

Beyond Controllers in Laravel Projects

Finding the right place for logic

Mücahit Cücen



```
<?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();
    }
}
```

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

- 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

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Mail\PostUpdatedMail;
6 use App\Models\Post;
7 use App\Models\User;
8 use Illuminate\Http\RedirectResponse;
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Mail;
11
12 class PostController extends Controller
13 {
14     public function update(
15         $id,
16         Request $request,
17     ): RedirectResponse {
18         /** @var User|null $user */
19         $user = auth()->user();
20
21         if ($user === null) {
22             return redirect()->route('login');
23         }
24
25         /** @var Post|null $post */
26         $post = Post::query()
27             ->where('id', $id)
28             ->first();
29
30         if ($post === null) {
31             return redirect()->route('dashboard');
32         }
33
34         if (!$user->isAdmin() || $user->id !== $post->author_id) {
35             return redirect()->route('dashboard');
36         }
37
38         $this->validate($request, [
39             'title' => 'required',
40             'cover_letter' => 'required',
41             'content' => 'required',
42             'categories' => 'array',
43         ]);
44
45         $post->title = $request->get('title');
46         $post->cover_letter = $request->get('cover_letter');
47         $post->content = $request->get('content');
48         $post->categories()=>sync($request->get('categories'));
49         $post->save();
50
51         cache()->delete('post_' . $post->id);
52
53         foreach ($request->get('categories') as $categoryId) {
54             cache()->delete('category_' . $categoryId);
55         }
56
57         Mail::to($user)->send(new PostUpdatedMail($post));
58
59         return redirect()->back();
60     }
61 }

```

How to clarify?

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

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Mail\PostUpdatedMail;
6 use App\Models\Post;
7 use App\Models\User;
8 use Illuminate\Http\RedirectResponse;
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Mail;
11
12 class PostController extends Controller
13 {
14     public function update(
15         $id,
16         Request $request,
17     ): RedirectResponse {
18         /** @var User|null $user */
19         $user = auth()->user();
20
21         if ($user === null) {
22             return redirect()>route('login');
23         }
24
25         /** @var Post|null $post */
26         $post = Post::query()
27             ->where('id', $id)
28             ->first();
29
30         if ($post === null) {
31             return redirect()>route('dashboard');
32         }
33
34         if (!$user->isAdmin() || $user->id !== $post->author_id) {
35             return redirect()>route('dashboard');
36         }
37
38         $this->validate($request, [
39             'title' => 'required',
40             'cover_letter' => 'required',
41             'content' => 'required',
42             'categories' => 'array',
43         ]);
44
45         $post->title = $request->get('title');
46         $post->cover_letter = $request->get('cover_letter');
47         $post->content = $request->get('content');
48         $post->categories()->sync($request->get('categories'));
49         $post->save();
50
51         cache()->delete('post_' . $post->id);
52
53         foreach ($request->get('categories') as $categoryId) {
54             cache()->delete('category_' . $categoryId);
55         }
56
57         Mail::to($user)->send(new PostUpdatedMail($post));
58
59         return redirect()->back();
60     }
61 }

```

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

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Actions\PostUpdateAction;
6 use App\Http\Requests\PostUpdateRequest;
7 use App\Models\Post;
8 use Illuminate\Http\RedirectResponse;
9
10 class PostController extends Controller
11 {
12     public function update(
13         Post $post,
14         PostUpdateRequest $request,
15         PostUpdateAction $postUpdateAction,
16     ): RedirectResponse {
17         $postUpdateAction->execute($post, $request->validated());
18
19         return redirect()->back();
20     }
21 }

```

Let Laravel make it done!


```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Mail\PostUpdatedMail;
6 use App\Models\Post;
7 use App\Models\User;
8 use Illuminate\Http\RedirectResponse;
9 use Illuminate\Http\Request;
10 use Illuminate\Support\Facades\Mail;
11
12 class PostController extends Controller
13 {
14     public function update(
15         $id,
16         Request $request,
17     ): RedirectResponse {
18         /** @var User|null $user */
19         $user = auth()->user();
20
21         if ($user === null) {
22             return redirect()->route('login');
23         }
24
25         /** @var Post|null $post */
26         $post = Post::query()
27             ->where('id', $id)
28             ->first();
29
30         if ($post === null) {
31             return redirect()->route('dashboard');
32         }
33
34         if (!$user->isAdmin() && $user->id !== $post->author_id) {
35             return redirect()->route('dashboard');
36         }
37
38         $this->validate($request, [
39             'title' => 'required',
40             'cover_letter' => 'required',
41             'content' => 'required',
42             'categories' => 'array',
43         ]);
44
45         $post->title = $request->get('title');
46         $post->cover_letter = $request->get('cover_letter');
47         $post->content = $request->get('content');
48         $post->categories()->sync($request->get('categories'));
49         $post->save();
50
51         cache()->delete('post_' . $post->id);
52
53         foreach ($request->get('categories') as $categoryId) {
54             cache()->delete('category_' . $categoryId);
55         }
56
57         Mail::to($user)->send(new PostUpdatedMail($post));
58
59         return redirect()->back();
60     }
61 }
```



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\Actions\PostUpdateAction;
6 use App\Http\Requests\PostUpdateRequest;
7 use App\Models\Post;
8 use Illuminate\Http\RedirectResponse;
9
10 class PostController extends Controller
11 {
12     public function update(
13         Post $post,
14         PostUpdateRequest $request,
15         PostUpdateAction $postUpdateAction,
16     ): RedirectResponse {
17         $postUpdateAction->execute($post, $request->validated());
18
19         return redirect()->back();
20     }
21 }
```

Route Model Binding

Auto inject

Generates 404

```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         // do some business...
8
9         /** @var Post|null $post */
10        $post = Post::query()
11            →where('id', $id)
12            →first();
13
14        if ($post === null) {
15            return redirect()→route('dashboard');
16        }
17
18        // do some business...
19
20        return redirect()→back();
21    }
22 }

```



```

1 Route::post('/posts/{post}/update', [PostController::class, 'update'])
2     →middleware('auth')
3     →name('post.edit');

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5     ): RedirectResponse {
6         // $post variable contains a Post object.
7         // do some business...
8     }
9 }

```



```
1 Route::post('/posts/{post}/update', [PostController::class, 'update'])
2     →middleware('auth')
3     →name('post.edit');
```



```
1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5     ): RedirectResponse {
6         // $post variable contains a Post object.
7         // do some business ...
8     }
9 }
```

Form Requests

Authorization

Validation

```
1 <?php
2
3 namespace App\Http\Requests;
4
5 use Illuminate\Foundation\Http\FormRequest;
6
7 class PostUpdateRequest extends FormRequest
8 {
9     /**
10      * Determine if the user is authorized to make this request.
11      */
12     public function authorize(): bool
13     {
14         return false;
15     }
16
17     /**
18      * Get the validation rules that apply to the request.
19      *
20      * @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array|string>
21      */
22     public function rules(): array
23     {
24         return [
25             //
26         ];
27     }
28 }
```

```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         /** @var User|null $user */
8         $user = auth()→user();
9
10        if ($user === null) {
11            return redirect()→route('login');
12        }
13
14        // do some business...
15
16        if (!$user→isAdmin() && $user→id === $post→author_id) {
17            return redirect()→route('dashboard');
18        }
19
20        $this→validate($request, [
21            'title' ⇒ 'required',
22            'cover_letter' ⇒ 'required',
23            'content' ⇒ 'required',
24            'categories' ⇒ 'array',
25        ]);
26
27        // do some business...
28    }
29 }

```



```

1 Route::post('/posts/{post}/update', [PostController::class, 'update'])
2     →middleware('auth')
3     →name('post.edit');

```

```

1 class PostUpdateRequest extends FormRequest
2 {
3     public function authorize(): bool
4     {
5         return $this→user()→can('update', $this→route('post'));
6     }
7
8     public function rules(): array
9     {
10        return [
11            'title' ⇒ 'required',
12            'cover_letter' ⇒ 'required',
13            'content' ⇒ 'required',
14            'categories' ⇒ 'array',
15        ];
16    }
17 }

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5         PostUpdateRequest $request,
6     ): RedirectResponse {
7         // do some business...
8     }
9 }

```

```

1 class PostUpdateRequest extends FormRequest
2 {
3     public function authorize(): bool
4     {
5         return $this->user()->can('update', $this->route('post'));
6     }
7
8     public function rules(): array
9     {
10         return [
11             'title' => 'required',
12             'cover_letter' => 'required',
13             'content' => 'required',
14             'categories' => 'array',
15         ];
16     }
17 }

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5         PostUpdateRequest $request,
6     ): RedirectResponse {
7         // do some business...
8     }
9 }

```


Policies

Authorize user actions

```

1 <?php
2
3 namespace App\Policies;
4
5 use App\Models\Post;
6 use App\Models\User;
7 use Illuminate\Auth\Access\Response;
8
9 class PostPolicy
10 {
11     /**
12      * Determine whether the user can update the model.
13      */
14     public function update(User $user, Post $post): bool
15     {
16         //
17     }
18 }

```

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

```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         /** @var User|null $user */
8         $user = auth()→user();
9
10        if (!$user→isAdmin() && $user→id !== $post→author_id) {
11            return redirect()→route('dashboard');
12        }
13
14        // do some business...
15    }
16 }

```



```

1 class PostPolicy
2 {
3     /**
4      * Determine whether the user can update the model.
5      */
6     public function update(User $user, Post $post): bool
7     {
8         return $user→isAdmin() || $user→id === $post→author_id;
9     }
10 }

```

```

1 class PostUpdateRequest extends FormRequest
2 {
3     public function authorize(): bool
4     {
5         return $this→user()→can('update', $this→route('post'));
6     }
7 }

```



```

1 class PostPolicy
2 {
3     /**
4      * Determine whether the user can update the model.
5      */
6     public function update(User $user, Post $post): bool
7     {
8         return $user->isAdmin() || $user->id === $post->author_id;
9     }
10 }

```



```

1 class PostUpdateRequest extends FormRequest
2 {
3     public function authorize(): bool
4     {
5         return $this->user()->can('update', $this->route('post'));
6     }
7 }

```

Action Class

Simple PHP class

Contains an execute method
for only ONE action

```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         // do some business ...
8         /** @var Post|null $post */
9         $post = Post::query()
10             →where('id', $id)
11             →first();
12
13         // do some business ...
14
15         $post→title = $request→get('title');
16         $post→cover_letter = $request→get('cover_letter');
17         $post→content = $request→get('content');
18         $post→categories()→sync($request→get('categories'));
19         $post→save();
20         // do some business ...
21     }
22 }

```



```

1 class PostUpdateAction
2 {
3     public function execute(Post $post, array $data): Post
4     {
5         $post→title = $data['title'];
6         $post→cover_letter = $data['cover_letter'];
7         $post→content = $data['content'];
8
9         $post→categories()→sync($data['categories']);
10
11         $post→save();
12
13         event(new PostUpdatedEvent($post));
14
15         return $post;
16     }
17 }

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5         PostUpdateRequest $request,
6         PostUpdateAction $postUpdateAction,
7     ): RedirectResponse {
8         $postUpdateAction→execute($post, $request→validated());
9
10         return redirect()→back();
11     }
12 }

```

```

1 class PostUpdateAction
2 {
3     public function execute(Post $post, array $data): Post
4     {
5         $post->title = $data['title'];
6         $post->cover_letter = $data['cover_letter'];
7         $post->content = $data['content'];
8
9         $post->categories()->sync($data['categories']);
10
11        $post->save();
12
13        event(new PostUpdatedEvent($post));
14
15        return $post;
16    }
17 }

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5         PostUpdateRequest $request,
6         PostUpdateAction $postUpdateAction,
7     ): RedirectResponse {
8         $postUpdateAction->execute($post, $request->validated());
9
10        return redirect()->back();
11    }
12 }

```

Service Class

Simple PHP class

Contains one or more
methods for related context


```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         // do some business ...
8         /** @var Post|null $post */
9         $post = Post::query()
10             →where('id', $id)
11             →first();
12
13         // do some business ...
14
15         $post→title = $request→get('title');
16         $post→cover_letter = $request→get('cover_letter');
17         $post→content = $request→get('content');
18         $post→categories()→sync($request→get('categories'));
19         $post→save();
20         // do some business ...
21     }
22 }

```



```

1 class PostService
2 {
3     public function create(array $data): Post
4     {
5         // create new post
6     }
7
8     public function update(Post $post, array $data): void
9     {
10         $post→title = $data['title'];
11         $post→cover_letter = $data['cover_letter'];
12         $post→content = $data['content'];
13
14         $post→save();
15
16         event(new PostUpdatedEvent($post));
17     }
18 }

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5         PostUpdateRequest $request,
6         PostService $postService,
7     ): RedirectResponse {
8         $postService→update($post, $request→validated());
9
10         return redirect()→back();
11     }
12 }

```

```

1 class PostService
2 {
3     public function create(array $data): Post
4     {
5         // create new post
6     }
7
8     public function update(Post $post, array $data): void
9     {
10         $post->title = $data['title'];
11         $post->cover_letter = $data['cover_letter'];
12         $post->content = $data['content'];
13
14         $post->save();
15
16         event(new PostUpdatedEvent($post));
17     }
18 }

```

```

1 class PostController extends Controller
2 {
3     public function update(
4         Post $post,
5         PostUpdateRequest $request,
6         PostService $postService,
7     ): RedirectResponse {
8         $postService->update($post, $request->validated());
9
10        return redirect()->back();
11    }
12 }

```

Action Class vs Service Class

- Each action has its own class.
 - All actions in one single service class.
-
- CreatePostAction
 - UpdatePostAction
 - DeletePostAction
-
- PostService::create()
 - PostService::update()
 - PostService::delete()

Event/Listeners

Dispatchable

One event multiple listeners

Queueable (Listeners)

```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         // do some business...
8
9         cache()→delete('post_' . $post→id);
10
11         foreach ($request→get('categories') as $categoryId) {
12             cache()→delete('category_' . $categoryId);
13         }
14
15         Mail::to($user)→send(new PostUpdatedMail($post));
16
17         // do some business...
18     }
19 }

```



```

1 class PostUpdateAction
2 {
3     public function execute(Post $post, array $data): Post
4     {
5         // post update action
6
7         event(new PostUpdatedEvent($post));
8
9         return $post;
10    }
11 }

```

```

1 class SendPostUpdatedMail
2 {
3     public function handle(PostUpdatedEvent $event): void
4     {
5         Mail::to($event→getPost()→author)→send(new PostUpdatedMail($event→getPost()));
6     }
7 }

```

```

1 class PostUpdatedCacheClear
2 {
3     public function handle(PostUpdatedEvent $event): void
4     {
5         dispatch(new ClearPostCache($event→getPost()));
6         dispatch(new ClearCategoryCache($event→getPost()→categories));
7     }
8 }

```

```

1 class PostUpdateAction
2 {
3     public function execute(Post $post, array $data): Post
4     {
5         // post update action
6
7         event(new PostUpdatedEvent($post));
8
9         return $post;
10    }
11 }

```

```

1 class PostUpdatedCacheClear
2 {
3     public function handle(PostUpdatedEvent $event): void
4     {
5         dispatch(new ClearPostCache($event→getPost()));
6         dispatch(new ClearCategoryCache($event→getPost()→categories));
7     }
8 }

```

```

1 class SendPostUpdatedMail
2 {
3     public function handle(PostUpdatedEvent $event): void
4     {
5         Mail::to($event→getPost()→author)→send(new PostUpdatedMail($event→getPost()));
6     }
7 }

```

Jobs

Dispatchable

Queueable

```

1 class PostController extends Controller
2 {
3     public function update(
4         $id,
5         Request $request,
6     ): RedirectResponse {
7         // do some business...
8
9         cache()→delete('post_' . $post→id);
10
11         foreach ($request→get('categories') as $categoryId) {
12             cache()→delete('category_' . $categoryId);
13         }
14
15         Mail::to($user)→send(new PostUpdatedMail($post));
16
17         // do some business...
18     }
19 }

```



```

1 class PostUpdatedCacheClear
2 {
3     public function handle(PostUpdatedEvent $event): void
4     {
5         dispatch(new ClearPostCache($event→getPost()));
6         dispatch(new ClearCategoryCache($event→getPost()→categories));
7     }
8 }

```

```

1 class ClearCategoryCache implements ShouldQueue, ShouldBeUnique
2 {
3     use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
4
5     private array $categoryIds;
6
7     public function __construct(Collection $categories)
8     {
9         $this→categoryIds = $categories→pluck('id')→toArray();
10    }
11
12    public function handle(): void
13    {
14        foreach ($this→categoryIds as $categoryId) {
15            cache()→delete('category_' . $categoryId);
16        }
17    }
18 }

```




```

1 class PostUpdatedCacheClear
2 {
3     public function handle(PostUpdatedEvent $event): void
4     {
5         dispatch(new ClearPostCache($event→getPost()));
6         dispatch(new ClearCategoryCache($event→getPost()→categories));
7     }
8 }

```



```

1 class ClearCategoryCache implements ShouldQueue, ShouldBeUnique
2 {
3     use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;
4
5     private array $categoryIds;
6
7     public function __construct(Collection $categories)
8     {
9         $this→categoryIds = $categories→pluck('id')→toArray();
10    }
11
12    public function handle(): void
13    {
14        foreach ($this→categoryIds as $categoryId) {
15            cache()→delete('category_' . $categoryId);
16        }
17    }
18 }

```

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 ?

Than**Ks**😊

kommunity.com/m

x.com/mcucen

mucahit@mcucen.com