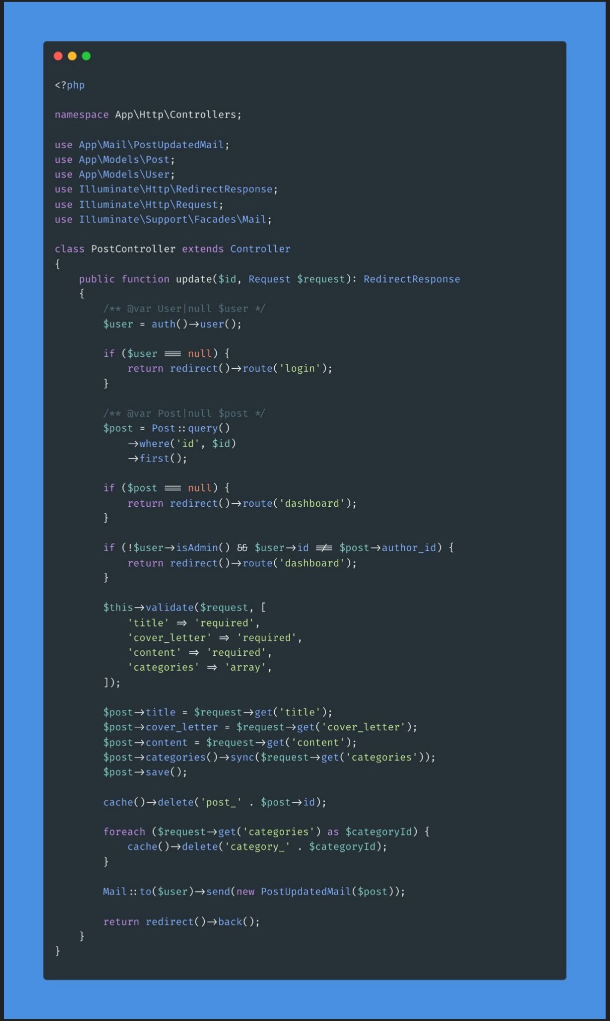


```php
<?php

namespace App\Http\Controllers;

use App\Mail\PostUpdatedMail;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class PostController extends Controller
{
    public function update($id, Request $request): RedirectResponse
    {
        /** @var User|null $user */
        $user = auth()->user();

        if ($user === null) {
            return redirect()->route('login');
        }

        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        if ($post === null) {
            return redirect()->route('dashboard');
        }

        if (!$user->isAdmin() && $user->id !== $post->author_id) {
            return redirect()->route('dashboard');
        }

        $this->validate($request, [
            'title' => 'required',
            'cover_letter' => 'required',
            'content' => 'required',
            'categories' => 'array',
        ]);

        $post->title = $request->get('title');
        $post->cover_letter = $request->get('cover_letter');
        $post->content = $request->get('content');
        $post->categories()->sync($request->get('categories'));
        $post->save();

        cache()->delete('post_' . $post->id);

        foreach ($request->get('categories') as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }

        Mail::to($user)->send(new PostUpdatedMail($post));

        return redirect()->back();
    }
}
```

TEKNASYON PHPkonf.

Beyond Controllers in Laravel Projects

Finding the right place for logic

Mücahit Cücen

Who am I? (Mücahit Cücen)

- Software Developer [at] Moneo
- 5+ years experience of professional software dev.
- Self-taught Software Developer
- Bahcesehir University / Computer Eng.
- Marmara University / Computer Prog.
- IstanbulPHP user group lead

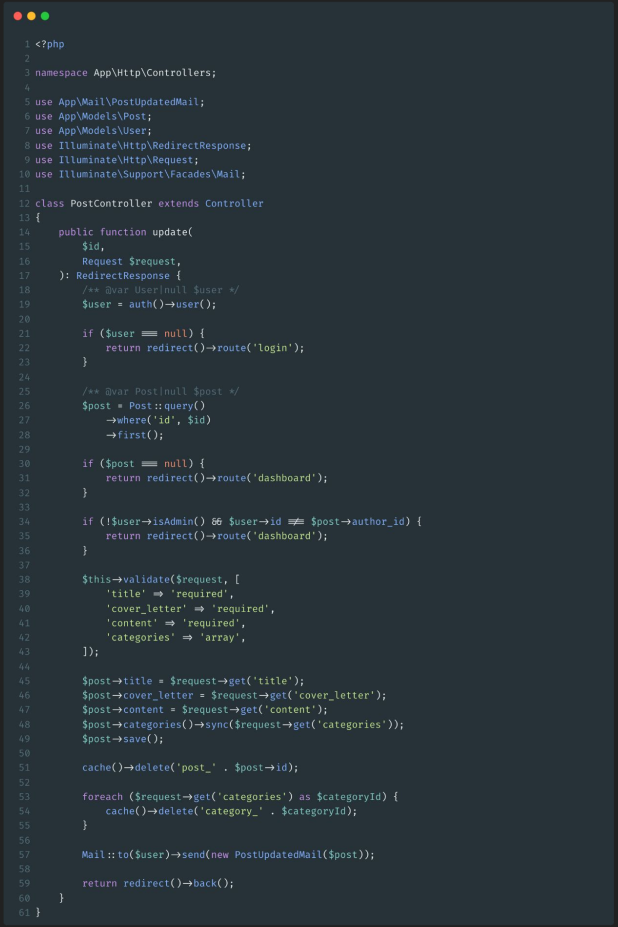
Headings

1. The disadvantages of doing all tasks within a single method.
2. How can we simplify complex methods?
3. The advantages of distributing responsibilities within a method?
4. The features that Laravel offers us

Main points

- Clean code
- Single responsibility
- Don’t Repeat Yourself
- Maintainability
- Reusability
- Tests

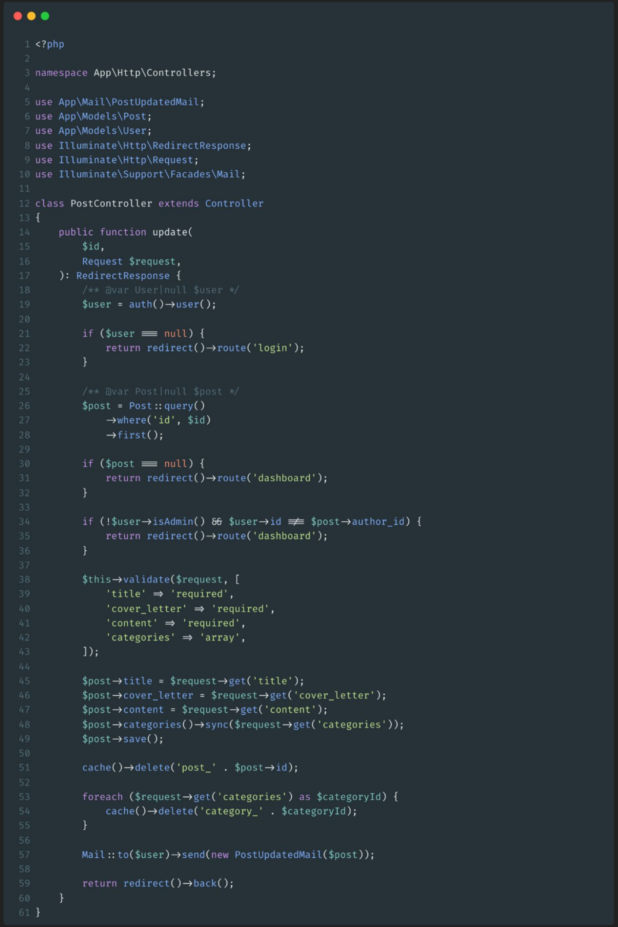
Disadvantages



```php
<?php

namespace App\Http\Controllers;

use App\Mail\PostUpdatedMail;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        /** @var User|null $user */
        $user = auth()->user();

        if ($user === null) {
            return redirect()->route('login');
        }

        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        if ($post === null) {
            return redirect()->route('dashboard');
        }

        if (!$user->isAdmin() && $user->id !== $post->author_id) {
            return redirect()->route('dashboard');
        }

        $this->validate($request, [
            'title' => 'required',
            'cover_letter' => 'required',
            'content' => 'required',
            'categories' => 'array',
        ]);

        $post->title = $request->get('title');
        $post->cover_letter = $request->get('cover_letter');
        $post->content = $request->get('content');
        $post->categories()->sync($request->get('categories'));
        $post->save();

        cache()->delete('post_' . $post->id);

        foreach ($request->get('categories') as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }

        Mail::to($user)->send(new PostUpdatedMail($post));

        return redirect()->back();
    }
}
```

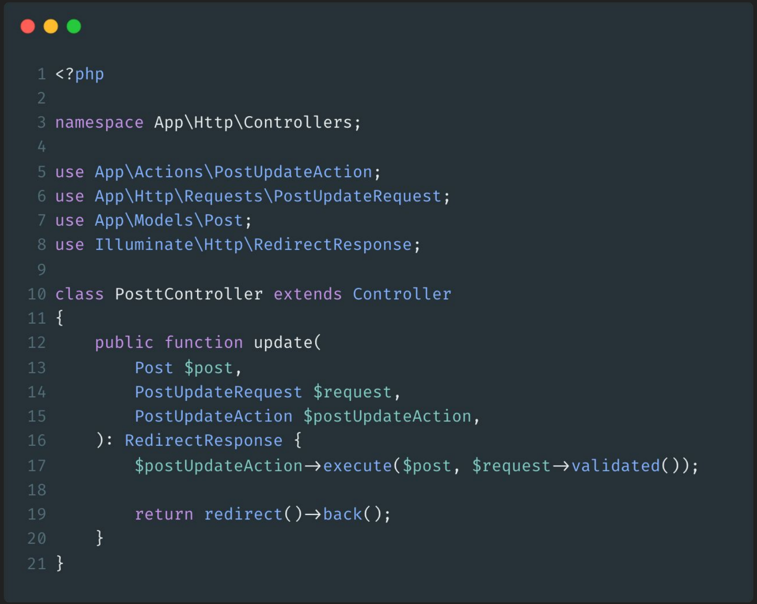
- Hard to read
- Repeated code
- Easy to make mistakes
- Lacks framework features
- High chance of problems
- High cost for maintenance and debug
- Can't write tests

How to clarify?



```php
<?php

namespace App\Http\Controllers;

use App\Mail\PostUpdatedMail;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        /** @var User|null $user */
        $user = auth()->user();

        if ($user === null) {
            return redirect()->route('login');
        }

        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        if ($post === null) {
            return redirect()->route('dashboard');
        }

        if (!$user->isAdmin() && $user->id !== $post->author_id) {
            return redirect()->route('dashboard');
        }

        $this->validate($request, [
            'title' => 'required',
            'cover_letter' => 'required',
            'content' => 'required',
            'categories' => 'array',
        ]);

        $post->title = $request->get('title');
        $post->cover_letter = $request->get('cover_letter');
        $post->content = $request->get('content');
        $post->categories()->sync($request->get('categories'));
        $post->save();

        cache()->delete('post_' . $post->id);

        foreach ($request->get('categories') as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }

        Mail::to($user)->send(new PostUpdatedMail($post));

        return redirect()->back();
    }
}
```

- Distribute responsibilities
- Use framework features
- Follow best practices
- Do not reinvent the wheel

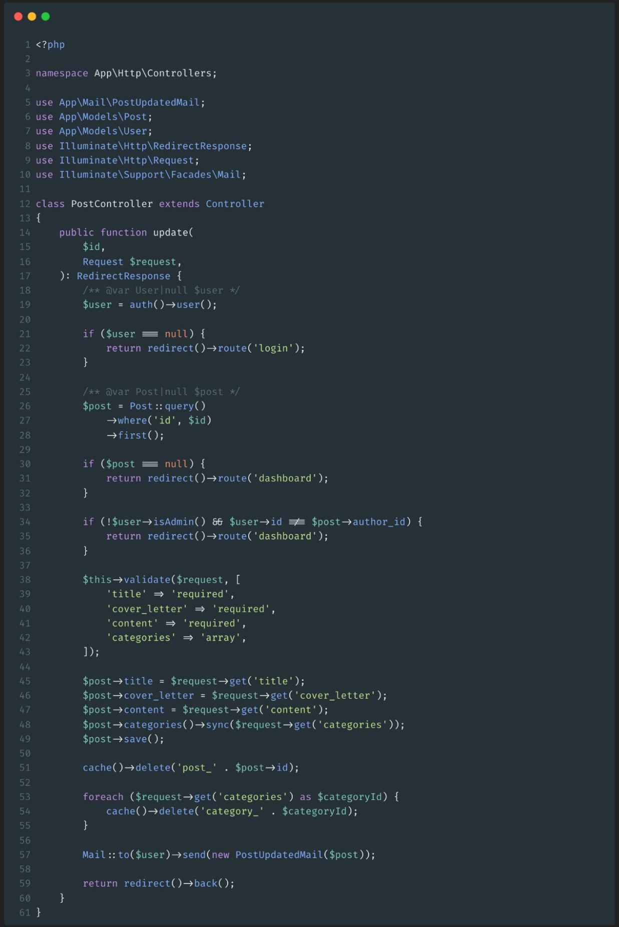


```php
<?php

namespace App\Http\Controllers;

use App\Actions\PostUpdateAction;
use App\Http\Requests\PostUpdateRequest;
use App\Models\Post;
use Illuminate\Http\RedirectResponse;

class PosttController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
        PostUpdateAction $postUpdateAction,
    ): RedirectResponse {
        $postUpdateAction→execute($post, $request→validated());

        return redirect()→back();
    }
}
```

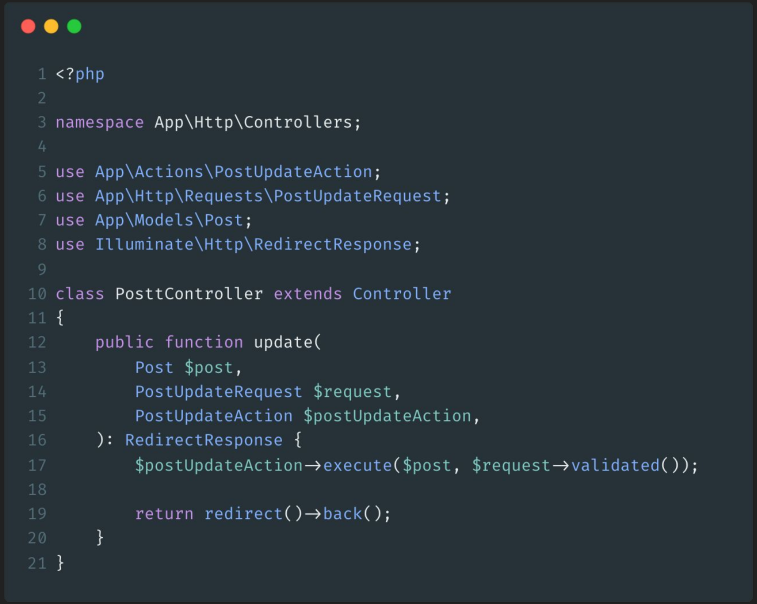
Advantages

- Easy to read
- Can be used again (D.R.Y.)
- Defensive
- Framework features
- Good for teamwork
- Easy to maintain and understand
- Can write tests

Let Laravel make it done!



```php
<?php

namespace App\Http\Controllers;

use App\Mail\PostUpdatedMail;
use App\Models\Post;
use App\Models\User;
use Illuminate\Http\RedirectResponse;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Mail;

class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        /** @var User|null $user */
        $user = auth()->user();

        if ($user === null) {
            return redirect()->route('login');
        }

        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        if ($post === null) {
            return redirect()->route('dashboard');
        }

        if (!$user->isAdmin() && $user->id !== $post->author_id) {
            return redirect()->route('dashboard');
        }

        $this->validate($request, [
            'title' => 'required',
            'cover_letter' => 'required',
            'content' => 'required',
            'categories' => 'array',
        ]);

        $post->title = $request->get('title');
        $post->cover_letter = $request->get('cover_letter');
        $post->content = $request->get('content');
        $post->categories()->sync($request->get('categories'));
        $post->save();

        cache()->delete('post_' . $post->id);

        foreach ($request->get('categories') as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }

        Mail::to($user)->send(new PostUpdatedMail($post));

        return redirect()->back();
    }
}
```

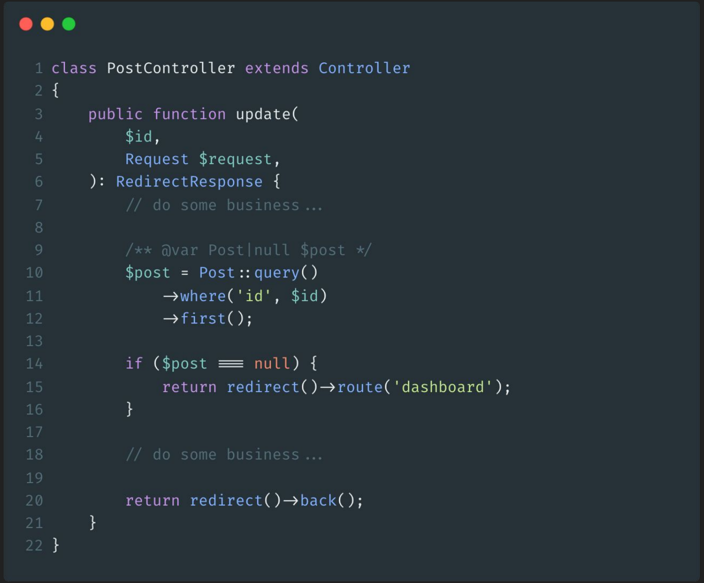


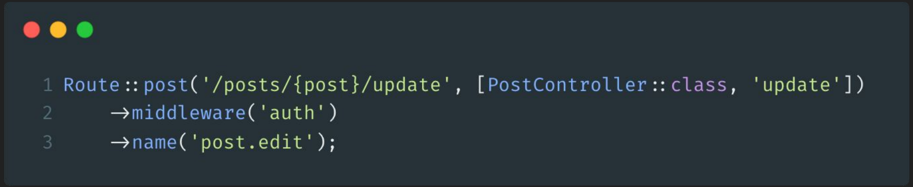
```php
<?php

namespace App\Http\Controllers;

use App\Actions\PostUpdateAction;
use App\Http\Requests\PostUpdateRequest;
use App\Models\Post;
use Illuminate\Http\RedirectResponse;

class PosttController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
        PostUpdateAction $postUpdateAction,
    ): RedirectResponse {
        $postUpdateAction->execute($post, $request->validated());

        return redirect()->back();
    }
}
```

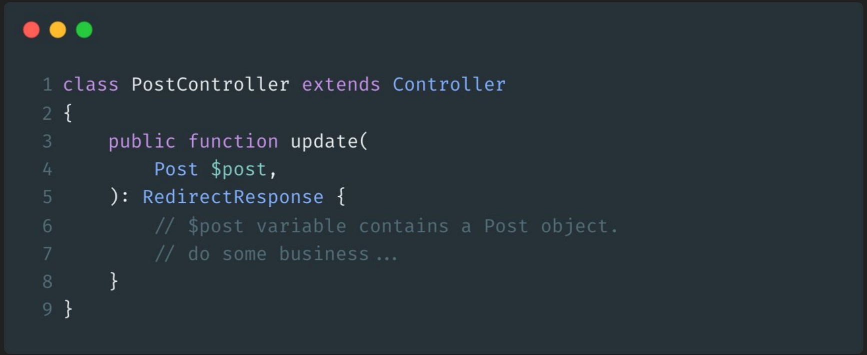
Route Model Binding

Auto inject

Generates 404


```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        // do some business...

        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        if ($post === null) {
            return redirect()->route('dashboard');
        }

        // do some business...

        return redirect()->back();
    }
}
```


```php
Route::post('/posts/{post}/update', [PostController::class, 'update'])
    ->middleware('auth')
    ->name('post.edit');
```


```php
class PostController extends Controller
{
    public function update(
        Post $post,
    ): RedirectResponse {
        // $post variable contains a Post object.
        // do some business...
    }
}
```

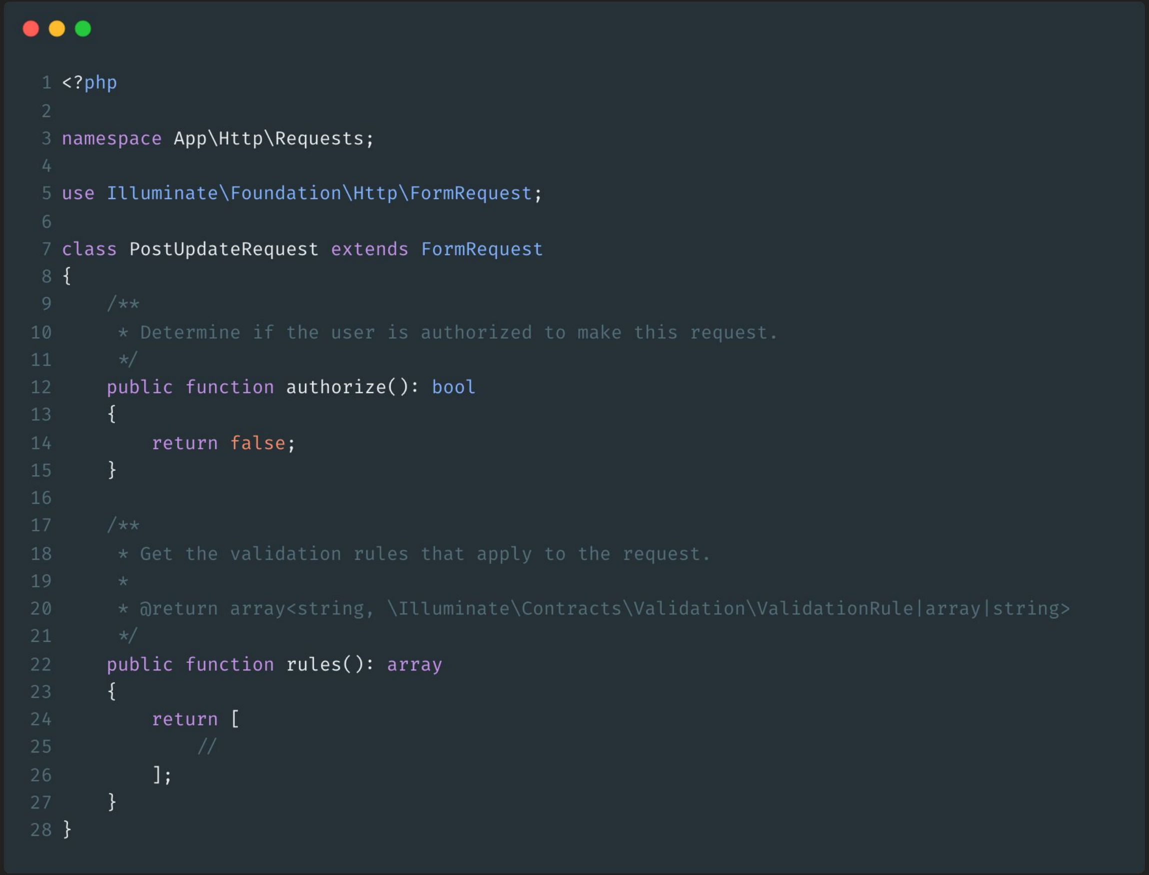
```php
Route::post('/posts/{post}/update', [PostController::class, 'update'])
    ->middleware('auth')
    ->name('post.edit');
```

```php
class PostController extends Controller
{
    public function update(
        Post $post,
    ): RedirectResponse {
        // $post variable contains a Post object.
        // do some business...
    }
}
```

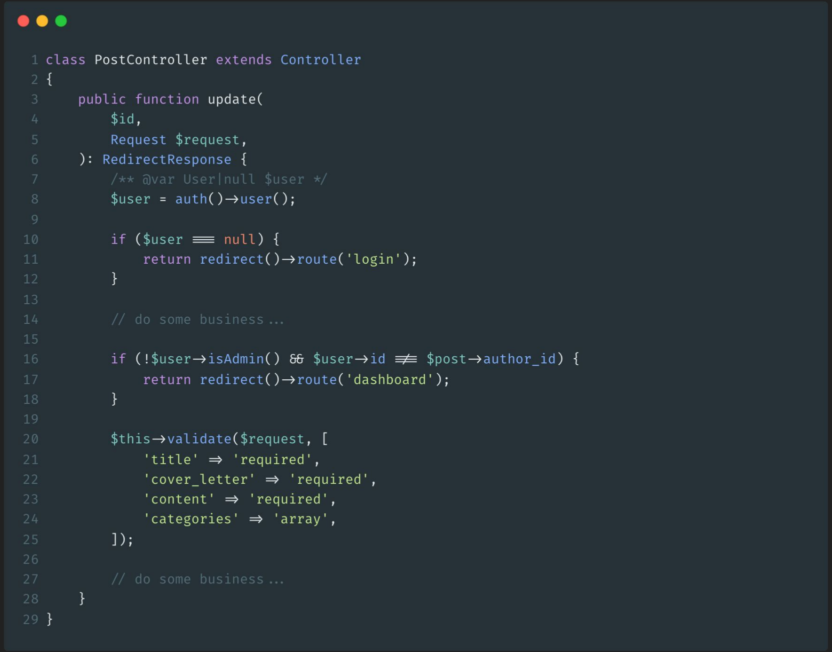
Form Requests

Authorization

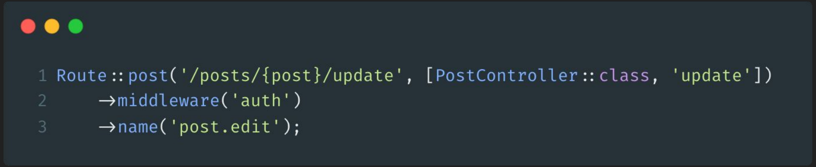
Validation



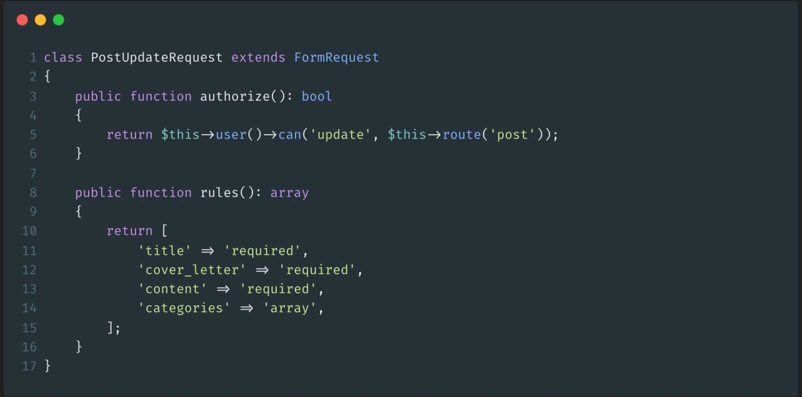
```php
<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class PostUpdateRequest extends FormRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return false;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>
     */
    public function rules(): array
    {
        return [
            //
        ];
    }
}
```



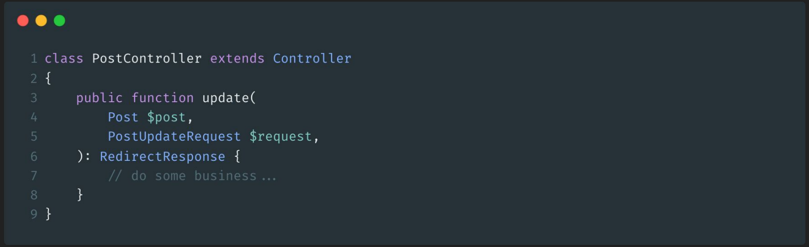
```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        /** @var User|null $user */
        $user = auth()->user();

        if ($user === null) {
            return redirect()->route('login');
        }

        // do some business...

        if (!$user->isAdmin() && $user->id !== $post->author_id) {
            return redirect()->route('dashboard');
        }

        $this->validate($request, [
            'title' => 'required',
            'cover_letter' => 'required',
            'content' => 'required',
            'categories' => 'array',
        ]);

        // do some business...
    }
}
```



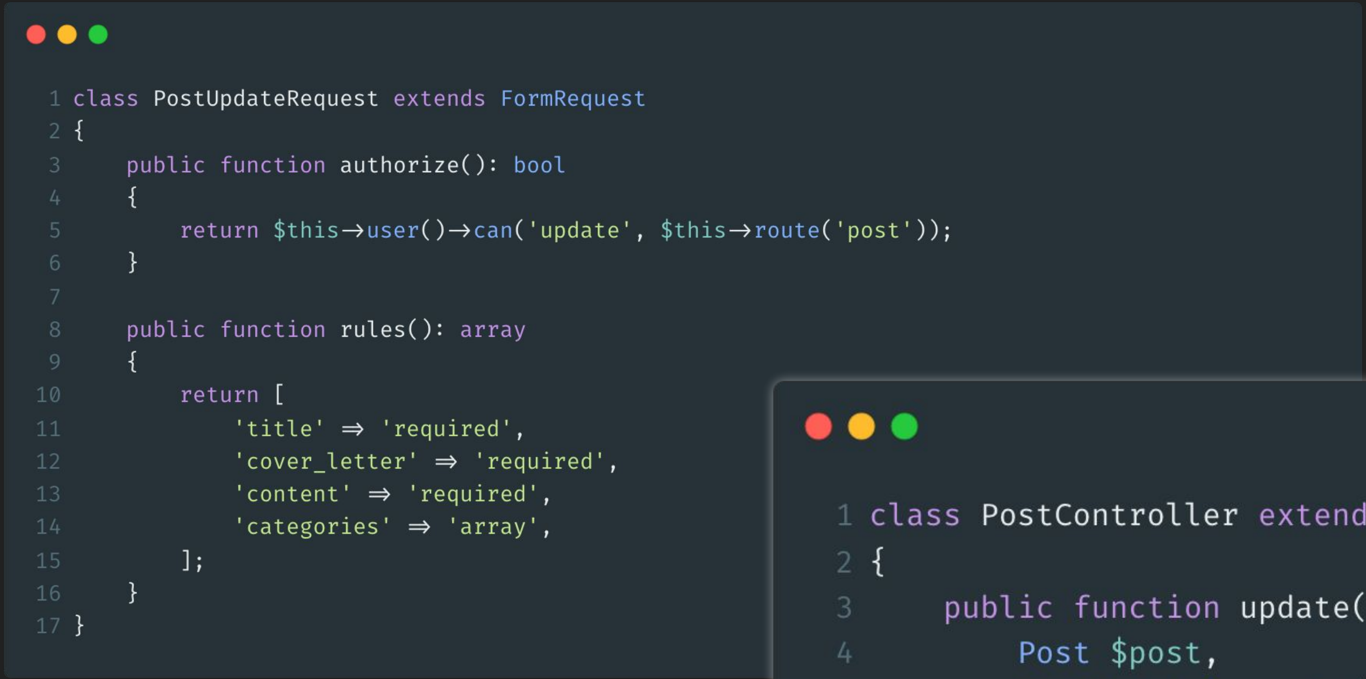
```php
Route::post('/posts/{post}/update', [PostController::class, 'update'])
    ->middleware('auth')
    ->name('post.edit');
```



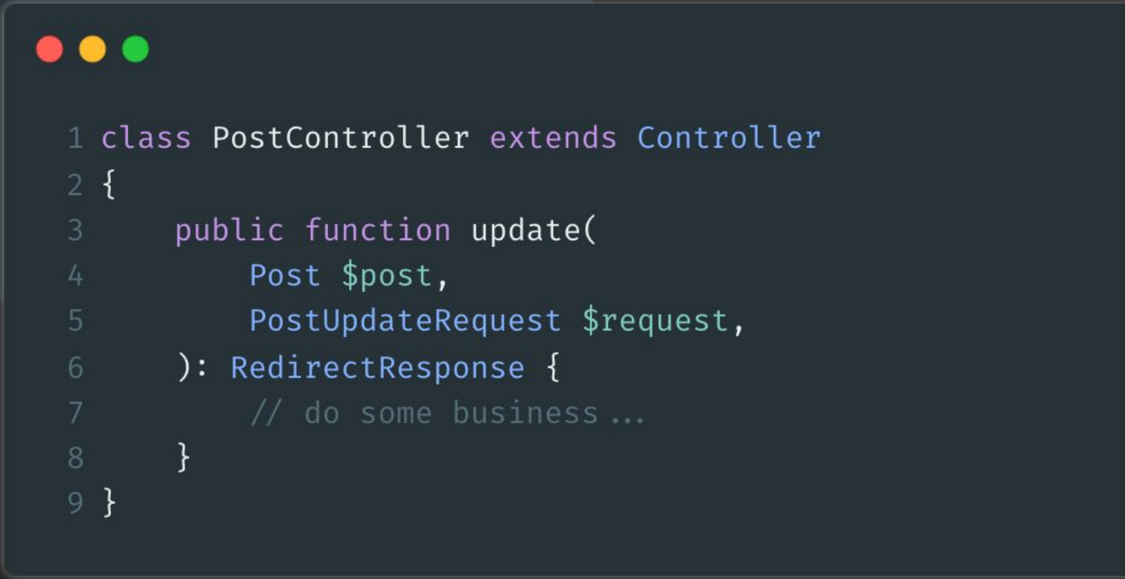
```php
class PostUpdateRequest extends FormRequest
{
    public function authorize(): bool
    {
        return $this->user()->can('update', $this->route('post'));
    }

    public function rules(): array
    {
        return [
            'title' => 'required',
            'cover_letter' => 'required',
            'content' => 'required',
            'categories' => 'array',
        ];
    }
}
```



```php
class PostController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
    ): RedirectResponse {
        // do some business...
    }
}
```



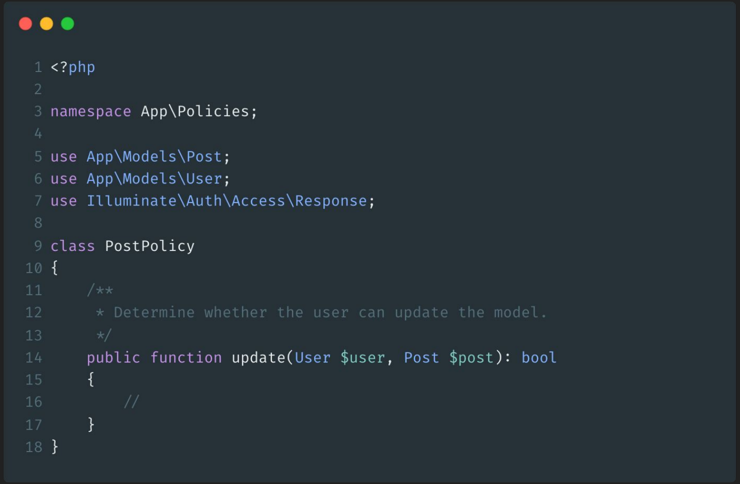
```php
class PostUpdateRequest extends FormRequest
{
    public function authorize(): bool
    {
        return $this→user()→can('update', $this→route('post'));
    }

    public function rules(): array
    {
        return [
            'title' ⇒ 'required',
            'cover_letter' ⇒ 'required',
            'content' ⇒ 'required',
            'categories' ⇒ 'array',
        ];
    }
}
```



```php
class PostController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
    ): RedirectResponse {
        // do some business...
    }
}
```

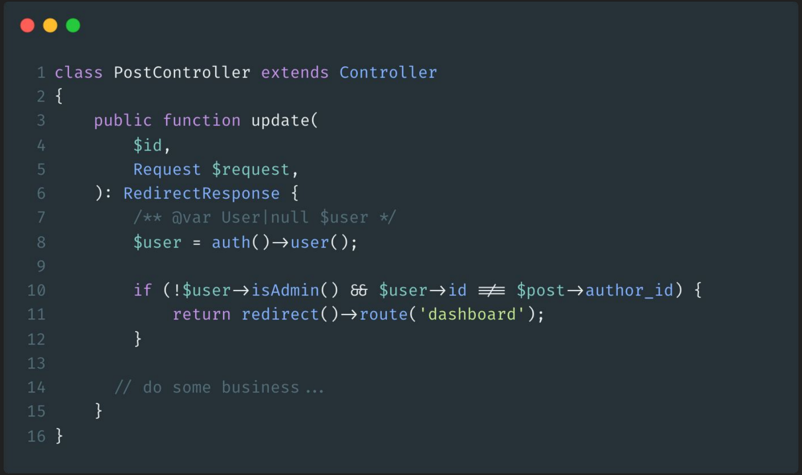
Policies

Authorize user actions

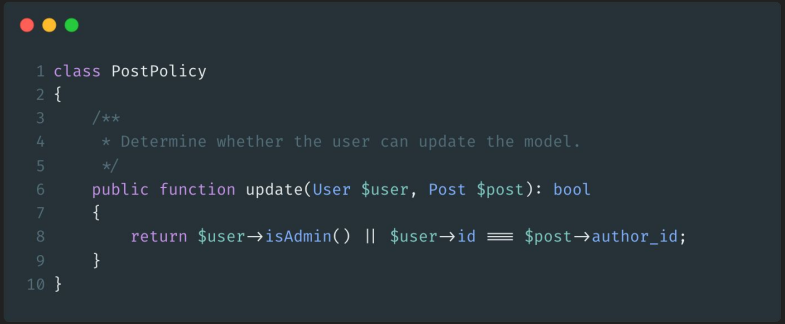


```php
<?php

namespace App\Policies;

use App\Models\Post;
use App\Models\User;
use Illuminate\Auth\Access\Response;

class PostPolicy
{
    /**
     * Determine whether the user can update the model.
     */
    public function update(User $user, Post $post): bool
    {
        //
    }
}
```

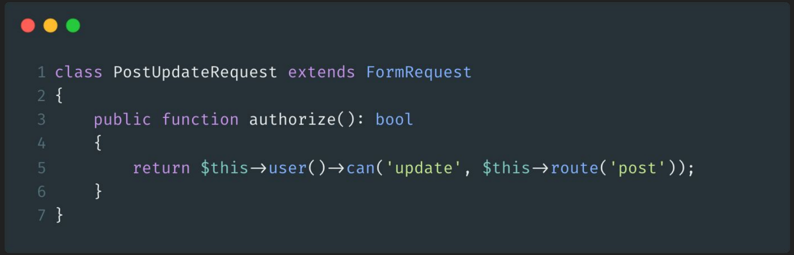
- viewAny
- view
- create
- update
- delete
- restore
- forceDelete



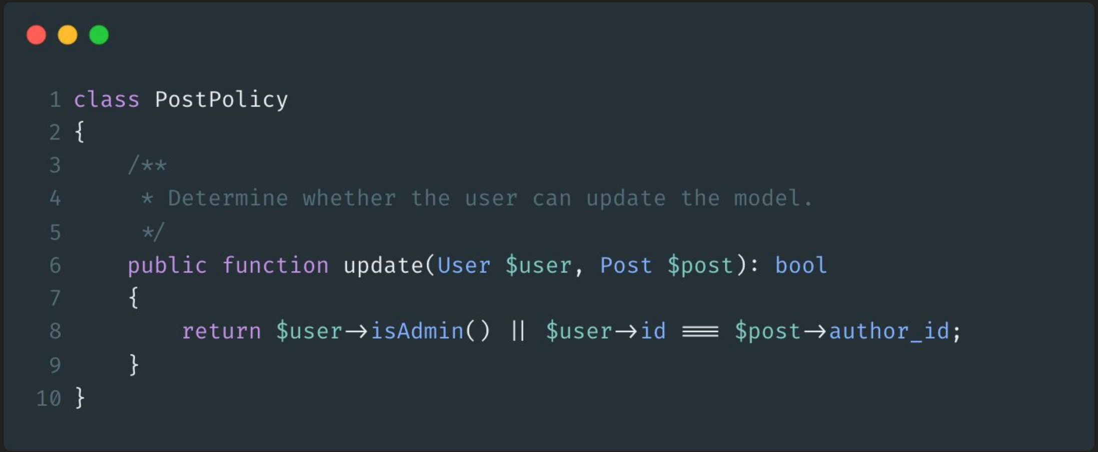
```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        /** @var User|null $user */
        $user = auth()->user();

        if (!$user->isAdmin() && $user->id !== $post->author_id) {
            return redirect()->route('dashboard');
        }

      // do some business ...
    }
}
```



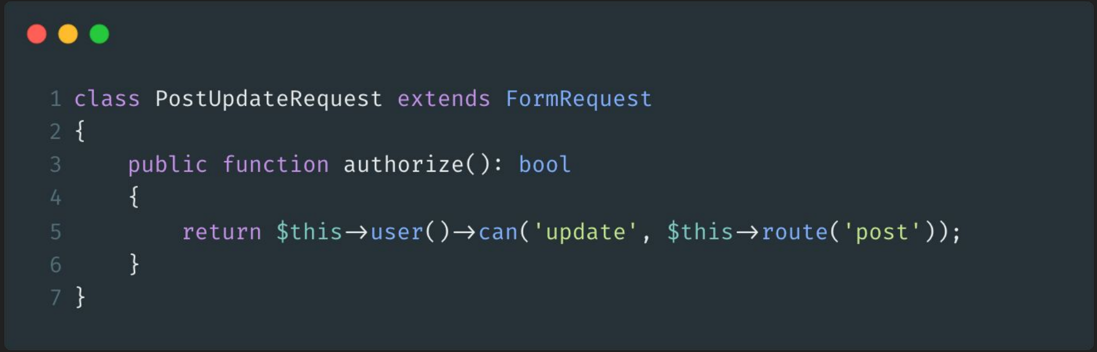
```php
class PostPolicy
{
    /**
     * Determine whether the user can update the model.
     */
    public function update(User $user, Post $post): bool
    {
        return $user->isAdmin() || $user->id === $post->author_id;
    }
}
```



```php
class PostUpdateRequest extends FormRequest
{
    public function authorize(): bool
    {
        return $this->user()->can('update', $this->route('post'));
    }
}
```



```php
class PostPolicy
{
    /**
     * Determine whether the user can update the model.
     */
    public function update(User $user, Post $post): bool
    {
        return $user->isAdmin() || $user->id === $post->author_id;
    }
}
```



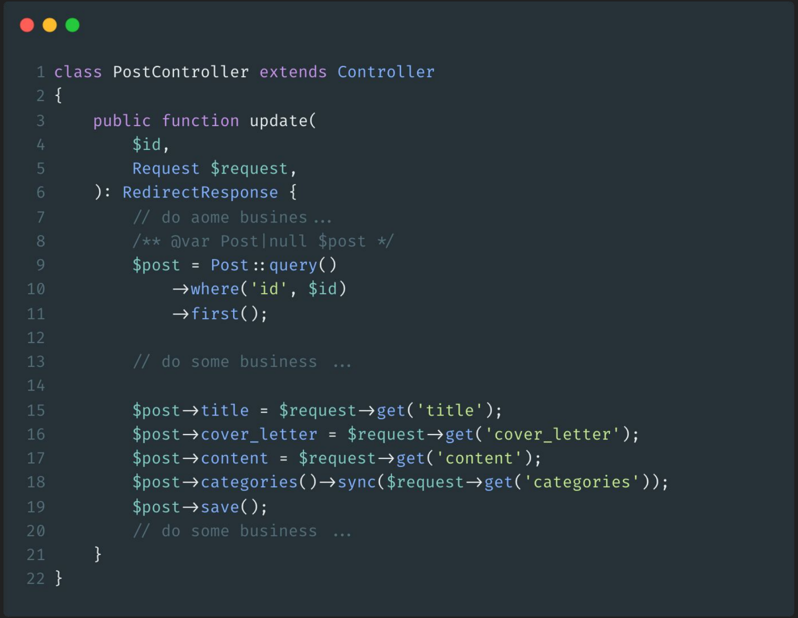
```php
class PostUpdateRequest extends FormRequest
{
    public function authorize(): bool
    {
        return $this->user()->can('update', $this->route('post'));
    }
}
```

Action Class

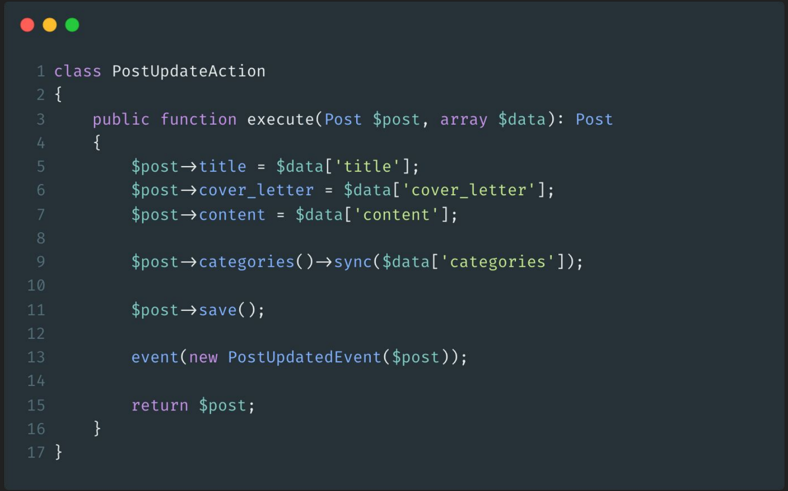
Simple PHP class

Contains an execute method
for only ONE action

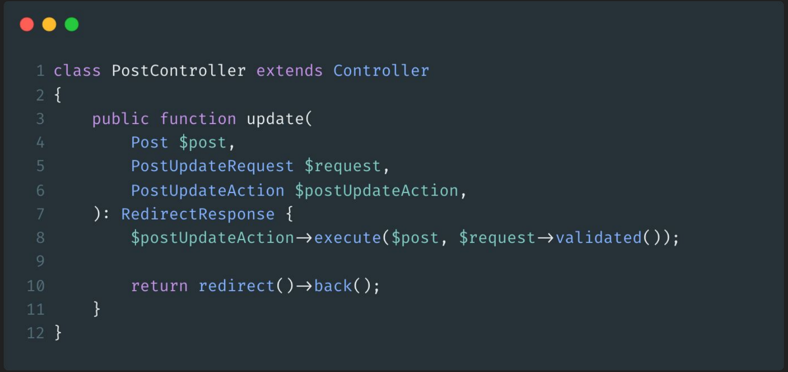
TEKNASYON PHPkonf.



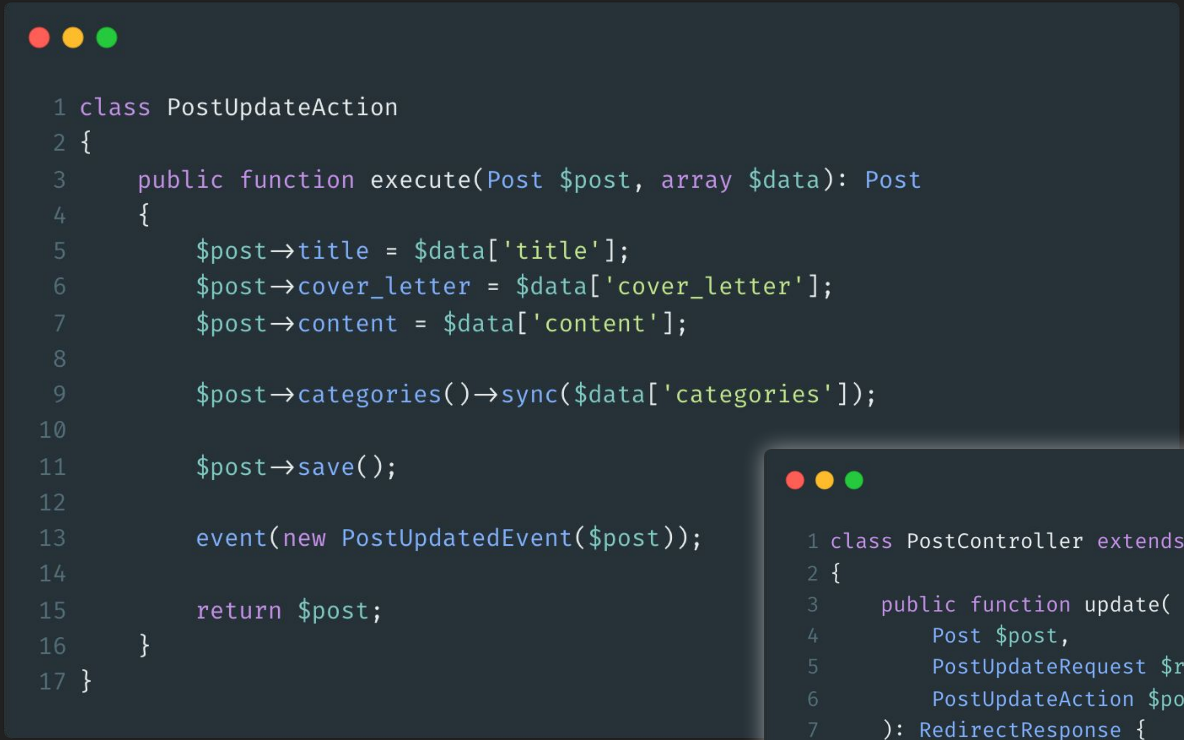
```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        // do aome busines ...
        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        // do some business ...

        $post->title = $request->get('title');
        $post->cover_letter = $request->get('cover_letter');
        $post->content = $request->get('content');
        $post->categories()->sync($request->get('categories'));
        $post->save();
        // do some business ...
    }
}
```



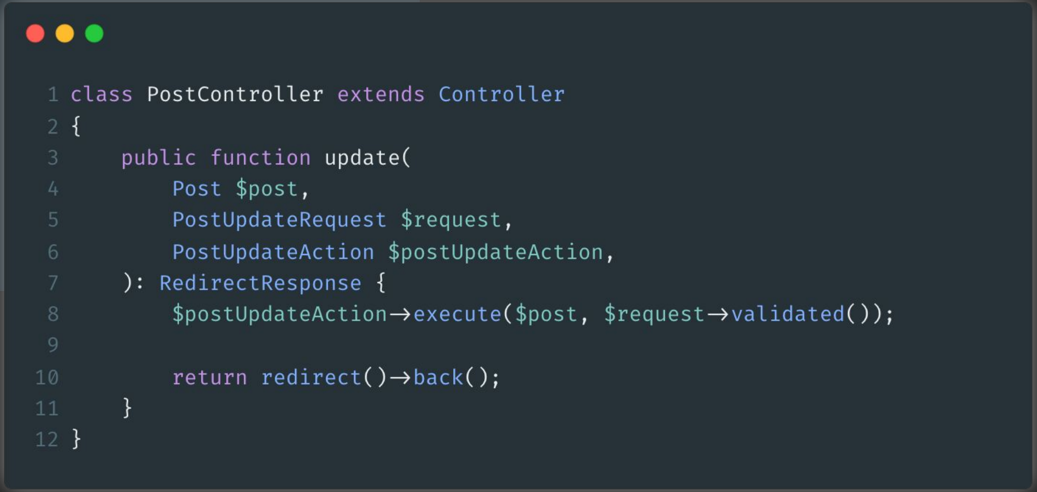
```php
class PostUpdateAction
{
    public function execute(Post $post, array $data): Post
    {
        $post->title = $data['title'];
        $post->cover_letter = $data['cover_letter'];
        $post->content = $data['content'];

        $post->categories()->sync($data['categories']);

        $post->save();

        event(new PostUpdatedEvent($post));

        return $post;
    }
}
```



```php
class PostController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
        PostUpdateAction $postUpdateAction,
    ): RedirectResponse {
        $postUpdateAction->execute($post, $request->validated());

        return redirect()->back();
    }
}
```



```php
class PostUpdateAction
{
    public function execute(Post $post, array $data): Post
    {
        $post->title = $data['title'];
        $post->cover_letter = $data['cover_letter'];
        $post->content = $data['content'];

        $post->categories()->sync($data['categories']);

        $post->save();

        event(new PostUpdatedEvent($post));

        return $post;
    }
}
```

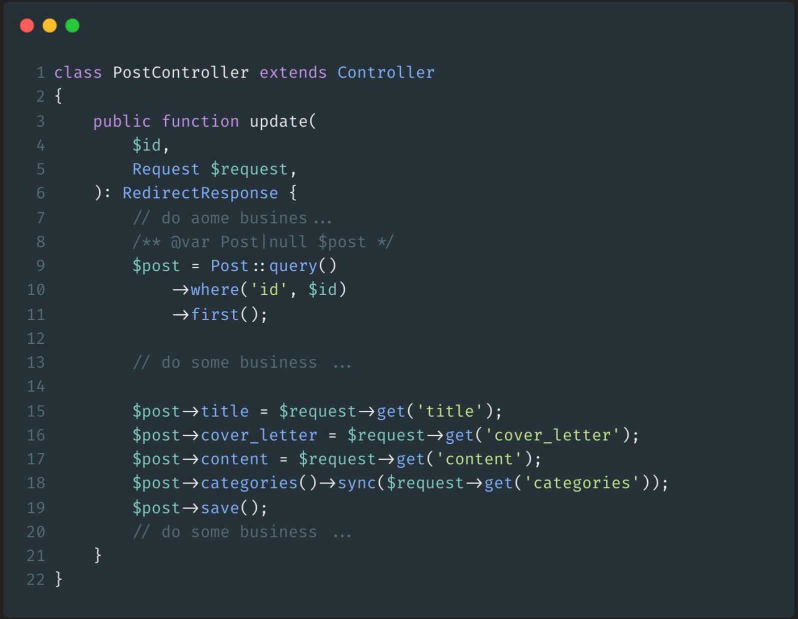


```php
class PostController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
        PostUpdateAction $postUpdateAction,
    ): RedirectResponse {
        $postUpdateAction->execute($post, $request->validated());

        return redirect()->back();
    }
}
```

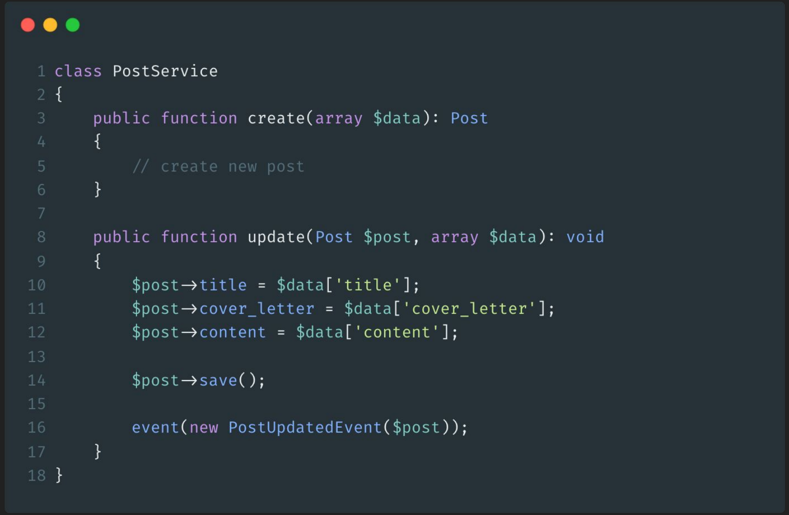
Service Class

Simple PHP class

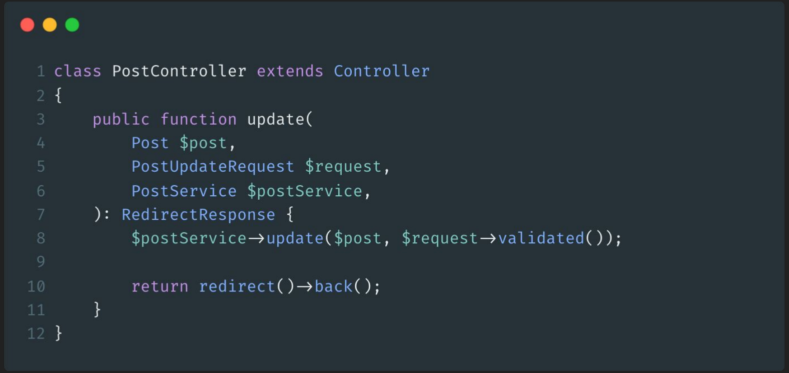
Contains one or more methods for related context



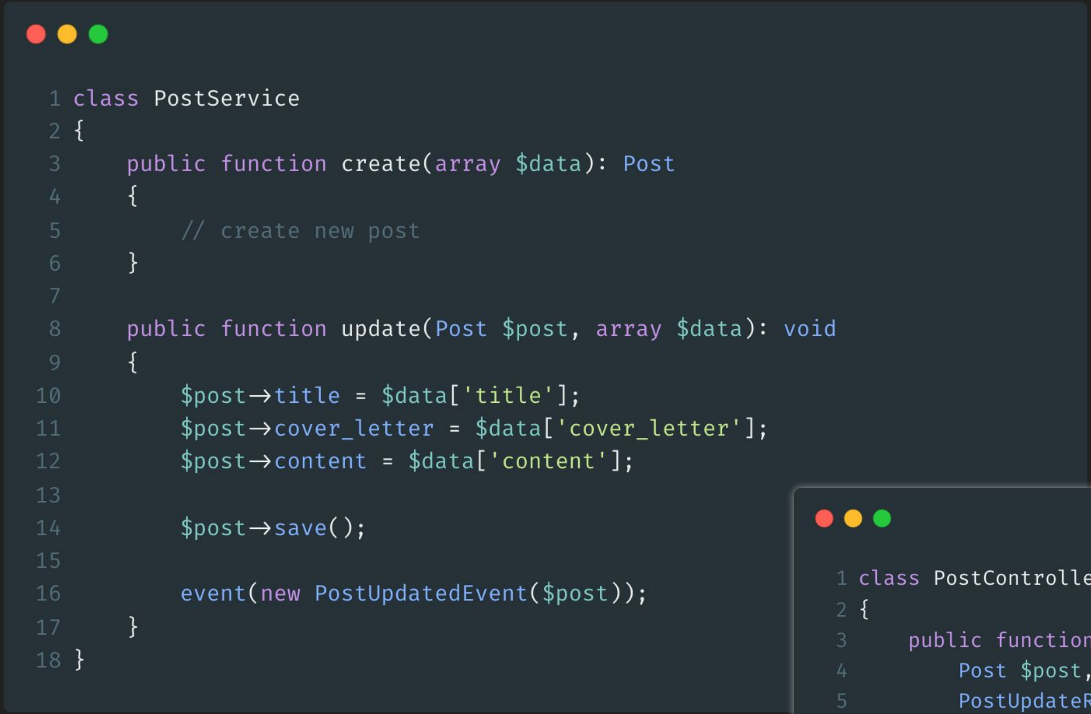
```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        // do aome busines...
        /** @var Post|null $post */
        $post = Post::query()
            ->where('id', $id)
            ->first();

        // do some business ...

        $post->title = $request->get('title');
        $post->cover_letter = $request->get('cover_letter');
        $post->content = $request->get('content');
        $post->categories()->sync($request->get('categories'));
        $post->save();
        // do some business ...
    }
}
```



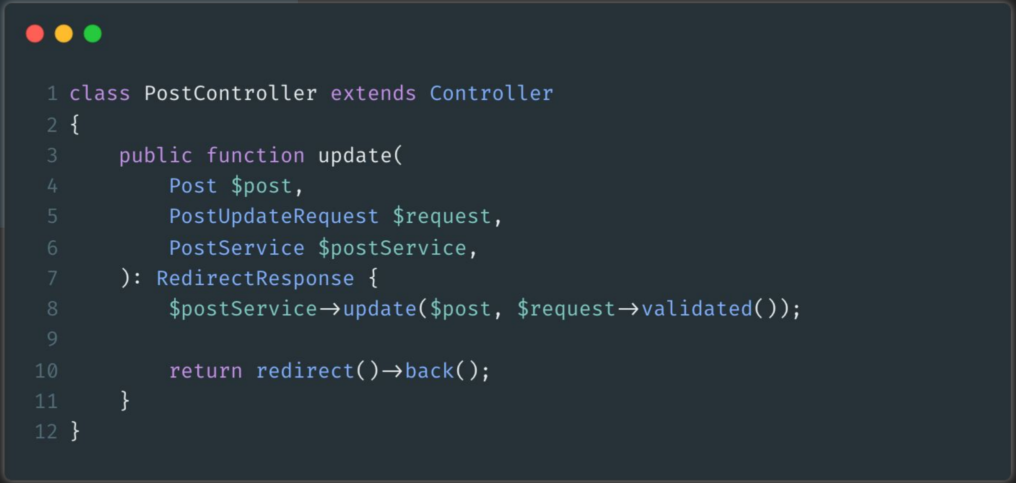
```php
class PostService
{
    public function create(array $data): Post
    {
        // create new post
    }

    public function update(Post $post, array $data): void
    {
        $post->title = $data['title'];
        $post->cover_letter = $data['cover_letter'];
        $post->content = $data['content'];

        $post->save();

        event(new PostUpdatedEvent($post));
    }
}
```



```php
class PostController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
        PostService $postService,
    ): RedirectResponse {
        $postService->update($post, $request->validated());

        return redirect()->back();
    }
}
```



```php
class PostService
{
    public function create(array $data): Post
    {
        // create new post
    }

    public function update(Post $post, array $data): void
    {
        $post->title = $data['title'];
        $post->cover_letter = $data['cover_letter'];
        $post->content = $data['content'];

        $post->save();

        event(new PostUpdatedEvent($post));
    }
}
```



```php
class PostController extends Controller
{
    public function update(
        Post $post,
        PostUpdateRequest $request,
        PostService $postService,
    ): RedirectResponse {
        $postService->update($post, $request->validated());

        return redirect()->back();
    }
}
```

Action Class vs Service Class

- Each action has its own class.

- CreatePostAction
- UpdatePostAction
- DeletePostAction

- All actions in one single service class.

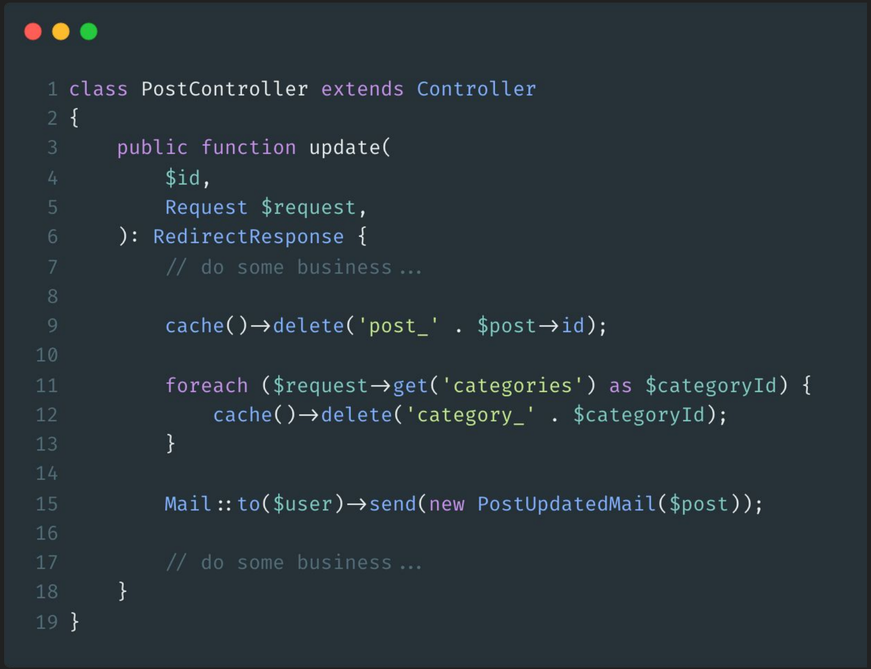
- PostService::create()
- PostService::update()
- PostService::delete()

Event/Listeners

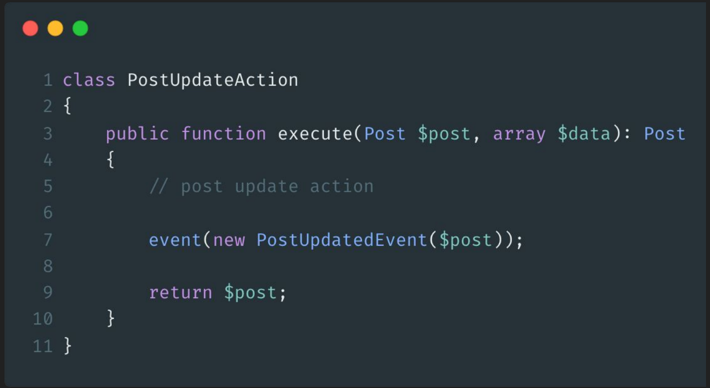
Dispatchable

One event multiple listeners

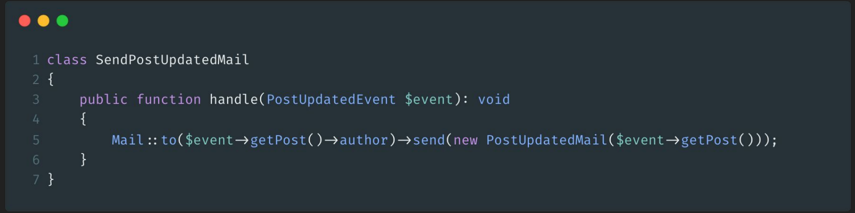
Queueable (Listeners)



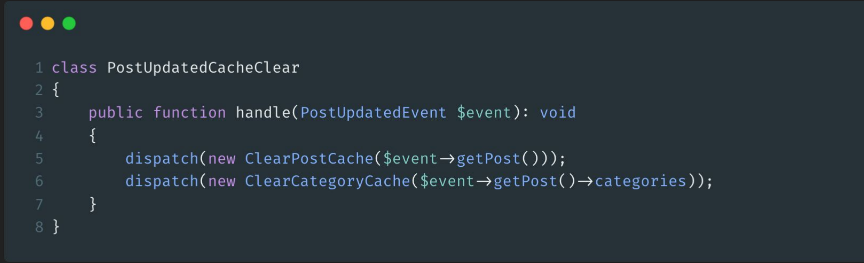
```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        // do some business...

        cache()→delete('post_' . $post→id);

        foreach ($request→get('categories') as $categoryId) {
            cache()→delete('category_' . $categoryId);
        }

        Mail::to($user)→send(new PostUpdatedMail($post));

        // do some business...
    }
}
```



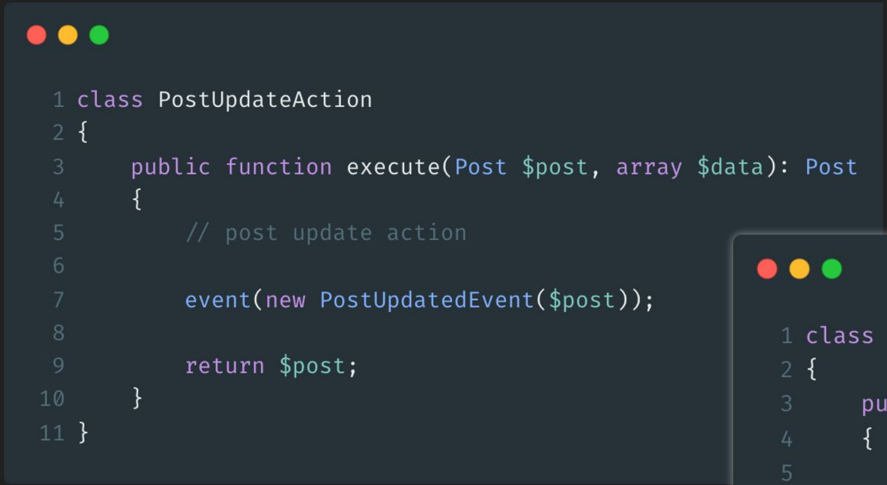
```php
class PostUpdateAction
{
    public function execute(Post $post, array $data): Post
    {
        // post update action

        event(new PostUpdatedEvent($post));

        return $post;
    }
}
```



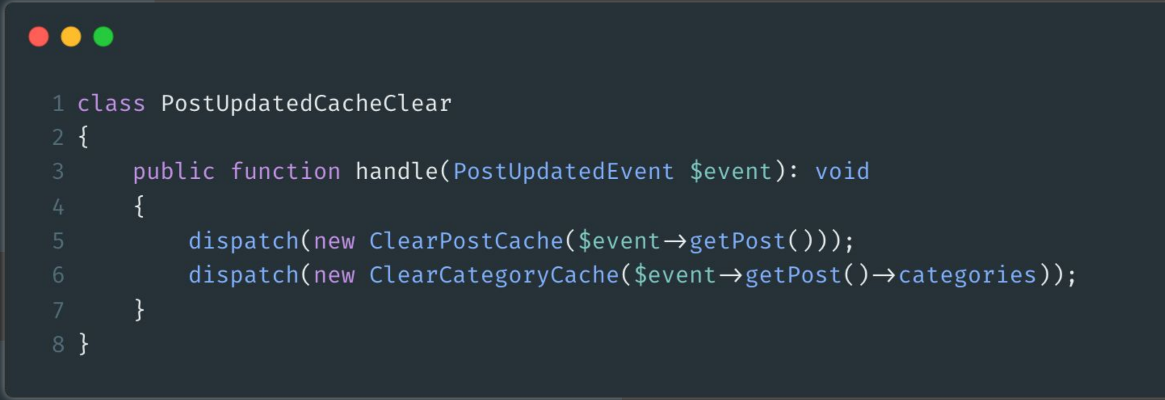
```php
class SendPostUpdatedMail
{
    public function handle(PostUpdatedEvent $event): void
    {
        Mail::to($event→getPost()→author)→send(new PostUpdatedMail($event→getPost()));
    }
}
```



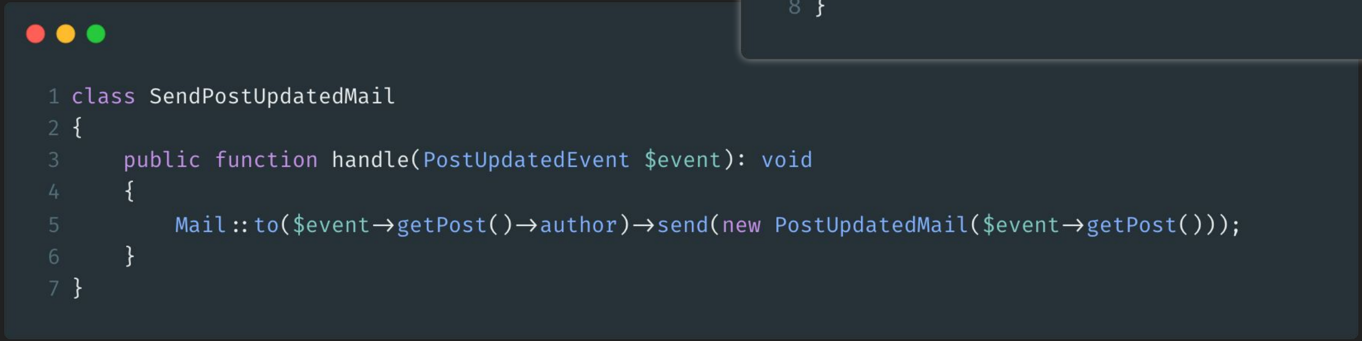
```php
class PostUpdatedCacheClear
{
    public function handle(PostUpdatedEvent $event): void
    {
        dispatch(new ClearPostCache($event→getPost()));
        dispatch(new ClearCategoryCache($event→getPost()→categories));
    }
}
```



```php
class PostUpdateAction
{
    public function execute(Post $post, array $data): Post
    {
        // post update action

        event(new PostUpdatedEvent($post));

        return $post;
    }
}
```



```php
class PostUpdatedCacheClear
{
    public function handle(PostUpdatedEvent $event): void
    {
        dispatch(new ClearPostCache($event→getPost()));
        dispatch(new ClearCategoryCache($event→getPost()→categories));
    }
}
```

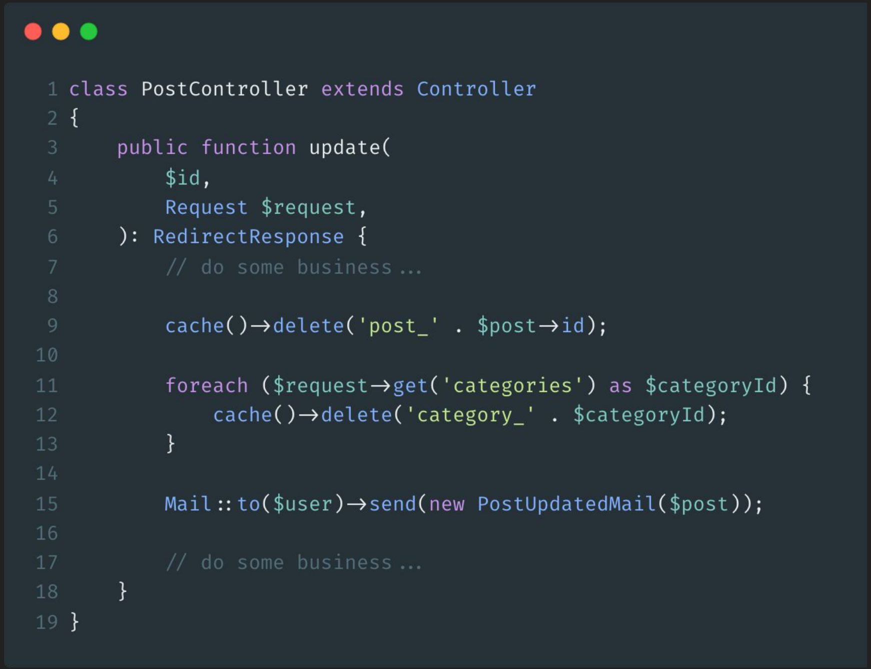


```php
class SendPostUpdatedMail
{
    public function handle(PostUpdatedEvent $event): void
    {
        Mail::to($event→getPost()→author)→send(new PostUpdatedMail($event→getPost()));
    }
}
```

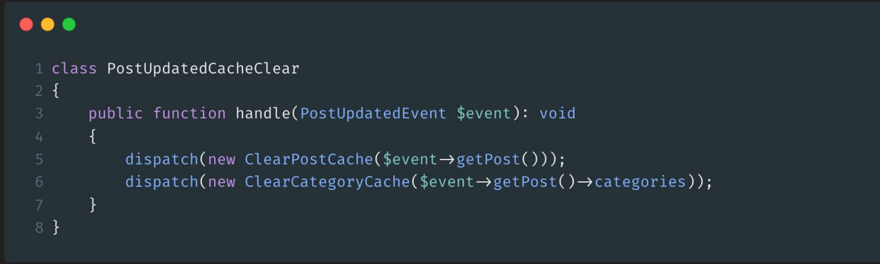
Jobs

Dispatchable

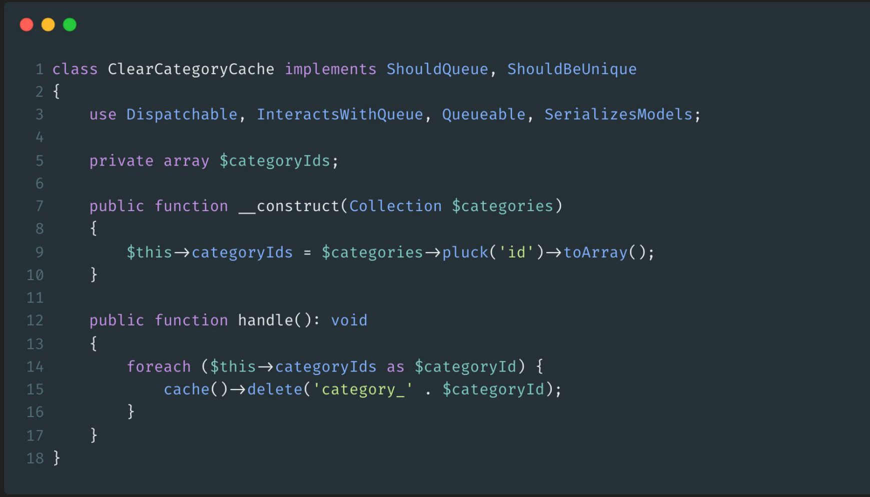
Queueable



```php
class PostController extends Controller
{
    public function update(
        $id,
        Request $request,
    ): RedirectResponse {
        // do some business...

        cache()->delete('post_' . $post->id);

        foreach ($request->get('categories') as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }

        Mail::to($user)->send(new PostUpdatedMail($post));

        // do some business...
    }
}
```



```php
class PostUpdatedCacheClear
{
    public function handle(PostUpdatedEvent $event): void
    {
        dispatch(new ClearPostCache($event->getPost()));
        dispatch(new ClearCategoryCache($event->getPost()->categories));
    }
}
```



```php
class ClearCategoryCache implements ShouldQueue, ShouldBeUnique
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    private array $categoryIds;

    public function __construct(Collection $categories)
    {
        $this->categoryIds = $categories->pluck('id')->toArray();
    }

    public function handle(): void
    {
        foreach ($this->categoryIds as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }
    }
}
```

TEKNASYON PHPkonf.

```php
class PostUpdatedCacheClear
{
    public function handle(PostUpdatedEvent $event): void
    {
        dispatch(new ClearPostCache($event->getPost()));
        dispatch(new ClearCategoryCache($event->getPost()->categories));
    }
}
```

```php
class ClearCategoryCache implements ShouldQueue, ShouldBeUnique
{
    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    private array $categoryIds;

    public function __construct(Collection $categories)
    {
        $this->categoryIds = $categories->pluck('id')->toArray();
    }

    public function handle(): void
    {
        foreach ($this->categoryIds as $categoryId) {
            cache()->delete('category_' . $categoryId);
        }
    }
}
```

The end?

What's Next?

- Observers
- Accessors
- Mutators
- Scopes
- Resources
- Global Helpers
- Value Objects
- Traits
- Abstract Classes
- Package Development

What about Repository Classes?

Summary

- Distribute responsibilities
- Use framework features
- Follow best practices
- Do not reinvent the wheel

Stay up-to-date

- Follow the documentation
- Follow what’s new content for versions
- Code-review (internal/external)
- Online contents (articles, blogs, tips and tricks videos)
- Community events / conferences

Questions❓

TEKNASYON PHPkonf.

ThanKs🙂

kommunity.com/m

x.com/mcucen

mucahit@mcucen.com