(http://baeldung.com)

Una guía de volteretas para la primavera

Última modificación: 23 de marzo de 2018

por Eric Goebelbecker (http://www.baeldung.com/author/eric-goebelbecker/)
(http://www.baeldung.com/author/eric-goebelbecker/)

Primavera (http://www.baeldung.com/category/spring/) +

1. Información general

En este tutorial, echaremos un vistazo a Flips, (https://github.com/Feature-Flip/flips) una biblioteca que implementa indicadores de funciones en forma de potentes anotaciones para las aplicaciones Spring Core, Spring MVC y Spring Boot.

Los indicadores de funciones (o alterna) son un patrón para entregar nuevas funciones de forma rápida y segura. Estos conmutadores nos permiten modificar el comportamiento de la aplicación **sin cambiar o implementar código nuevo.** El blog de Martin Fowler tiene un artículo muy informativo sobre las banderas de características aquí (https://martinfowler.com/articles/feature-toggles.html) .

2. Dependencia de Maven

Antes de comenzar, necesitamos agregar la biblioteca Flips a nuestro *pom.xml:*

```
<dependency>
    <groupId>com.github.feature-flip</groupId>
    <artifactId>flips-core</artifactId>
    <version>1.0.1</version>
</dependency>
```

Maven Central tiene la última versión de la biblioteca (https://search.maven.org/#search%7Cga%7C1%7Cg%3A%22com.github.feature-flip%22) , y el proyecto Github está aquí (https://github.com/Feature-Flip/flips) .

Por supuesto, también debemos incluir un Spring:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <version>1.5.10.RELEASE</version>
</dependency>
```

Desde tirones todavía no es compatible con la versión 5.x primavera, vamos a usar la versión más reciente de la primavera de arranque en la rama 4.x (https://search.maven.org/#artifactdetails%7Corg.springframework.boot%7Cspring-boot-starter-web%7C1.5.10.RELEASE%7Cjar) .

3. Un servicio REST simple para flips

Vamos a armar un proyecto simple de Spring Boot para agregar y alternar nuevas características y banderas.

Nuestra aplicación REST proporcionará acceso a los recursos de *Foo* :

```
public class Foo {
    private String name;
    private int id;
}
```

Simplemente crearemos un *Servicio* que mantenga una lista de *Foos* :

```
@Service
public class FlipService {

    private List<Foo> foos;

    public List<Foo> getAllFoos() {
        return foos;
    }

    public Foo getNewFoo() {
        return new Foo("New Foo!", 99);
    }
}
```

Nos referiremos a los métodos de servicio adicionales a medida que avanzamos, pero este fragmento debería ser suficiente para ilustrar lo que *FlipService* hace en el sistema.

Y, por supuesto, tenemos que crear un controlador:

```
@RestController
public class FlipController {

    private FlipService flipService;

    // constructors

    @GetMapping("/foos")
    public List<Foo> getAllFoos() {
        return flipService.getAllFoos();
    }
}
```

4. Funciones de control basadas en la configuración

El uso más básico de Flips es habilitar o deshabilitar una función en función de la configuración. Flips tiene varias anotaciones para esto.

4.1. Propiedad del entorno

Imaginemos que agregamos una nueva capacidad a *FlipService* ; recuperando *foos* por su id.

Agreguemos la nueva solicitud al controlador:

```
@GetMapping("/foos/{id}")
@FlipOnEnvironmentProperty(
  property = "feature.foo.by.id",
  expectedValue = "Y")
public Foo getFooById(@PathVariable int id) {
    return flipService.getFooById(id)
      .orElse(new Foo("Not Found", -1));
}
```

La *propiedad @FlipOnEnvironmentProperty* controla si esta API está disponible o no.

En pocas palabras, cuando *feature.foo.by.id* es *Y* , podemos hacer solicitudes por Id. Si no está (o no está definido en absoluto) Flips desactivará el método API.

Si una característica no está habilitada, Flips lanzará *FeatureNotEnabledException* y Spring devolverá "No implementado" al cliente REST.

Cuando llamamos a la API con la propiedad establecida en *N* , esto es lo que vemos:

```
Status = 501
Headers = {Content-Type=[application/json;charset=UTF-8]}
Content type = application/json;charset=UTF-8
Body = {
    "errorMessage": "Feature not enabled, identified by method
      public com.baeldung.flips.model.Foo
      com.baeldung.flips.controller.FlipController.getFooById(int)",
    "className":"com.baeldung.flips.controller.FlipController",
    "featureName":"getFooById"
}
```

Como se esperaba, Spring atrapa la *FeatureNotEnabledException* y devuelve el estado 501 al cliente.

4.2. Perfil activo

Spring nos ha dado la capacidad de mapear beans en diferentes perfiles (http://www.baeldung.com/spring-profiles) , como *dev* , *test* o *prod* . Ampliar esta capacidad para mapear banderas de características para el perfil activo tiene sentido intuitivo.

Veamos cómo las funciones están habilitadas o deshabilitadas en función del perfil (http://www.baeldung.com/spring-profiles) activo de Spring (http://www.baeldung.com/spring-profiles) :

```
@RequestMapping(value = "/foos", method = RequestMethod.GET)
@FlipOnProfiles(activeProfiles = "dev")
public List getAllFoos() {
    return flipService.getAllFoos();
}
```

La anotación *@FlipOnProfiles* acepta una lista de nombres de perfil. Si el perfil activo está en la lista, se puede acceder a la API.

4.3. Expresiones de primavera

El lenguaje de expresión de Spring (SpEL) (http://www.baeldung.com/spring-expression-language) es el poderoso mecanismo para manipular el entorno de tiempo de ejecución. Flips nos ayuda a alternar funciones también.

@FlipOnSpringExpression alterna un método basado en una expresión SpEL que devuelve un booleano.

Usemos una expresión simple para controlar una nueva característica:

```
@FlipOnSpringExpression(expression = "(2 + 2) == 4")
@GetMapping("/foo/new")
public Foo getNewFoo() {
    return flipService.getNewFoo();
}
```

4.4. Inhabilitar

Para deshabilitar una característica por completo, use *@FlipOff* :

```
@GetMapping("/foo/first")
@FlipOff
public Foo getFirstFoo() {
    return flipService.getLastFoo();
}
```

En este ejemplo, *getFirstFoo ()* es completamente inaccesible.

Como veremos a continuación, podemos combinar anotaciones de Flips, lo que posibilita el uso de *@FlipOff* para deshabilitar una función en función del entorno u otros criterios.

5. Funciones de control con fecha / hora

Los flips pueden alternar una función según una fecha / hora o el día de la semana. Atar la disponibilidad de una nueva función para el día o la fecha tiene ventajas obvias.

5.1. Fecha y hora

@FlipOnDateTime acepta el nombre de una propiedad que está formateada en formato ISO 8601 (https://www.iso.org/iso-8601-date-and-time-format.html) .

Establezcamos una propiedad que indique una nueva característica que estará activa el 1 de marzo:

```
first.active.after=2018-03-01T00:00:00Z
```

Luego, escribiremos una API para recuperar el primer Foo:

```
@GetMapping("/foo/first")
@FlipOnDateTime(cutoffDateTimeProperty = "first.active.after")
public Foo getFirstFoo() {
    return flipService.getLastFoo();
}
```

Flips comprobará la propiedad nombrada. Si la propiedad existe y ha pasado la fecha / hora especificada, la característica está habilitada.

5.2. Día de la semana

La biblioteca proporciona *@FlipOnDaysOfWeek* , que es útil para operaciones como las pruebas A / B:

```
@GetMapping("/foo/{id}")
@FlipOnDaysOfWeek(daysOfWeek={DayOfWeek.MONDAY, DayOfWeek.WEDNESDAY})
public Foo getFooByNewId(@PathVariable int id) {
    return flipService.getFooById(id).orElse(new Foo("Not Found", -1));
}
```

getFooByNewId () solo está disponible los lunes y miércoles.

6. Reemplazar un frijol

La activación y desactivación de métodos es útil, pero es posible que deseemos introducir un nuevo comportamiento a través de nuevos objetos. *@FlipBean* dirige a Flips para llamar a un método en un nuevo bean.

Una anotación Flips puede funcionar en cualquier Spring *@Component.* Hasta ahora, solo hemos modificado nuestro *@RestController* , intentemos modificar nuestro *Servicio.*

Creamos un nuevo servicio con un comportamiento diferente de *FlipService* :

```
@Service
public class NewFlipService {
    public Foo getNewFoo() {
        return new Foo("Shiny New Foo!", 100);
    }
}
```

Reemplazaremos el *getNewFoo ()* del servicio *anterior* con la nueva versión:

```
@FlipBean(with = NewFlipService.class)
public Foo getNewFoo() {
    return new Foo("New Foo!", 99);
}
```

Flips dirigirá las llamadas a *getNewThing ()* a *NewFlipService. @FlipBean* es otro *botón* que es más útil cuando se combina con otros. Miremos eso ahora.

7. Combinación de Alternar

Combinamos los botones al especificar más de uno. Flips los evalúa en secuencia, con la lógica implícita "Y". Por lo tanto, todos deben ser verdaderos para alternar la función.

Combinemos dos de nuestros ejemplos anteriores:

```
@FlipBean(
  with = NewFlipService.class)
@FlipOnEnvironmentProperty(
  property = "feature.foo.by.id",
  expectedValue = "Y")
public Foo getNewFoo() {
    return new Foo("New Foo!", 99);
}
```

Hemos hecho uso del nuevo servicio configurable.

8. Conclusión

En esta breve guía, creamos un servicio Spring Boot simple y activamos y desactivamos las API usando anotaciones Flips. Vimos cómo se alternaban las características usando la información de configuración y la fecha / hora, y también cómo las características pueden alternarse intercambiando los beans en el tiempo de ejecución.

Las muestras de código, como siempre, pueden encontrarse en GitHub (https://github.com/eugenp/tutorials/tree/master/spring-4) .

Ricardo

Mi nueva anotación favorita - @FlipOff

Huésped

+ 0 − Responder

Hace 15 días

Yo (http://www.baeldung.com/author/thefather/)

Si exacto 😉

(http://www.baeldung.com/author/thefather/)

Administración

+ 0 − Responder

Hace 15 días

CATEGORÍAS

PRIMAVERA (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
DESCANSO (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SEGURIDAD (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCIA (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)
HTTPCLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
KOTLIN (HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

TUTORIAL "VOLVER A LO BÁSICO" DE JAVA (HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL)
JACKSON JSON TUTORIAL (HTTP://WWW.BAELDUNG.COM/JACKSON)
TUTORIAL DE HTTPCLIENT 4 (HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE)
REST CON SPRING TUTORIAL (HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/)
TUTORIAL DE SPRING PERSISTENCE (HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/)

SEGURIDAD CON SPRING (HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING)

ACERCA DE

ACERCA DE BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)
LOS CURSOS (HTTP://COURSES.BAELDUNG.COM)
TRABAJO DE CONSULTORÍA (HTTP://WWW.BAELDUNG.COM/CONSULTING)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
EL ARCHIVO COMPLETO (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)
ESCRIBIR PARA BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)
CONTACTO (HTTP://WWW.BAELDUNG.COM/CONTACT)
INFORMACIÓN DE LA COMPAÑÍA (HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
TÉRMINOS DE SERVICIO (HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
POLÍTICA DE PRIVACIDAD (HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY)
EDITORES (HTTP://WWW.BAELDUNG.COM/EDITORS)
KIT DE MEDIOS (PDF) (HTTPS://S3.AMAZONAWS.COM/BAELDUNG.COM/BAELDUNG+-+MEDIA+KIT.PDF)