










Anuncios

 **You don't need to go to an office to write code. Work with us!** [APPLY](#)

REPORT THIS AD

**devs4j**

EL MEJOR SITIO WEB SOBRE PROGRAMACIÓN EN ESPAÑOL.

ESPAÑOL

Spring framework 5 : AOP After returning Advice

BY RAIDENTRANCE ON MARZO 4, 2019 · ([DEJA UN COMENTARIO](#))[Rate This](#)

Como vimos en el post pasado <https://devs4j.com/2019/02/28/spring-framework-5-aop-conceptos-basicos/> (<https://devs4j.com/2019/02/28/spring-framework-5-aop-conceptos-basicos/>) existen diferentes tipos de advices, en este post nos enfocaremos en After returning Advice.

Para poder seguir estos ejemplos es necesario crear un proyecto spring boot simple.

Creación de un servicio de spring

El primer paso para entender como funcionan los advices será crear un servicio de spring, este objeto será nuestro **Target object**.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;

/**
```

```
* @author raidentrance
*
*/
@Service
public class HelloWorldService {

    private static final Logger log = LoggerFactory.getLogger(HelloWorldService.class);

    public void print() {
        log.info("Hello world");
    }

}
```

Como vemos nuestro servicio es solo una clase llamada **HelloWorldService** con un método llamado **print()**.

After returning Advice

En este ejemplo interceptaremos las peticiones a la clase HelloWorldService en su método print utilizando un After Returning Advice, veamos el siguiente ejemplo:

```
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.AfterReturning;
import org.aspectj.lang.annotation.Aspect;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;

/**
 * @author raidentrance
 *
 */
@Aspect
@Component
public class AfterReturningAdviceExample {

    private static final Logger log = LoggerFactory.getLogger(AfterReturningAdviceExample.class);

    @AfterReturning("execution(* com.devs4j.service.HelloWorldService.print())")
    public void logAfterReturning(JoinPoint joinPoint) {
        log.info("AfterReturningAdviceExample: " + joinPoint.toString());
    }

}
```

```
        log.info("After returning example");  
    }  
  
}
```

Del código anterior podemos analizar los siguientes puntos:

- La clase esta anotada con **@Component** y **@Aspect** esto permite a spring identificarlo como un bean y como un aspecto.
- La anotación **@AfterReturning** nos permite utilizar un After returning advice.
- Los advices reciben como parámetro un **Pointcut** el cual define los objetos que serán afectados por el Advice (Explicaremos Pointcut expression language en otro post).
- El método recibe como parámetro un objeto que implementa la interfaz **JoinPoint**, esto nos permite acceder a información del JoinPoint que se interceptó.
- Lo único que hace nuestro aspecto es imprimir el mensaje **After returning example**
- Los Advices After returning se ejecutan después de que se ejecutó el método interceptado siempre y cuando no haya habido una excepción.

Una vez que tenemos listo nuestro aspecto el siguiente paso será probarlo, para esto crearemos la siguiente clase:

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ConfigurableApplicationContext;  
  
import com.devs4j.service>HelloWorldService;  
  
@SpringBootApplication  
public class Devs4jSpringAopApplication {  
  
    public static void main(String[] args) {  
        ConfigurableApplicationContext appli  
            args);  
        HelloWorldService helloWorldService  
            helloWorldService.print();  
    }  
  
}
```

Del código anterior podemos analizar los siguientes puntos:

- Obtenemos un bean del tipo **HelloWorldService**
- Ejecutamos el método `print()`

Salida:

```
2019-03-01 10:57:37.307 INFO 72532 --- [
2019-03-01 10:57:37.308 INFO 72532 --- [
```

Como vemos se imprimió el mensaje **After returning example** después de **Hello world**, esto nos indica que el aspecto que utiliza after returning advice se ejecuta después del método `print` que se invocó de la clase `HelloWorldService`.

Para estar al pendiente sobre nuestro contenido nuevo síguenos en nuestras redes sociales <https://www.facebook.com/devs4j/>

(<https://www.facebook.com/devs4j/>) y <https://twitter.com/devs4j> (<https://twitter.com/devs4j>).

Autor: Alejandro Agapito Bautista

Twitter: @raidentrance

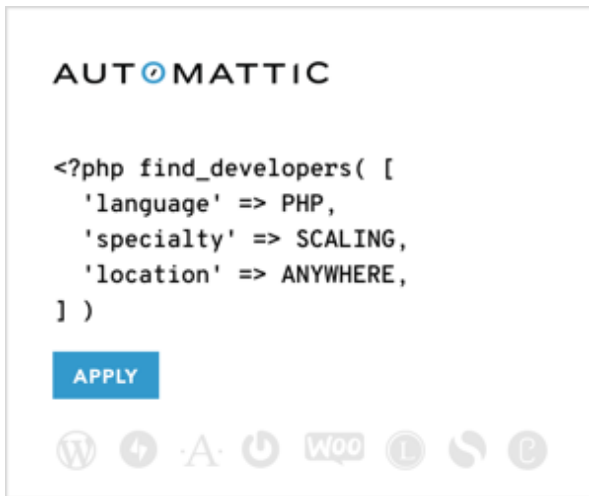
(<https://geeksjavamexico.wordpress.com/mentions/raidentrance/>)

Contacto:raidentrance@gmail.com

Anuncios



REPORT THIS AD



REPORT THIS AD



Publicado por raidentrance

Soy @raidentrance en Twitter y en Github, soy egresado de la Facultad de Ingeniería de la UNAM, cuento con 8 certificaciones en diferentes áreas del desarrollo de software, me gustan las cervezas y soy Geek. Ver todas las entradas de raidentrance (<https://devs4j.com/author/raidentrance/>)

