Your Code Is

SIMPLIFY Or COMPLEXITY?

with **Dinh Quoc Han**

Code Của Bạn

ĐƠN GIẢN *Hay* PHỨC TẠP?

với **Đinh Quốc Hân**

Who Am I?

PHP Lover, Artisan

6 years experience

Currently working at SHIFT ASIA

Admin at Laravel Viet Nam

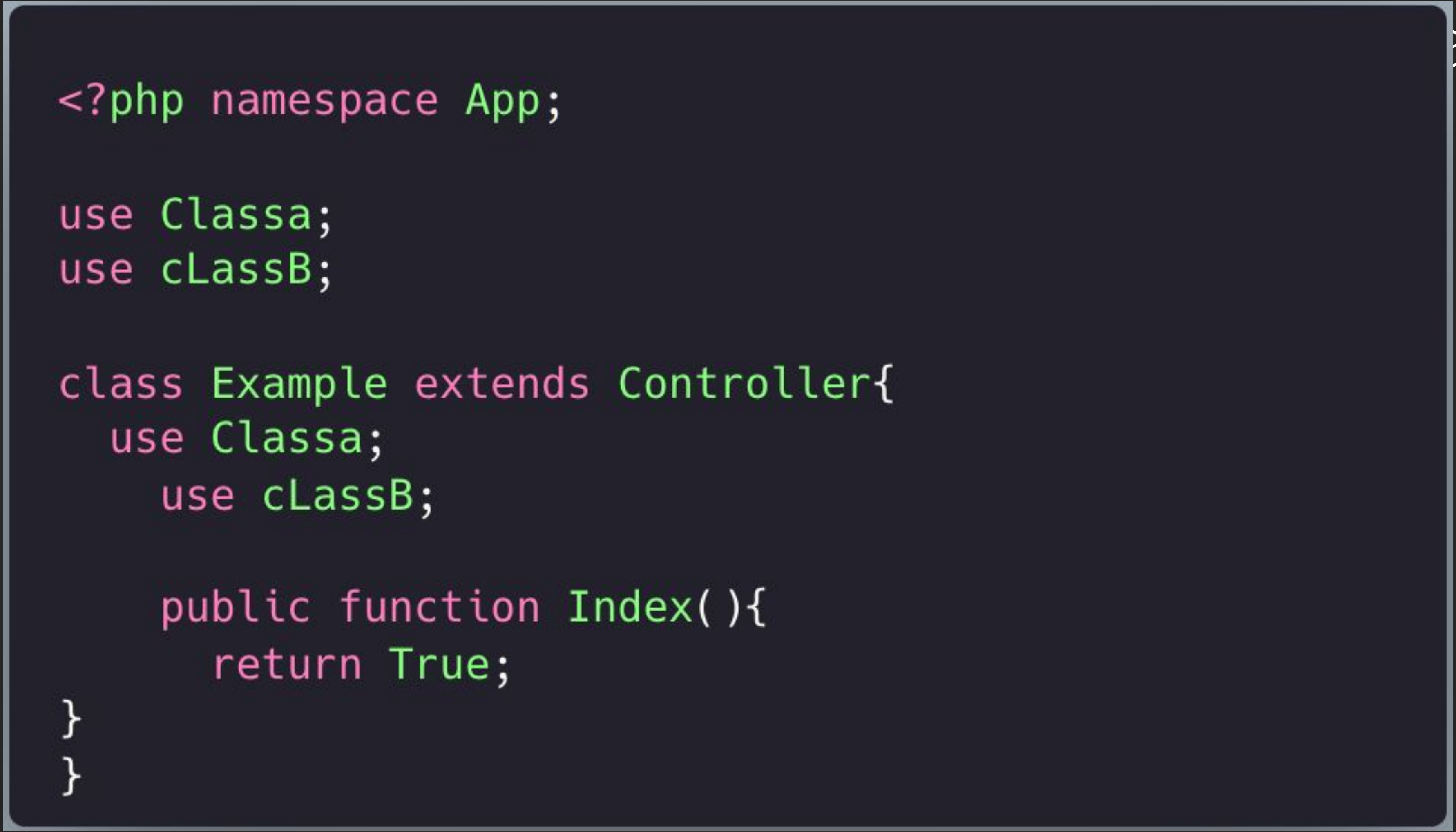
Overview

1. Problems
2. Coding Standards
3. CRUD
4. Complexity?
5. Q&A

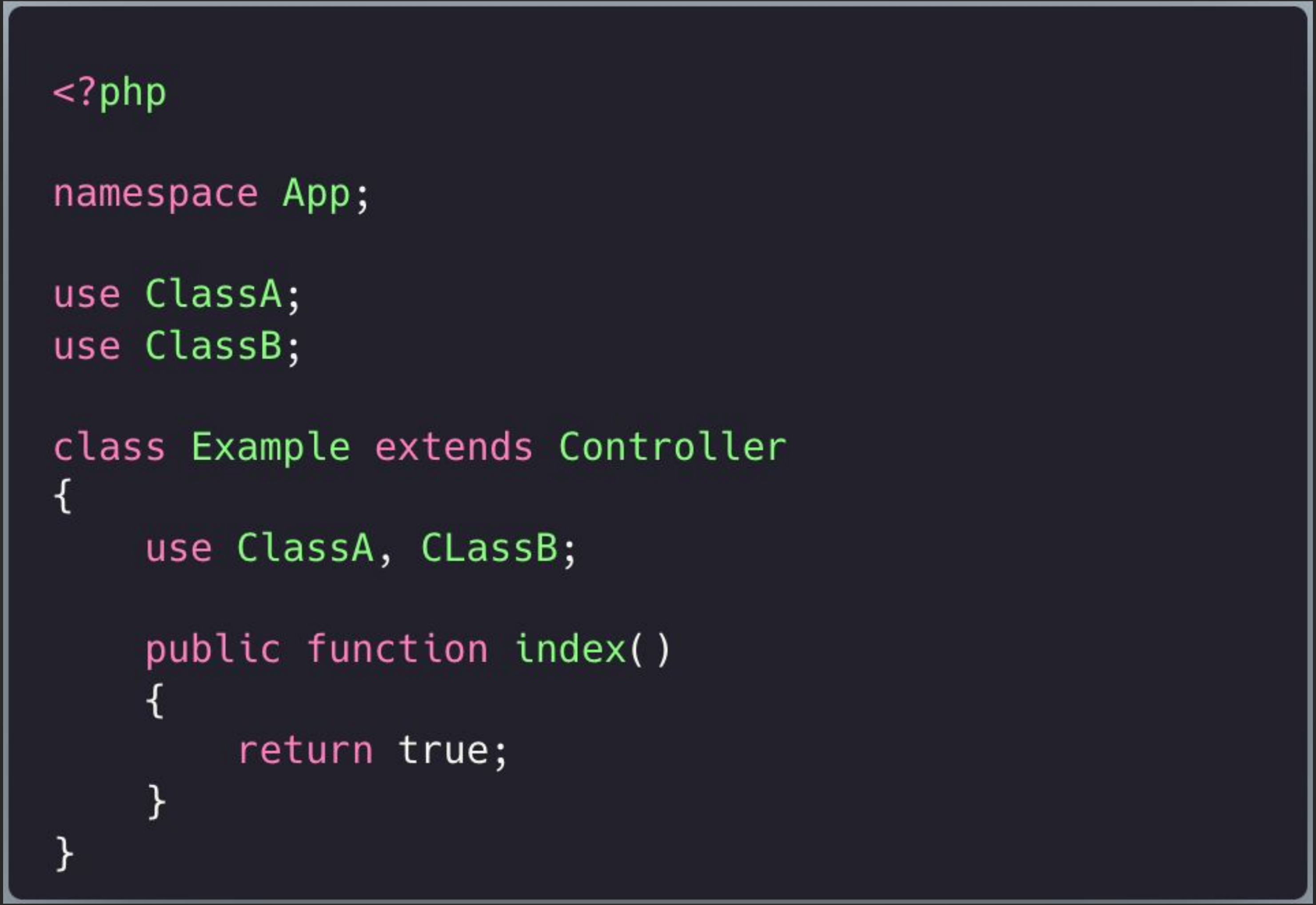
1. Problems!

2. Coding Standards

PSR-12



```php
<?php namespace App;

use Classa;
use cLassB;

class Example extends Controller{
  use Classa;
    use cLassB;

    public function Index(){
      return True;
}
}
```



```php
<?php

namespace App;

use ClassA;
use ClassB;

class Example extends Controller
{
    use ClassA, CLassB;

    public function index()
    {
        return true;
    }
}
```

How to follow PSR-12?

We don't do that here

PSR-12

```
composer require --dev laravel/pint
```

3. CRUD

Create - Read - Update - Delete 😎

How Many?

M-V-C

M-V-C but by Packages

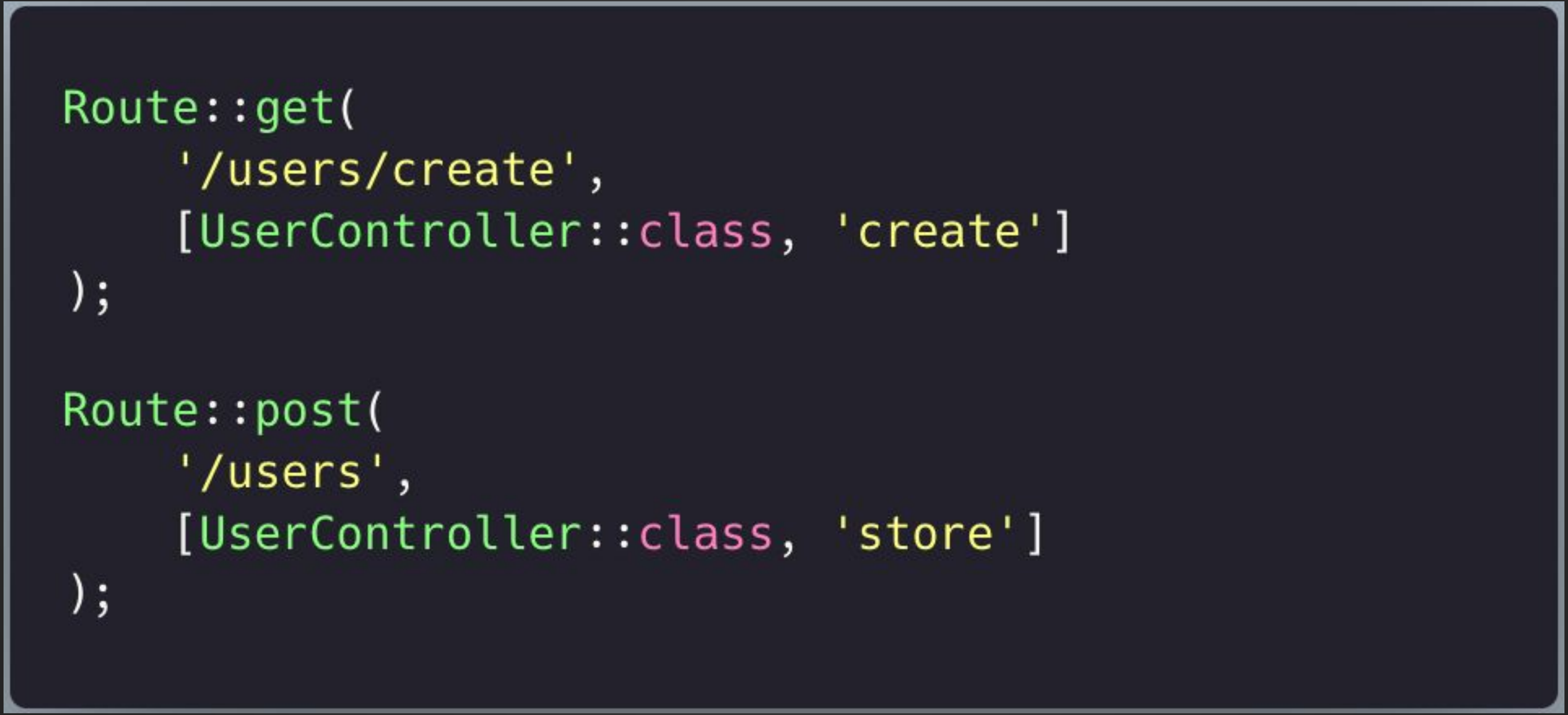
Domain Oriented Design

Not ‘DDD’ - Domain Driven Design 🤣

...

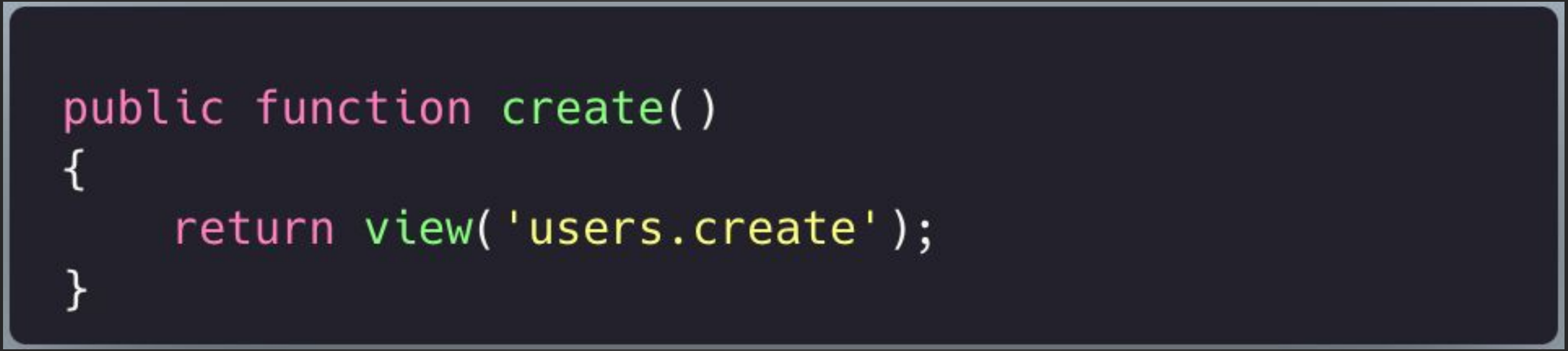
Basic Example

routes/web.php



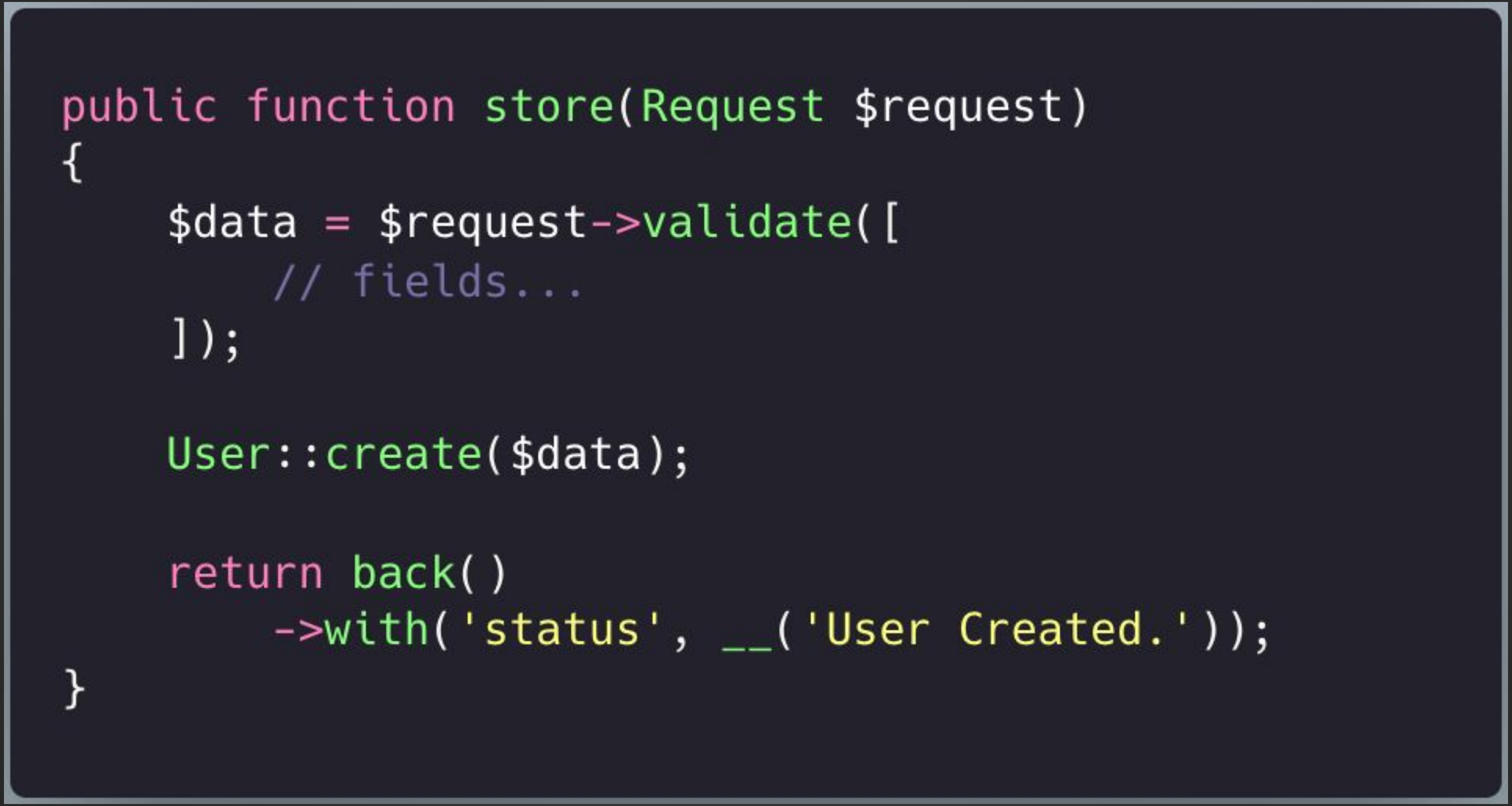
```
Route::get(
    '/users/create',
    [UserController::class, 'create']
);

Route::post(
    '/users',
    [UserController::class, 'store']
);
```

app/Http/Controllers/UserController.php



```php
public function create()
{
    return view('users.create');
}
```

app/Http/Controllers/UserController.php



```php
public function store(Request $request)
{
    $data = $request->validate([
        // fields...
    ]);

    User::create($data);

    return back()
        ->with('status', __('User Created.'));
}
```

Write all business logic in controller?

NO! NO GOD PLEASE NO!

Why?

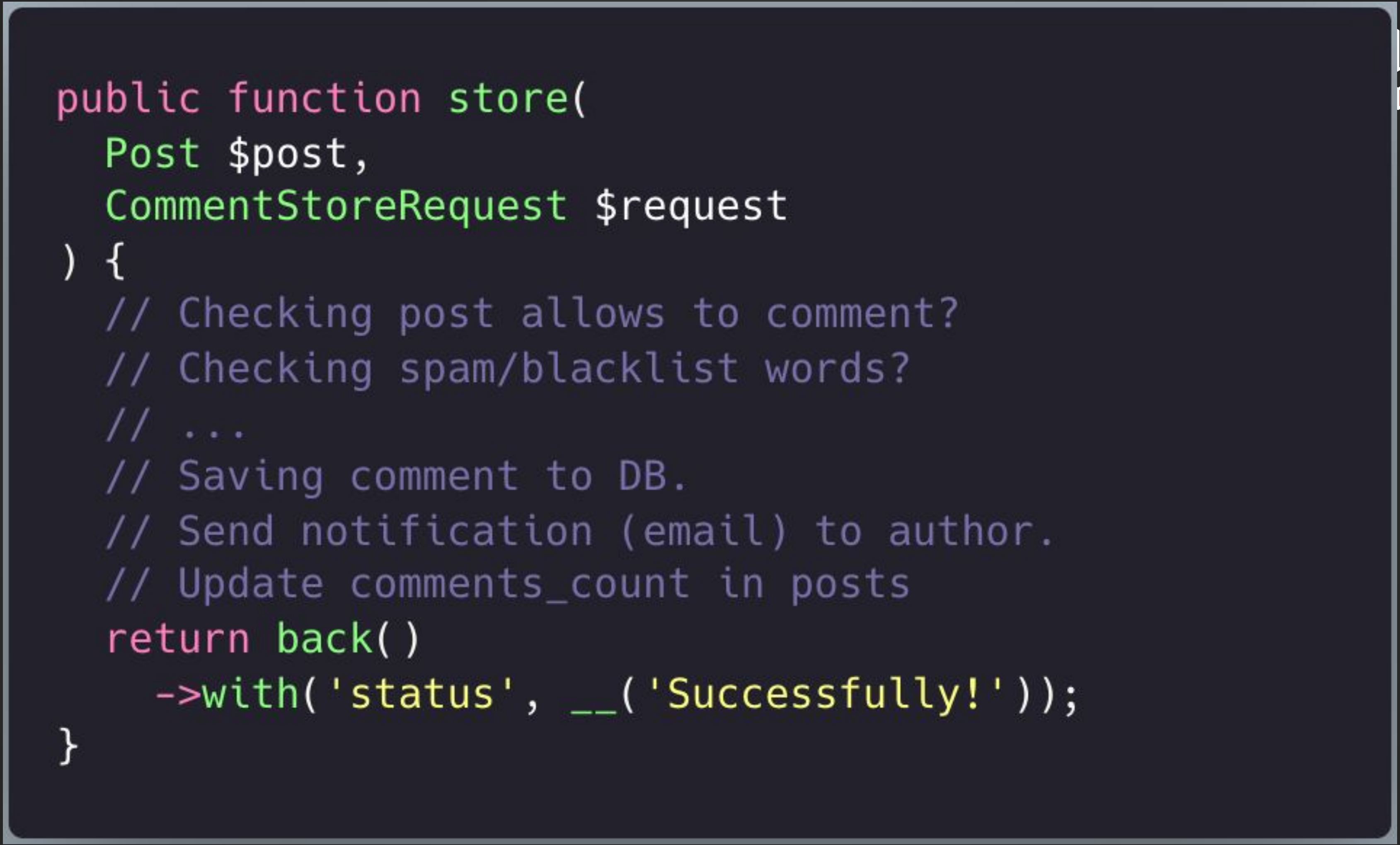
Controller should hold ZERO logic!

Controller only bring logic together!

Easy apply design patterns, structures!

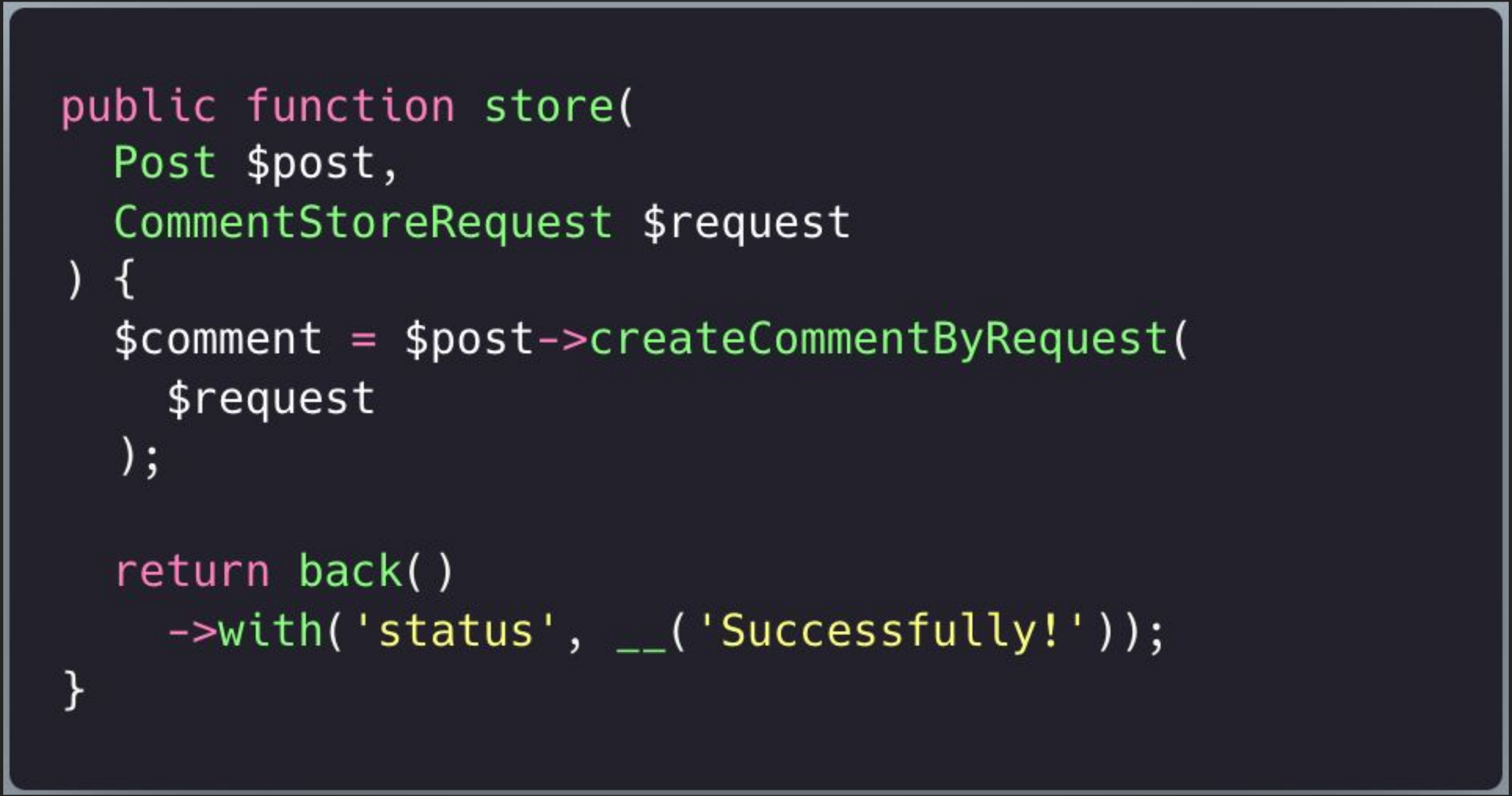
Easy to reuse, maintenance, testing

1. Fat Model - Thin Controller



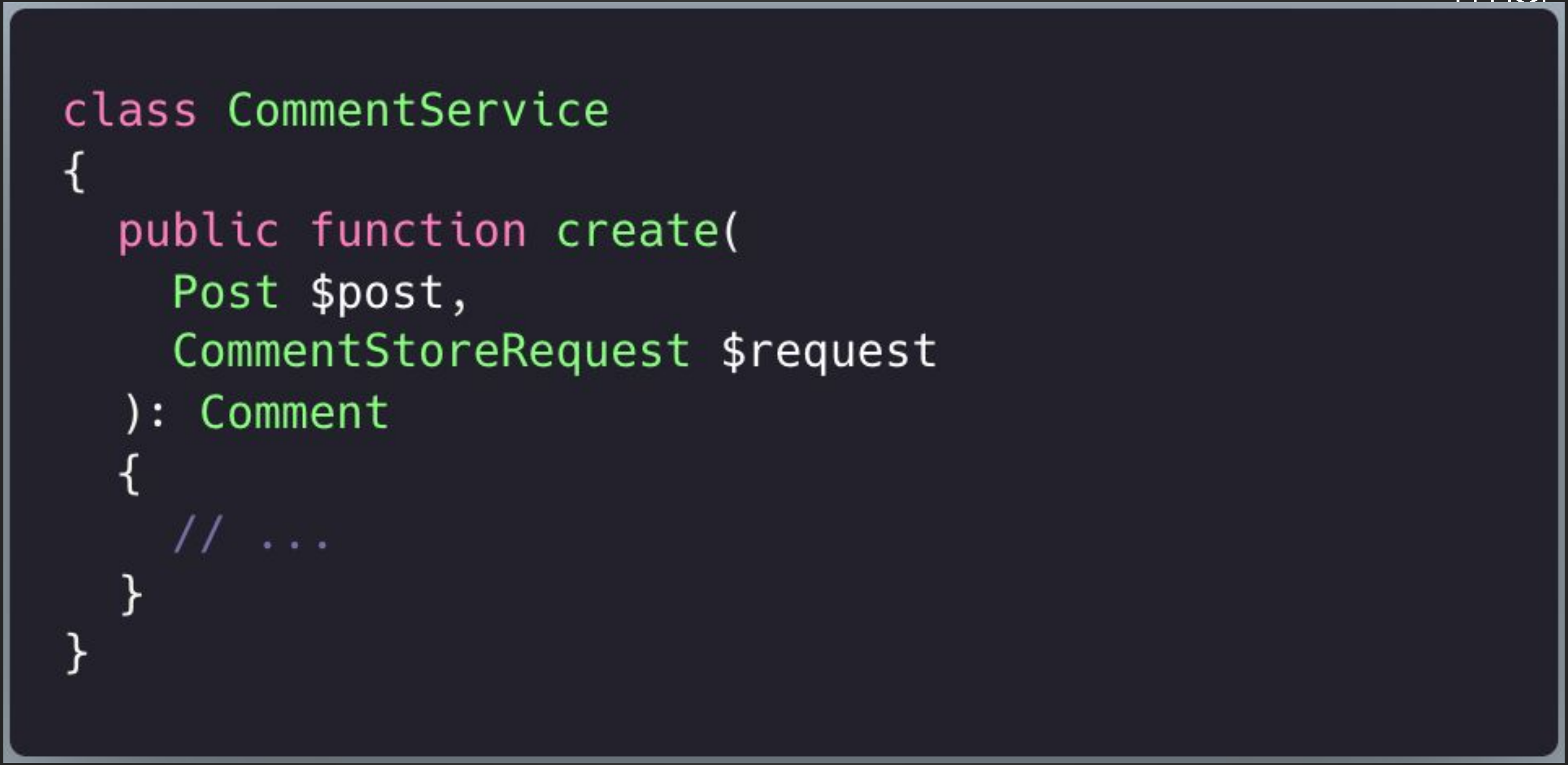
```php
public function store(
  Post $post,
  CommentStoreRequest $request
) {
  // Checking post allows to comment?
  // Checking spam/blacklist words?
  // ...
  // Saving comment to DB.
  // Send notification (email) to author.
  // Update comments_count in posts
  return back()
    ->with('status', __('Successfully!'));
}
```

```
class Post extends Model
{
  ...
  public function createCommentByRequest(
    CommentStoreRequest $request
  ): Comment {
    // Checking post allows to comment?
    // Checking spam/blacklist words?
    // ...
    // Saving comment to DB.
    // Send notification (email) to author.
    // Update comments_count in posts
  }
  ...
}
```

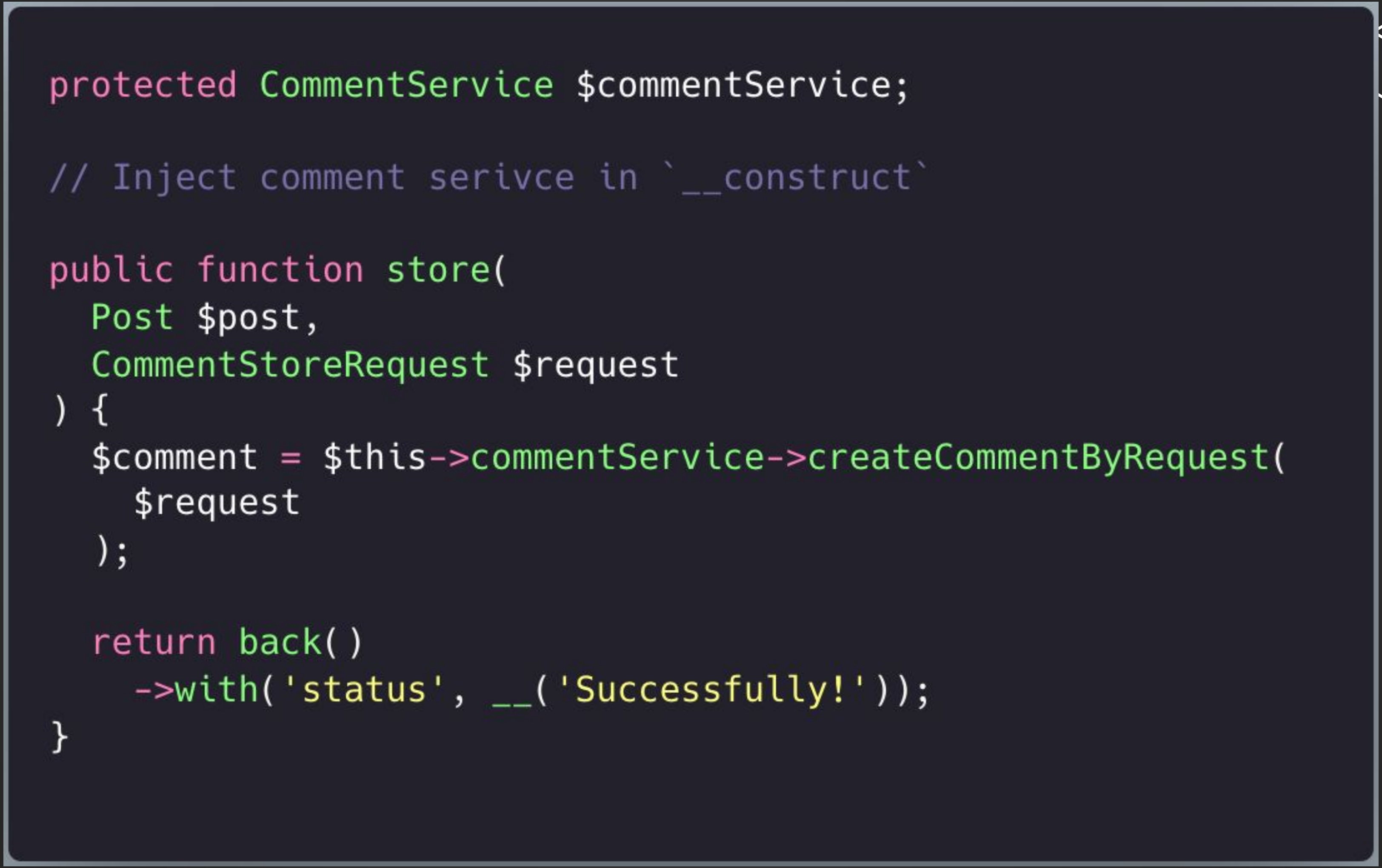


```php
public function store(
  Post $post,
  CommentStoreRequest $request
) {
  $comment = $post->createCommentByRequest(
    $request
  );

  return back()
    ->with('status', __('Successfully!'));
}
```

2. Repository Class or Service Class



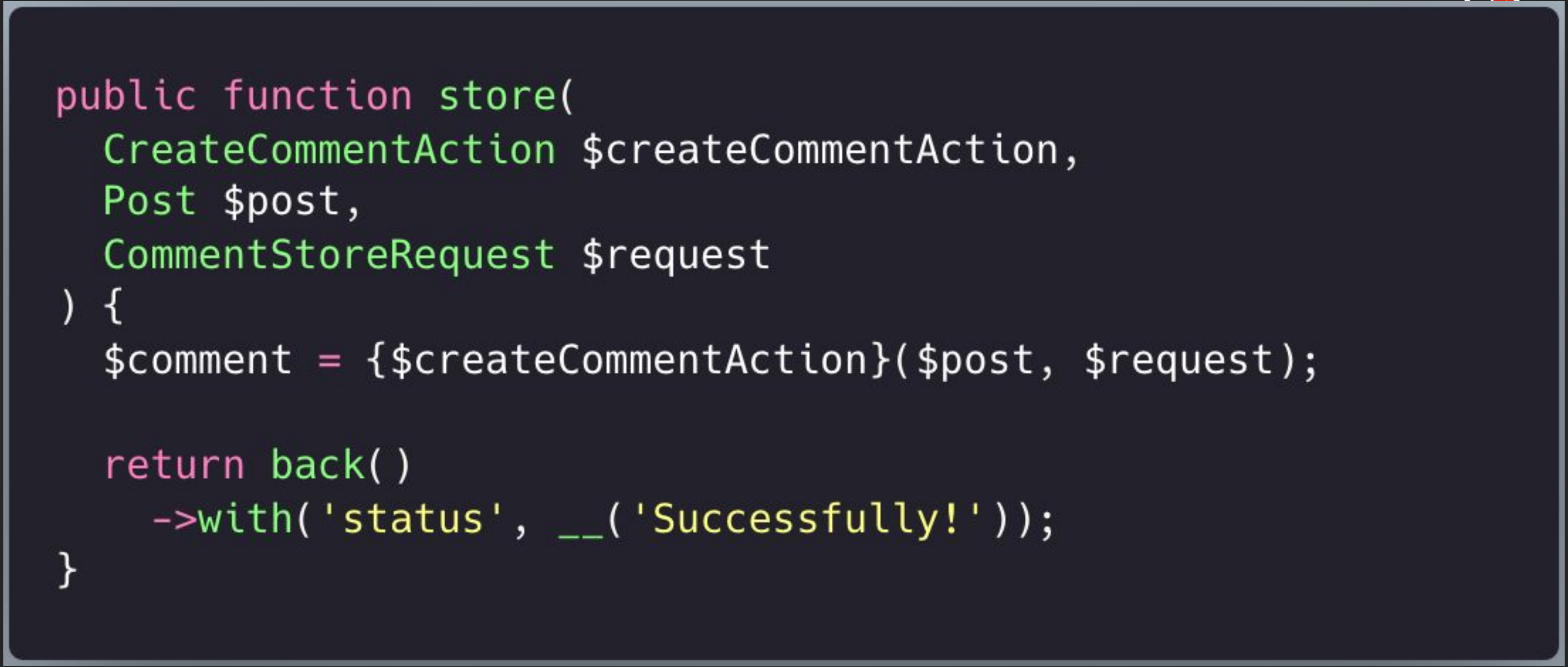
```
class CommentService
{
  public function create(
    Post $post,
    CommentStoreRequest $request
  ): Comment
  {
    // ...
  }
}
```



```php
protected CommentService $commentService;

// Inject comment serivce in `__construct`

public function store(
  Post $post,
  CommentStoreRequest $request
) {
  $comment = $this->commentService->createCommentByRequest(
    $request
  );

  return back()
    ->with('status', __('Successfully!'));
}
```

3. Actions/Commands Class (CQRS)

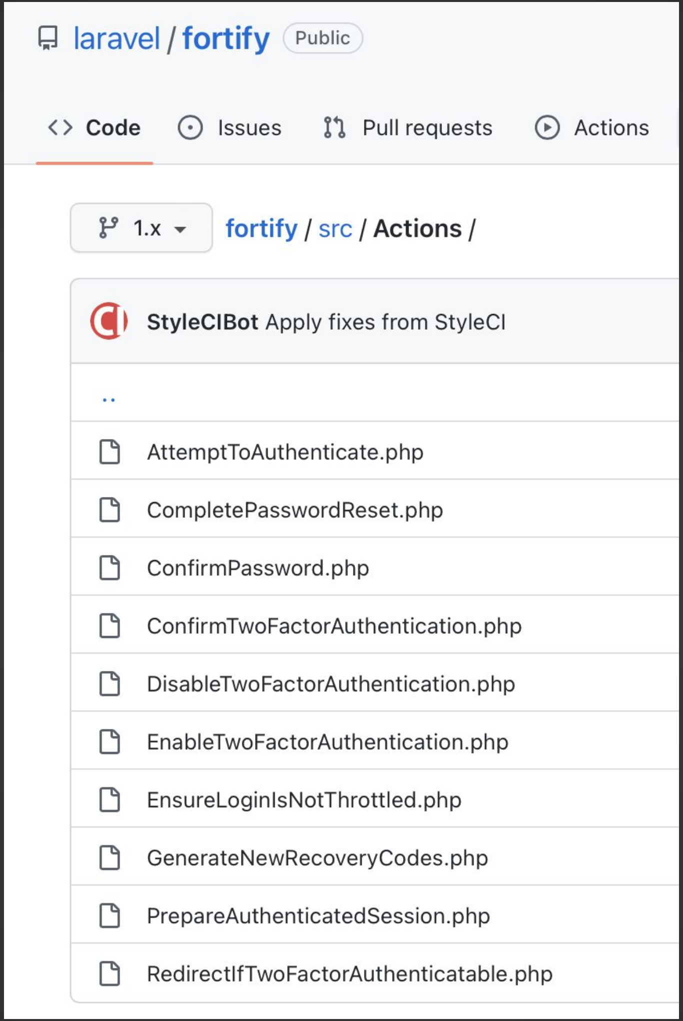
```php
class CreateCommentAction
{
  public function __invoke(
    Post $post,
    CommentStoreRequest $request
  ): Comment
  {
    // ...
  }
}
```



```php
public function store(
  CreateCommentAction $createCommentAction,
  Post $post,
  CommentStoreRequest $request
) {
  $comment = {$createCommentAction}($post, $request);

  return back()
    ->with('status', __('Successfully!'));
}
```

4. Together

Example From Laravel


laravel / fortify Public

Code Issues Pull requests Actions

1.x fortify / src / Actions /

StyleCIBot Apply fixes from StyleCI

..

AttemptToAuthenticate.php

CompletePasswordReset.php

ConfirmPassword.php

ConfirmTwoFactorAuthentication.php

DisableTwoFactorAuthentication.php

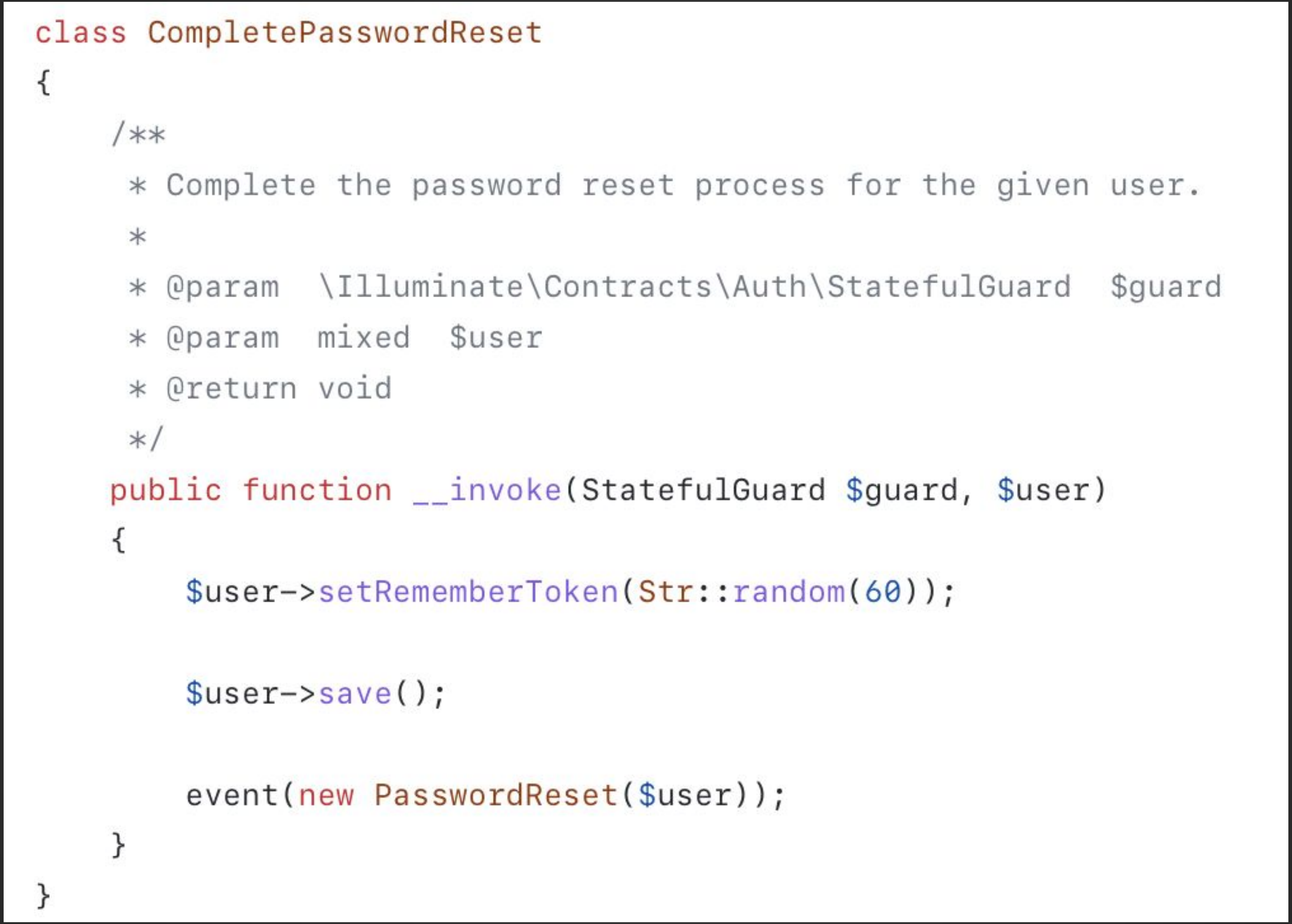
EnableTwoFactorAuthentication.php

EnsureLoginIsNotThrottled.php

GenerateNewRecoveryCodes.php

PrepareAuthenticatedSession.php

RedirectIfTwoFactorAuthenticatable.php



```php
class CompletePasswordReset
{
    /**
     * Complete the password reset process for the given user.
     *
     * @param  \Illuminate\Contracts\Auth\StatefulGuard  $guard
     * @param  mixed  $user
     * @return void
     */
    public function __invoke(StatefulGuard $guard, $user)
    {
        $user->setRememberToken(Str::random(60));

        $user->save();

        event(new PasswordReset($user));
    }
}
```

Next: Controller!

Cruddy by Design

Adam Wathan

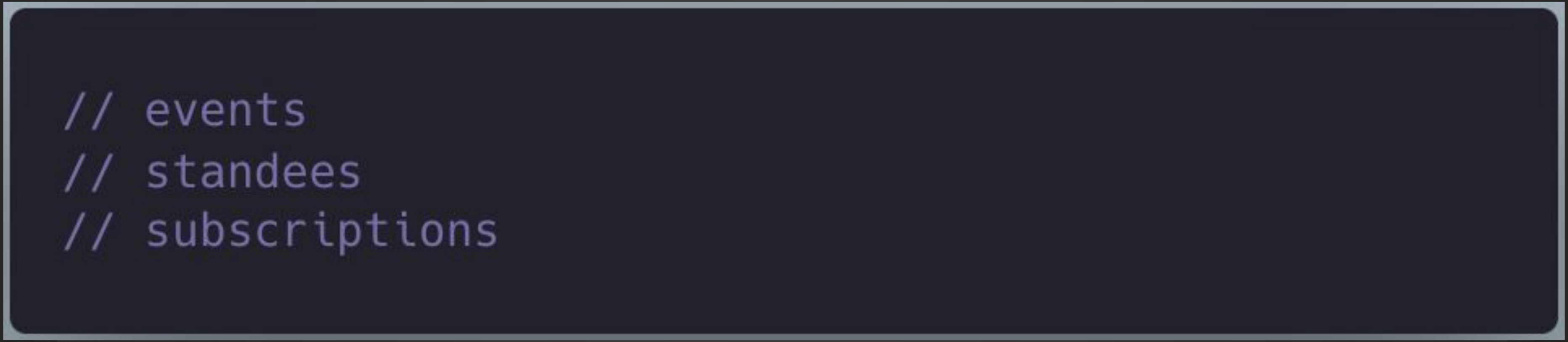
"More controllers doing less work obviates need for many other fancy patterns!"

David Heinemeier Hansson - Creator of RoR

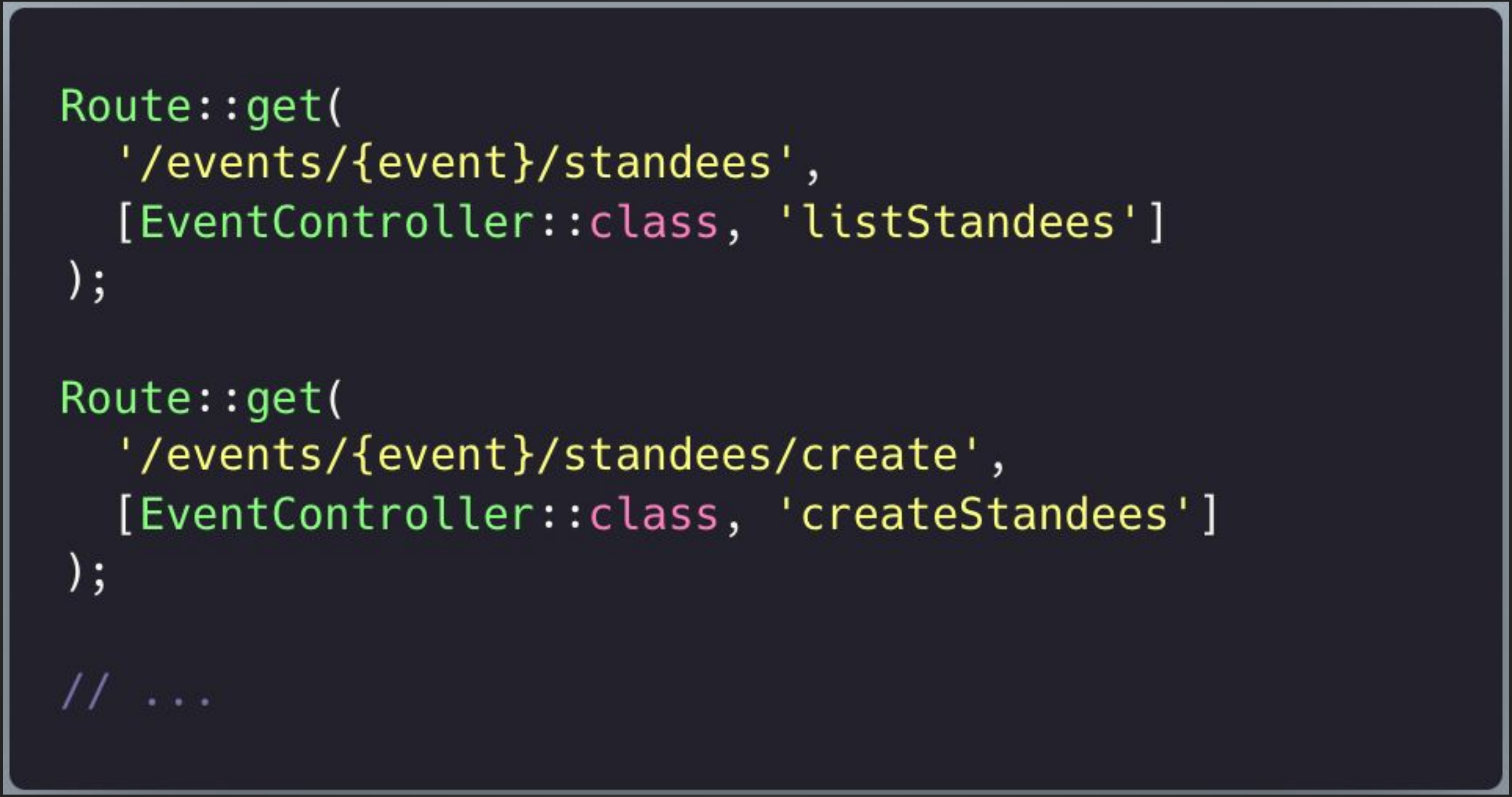
7 Standard Actions in Laravel

1. Index
2. Create
3. Store
4. Show
5. Edit
6. Update
7. Destroy

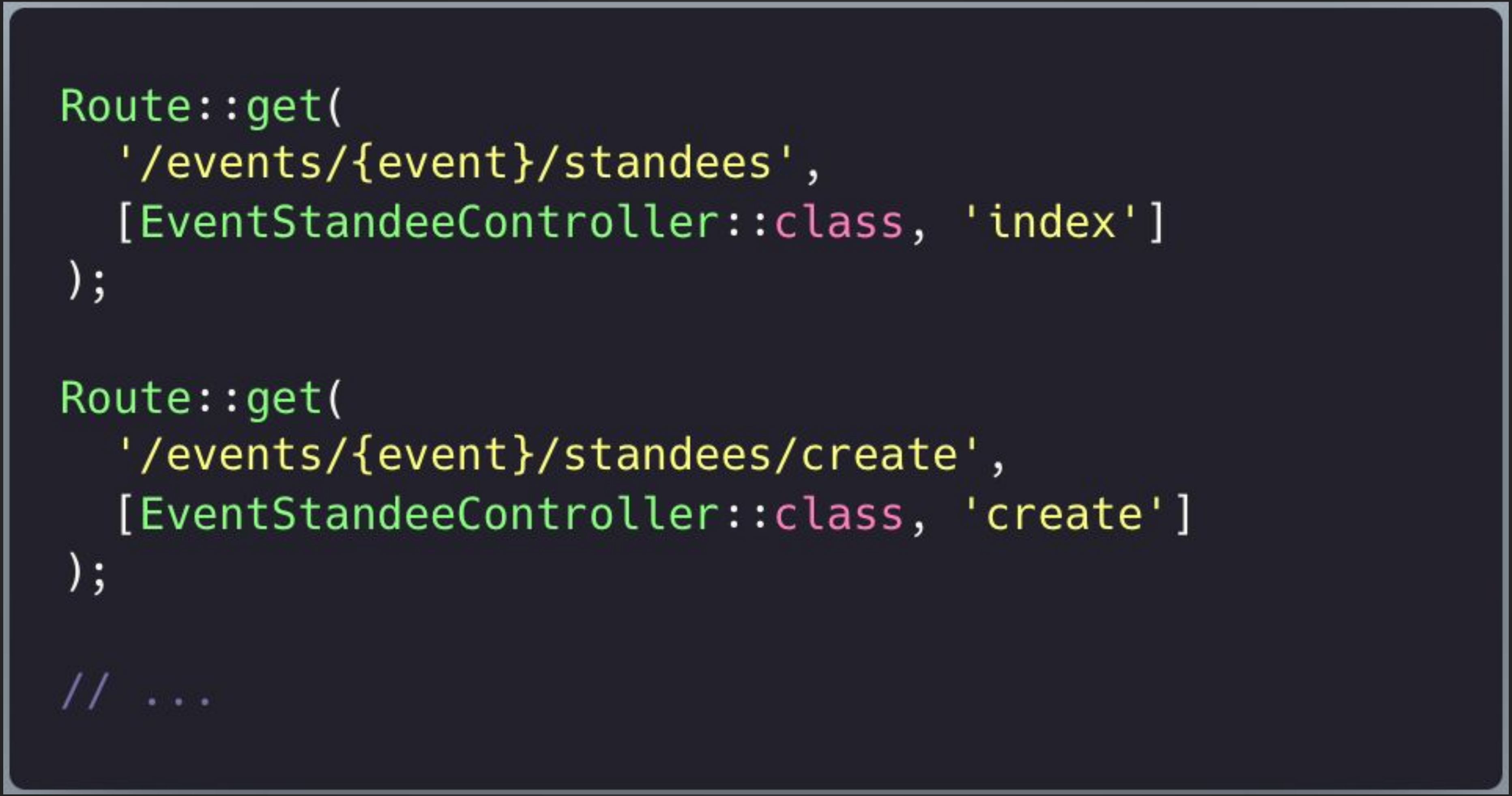
NEVER write **CUSTOM ACTION**



```
// events
// standees
// subscriptions
```



```php
Route::get(
  '/events/{event}/standees',
  [EventController::class, 'listStandees']
);

Route::get(
  '/events/{event}/standees/create',
  [EventController::class, 'createStandees']
);


// ...
```



```php
Route::get(
  '/events/{event}/standees',
  [EventStandeeController::class, 'index']
);

Route::get(
  '/events/{event}/standees/create',
  [EventStandeeController::class, 'create']
);

// ...
```

Tip #1: Nested Resource?

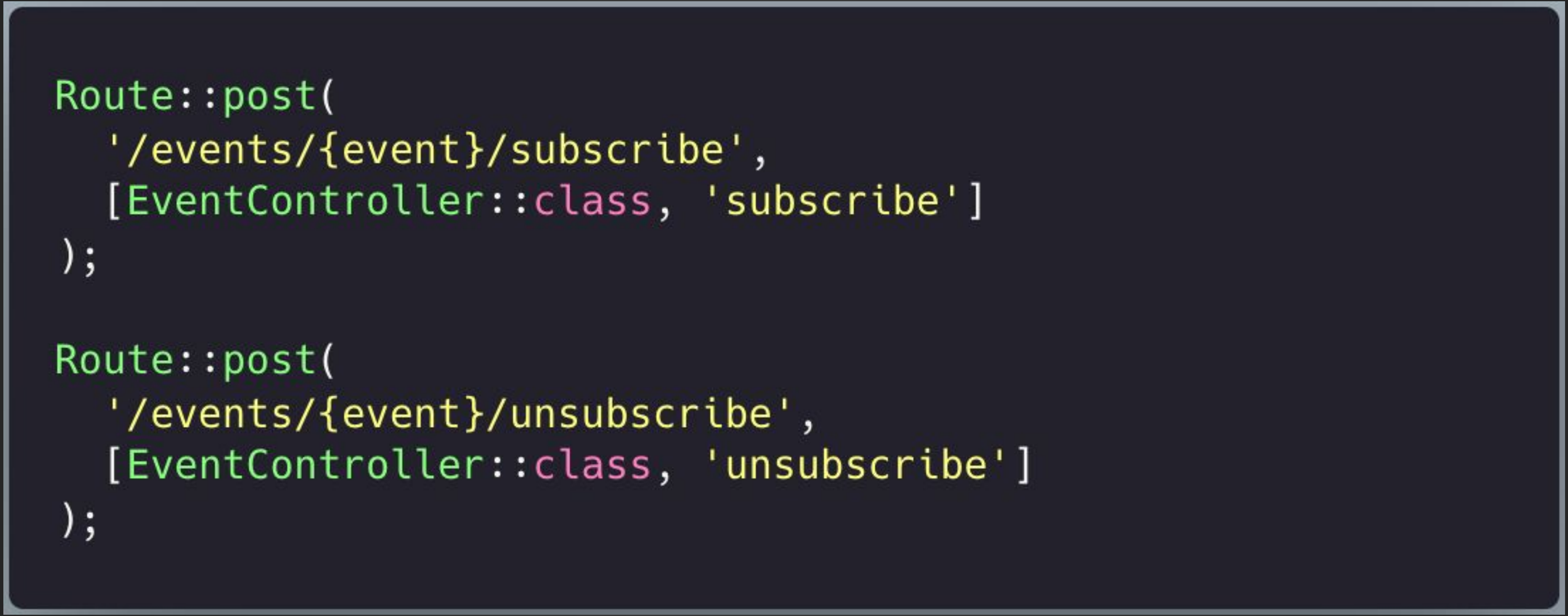
Create New Controller

```php
Route::post(
  '/events/{event}/cover-image',
  [EventController::class, 'updateCoverImage']
);

Route::delete(
  '/events/{event}/cover-image',
  [EventController::class, 'destroyCoverImage']
);
```

```php
Route::post(
    '/events/{event}/cover-image',
    [EventCoverImageController::class, 'store']
);

Route::delete(
    '/events/{event}/cover-image',
    [EventCoverImageController::class, 'destroy']
);
```

Tip #2: Edited Dependently?

Create New Controller

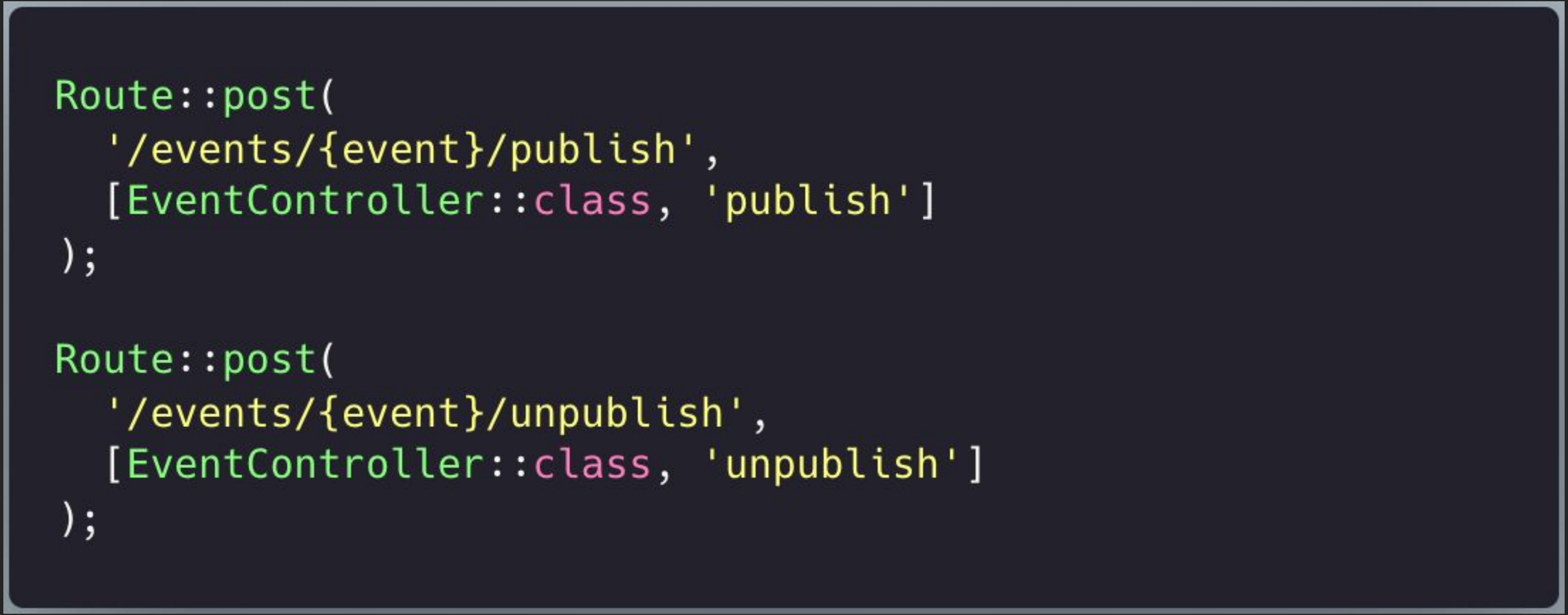


```php
Route::post(
   '/events/{event}/subscribe',
   [EventController::class, 'subscribe']
);

Route::post(
   '/events/{event}/unsubscribe',
   [EventController::class, 'unsubscribe']
);
```

```php
Route::post(
   '/events/{event}/subscriptions',
   [EventSubscriptionController::class, 'store']
);

Route::delete(
   '/events/{event}/subscriptions',
   [EventSubscriptionController::class, 'destroy']
);
```

Tip #3: Touches Pivot Records?

Create New Controller

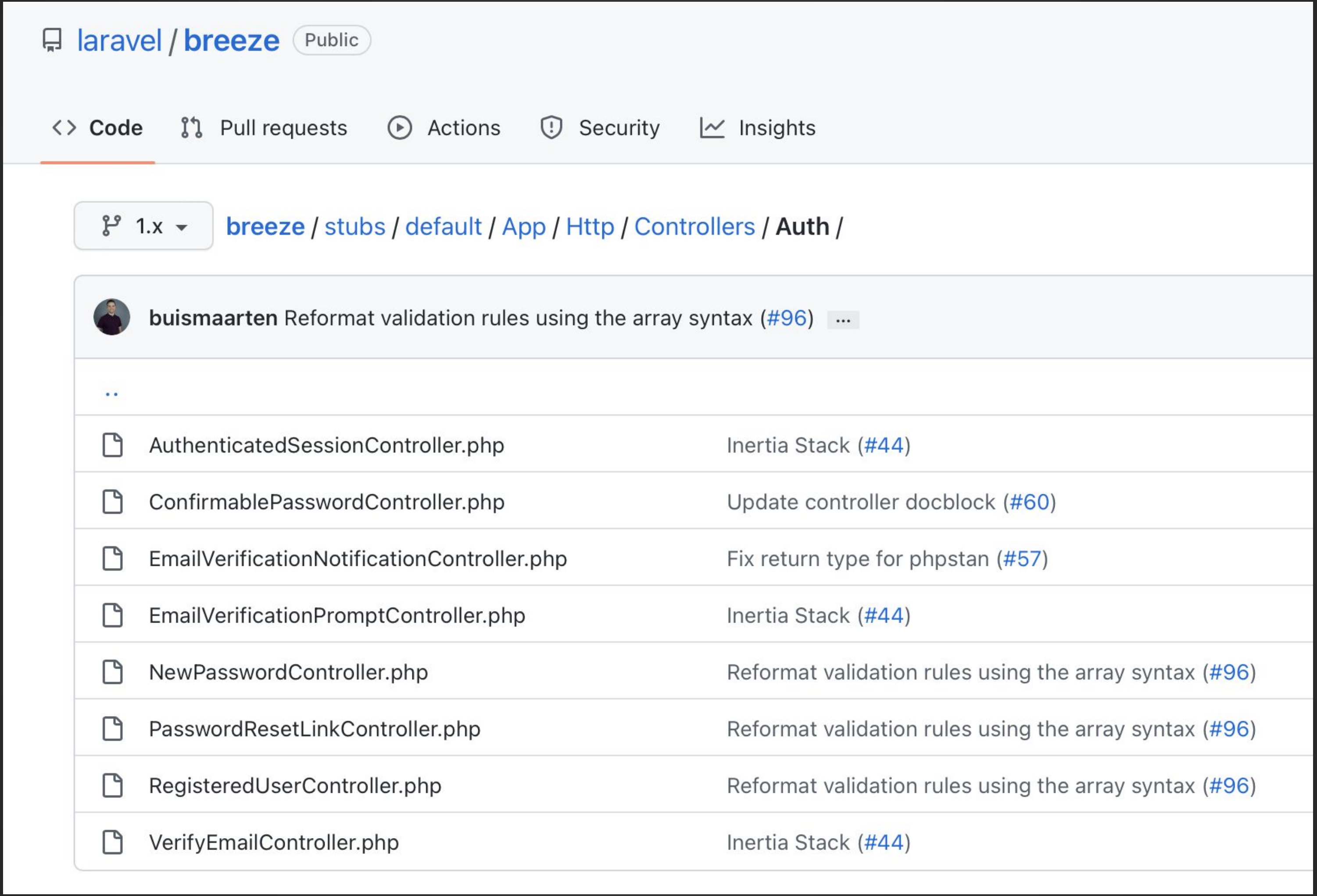

```php
Route::post(
  '/events/{event}/publish',
  [EventController::class, 'publish']
);

Route::post(
  '/events/{event}/unpublish',
  [EventController::class, 'unpublish']
);
```

```php
Route::post(
    '/published-events/{event}',
    [PublishedEventController::class, 'store']
);

Route::delete(
    '/published-events/{event}',
    [PublishedEventController::class, 'destroy']
);
```

Tip #4: Transitions State?

Create New Controller

Example From Laravel

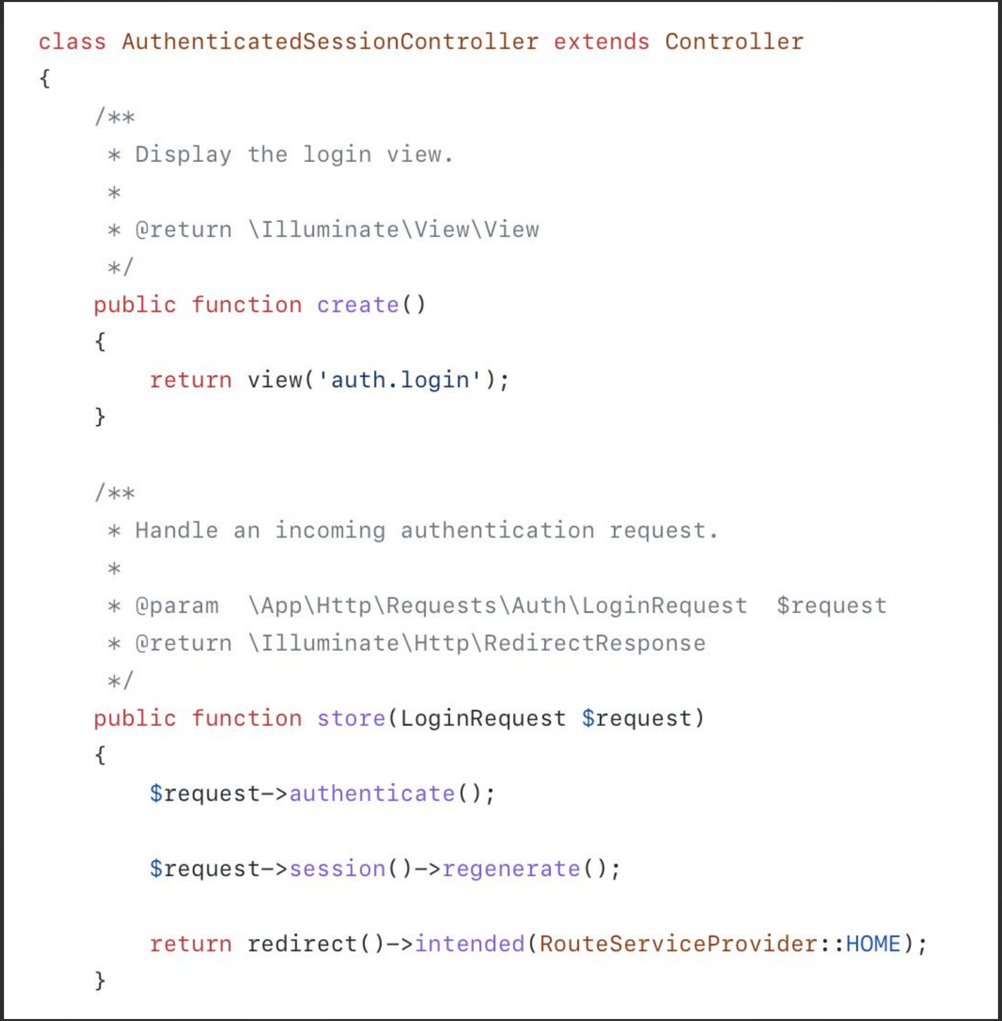


breeze / stubs / default / App / Http / Controllers / Auth /

buismaarten Reformat validation rules using the array syntax (#96) ...

..

AuthenticatedSessionController.php	Inertia Stack (#44)
ConfirmablePasswordController.php	Update controller docblock (#60)
EmailVerificationNotificationController.php	Fix return type for phpstan (#57)
EmailVerificationPromptController.php	Inertia Stack (#44)
NewPasswordController.php	Reformat validation rules using the array syntax (#96)
PasswordResetLinkController.php	Reformat validation rules using the array syntax (#96)
RegisteredUserController.php	Reformat validation rules using the array syntax (#96)
VerifyEmailController.php	Inertia Stack (#44)

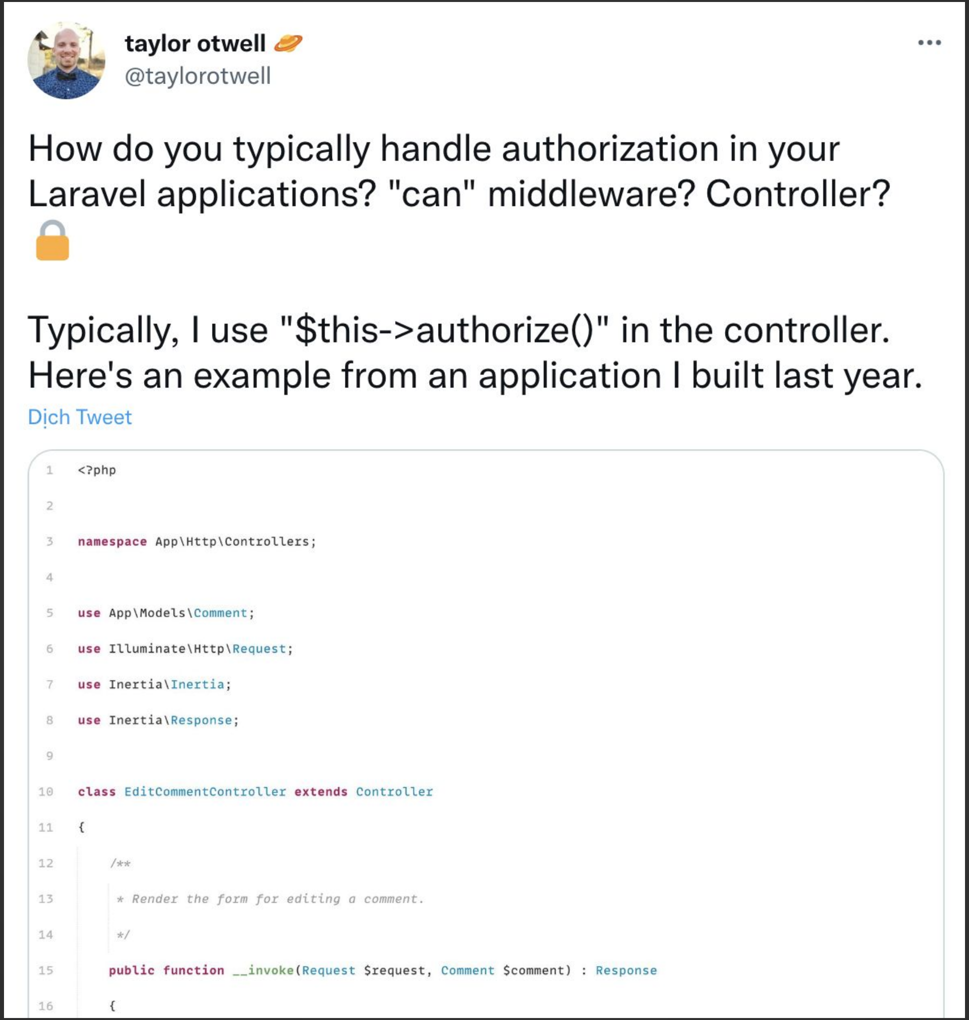


```php
class AuthenticatedSessionController extends Controller
{
    /**
     * Display the login view.
     *
     * @return \Illuminate\View\View
     */
    public function create()
    {
        return view('auth.login');
    }

    /**
     * Handle an incoming authentication request.
     *
     * @param  \App\Http\Requests\Auth\LoginRequest  $request
     * @return \Illuminate\Http\RedirectResponse
     */
    public function store(LoginRequest $request)
    {
        $request->authenticate();

        $request->session()->regenerate();

        return redirect()->intended(RouteServiceProvider::HOME);
    }
```

Or Single Action Controller?

Invokable Controller

```
Comments/CommentListController.php
Comments/CommentDetailController.php
Comments/CommentCreateController.php
Comments/CommentStoreController.php
Comments/CommentEditController.php
Comments/CommentUpdateController.php
Comments/CommentDestroyController.php
```



taylor otwell 🪐
@taylorotwell

How do you typically handle authorization in your Laravel applications? "can" middleware? Controller? 🔒

Typically, I use "$this->authorize()" in the controller. Here's an example from an application I built last year.

Dịch Tweet

```php
<?php

namespace App\Http\Controllers;

use App\Models\Comment;
use Illuminate\Http\Request;
use Inertia\Inertia;
use Inertia\Response;

class EditCommentController extends Controller
{
    /**
     * Render the form for editing a comment.
     */
    public function __invoke(Request $request, Comment $comment) : Response
    {
```

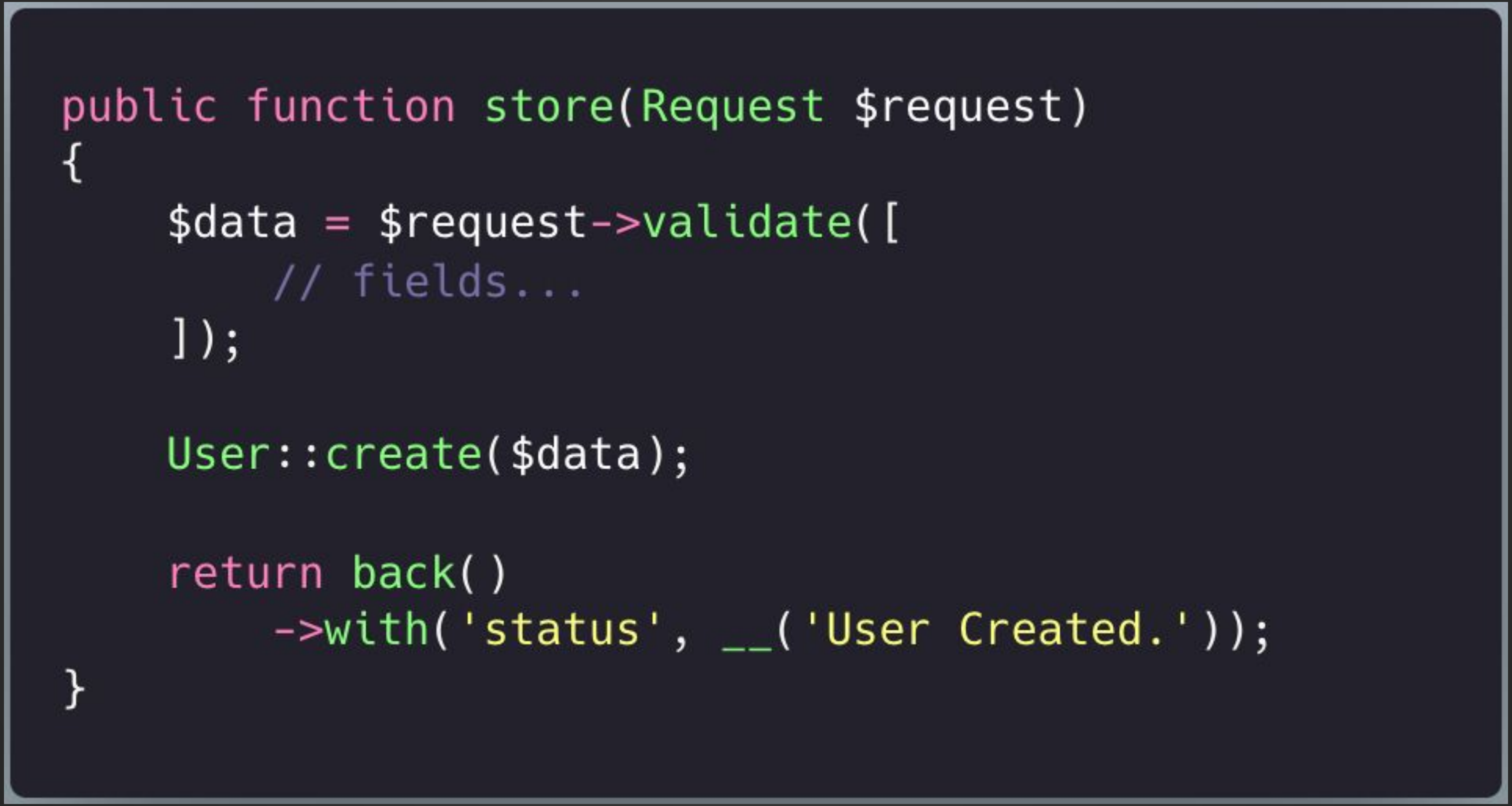
4. Complexity?

YAGNI

"You aren't gonna need it"

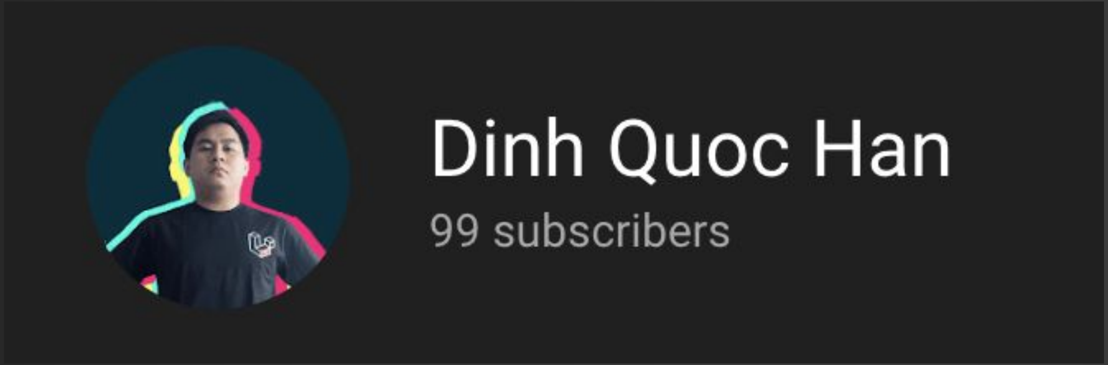
Back To The **BASIC**

app/Http/Controllers/UserController.php



```php
public function store(Request $request)
{
    $data = $request->validate([
        // fields...
    ]);

    User::create($data);

    return back()
        ->with('status', __('User Created.'));
}
```

1 Phút Quảng Cáo!



Dinh Quoc Han

99 subscribers

Thank You
Q&A