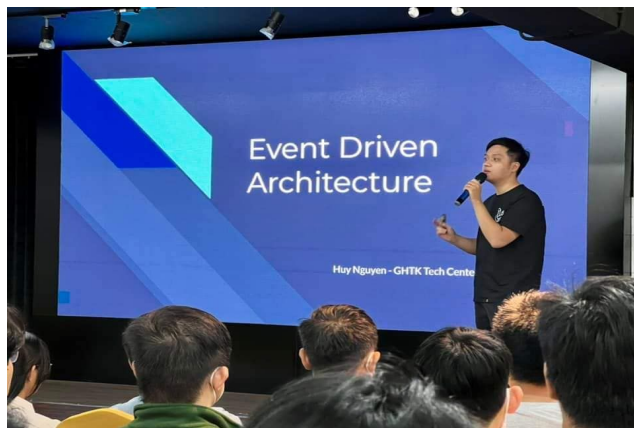


# Make MVC great again

HuyNT - GHTK Tech Center

# Giới thiệu

- Technical Leader @ GHTK Tech
- PHP Lover
- Tôi chia sẻ mọi điều tôi biết



MVC, Chúng tôi đã thử và



# 80%

Thời gian maintain được tiết kiệm





# 100%

Trải nghiệm lập trình được cải thiện





# 100%

No over engineer, no OOP trash, ...





# 100%

Laravel way







Trygve Mikkjel Heyerdahl  
Reenskaug  
đề xuất MVC năm 1979

Trở thành tiêu chuẩn cho hầu hết các  
web framework ngày nay



Rails tái định nghĩa cách một framework web MVC được triển khai

Laravel được truyền cảm hứng rất nhiều từ Rails

```
graph TD; Model[Model] --- Controller[Controller]; Controller --- View[View];
```

Model

Controller

View



Bạn sẽ đặt business logic ở đâu ?

Model

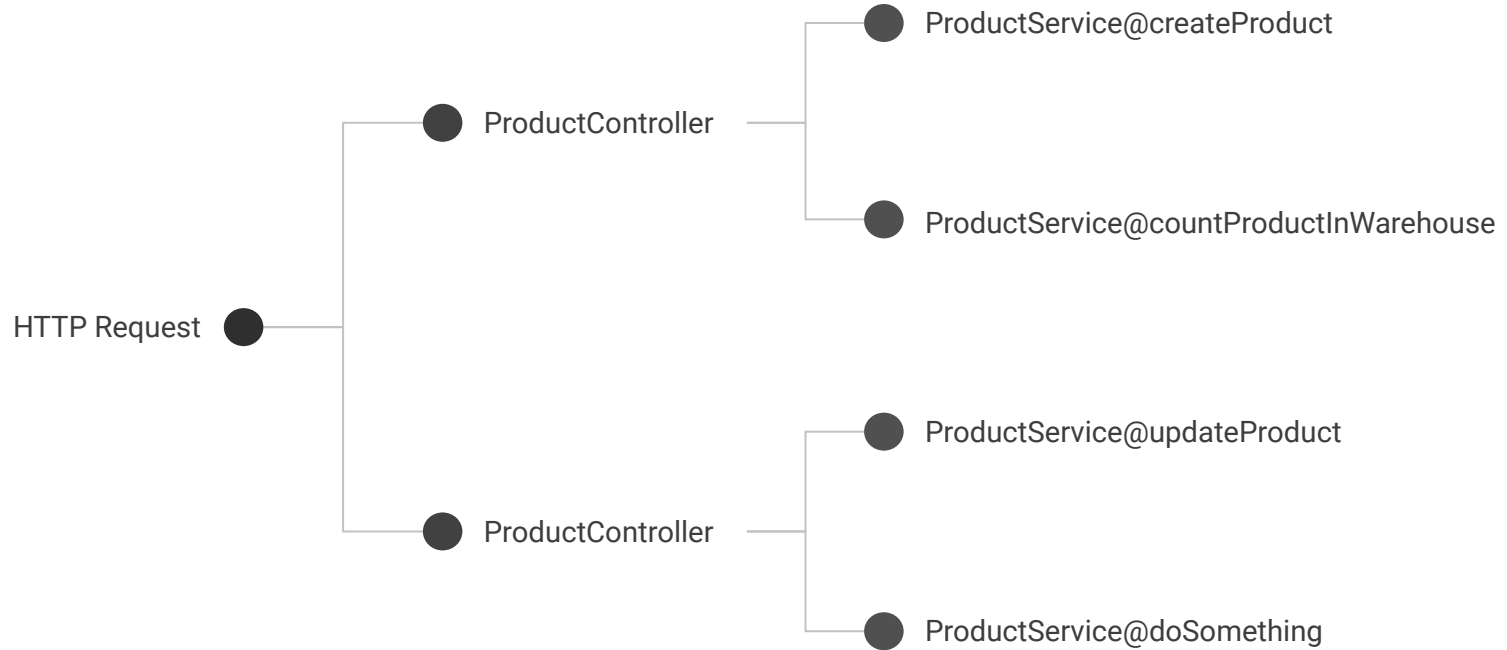
Service

Controller

View

Nice, hãy cùng xem vấn đề

Khó scale up logic





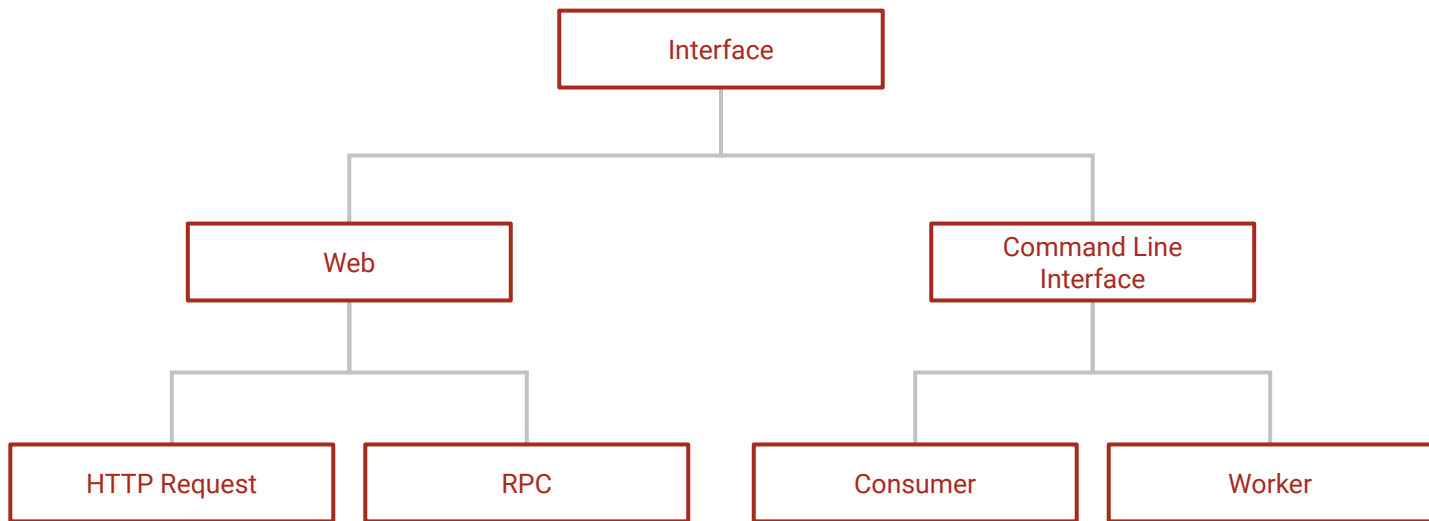
Project của bạn nên phản ánh các  
use case cụ thể (chứ không phải  
trong routes/web.php)

Khó test



Class / function cần nhỏ gọn để  
test được, hạn chế tối đa phụ  
thuộc thừa

Scale cho các user interface khác nhau





```
class ProductController {  
    public function create(Request $request) {  
        $this->productService->create($request->input('name'), $request->input('desc'));  
    }  
}  
  
class ProductCommand {  
    public function handle(string $name, string $desc) {  
        $this->productService->create($name, $desc);  
    }  
}
```

# Array Driven Design



Bạn cần biết nhiều hơn về dữ liệu chứ không chỉ  
là array  
[“don’t do” => “ this”]



```
class ProductController {  
    public function create(Request $request) {  
        //what the hell inside data ???  
        $data = $request->all();  
  
        $result = $this->producerService->create($data);  
  
        //what the hell inside result ???  
        return response($result);  
    }  
}
```

Decouple from framework



```
class ProductController extends Controller {  
  
    public function create(Request $request) {  
        $this->performCreate($request->get('some_input'));  
        return response($someResponse);  
    }  
  
}
```

# Command Bus Design Pattern



Command Bus giúp chia tách dự  
án thành các use case cụ thể

Command



```
class CreateProductCommand {  
    private string $name;  
  
    private int $price;  
  
    public function setName(string $name) {  
        $this->name = $name;  
    }  
  
    public function setPrice(int $price) {  
        $this->price = $price;  
    }  
  
    public function getName() {  
        return $this->name;  
    }  
  
    public function getPrice() {  
        return $this->price;  
    }  
}
```





Command đảm bảo bạn hiểu rõ  
những gì ứng dụng cần

Handler



```
class CreateProductHandler {  
    public function handle(CreateProductCommand $createProductCommand) {  
        $name = $createProductCommand->getName();  
        $price = $createProductCommand->getPrice();  
  
        $this->productRepository->create($name, $price);  
    }  
}
```



```
class ProductController {  
    public function create(Request $request) {  
        $command = app()->make(CreateProductCommand::class);  
        $command->setName($request->get('name'));  
        $command->setPrice($request->get('price'));  
        $handler = app()->make(CreateProductHandler::class);  
        Bus::dispatchNow($command, $handler);  
    }  
}
```

Model

Command /  
Handler

Controller

View

Action

Business logic cần được chia thành các class với duy nhất một method phục vụ một nghiệp vụ cụ thể (nếu  $> 1$ , thì chỉ nên là các phương thức hỗ trợ logic đó)

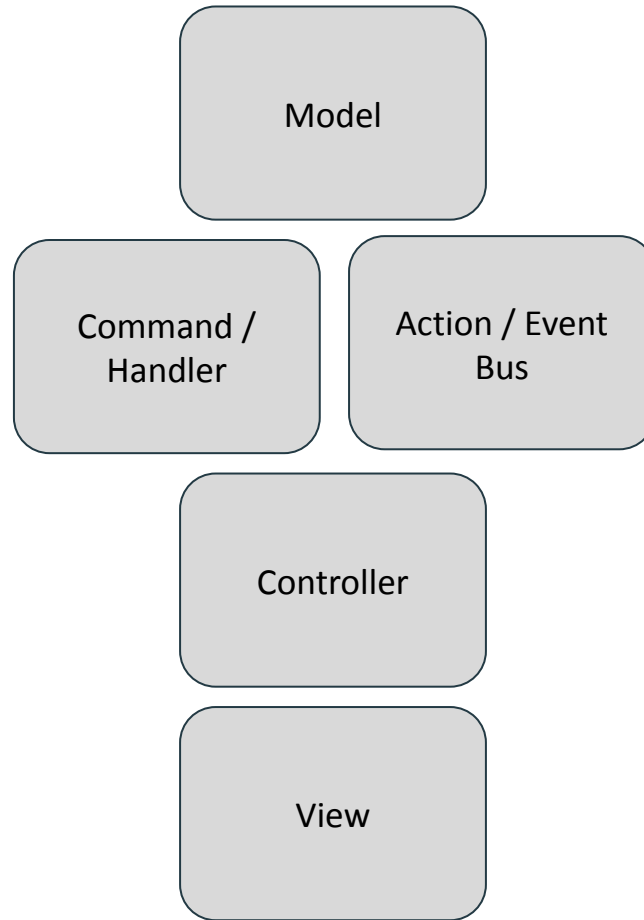


```
class ProductService {  
    public function create($name, $price) {  
        $this->productRepository->create($name, $price);  
        $this->sendNotification();  
        $this->logging();  
        $this->doSomethingYouKnowIDontKnow();  
    }  
}
```





```
class CreateProductAction {  
    public function handle($name, $price) {  
        $this->productRepository->create($name, $price);  
        event(new ProductCreated())  
    }  
}  
  
class SendEmailAction {  
    public function handle(ProductCreated $event) {  
        //I know I know  
    }  
}  
  
class LoggingAction {  
    public function handle(ProductCreated $event) {  
        //I know I know  
    }  
}  
  
class SomethingSomething {  
    public function handle(ProductCreated $event) {  
        //I know I know  
    }  
}
```



DTO / ViewModel



```
class CustomerData extends DataTransferObject
{
    // ...

    public static function fromRequest(
        CustomerRequest $request
    ): self {
        return new self([
            'name' => $request->get('name'),
            'email' => $request->get('email'),
            'birth_date' => Carbon::make(
                $request->get('birth_date')
            ),
        ]);
    }
}
```

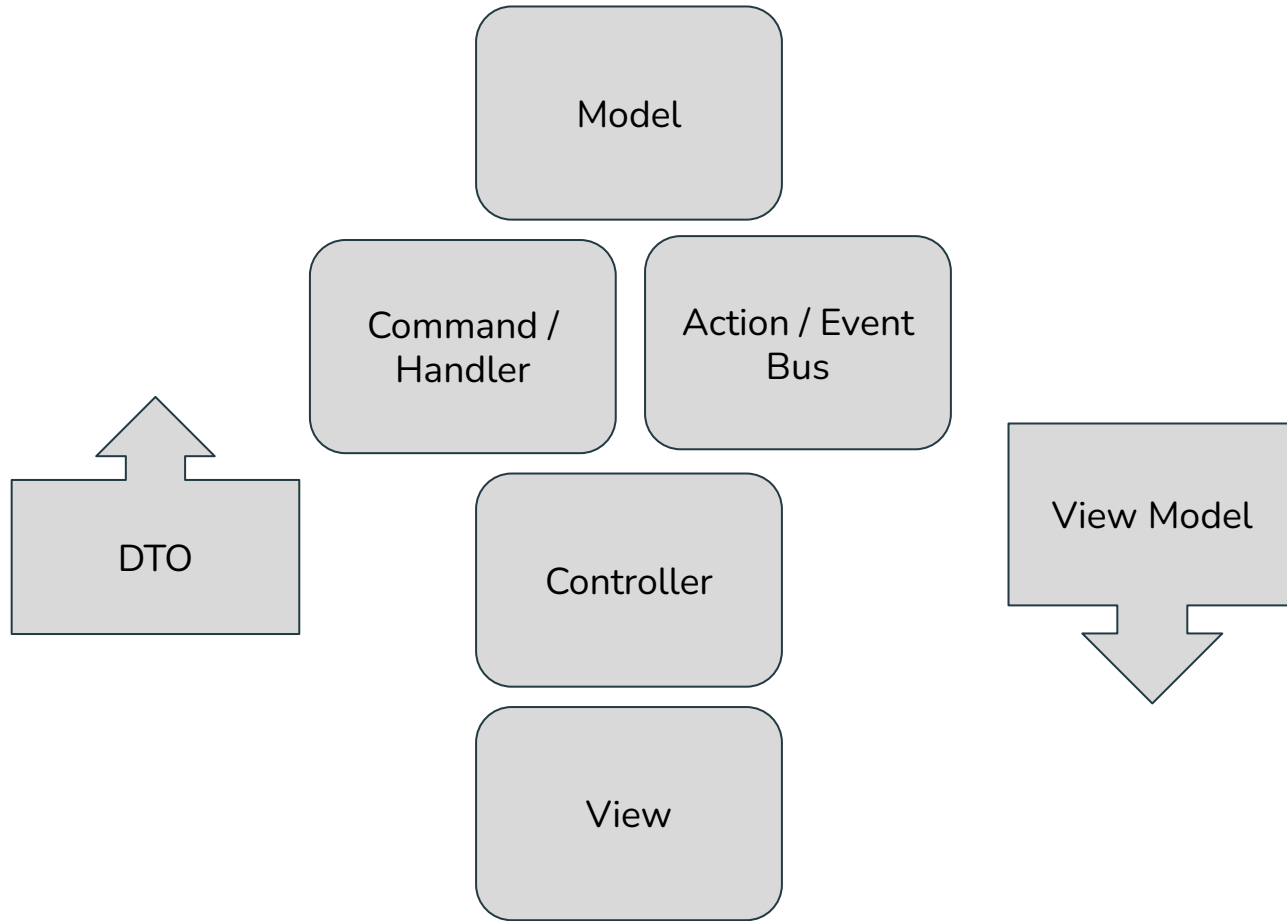



```
class ReportViewModel
{
    public function __construct(array $someBigResult)
    {
        $this->firstResult = $someBigResult['first'] ?? [];
        $this->secondResult = $someBigResult['second'] ?? [];
    }

    public function getFirstResult(): array
    {
        return $this->firstResult;
    }


    public function transformForWeb(): array
    {
        return [
            // your custom result here
        ];
    }

    public function transformForOtherPurpose(): array
    {
        return [
            // your custom result here
        ];
    }
}
```





Giảm sự kế thừa tối đa  
Áp dụng dễ dàng mọi Design Pattern



# Module hoá HMVC - Packaging



Wordpress action / filter ?



HMVC + Service Provider + Event Listener



# Thank you Q/A

Cám ơn mọi người đã lắng nghe

