¿Qué hace un Aspecto?

La programación orientada a aspecto hace un excelente trabajo para que el código sea más reutilizable, compacto, desacoplado y modular.

Se ocupa de estas tareas que son transversales a nuestro código y lógica de negocio.

Es decir son fragmentos de códigos reutilizables que interceptan un método en cuestión para agregar funcionalidad transversal antes o después.

Autenticación

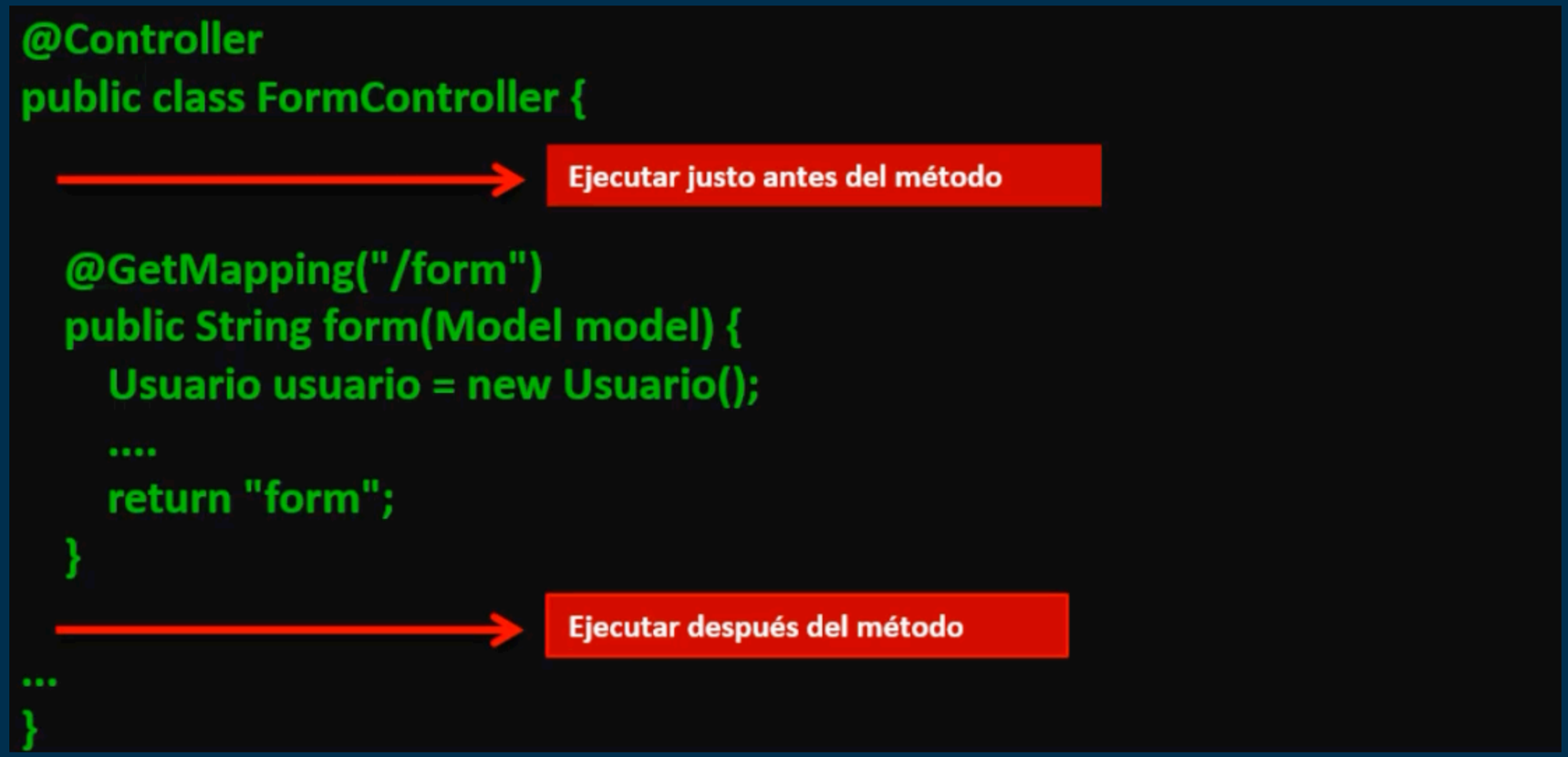
Autorización

Logger

Transacción

Se pueden aplican a los métodos del controlador, servicios, repositorios o Daos, etc.

¿Qué hace la AOP?



Envuelve al método del controlador

Conceptos y terminología

- Aspects (Aspectos): Un Aspects es una modularización de una preocupación (tarea o proceso) que corta a través de múltiples objetos y métodos, un interceptor.

- Join point (Punto de Cruce o Unión): es un punto en la ejecución del programa, ejemplo en una invocación de método o manejo de una excepción En Spring AOP, un join point representa una ejecución de un método.

- Advice (Consejo o asesoramiento): Código que se ejecuta en un particular punto de unión (joinpoint), hay diferentes tipos, antes/before advice, se ejecuta antes del punto de unión (joinpoint), después/after advice, se ejecuta después del punto de unión (joinpoint), alrededor/around advice, se ejecuta alrededor del punto de unión (joinpoint).

- Pointcuts (Puntos de Corte): Un punto de corte (pointcut) es una expresión que agrupa uno o más puntos de unión (join points). Un Pointcuts puede estar compuesto en base a relaciones complejas definidas por expreciones regulares (patrón), para restringir y afinar aún más cuando se deba ejecutar un advice o consejo.

Qué es un Advice

- Código que se ejecuta en un particular punto de unión (joinpoint) y esta asociado a una expresión pointcut (similar a las expresiones regulares), y es ejecutado antes, después, o alrededor del método que coincida con ese patrón (match) definido en el pointcut.

- La expresión pointcut pueden ser definidas en dos formas, una expresión pointcut definida en mismo lugar que el Advice o una referencia hacia un pointcut reusable o nombrado (named).

Tipos de Advice

- Before advice: Consejo que se ejecuta antes de una ejecución del método.

- After returning advice: Consejo a ser ejecutado después de que una ejecución de método sea completado con normalidad, es decir haya retornado el valor/objeto.

- After throwing advice: Consejo a ser ejecutado si un método lanza una excepción.

- After (finally) advice: Consejos a ser ejecutado independientemente de si un método retorna/finaliza normalmente o si lanza una excepción.

- Around advice: Consejo que envuelven a una invocación de método, este es el más poderoso de los tipos de advice. Around advice puede realizar tareas personalizadas antes y después de la invocación del método.

Cómo funciona

Before Advice (antes del método)

Before advice es definido en una clase aspect usando la anotación @Before

Ejemplo opción #1 – La expresión Pointcut es definida en el mismo lugar

```
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
public class EjemploAntes {

   @Before("execution(* com.abc.miapp.dao.*.*(..))")
   public void compruebaAcceso() {
      // ...
   }
}
```

aspecto

Expresión Pointcut

advice