

# アスペクト指向プログラミング in Spring Boot

# Introduction

## アスペクト指向プログラミング

本質的関心事（Core Concern）と横断的関心事（Crosscutting Concern）に分けて、プログラミングを行う手法。

横断的関心事を、アスペクトというモジュールに分離する。

オブジェクト指向の限界を、補完するために考案された。  
現在では、他のプログラミング手法へも適用可能となっている。

# 目次

- 1, アスペクト指向のメリット
- 2, 本質的関心事と横断的関心事
- 3, オブジェクト指向とアスペクト指向
- 4, アスペクトクラスの作り方
- 5, アスペクトな関数の実行タイミング
- 6, アスペクト指向によるログ出力サンプル

# 1, アスペクト指向のメリット

## 複雑性の軽減

横断的関心事を1カ所にまとめることが出来る。

それにより、オブジェクト指向の弱点であった、横断的関心事による複雑化を軽減することが出来る。

## 変更に強い

横断的関心事が1カ所にまとまっているため、変更時でも1カ所の修正を行うだけで、プログラム全体の修正を行うことが出来る。

## 2, 本質的関心事と横断的関心事

### 本質的関心事 : Core Concern

プログラムが処理を行うメインタスク。

オブジェクト指向を用いて構築することに向いている。

### 横断的関心事 : Crosscutting Concern

メインタスクを補助するために、共通的に処理するタスク。

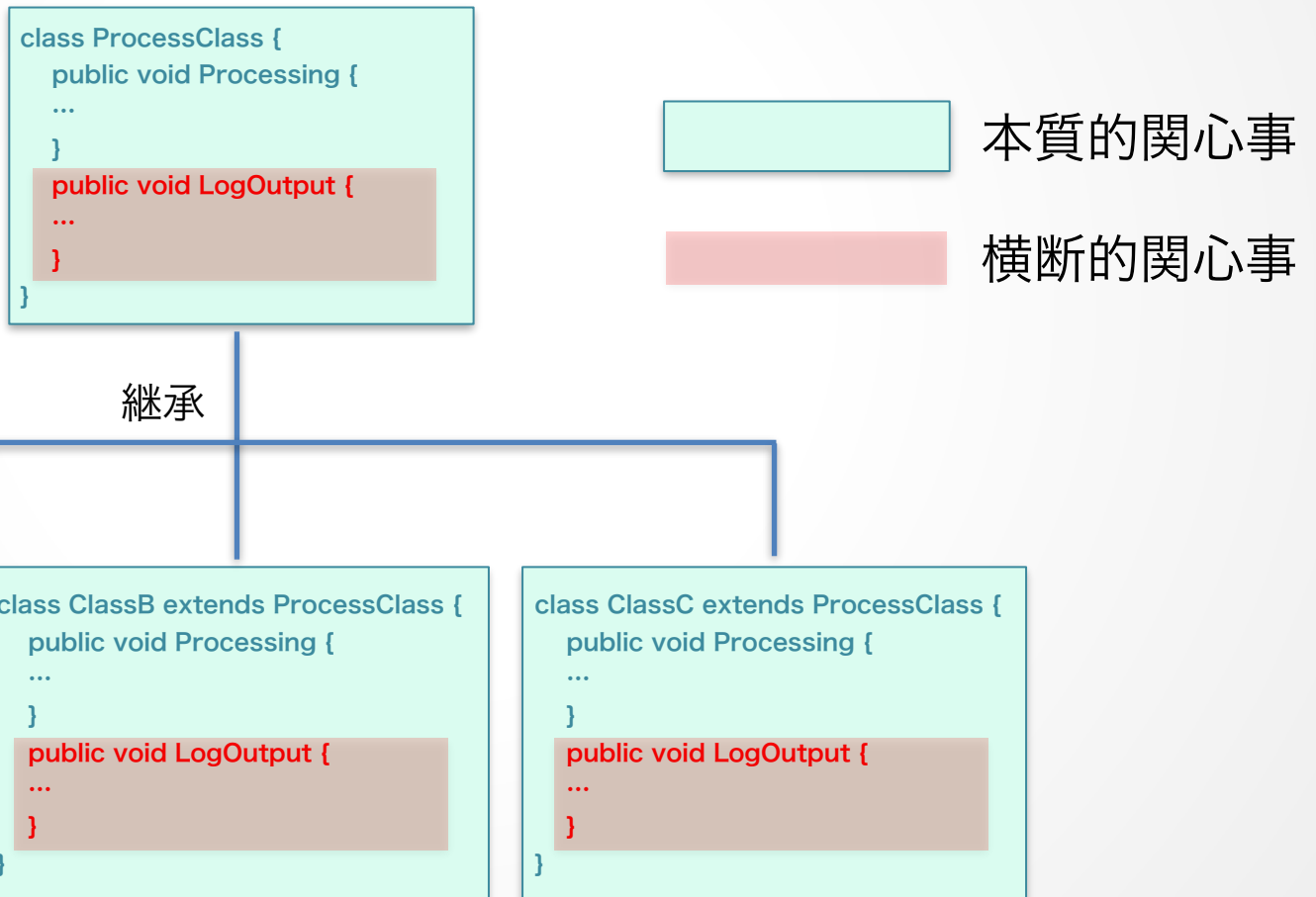
オブジェクト指向を用いることに向いていない。

**アスペクト指向において、中核となる部分。**

# 3, オブジェクト指向とアスペクト指向

## オブジェクト指向

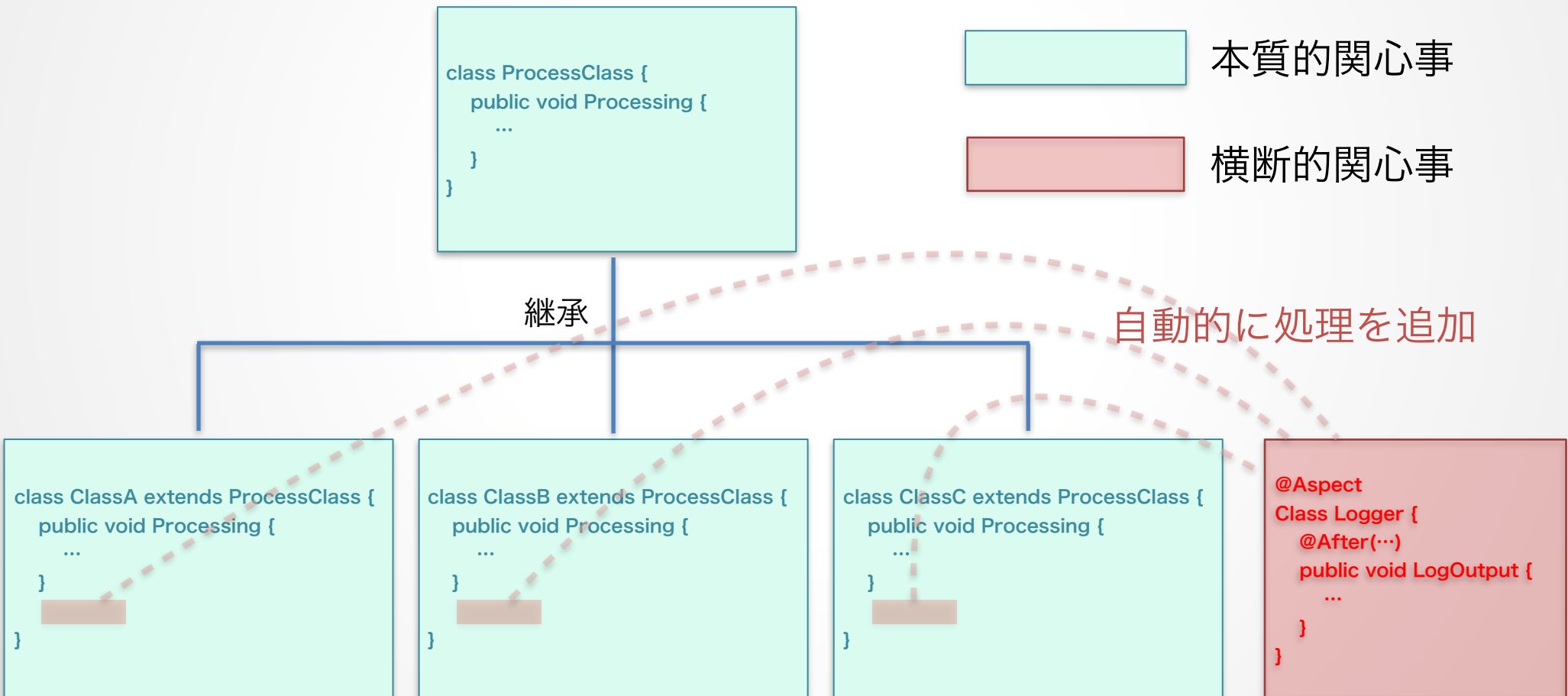
本質的関心事と横断的関心事が混在している。  
ロジックが複雑化して、保守性も悪い。



# 3, オブジェクト指向とアスペクト指向

## アスペクト指向

本質的関心事と横断的関心事を分けることが出来る。  
ロジックがシンプルになり、保守性も高まる。



## 4, アスペクトクラスの作り方

1, ビルドツールに、アスペクト指向プログラミング(AOP)の設定を追加する。

Gradleの場合

```
dependencies {  
    compile 'org.springframework.boot:spring-boot-starter-aop'  
    ...  
}
```

Mavenの場合

```
<dependency>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-aop</artifactId>  
</dependency>
```

2, クラスに対して、@Aspectアノテーションを付ける。

```
@Aspect  
public class AOPLogger {  
  
}
```



## 5, アスペクトな関数の実行タイミング

Spring Bootにおいて、アスペクトクラスの関数を実行タイミングを制御するアノテーションは、以下の5つである。

**@Before**

関数実行前

**@After**

関数実行後

**@Around**

関数処理に割り込み

**@AfterReturning**

関数処理の正常時

**@AfterThrowing**

関数処理の異常時

# 6, アスペクト指向によるログ出力サンプル

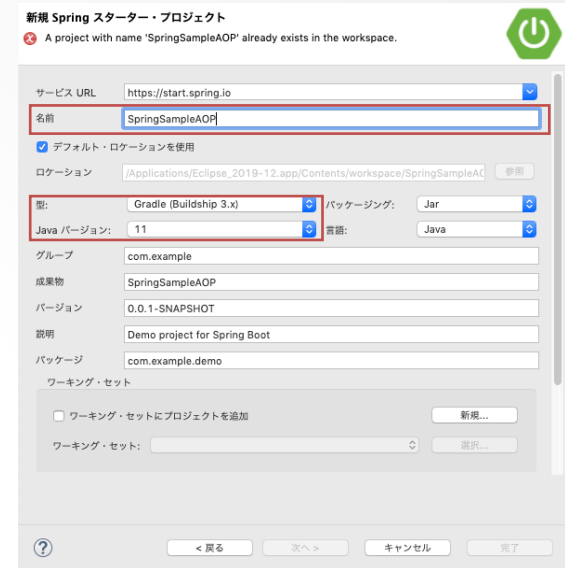
## 6-1, プロジェクトの作成

### 1, Spring スターター・プロジェクト作成

名前: SpringSampleAOP

型: Gradle (Buildship 3.x)

Java バージョン: 11



### 2, 新規 Spring スターター・プロジェクト依存関係

下記の項目をチェック。

- Spring Web
- Spring Boot DevTools

※画面上部に表示されていない場合は、検索欄を使用。



# 6, アスペクト指向によるログ出力サンプル

## 6-1, プロジェクトの作成

### 3, ファイル修正

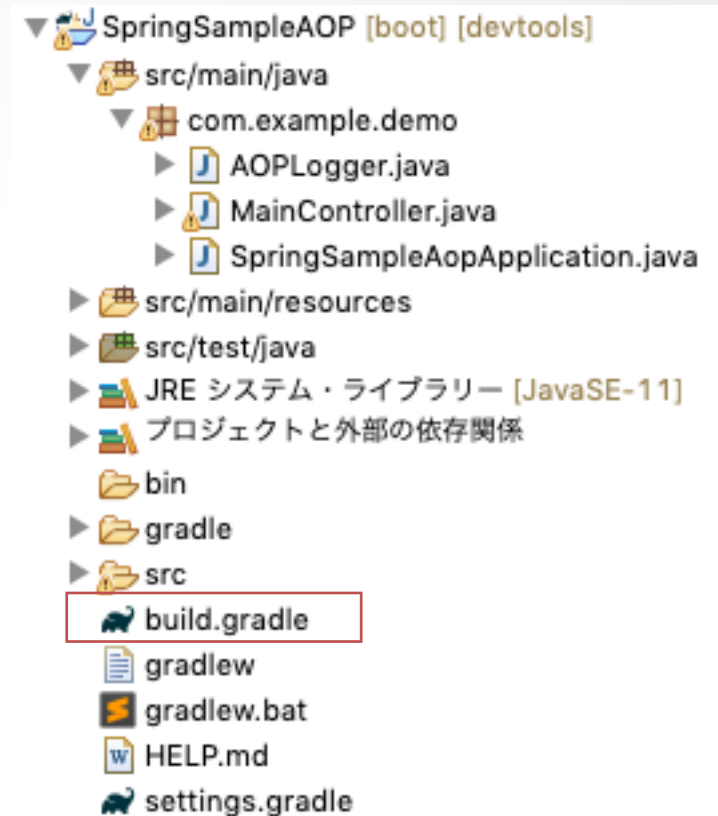
対象ファイル

build.gradle

修正内容

以下の内容を、dependenciesに追記

```
dependencies {  
    compile 'org.springframework.boot:spring-boot-starter-aop'  
    ...  
}
```



# 6, アスペクト指向によるログ出力サンプル

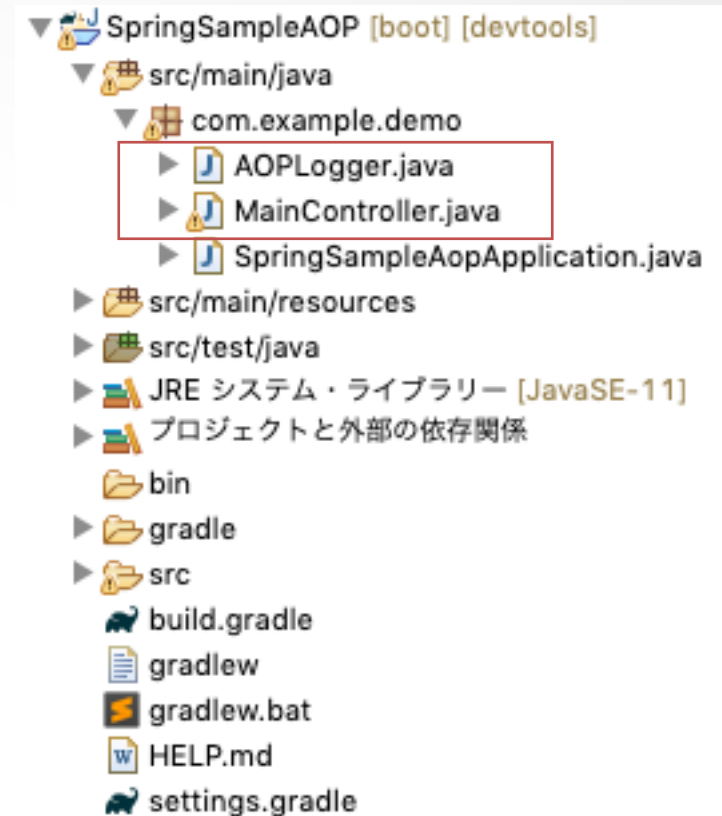
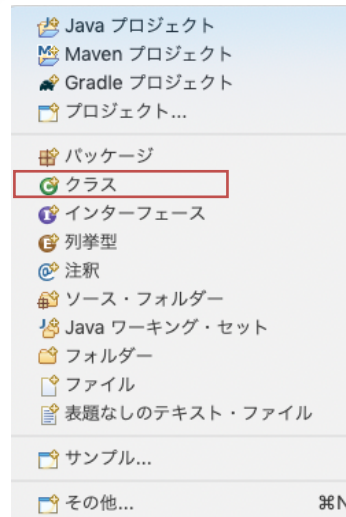
## 6-1, プロジェクトの作成

### 4, 追加ファイル

Javaクラスファイル

AOPLogger.java

MainController.java



※ファイル内容は、サンプルファイルよりコピー&ペースト。

# 6, アスペクト指向によるログ出力サンプル

## 6-2, プロジェクトの実行

### サンプルプロジェクト概要

"http://localhost:8080/hello1～3"にアクセスすることにより、  
「Hello World 1～3」を出力する関数が、それぞれ実行される。

その関数が実行される際に、ログが出力される。

ログ出力関数には、アスペクト指向のアノテーションが付いているため、  
それぞれの挙動を確認する。

確認するアノテーションは、以下の5つ。

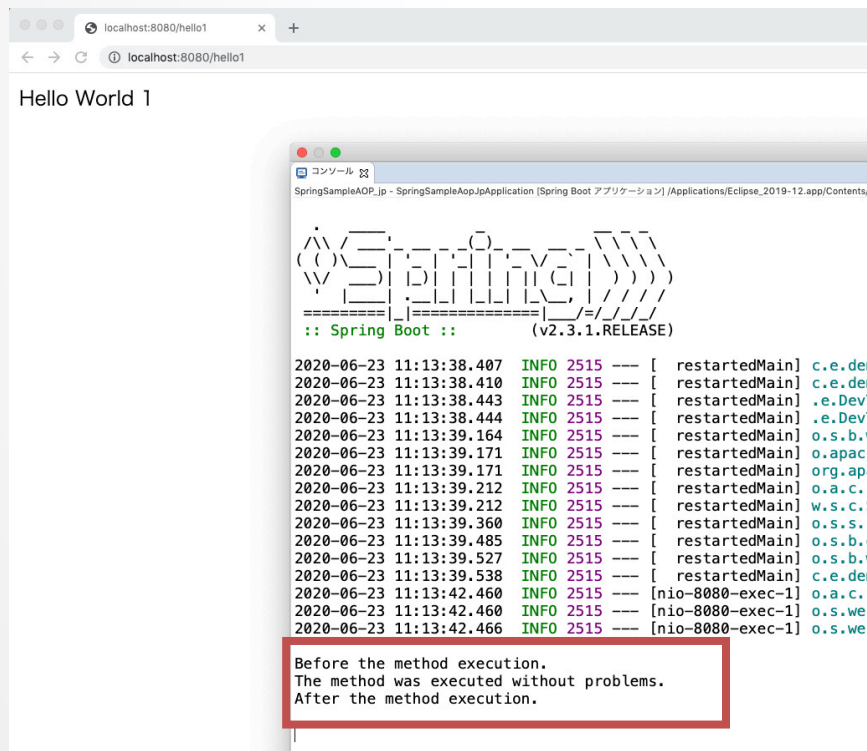
@Before, @After, @Around, @AfterReturning, @AfterThrowing

# 6, アスペクト指向によるログ出力サンプル

## 6-2, プロジェクトの実行

### 1, Hello1関数実行の場合

- 1, プロジェクトを実行する。
- 2, "http://localhost:8080/hello1"へ、アクセスする。
- 3, **コンソールにログが出力される。**
- 4, ブラウザに、「Hello World 1」が出力される。



### 処理の流れ



# 6, アスペクト指向によるログ出力サンプル

## 6-2, プロジェクトの実行

### 2, Hello2関数実行の場合

- 1, プロジェクトを実行する。
- 2, "http://localhost:8080/hello2"へ、アクセスする。
- 3, **コンソールにログが出力される。**
- 4, ブラウザに、何も出力されない。

```
SpringSampleAOP.jp - SpringSampleAop.jpApplication [Spring Boot アプリケーション] /Applications/Eclipse_2019-12.app/Contents/
:: Spring Boot :: (v2.3.1.RELEASE)

2020-06-23 11:14:09.226 INFO 2522 --- [ restartedMain] c.e.del
2020-06-23 11:14:09.228 INFO 2522 --- [ restartedMain] c.e.del
2020-06-23 11:14:09.263 INFO 2522 --- [ restartedMain] .e.Dev
2020-06-23 11:14:09.263 INFO 2522 --- [ restartedMain] .e.Dev
2020-06-23 11:14:09.951 INFO 2522 --- [ restartedMain] o.s.b.i
2020-06-23 11:14:09.958 INFO 2522 --- [ restartedMain] o.apacl
2020-06-23 11:14:09.959 INFO 2522 --- [ restartedMain] org.ap
2020-06-23 11:14:10.005 INFO 2522 --- [ restartedMain] o.a.c.
2020-06-23 11:14:10.005 INFO 2522 --- [ restartedMain] w.s.c.
2020-06-23 11:14:10.167 INFO 2522 --- [ restartedMain] o.s.s.
2020-06-23 11:14:10.277 INFO 2522 --- [ restartedMain] o.s.b.i
2020-06-23 11:14:10.337 INFO 2522 --- [ restartedMain] o.s.b.i
2020-06-23 11:14:10.349 INFO 2522 --- [ restartedMain] c.e.del
2020-06-23 11:14:12.587 INFO 2522 --- [nio-8080-exec-1] o.a.c.
2020-06-23 11:14:12.587 INFO 2522 --- [nio-8080-exec-1] o.s.wel
2020-06-23 11:14:12.593 INFO 2522 --- [nio-8080-exec-1] o.s.wel

The interrupt process was performed.
```

### 処理の流れ



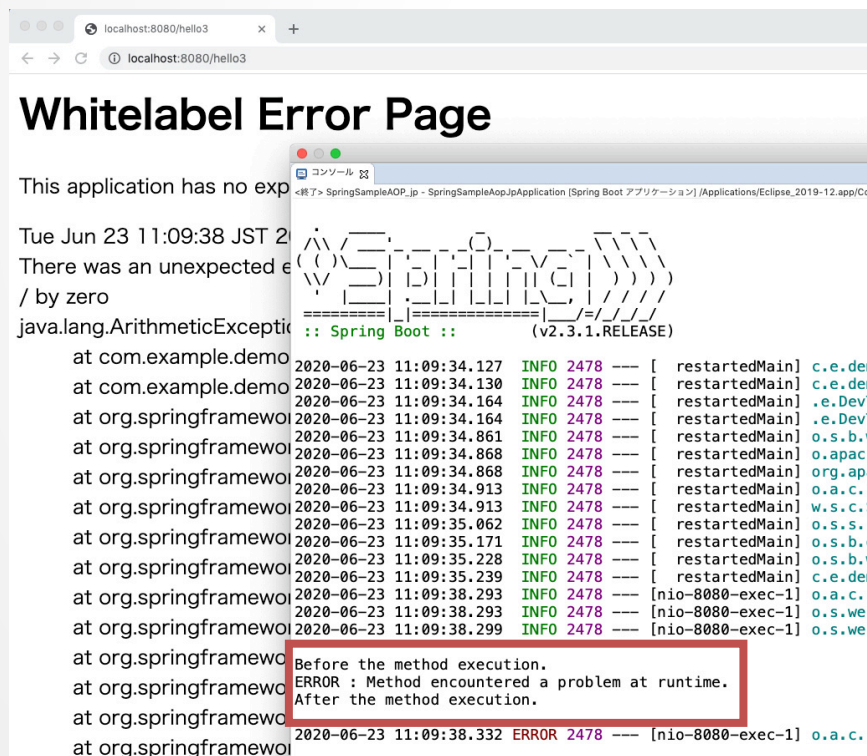


# 6, アスペクト指向によるログ出力サンプル

## 6-2, プロジェクトの実行

### 3, Hello3関数実行の場合

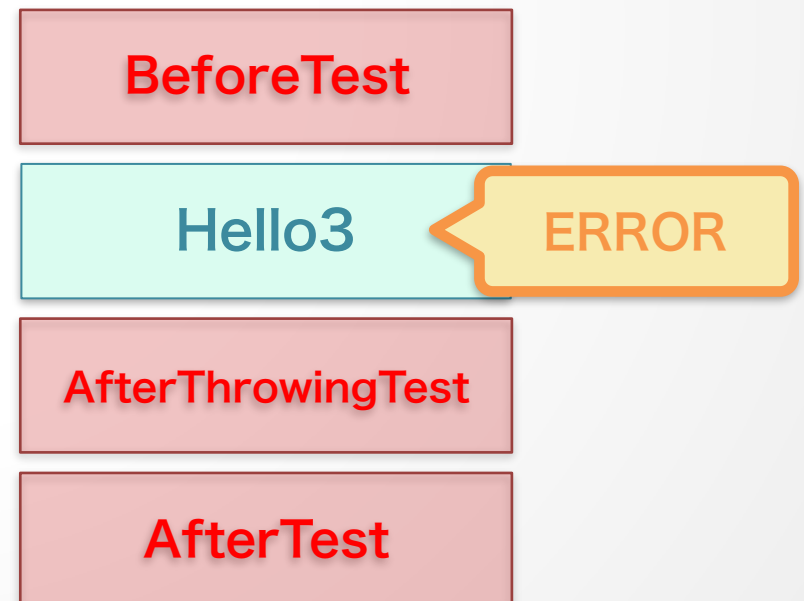
- 1, プロジェクトを実行する。
- 2, "http://localhost:8080/hello3"へ、アクセスする。
- 3, **コンソールにログが出力される。**
- 4, ブラウザとコンソールに、エラー文言が出力される。



The screenshot shows a web browser at localhost:8080/hello3 displaying a 'Whitelabel Error Page'. The message states: 'This application has no explicit handler for the request. There was an unexpected error (type=java.lang.ArithmeticException, message=division by zero)'. Below the error message, a terminal window shows log output from Spring Boot. The logs indicate that the application restarted multiple times due to an 'ArithmeticException: division by zero'. The final log entry shows the error occurring at 11:09:38.332.

```
2020-06-23 11:09:34.127 INFO 2478 --- [ restartedMain] c.e.de
2020-06-23 11:09:34.130 INFO 2478 --- [ restartedMain] c.e.de
2020-06-23 11:09:34.164 INFO 2478 --- [ restartedMain] .e.Dev
2020-06-23 11:09:34.164 INFO 2478 --- [ restartedMain] .e.Dev
2020-06-23 11:09:34.861 INFO 2478 --- [ restartedMain] o.s.b.a
2020-06-23 11:09:34.868 INFO 2478 --- [ restartedMain] org.apa
2020-06-23 11:09:34.913 INFO 2478 --- [ restartedMain] o.a.c
2020-06-23 11:09:34.913 INFO 2478 --- [ restartedMain] w.s.c
2020-06-23 11:09:35.062 INFO 2478 --- [ restartedMain] o.s.s
2020-06-23 11:09:35.171 INFO 2478 --- [ restartedMain] o.s.b
2020-06-23 11:09:35.228 INFO 2478 --- [ restartedMain] o.s.b
2020-06-23 11:09:35.239 INFO 2478 --- [ restartedMain] c.e.de
2020-06-23 11:09:38.293 INFO 2478 --- [nio-8080-exec-1] o.a.c
2020-06-23 11:09:38.293 INFO 2478 --- [nio-8080-exec-1] o.s.wel
2020-06-23 11:09:38.299 INFO 2478 --- [nio-8080-exec-1] o.s.wel
2020-06-23 11:09:38.332 ERROR 2478 --- [nio-8080-exec-1] o.a.c
```

### 処理の流れ





# アスペクト指向プログラミング in Spring Boot

Fin.