boot Admin

Última modificación: 18 de agosto de 2019

por baeldung (https://www.baeldung.com/author/baeldung/)
(https://www.baeldung.com/author/baeldung/)

Spring Boot (https://www.baeldung.com/category/spring/spring-boot/)

1. Info

Spring Boot Admin (https://github.com/codecentric/spring-boot-admin) es una aplicación web, utilizada para administrar y monitorear aplicaciones Spring Boot. Cada aplicación se considera un cliente y se registra en el servidor de administración. Detrás de escena, la magia viene dada por los puntos finales de Spring Boot Actuator.

En este artículo, vamos a describir los pasos para configurar un servidor Spring Boot Admin y cómo una aplicación se convierte en cliente.

2. Configuración del servidor de administración

En primer lugar, necesitamos crear una aplicación web simple Spring Boot y agregar las siguientes dependencias de Maven (https://search.maven.org/classic/#search%7Cga%7C1%7Cspring-boot-admin-server) :

```
<dependency>
    <groupId>de.codecentric</groupId>
    <artifactId>spring-boot-admin-server</artifactId>
    <version>1.5.4</version>
</dependency>
<dependency>
    <groupId>de.codecentric</groupId>
    <artifactId>spring-boot-admin-server-ui</artifactId>
    <version>1.5.4</version>
</dependency>
```

Después de esto, el *@EnableAdminServer* estará disponible, por lo que lo *agregaremos* a la clase principal, como se muestra en el siguiente ejemplo:

```
@EnableAdminServer
@SpringBootApplication
public class SpringBootAdminServerApplication {

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

En este punto, el servidor está listo para iniciarse y puede registrar aplicaciones de cliente.

3. Configuración de un cliente

Ahora, después de configurar nuestro servidor de administración, podemos registrar nuestra primera aplicación Spring Boot como cliente. Debemos agregar la siguiente dependencia de Maven (https://se

```
<dep

    <artifactId>spring-boot-admin-starter-client</artifactId>
    <version>1.5.4</version>
</dependency>
```

Lo único que queda es configurar el cliente para saber acerca de la URL base del servidor de administración. Para que esto suceda, solo agregamos las siguientes propiedades:

```
spring.boot.admin.url=http://localhost:8080 (http://localhost:8080)
management.security.enabled=false
```

4. Configuración de seguridad

El servidor Spring Boot Admin tiene acceso a los puntos finales sensibles de la aplicación, por **lo que se recomienda que agreguemos alguna configuración de seguridad tanto a la aplicación de administrador como a la de cliente.**

Al principio, nos centraremos en configurar la seguridad del servidor de administración. Debemos agregar las siguientes dependencias de Maven (https://search.maven.org/classic/#search%7Cga%7C1%7Cg%3A%22org.springframework.boot%22%20AND%20a%3A%22spring-boot-starter-security%22) :

```
<dependency>
    <groupId>de.codecentric</groupId>
    <artifactId>spring-boot-admin-server-ui-login</artifactId>
    <version>1.5.4</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Esto habilitará la seguridad y agregará una interfaz de inicio de sesión a la aplicación de administración.

Después, agregaremos una clase de configuración de seguridad como puede ver a continuación:

```
@Configuration
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
          .formLogin()
          .loginPage("/login.html")
          .loginProcessingUrl("/login")
          .permitAll();
        http
          .logout().logoutUrl("/logout");
        http
          .csrf().disable();
        http
          .authorizeRequests()
          .antMatchers("/login.html", "/**/*.css", "/img/**", "/third-party/**")
          .permitAll();
        http
          .authorizeRequests()



}
```

Hay una configuración de seguridad simple, pero después de agregarla, notaremos que el cliente ya no puede registrarse en el servidor.

Para registrar el cliente en el servidor recién asegurado, debemos agregar algo más de configuración en el archivo de propiedades del cliente:

```
spring.boot.admin.username=admin
spring.boot.admin.password=admin
```

Estamos en el punto, donde nuestro servidor de administración está protegido, pero el cliente no. En un sistema de producción, naturalmente, las aplicaciones que estamos tratando de monitorear estarán aseguradas.

Por lo tanto, también agregaremos seguridad al cliente, y notaremos en la interfaz de UI del servidor de administración que la información del cliente ya no está disponible.

Tenemos que agregar algunos metadatos que se enviarán al servidor de administración. El servidor utiliza esta información para conectarse a los puntos finales del cliente:

```
management.security.enabled=true
security.user.name=client
security.user.password=client
spring.boot.admin.client.metadata.user.name=${security.user.name}
spring.boot.admin.client.metadata.user.password=${security.user.password}
```

El envío de credenciales a través de HTTP, por supuesto, no es seguro, por lo que la comunicación debe pasar por HTTPS.

5. Funciones de monitoreo y gestión

Spring Boot Admin se puede configurar para mostrar solo la información que consideramos útil. Solo tenemos que modificar la configuración predeterminada y agregar nuestras propias métricas necesarias:

```
spring.boot.admin.routes.endpoints=env, metrics, trace, jolokia, info, configprops
```

A medida que avanzamos, veremos que hay otras características que se pueden explorar. Estamos hablando de la *administración de beans JMX* usando *Jolokia* y también la administración de *Loglevel* .

Spring Boot Admin también admite la replicación de clúster utilizando Hazelcast. Solo tenemos que agregar la siguiente dependencia de Maven (https://search.maven.org/classic/#search%7Cga%7C1%7Cg%3A%22com.hazelcast%22%20AND%20a%3A%22hazelcast%22) y dejar que la configuración automática haga el resto:

```
<dependency>
    <groupId>com.hazelcast</groupId>
    <artifactId>hazelcast</artifactId>
</dependency>
```

Si queremos [illegible] ada:

```
@Co
public class HazelcastConfig {

    @Bean
    public Config hazelcast() {
        return new Config()
          .setProperty("hazelcast.jmx", "true")
          .addMapConfig(new MapConfig("spring-boot-admin-application-store")
            .setBackupCount(1)
            .setEvictionPolicy(EvictionPolicy.NONE))
          .addListConfig(new ListConfig("spring-boot-admin-event-store")
            .setBackupCount(1)
            .setMaxSize(1000));
    }
}
```

6. Notificaciones

A continuación, analicemos la posibilidad de recibir notificaciones del servidor de administración si sucede algo con nuestro cliente registrado. Los siguientes notificadores están disponibles para la configuración:

- Email
- PagerDuty
- OpsGenie
- Hipchat
- Flojo
- Charlemos

6.1. Notificaciónes de Correo Electrónico

Nos centraremos en configurar notificaciones de correo para nuestro servidor de administración. Para que esto suceda, debemos agregar la dependencia del iniciador de correo (https://search.maven.org/classic/#search%7Cga%7C1%7Cg%3A%22org.springframework.boot%22%20AND%20a%3A%22spring-boot-starter-mail%22) como se muestra a continuación:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
    <version>1.5.4</version>
</dependency>
```

Después de esto, debemos agregar alguna configuración de correo:

```
spring.mail.host=smtp.example.com
spring.mail.username=smtp_user
spring.mail.password=smtp_password
spring.boot.admin.notify.mail.to=admin@example.com
```

Ahora, cada vez que nuestro cliente registrado cambia su estado de UP a OFFLINE o de lo contrario, se envía un correo electrónico a la dirección configurada anteriormente. Para los otros notificadores, la configuración es similar.

6.2. Notificaciones de Hipchat

Como veremos, la integración con Hipchat es bastante sencilla; solo hay algunas propiedades obligatorias para establecer:

```
spring.boot.admin.notify.hipchat.auth-token=<generated_token>
spring.boot.admin.notify.hipchat.room-id=<room-id>
spring.boot.admin.notify.hipchat.url=https://yourcompany.hipchat.com/v2/ (https://yourcompany.hipchat
```

Una vez definidos, notaremos en la sala Hipchat que recibimos notificaciones cada vez que cambia el estado del cliente.

6.3. Configuración de notificaciones personalizadas

Podemos configurar un sistema de notificación personalizado teniendo a nuestra disposición algunas herramientas poderosas para esto. Podemos usar un *notificador recordatorio* para enviar una notificación programada hasta que cambie el estado del cliente.

O tal vez queremos enviar notificaciones a un conjunto filtrado de clientes. Para esto, podemos usar un *notificador de filtrado:*

```java
@Configuration
@EnableScheduling
public class NotifierConfiguration {

    @Autowired private Notifier notifier;

    @Bean
    public FilteringNotifier filteringNotifier() {
        return new FilteringNotifier(notifier);
    }

    @Bean
    @Primary
    public RemindingNotifier remindingNotifier() {
        RemindingNotifier remindingNotifier
          = new RemindingNotifier(filteringNotifier());
        remindingNotifier.setReminderPeriod(TimeUnit.MINUTES.toMillis(5));
        return remindingNotifier;
    }

    @Scheduled(fixedRate = 60_000L)
    public void remind() {
        remindingNotifier().sendReminders();
    }
}
```

7. Conclusión

Este tutorial de introducción cubre los pasos simples que uno tiene que hacer para monitorear y administrar sus aplicaciones Spring Boot usando Spring Boot Admin.

La configuración automática nos permite agregar solo algunas configuraciones menores y, al final, tener un servidor de administración totalmente funcional.

Y, como siempre, el código de muestra de esta guía se puede encontrar en Github (https://github.com/eugenp/tutorials/tree/master/spring-boot-admin) .

▲ el más nuevo ▲ más antiguo ▲ más votado

Huésped

Yarin Zimmerman

Guía genial!
Todos mis clientes están protegidos con seguridad de arranque de primavera y están respaldados por Auth0, lo que significa que uso un token de acceso para acceder a sus API. ¿Puede proporcionar más información sobre cómo SBA debería interactuar con dichos clientes seguros? en este momento puedo registrar mi aplicación pero obtengo 401 para registradores, métricas, env, etc

✚ 0 ━ hace 1 año

Administración

Yo

Hola, Yarin:
Entonces, Auth0 es un sistema administrado con el que te estás integrando, con su propio conjunto de reglas, prácticas y documentación. En general, te diría a sus documentos oficiales: harán un trabajo mucho mejor para ayudarte a integrarte con su sistema.
Saludos,
Eugen.

+ 0 −

hace 1 año

Huésped

Daniel

Hola baeldung, en
primer lugar gracias por el tutorial. No puedo enviar notificaciones por correo electrónico o por falta de tiempo, he descomentado la sección de notificaciones por correo en su ejemplo con mis propios valores de gmail pero todavía no funciona, ¿Alguna idea?

+ 0 −

hace 1 año

(https://www.baeldung.com/author/grzegorz-author/)

Miembro

Grzegorz Piwowarek (http://4comprehension.com)

¿Hay alguna información potencialmente útil en los registros?

+ 0 −

hace 1 año

Huésped

Daniel

Hola grzegorz

finalmente pude hacerlo funcionar hace unos días, pero gracias por tu respuesta

+ 0 −

hace 1 año

¡Los comentarios están cerrados en este artículo!

CATEGORÍAS

PRIMAVERA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTPS://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTPS://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTPS://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTPS://WWW.BAELDUNG.COM/CATEGORY/JSON/JACKSON/)
HTTP DEL LADO DEL CLIENTE (HTTPS://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

KOTLIN (HTTPS://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL DE JAVA "VOLVER A LO BÁSICO" (/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (/JACKSON)
HTTPCLIENT 4 TUTORIAL (/HTTPCLIENT-GUIDE)
RESTO CON SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)
TUTORIAL SPRING PERSISTENCE (/PERSISTENCE-WITH-SPRING-SERIES)
SEGURIDAD CON PRIMAVERA (/SECURITY-SPRING)

ACERCA DE

SOBRE BAELDUNG (/ABOUT)
LOS CURSOS (HTTPS://COURSES.BAELDUNG.COM)
TRABAJO DE CONSULTORÍA (/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
EL ARCHIVO COMPLETO (/FULL_ARCHIVE)
ESCRIBIR PARA BAELDUNG (/CONTRIBUTION-GUIDELINES)
EDITORES (/EDITORS)
NUESTROS COMPAÑEROS (/PARTNERS)
ANUNCIE EN BAELDUNG (/ADVERTISE)

TÉRMINOS DE SERVICIO (/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (/PRIVACY-POLICY)
INFORMACIÓN DE LA COMPAÑÍA (/BAELDUNG-COMPANY-INFO)
CONTACTO (/CONTACT)