

الجوانب لـSpring برمجة إلـى الـTوجه

بتقسيم لك تسمح الـTوجه عمل إطار في قوية ميزة هو استخدام أساسيات عبر سارشـدـك الـMـاـعـاـمـلـاتـ. إدارة أو الأمان، الـTـسـجـيـلـ، مثل الـMـتـقـاطـعـةـ الـCـضـائـيـ. عمليـنـجـ

1. الرؤية المفاهيم فـهـمـ

الـTـسـجـيـلـ. مثل مـتـقـاطـعـةـ قـضـيـةـ تـحـوـيـ وـحـدـةـ: إـلـىـ تـحـتـاجـ مـاـ إـلـيـكـ،ـ فـيـ الـغـوصـ قـبـلـ يـجـبـ أـيـنـ يـحـدـدـ شـرـطـ: طـرـيـقـةـ. تـنـفـيـذـ بـعـدـ أوـقـبـلـ مـثـلـ مـعـيـنـةـ نـقـطـةـ فـيـ الـجـانـبـ يـتـخـذـهـ الـذـيـ الـجـاءـ مـثـلـ الـجـانـبـ تـطـبـيـقـيـمـكـنـ حـيـثـ الـBـرـنـاـمـجـ تـنـفـيـذـ فـيـ نـقـطـةـ: مـحـدـدـةـ. فـيـاتـ أوـطـرـقـ مـثـلـ الـNـaـvـeـ مـحـدـدـةـ. اـسـتـدـعـاءـ طـرـيـقـةـ. اـسـتـدـعـاءـ

جـوانـبـهـاـ. لـتـطـبـيـقـ بـالـوـكـالـاتـ بـكـ الـخـاصـةـ الـكـائـنـاتـ يـلـفـ أـنـهـ يـعـنـيـ مـمـاـ الـوـكـالـةـ، عـلـىـ مـبـنـيـ

2. مشروعك إعدادـ

الـA~u~t~o~f~. اـعـتـمـادـ أـضـفـ مـعـ الـP~o~m~. مـشـرـوعـ أـوـ الـP~o~m~ مـشـرـوعـ إـلـىـ سـتـخـدـامـ: xml <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-aop</artifactId> </dependency> تمـكـيـنـ مـاـ عـادـةـ تـكـوـيـنـكـ فـيـ تـمـكـيـنـ مـاـ بـاسـتـخـدـامـ صـرـيـحـ بـشـكـلـ تـمـكـيـنـهـ يـمـكـنـكـ وـلـكـنـ،ـ @EnableAspectJAutoProxyـ.

3. جانب إنـشـاءـ

بـاسـتـخـدـامـ جـانـبـ تـعـرـيـفـ يـمـكـنـكـ هـكـذاـ:

الـTـسـجـيـلـ جـانـبـ مـثـالـ:

```
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class LoggingAspect {

    /**
     * Before advice
     */
    @Before("execution(* com.example.myapp.service.*.*(..))")
    public void logBeforeMethod() {
        System.out.println("Logging Aspect: " + new Date());
    }
}
```

```

        System.out.println( "           " ; (
}

// :
@After("execution(* com.example.myapp.service.*.*(..))")
public void logAfterMethod() {
    System.out.println( "           " ; (
}
}

```

- @Aspect: الـ Aspect يحدد كجـانـبـ الـ فـيـةـ هـذـهـ يـحـددـ
- @Component: كـكـيـانـ يـسـجـلـهـاـ
- execution(* com.example.myapp.service.*.*(..)) يـحـددـ نـقـطـةـ تـعـبـيـرـ serviceـ حـزـمـةـ تـحـتـ فـيـةـ أـيـ طـرـيـقـةـ أـيـ مـتـغـيـرـاتـ أـيـ وـعـوـدـةـ نـوـعـ أـيـ مـعـ

4. أنواع الصـائـحـ الـنـصـائـحـ

بعد دـيـعـمـلـ @After: الـ Afterـ طـابـقـةـ الـ طـرـيـقـةـ تـنـفـيـذـ قـبـلـ يـعـمـلـ @Before: الـ Beforeـ مـنـ أـنـ وـاعـ عـدـةـ يـقـدـمـ

أـلـقـتـ إـذـاـ يـعـمـلـ @AfterReturning: بـنـجـاحـ الـ طـرـيـقـةـ عـوـدـةـ بـعـدـ يـعـمـلـ @AfterThrowing: أـلـفـشـلـ أوـ الـ نـجـاحـ عـنـ الـ نـظـرـ بـغـضـ

أـقـوـىـ الـ تـنـفـيـذـ فـيـ بـتـحـكـمـ لـ كـيـسـمـ مـمـاـ الـ طـرـيـقـةـ،ـ يـلـفـ @Around: اـسـتـثـنـاءـ الـ طـرـيـقـةـ

حـولـ نـصـيـحـةـ مـثـالـ:

```

import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class PerformanceAspect {

    @Around("execution(* com.example.myapp.service.*.*(..))")
    public Object measureTime(ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        Object result = joinPoint.proceed(); //
        long end = System.currentTimeMillis();
        System.out.println( " : " + (end - start) + "ms");
        return result;
    }
}

```

- اعترافات. يتم الالتفاف حول الطريقة بمثابة `ProceedingJoinPoint`.
- التنفيذ `proceed()`: الـ `proceed()` يستدعي الطريقة الأصلية.

نقطة عبارات 5.

الشائع: الصيغة النصية. تطبيق يتم حيث النقطة تعريف `execution(modifiers? return-type declaring-type? method-name(params) throws?)`. مثال: `execution(public String com.example.myapp.service.MyService.get*(...))` إلی تعود والتي `MyService` في `com.example.myapp.service` بـ `get*`. تبدأ الـ `MyService` العامة الـ `String` طرق مع يتطابق.

النقطة: دمج أي صيغة ممكن.

```
@Pointcut("execution(* com.example.myapp.service.*.*(..))")
```

```
public void serviceMethods() {}
```

```
@Before("serviceMethods()")
```

```
public void logBeforeService() {
    System.out.println( "           " ; (
}
```

الطريقة تفاصيل إلی الوصول 6.

البيانات: أو التوقيعات، الطريقة متغيرات إلی الوصول يمكن.

```
@Before("execution(* com.example.myapp.service.*.*(..))")
```

```
public void logMethodDetails(JoinPoint joinPoint) {
    String methodName = joinPoint.getSignature().getName();
    Object[] args = joinPoint.getArgs();
    System.out.println( "           " + methodName + " " + args.length + " " ; (
}
```

والتجربة العمل 7.

- الطرق. بعض مع `MyService` مثل خدمة فئة إنشاء.
- بـ `public`. الخواص `modifiers` تطبيق تشغيل.
- المطابقة. الطرق على تلقائيًا الـ `joinPoint` تطبيق سيتم.

خدمة: مثال

```
@Service
```

```
public class MyService {
    public String sayHello(String name) {
```

```
    return " " + name;  
}  
}
```

الأداء. أو الـ `sayHello("Alice")` تـ `myService` تـ `شـ يـ طـ سـ يـ تـ مـ` عـ `نـ دـ مـ`

نـ صـ اـيـ حـ

بـ `حـ كـ مـ`. اـ `سـ تـ خـ دـ مـ` لـ `ذـ اـ لـ وـ كـ الـ اـتـ`، بـ `سـ بـ بـ عـ بـ ءـ اـ` يـ `ضـ يـ فـ` :`اـ لـ أـ دـاءـ` `تـ دـ يـ رـ هـ اـ لـ اـ لـ تـ يـ لـ لـ أـ شـ يـ اـءـ بـ الـ نـ سـ بـ ةـ` فـ `قـ طـ`. `يـ دـ يـ رـ هـ اـ لـ اـ لـ تـ يـ اـ لـ كـ اـئـ نـ اـتـ عـ لـ لـ يـ` يـ `عـ مـ لـ` :`اـ لـ مـ نـ طـ قـةـ` `أـ قـ وـ وـ يـ`. `بـ دـ يـ لـ اـ لـ تـ شـ خـ يـ صـ تـ سـ جـ يـ لـ` تـ `مـ كـ يـ نـ` :`اـ لـ تـ شـ خـ يـ صـ` `اـ لـ وـ كـ الـ اـتـ`. تـ `طـ بـ يـ قـ كـ يـ فـ يـةـ لـ رـ ؤـ يـ ةـ` لـ `org.aop.framework.spring` تـ `سـ جـ يـ لـ` تـ `مـ كـ يـ نـ` :`اـ لـ تـ شـ خـ يـ صـ`

أـ `كـ بـ رـ`! بـ `شـ كـ لـ` الـ `تـ فـ سـ يـ رـ` وـ `سـ أـ عـ دـ لـ` أـ `خـ بـ رـ نـ يـ` معـ `يـ نـ`، تـ `نـ فـ يـ ذـ فـ يـ` مـ `سـ اـعـ دـةـ` إـ `لـ يـ` تـ `حـ تـ اـجـ` أو مـ `حـ دـ دـةـ` اـ `سـ تـ خـ دـ اـمـ` حـ `اـ لـ ةـ` لـ `دـ يـ كـ` كـ `انـ` إـ `ذـ`