AOP in PHP

홍성범(sungbumoo@gmail.com)

AOP(Aspect Oriented Programming)는 OOP처럼
프로그래밍 패러다임중 하나입니다.

(아직까지는 OOP가 끝판왕 아니었어?)

그럼 AOP가 OOP를 대체하는건가?

아닙니다.

OOP는 그대로 쓰면서 AOP를 적용해서 개발합니다.

OOP가 뭐가 모자라서 AOP를 더 써야하나요?

OOP도 조금 문제가 있습니다.

코드로 살펴볼까요?

```php
<?php

class DocumentHandler
{
    public function insertDocument($doc)
    {

        Mail::send($to, $doc->title);

        Log::info($doc->title.' added');

        DB::beginTransaction();

        $result = $documentRepository->insertDoc($doc);

        return $result;

        DB::commit();
    }

     ...
}
```

버섯다 버꾸이...

DocumentHandler는 문서를 처리하는 비지니스 로직을 담당합니다.
근데 여러가지 부가적인 기능을 DocumentHandler클래스가 신경쓰고 있네요

DocumentHandler:

"난 문서 처리만 하고싶어.. 다른건 신경 쓰기 싫어.."

보통 이럴 땐 이벤트를 씁니다.

```php
<?php

class DocumentHandler
{
    public function insertDocument($doc)
    {

        Event::fire('insert_doc', $doc);

        $this->insertDoc($doc);

        return $doc;

    }
}
```

그래도 많이 깔끔해졌네..

그런데.. 이것조차 이벤트를 파이어하는 별도의 코드가 들어가야 합니다.

(뭔가 찜찜해..)

```php
<?php

class DocumentHandler
{
    public function insertDocument($doc)
    {

        $this->insertDoc($doc);

         return $doc;

    }
}
```

오우.. 완전 깔끔!!

AOP는 비지니스 로직이 아닌 부가기능을 다 빼버립니다.

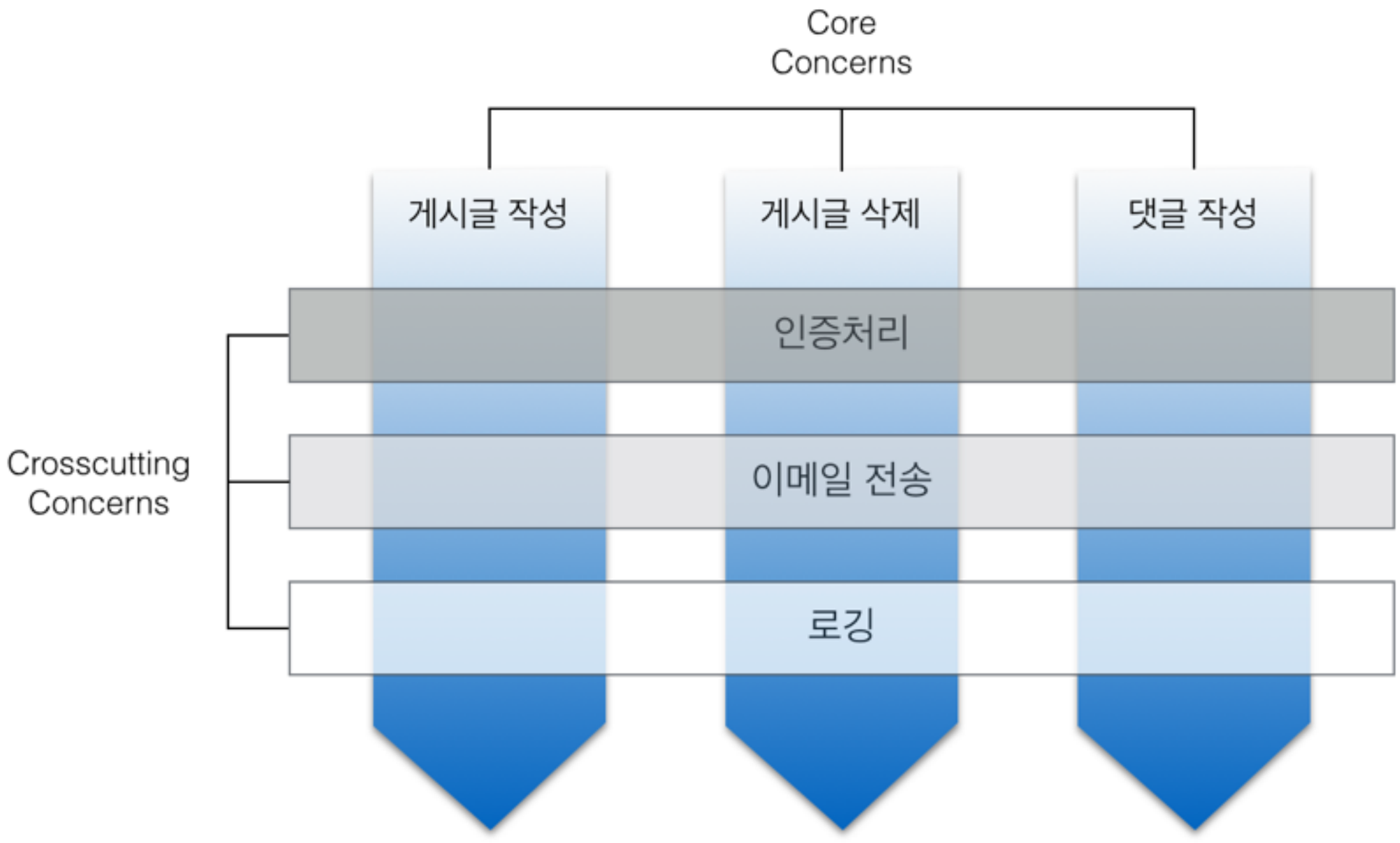
부가기능은 각각의 담당자가 "알아서" 처리합니다.

여기서 고민거리가 하나 생깁니다

"어떻게 비지니스로직이 실행될 때, 부가기능을
각각의 담당자가 "알아서" 처리하도록 할까?"

이걸 처리하는게
AOP의 핵심기술입니다.(weaving이라고 합니다)

(물론 개발자가 언제언제 부가기능을 실행하라고 미리 지정해줘야 합니다.)


Core
Concerns
게시글 작성
게시글 삭제
댓글 작성
인증처리
Crosscutting
Concerns
이메일 전송
로깅

여기서 잠깐!! 용어 정리를 하고 갑니다.

core concern - 비지니스 로직.

target - core concern을 담당하는 녀석.

cross-cutting concerns - 부가 기능들. 로깅, 이메일 전송 등등

용어 정리 (이어서)

Join Point - 비지니스 로직이 실행되는 중 부가기능이 끼어들수 있는 시점.
함수실행전, 실행후, 예외발생시, 맴버필드 변경시 등등

Point cut - Join Point가 끼어들기를 할 수 있는 시점의 후보들이라면, Point cut은 실제로 선택된 Join Point

Advice - 실행하려는 부가기능

Aspect or Advisor - Pointcut(언제)과 Advice(무엇을)를 합쳐서 Aspect 또는 Advisor라고 표현

weaving - Point cut에 Advice가 실행되도록 하는 기능

위빙(weaving)은 몇가지 방법이 있습니다.

컴파일시 위빙 - 비지니스 로직의 소스코드를 컴파일할 때, 부가 기능을 처리하는 코드를 추가합니다.

클래스 로드시 위빙 - 클래스가 로딩될 때 부가 기능을 처리하는 바이너리 코드를 추가합니다.

런타임시 위빙 - 비지니스 로직을 담당하는 클래스의 proxy를 두고, 그 클래스 대신 proxy가 호출을 받아서 부가 기능을 처리하고, 그 클래스를 호출합니다.

PHP에서 AOP를 구현하는 몇가지 방법이 나와있습니다.

AOP PECL extension

Go! AOP-PHP Library

TYPO3 Flow

코드로 한번 살펴볼까요?

```php
function sendMailBeforeInsertDocument (AopJoinPoint $object)
{
    $doc = $object->getArguments()[0];

    Log::info($doc->title.' added');
}

aop_add_before('DocumentHandler->insertDocument()',
'sendMailBeforeInsertDocument');
```

ㄴ AOP PECL extension (http://aop-php.github.io/)

코드로 한번 살펴볼까요?

```php
<?php

class MonitorAspect implements Aspect
{

    /**
     * Method that will be called before real method
     *
     * @param MethodInvocation $invocation Invocation
     * @Before("execution(public DocumentHandler->insertDocument(*))")
     */
    public function sendMailBeforeInsertDocument(MethodInvocation $invocation)
    {

        $doc = $invocation->getArguments()[0];
        Log::info($doc->title.' added');

    }
}
```

↳ Go! AOP PHP Library (http://go.aopphp.com/)

Demo

Proxy를 이용한 AOP를 적용해 볼 계획입니다.

```php
class PluginServiceProvider extends ServiceProvider {

    public function register()
    {
        $this->app->bindShared('plugin', function($app)
        {

            $pluginHandler = new PluginHandler($app));

            $plugin = \AOP::newProxy($pluginHandler);

            return $plugin;
        });
    }
}
```

```php
aop('Plugin@addPluginRoutes', 'route_for_plugin',
    function ($target, &$args) {

        $routes = $args[0];

        $args[0] = 'change args';

        return $target($args);
    }
);
```

Thank you.