الجوانب□ إلى التوجه□ برمجة □□□ □□□□□□

□□□□□□ □□□ □□□□□□□□□□□□□□□□ □□□□□□□□□□□ هو ميزة قوية في إطار عمل □□□□□□ التي تسمح لك بتقسيم القضايا المتقاطعة مثل التسجيل، الأمن، أو إدارة المعاملات. سأرشدك عبر أساسيات استخدام □□□□ □□□□□□ □□□ باستخدام نهج عملي.

1. فهم المفاهيم الرئيسية

قبل الغوص في □□□□، إليك ما تحتاج إلى معرفته: □ □□□□□□: وحدة تحوي قضية متقاطعة مثل□ التسجيل□. □ □□□□□□: الإجراء الذي يتخذه الجانب في نقطة معينة مثل□ □قبل□ أو□ بعد□ تنفيذ□ طريقة□. □ □□□□□□□□□: شرط يحدد أين يجب تطبيق النصيحة مثل□ طرق أو فئات محددة□. □ □□□□ □□□□□: نقطة في تنفيذ البرنامج حيث يمكن تطبيق الجانب مثل□ استدعاء طريقة□.

□□□□□□ □□□ مبني على الوكالة، مما يعني أنه يلف الكائنات الخاصة بك بالوكالات لتطبيق جوانبها.

2. إعداد مشروعك

لاستخدام □□□ □□□□□□، ستحتاج إلى: □ مشروع □□□□□□ □□□□ أو□ مشروع □□□□□□ مع اعتمادات □□□□. □ أضف الاعتماد إلى pom.xml إذا كنت تستخدم □□□□□: xml <dependency> <groupId>org.springframework.boot</groupId> <artifactId>spring-boot-starter-aop</artifactId> </dependency> □ تمكين □□□ في تكوين عادة□ ما يكون تلقائيًا مع □□□□ □□□□□□، ولكن يمكنك تمكينه بشكل صريح باستخدام @EnableAspectJAutoProxy□.

3. إنشاء جانب

هكذا يمكنك تعريف جانب باستخدام □□□ □□□□□□ □□□□:

مثال جانب التسجيل:

```java
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("execution(* com.example.myapp.service.*.*(..))")
    public void logBeforeMethod() {
```

```
        System.out.println( "                    " ;(
    }

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

- @Aspect: يحدد هذه الفئة كجانب.
- @Component: يسجلها ككيان □□□□□□.
- execution(* com.example.myapp.service.*.*(..)): يحدد نقطة تعبير□أي طريقة في أي فئة تحت حزمة service مع أي نوع عودة و أي متغيرات.□

4. أنواع النصائح الشائعة

توجد عدة □□□ □□□□□□ أنواع من نصائح :التسمية□ @Before: يعمل قبل تنفيذ الطريقة. المتطابقة□ @After: يعمل بعد الطريقة بغض□النظر عن النجاح أو□ الفشل. □ @AfterReturning: يعمل بعد عودة الطريقة بنجاح. □ @AfterThrowing: يعمل إذا ألقت الطريقة استثناء. □ @Around: يلف الطريقة، مما يسمح لك بتحكم في التنفيذ. □أقوى□

مثال نصيحة حول:

```
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;
    }
}
```

- ProceedingJoinPoint: لمثل يمتي التي الطريقة يمثل اعتراضها.
- proceed(): الأصلية. الطريقة يستدعي

5. نقطة عبارات

الشائعة: الصيغة النحوية. تطبيق يمتي حيث النقاط تعريف execution(modifiers? return-type declaring-type? method-name(params) throws?) مثال: execution(public String com.example.myapp.service.MyService.get*(..)) ب تبدأ التي العامة الطرق مع يتطابق في MyService والتي تعود إلى String.

النقاط: دمج أيضًا يمكنك

```
@Pointcut("execution(* com.example.myapp.service.*.*(..))")
public void serviceMethods() {}

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

6. الطريقة تفاصيل إلى الوصول

البيانات: أو التوقيعات، الطريقة، متغيرات إلى الوصول يمكنك

```
@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 + " "    ;(
}
```

7. والتجربة العمل

- إنشاء فئة خدمة لمثل MyService مع بعض الطرق.
- تطبيق تشغيل الخاص بك.
- سيتم تطبيق الجانب الذي تلقائيًا على الطرق المتطابقة.

خدمة: مثال

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

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

امدنع يعدتست myService.sayHello("Alice")، متيس طيشنت بنوانج ليجستال أو الأداء.

نصائح

- **الأداء**: فيضي [illegible] بحكمة. استخدمه لذا الوكالات، بسبب عبئا [illegible]
- **المناطقة**: لمعي [illegible] على [illegible] يديه الكائنات التي الها لا التي للأشياء بالنسبة فقط. [illegible]، [illegible] أقوى. [illegible] ليدب [illegible] في فكر
- **التشخيص**: لـ org.springframework.aop لرؤية فيفي كيف تطبيق الوكالات. تمكين تسجيل تشخيص التال

أكبر! بشكل التفسير وسأدل أخبرني معين، تنفيذ في مساعدة إلى تحتاج أو محددة استخدام حالة لديك كان إذا