WORDCAMP
Sofia 2024

Launchpad: a bunch of accessible good practices

COQUARD Cyrille

Who is using?

- Composer

Who is using?

- Composer
- Namespaces

Who is using?

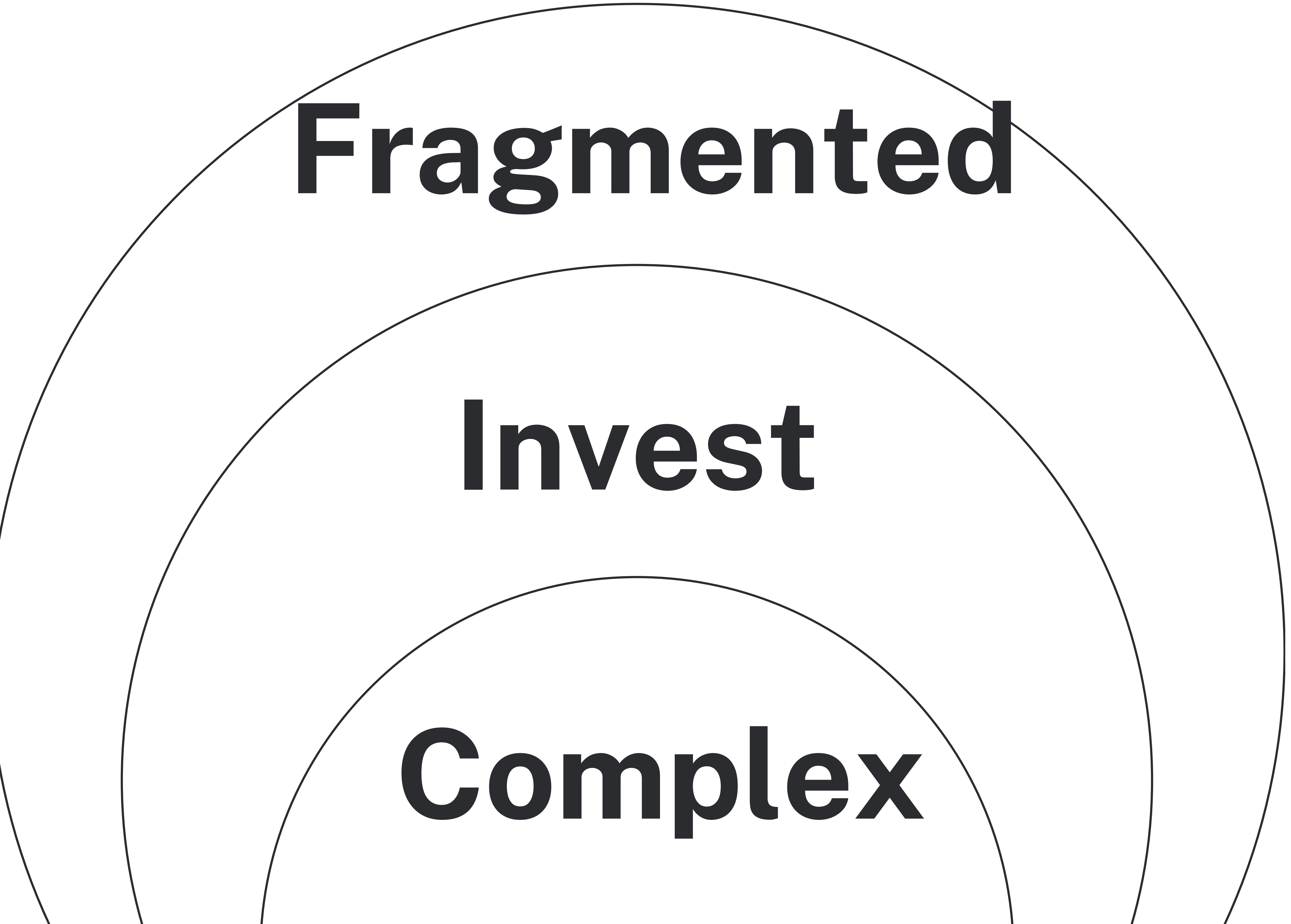
- Composer
- Namespaces
- Dependency injection

The garden is always greener

- Good practices are **the base**
- **Not the case** in WordPress

You are not bad, your tools are broken

- WordPress is end user centered
- Plugin devs are left aside



Layers of problems

Fragmented

No centralized documentation

Invest

Initial investment

Complex

Knowing all is a requirement

Good practices is reserved to an elite

- Each problem **excluded developers**
- Elite is created **by excluding**

WordPress is inclusive

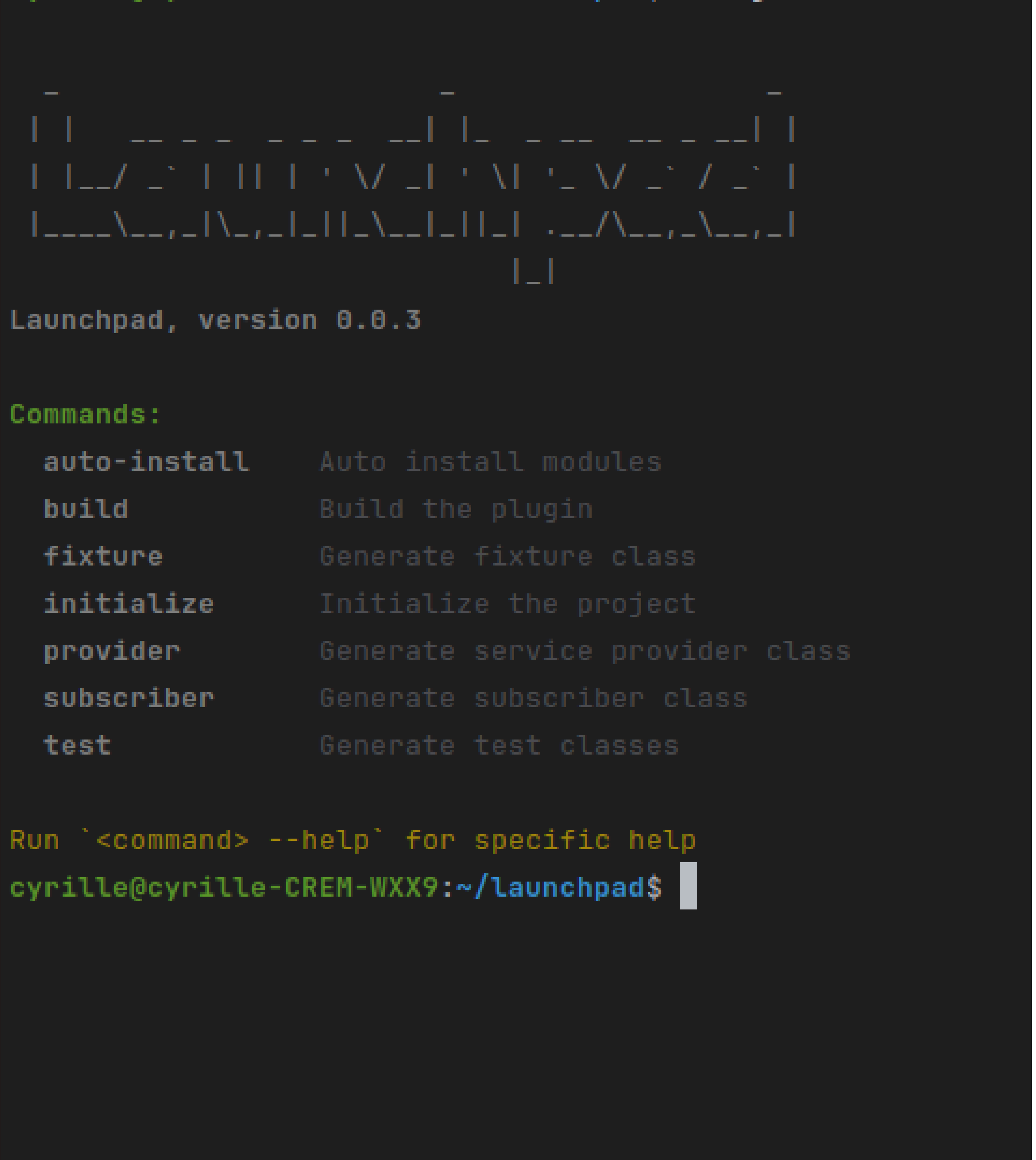
- **Excluding** is an issue
- **Inclusivity** is a main value

LAUNCHPAD

Lower the requirements to make good practices accessible.

Methods :

- **Abstract maximum of notions**
- **Offer a base**
- **Document notions**

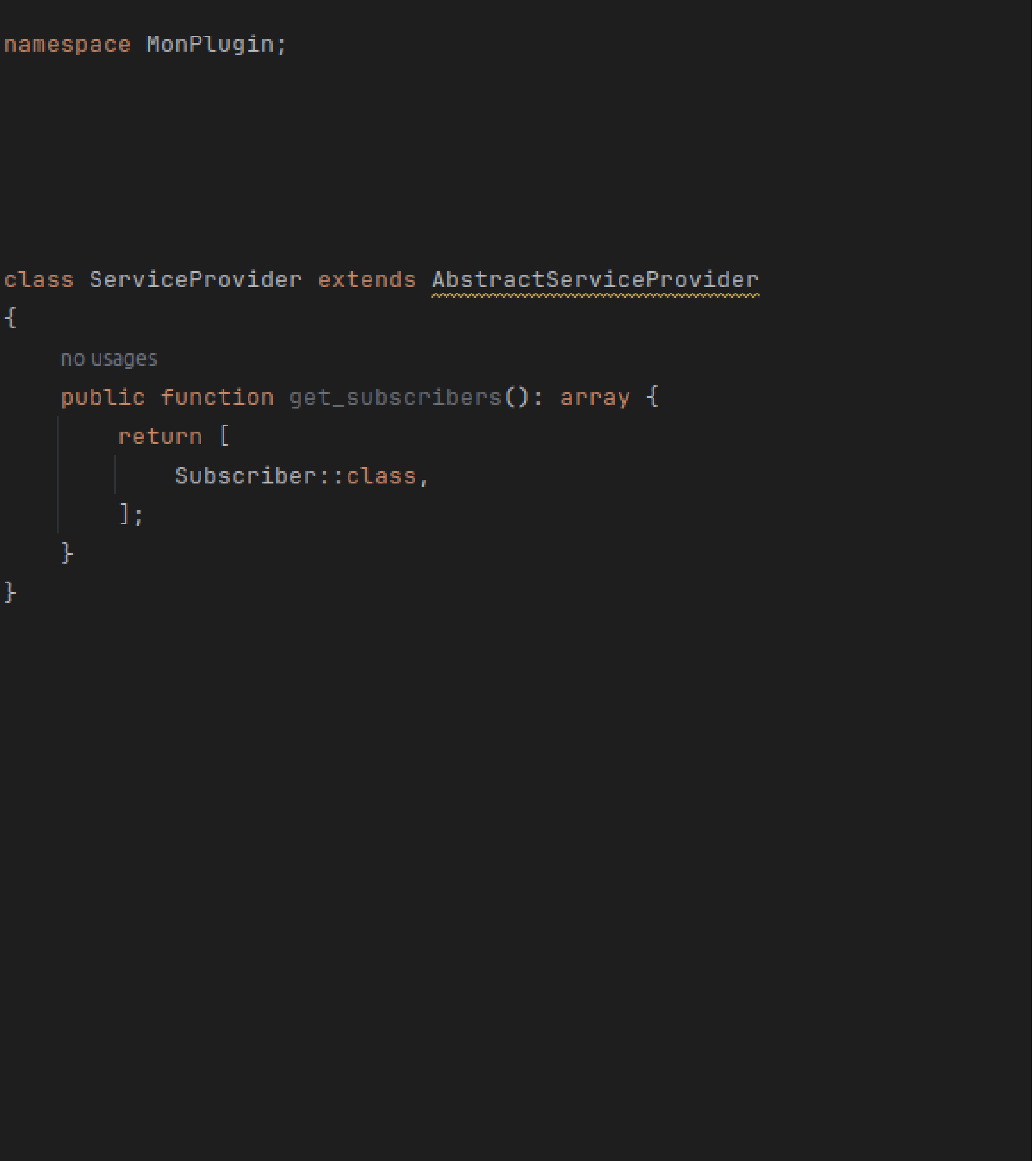


```
 _                       _                 _ 
| |   __ _ _  _ _ _  __| |_  _ __  __ _ __| |
| |__/ _` | || | ' \/ _| ' \| '_ \/ _` / _` |
|____\__,_|\_,_|_||_\__|_||_| .__/\__,_\__,_|
                            |_|              
Launchpad, version 0.0.3

Commands:
  auto-install    Auto install modules
  build           Build the plugin
  fixture         Generate fixture class
  initialize      Initialize the project
  provider        Generate service provider class
  subscriber      Generate subscriber class
  test            Generate test classes

Run `<command> --help` for specific help
cyrille@cyrille-CREM-WXX9:~/launchpad$
```

INVEST

Provide a base:

- Up to date
- 2 commands to start

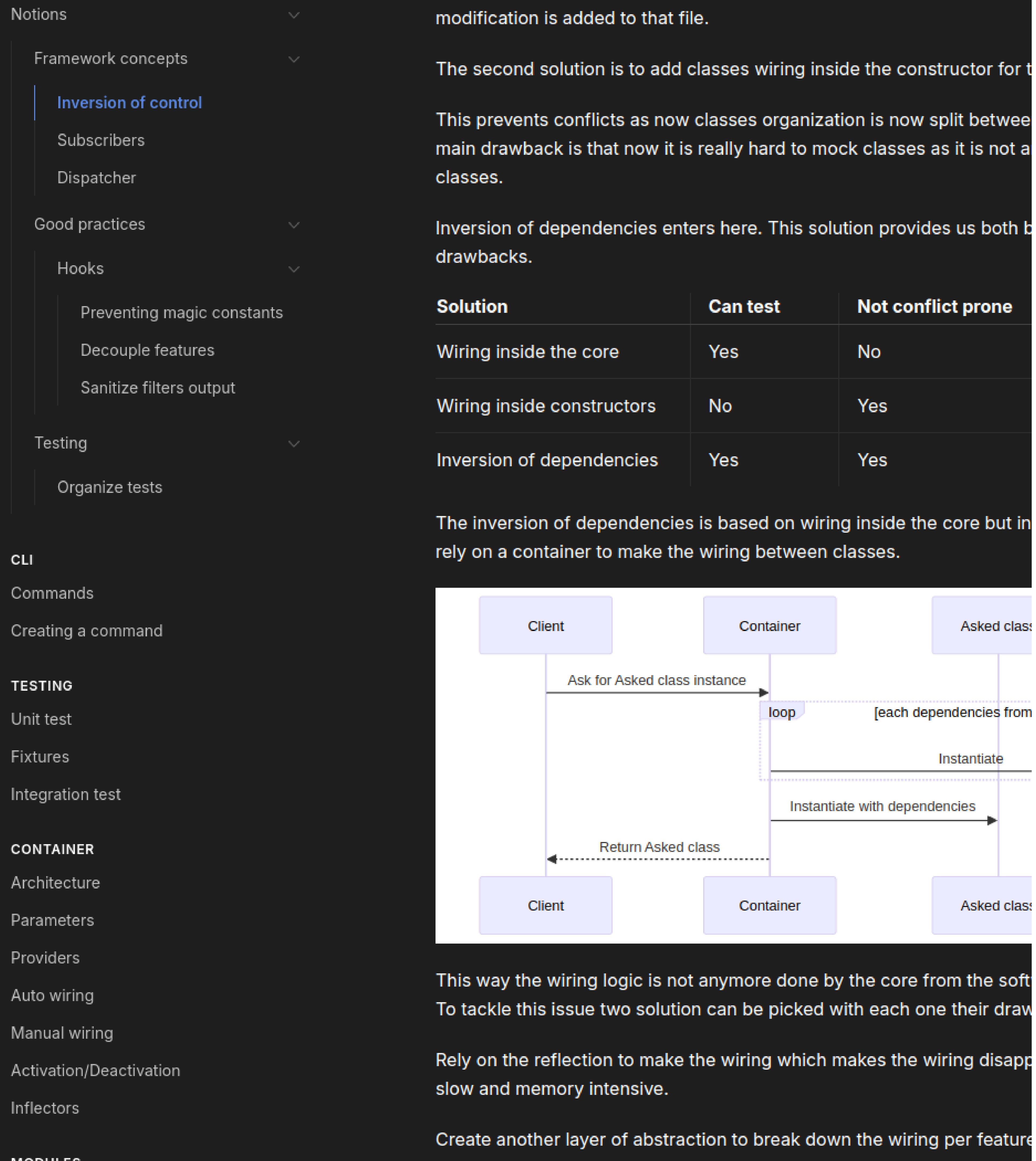


```
namespace MonPlugin;



class ServiceProvider extends AbstractServiceProvider
{
    no usages
    public function get_subscribers(): array {
        return [
            Subscriber::class,
        ];
    }
}
```

COMPLEXITY

Minimize entrance barrier:

- Provider
- Subscriber



Notions

Framework concepts

Inversion of control

Subscribers

Dispatcher

Good practices

Hooks

Preventing magic constants

Decouple features

Sanitize filters output

Testing

Organize tests

CLI

Commands

Creating a command

TESTING

Unit test

Fixtures

Integration test

CONTAINER

Architecture

Parameters

Providers

Auto wiring

Manual wiring

Activation/Deactivation

Inflectors

modification is added to that file.

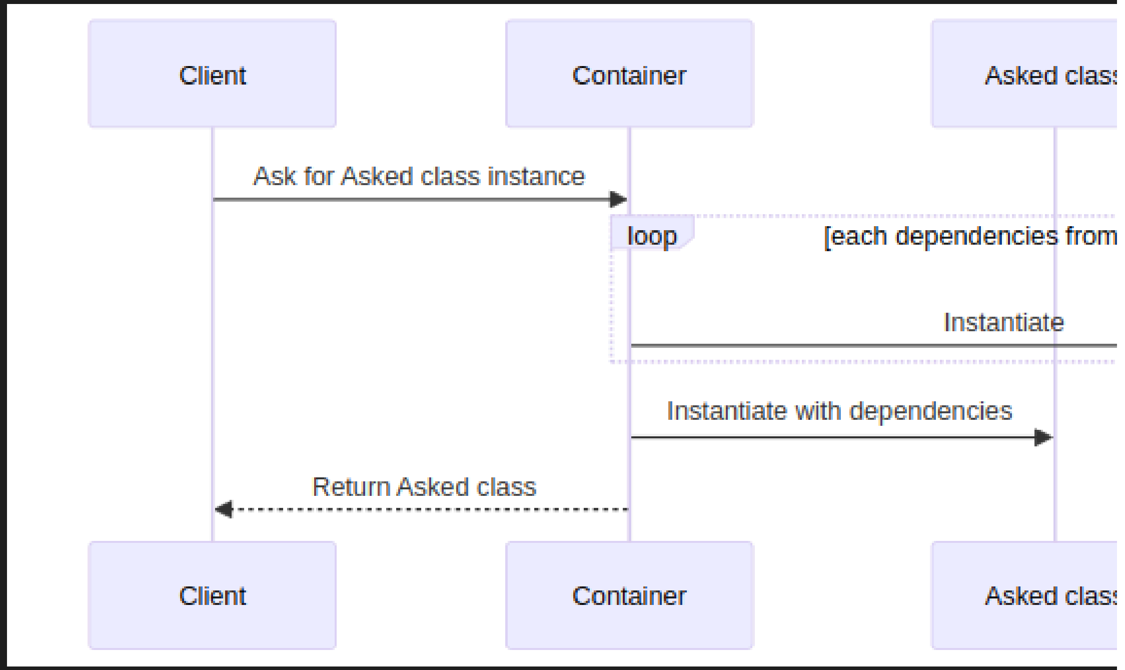
The second solution is to add classes wiring inside the constructor for t

This prevents conflicts as now classes organization is now split betwee
main drawback is that now it is really hard to mock classes as it is not a
classes.

Inversion of dependencies enters here. This solution provides us both b
drawbacks.

Solution	Can test	Not conflict prone
Wiring inside the core	Yes	No
Wiring inside constructors	No	Yes
Inversion of dependencies	Yes	Yes

The inversion of dependencies is based on wiring inside the core but in
rely on a container to make the wiring between classes.



This way the wiring logic is not anymore done by the core from the soft
To tackle this issue two solution can be picked with each one their draw

Rely on the reflection to make the wiring which makes the wiring disapp
slow and memory intensive.

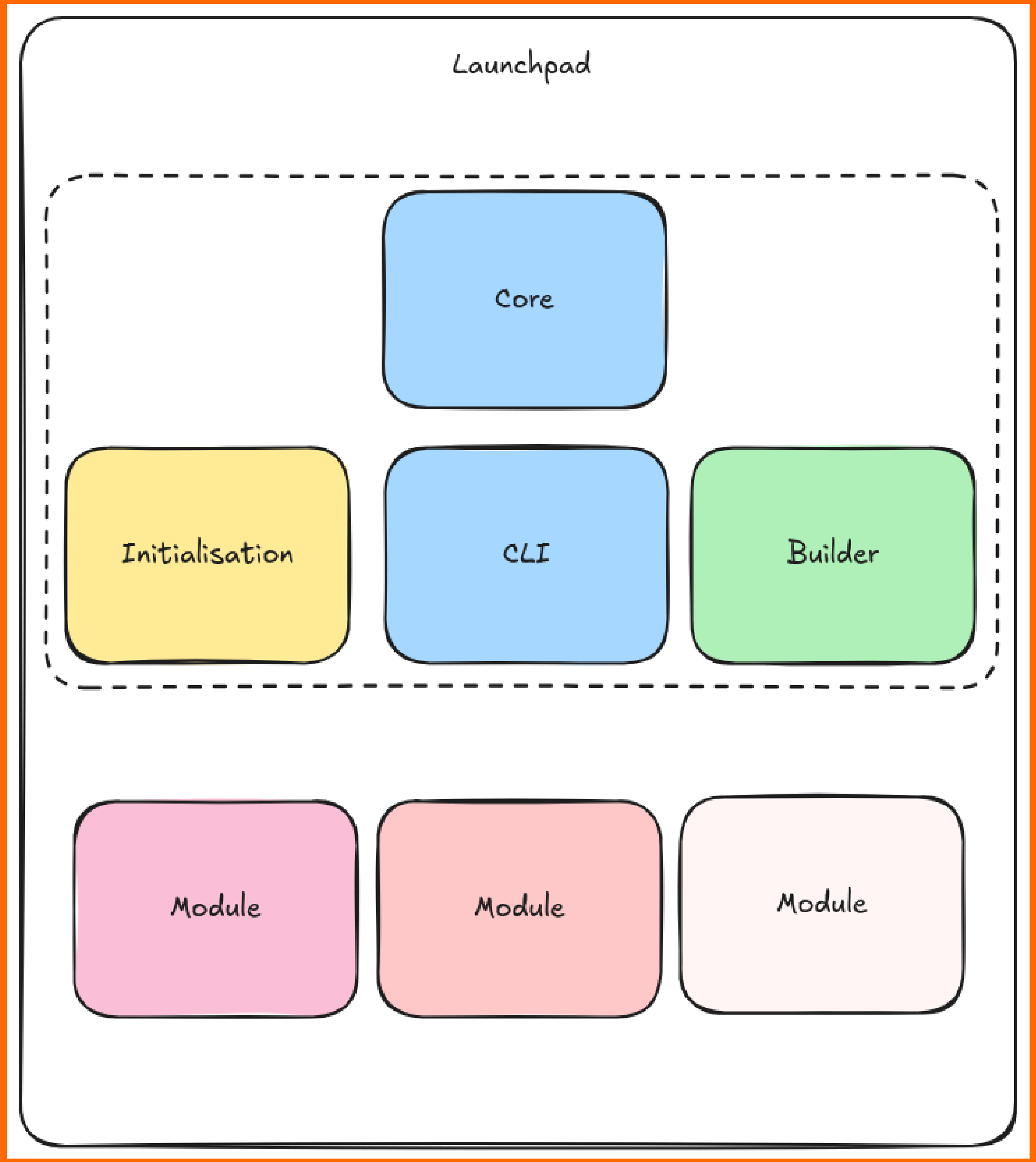
Create another layer of abstraction to break down the wiring per feature

FRAGMENTED

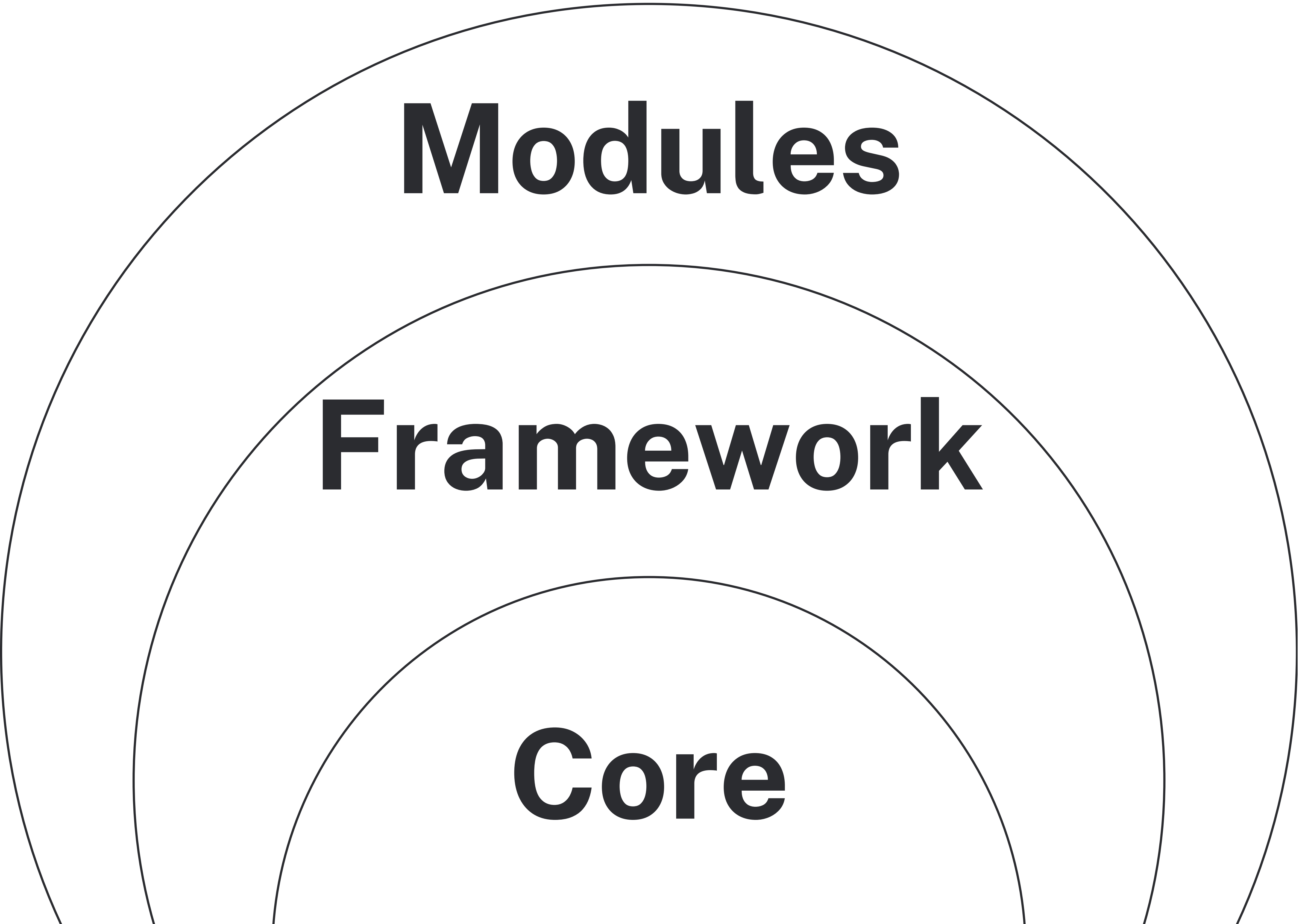
Make progress linear:

- Arrange by level
- Play with curiosity

A variety of developers



To similar problems,
Similar solutions



The onion

Core

The bare minimum

Framework

Simplify:
- Base
- Automate

Allow development environment:
- Code to ease development
- Build command for release

Modules

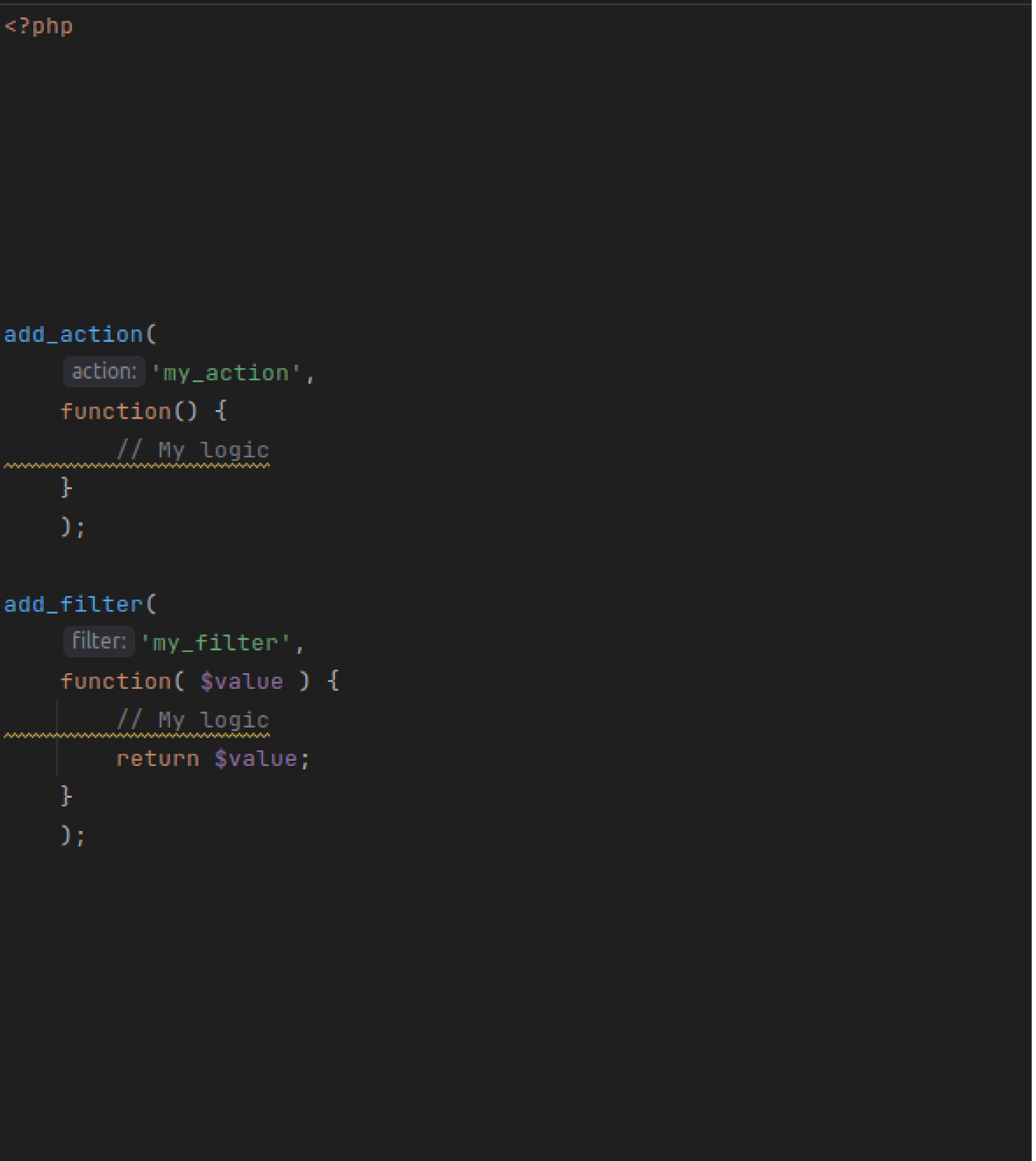
A constellation of modules:
- Liberty of choice
- Level adaptation
- Need adaptation

Simplicity, not *simplistic*
that is true **complexity**

The 2 approaches of DX

Developer e**X**perience:

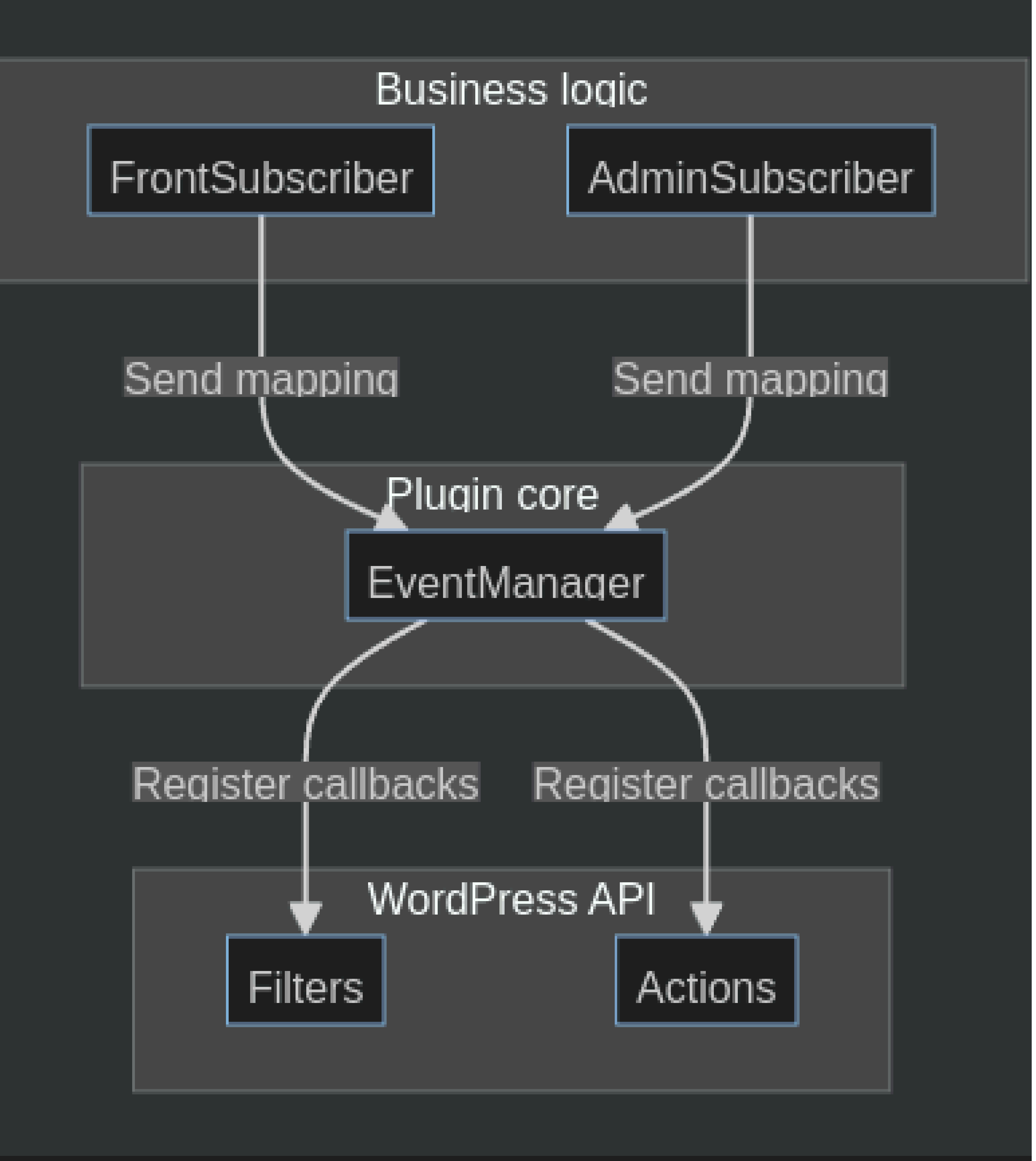
- It **doesn't** have to be complex
- Learn to **delegate**



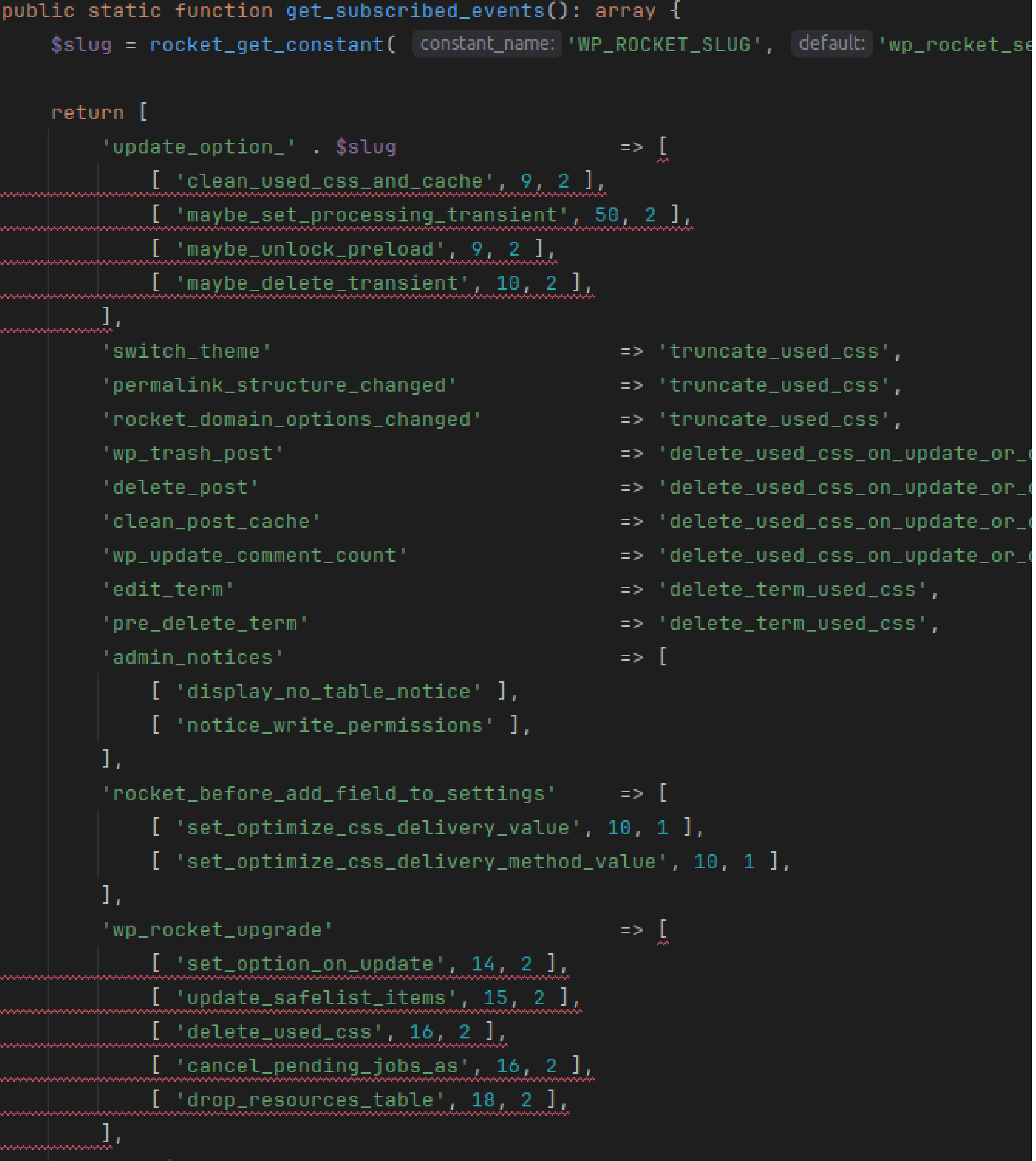
```
<?php

add_action(
    action: 'my_action',
    function() {
        // My logic
    }
    );

add_filter(
    filter: 'my_filter',
    function( $value ) {
        // My logic
        return $value;
    }
    );
```

Where to put them ?

- Core: hard to find
- Callback: hard to test



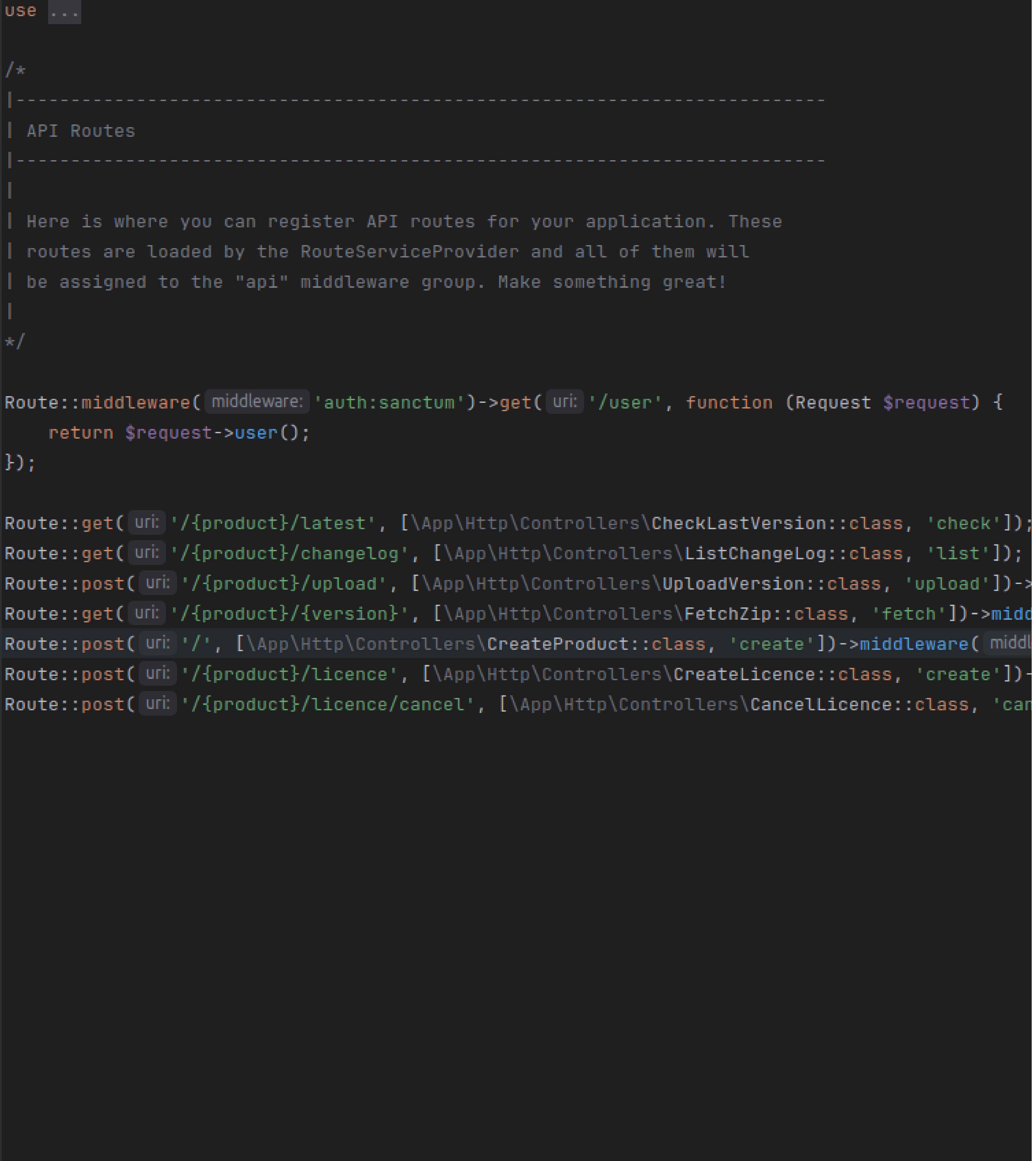
The subscriber pattern



```
public static function get_subscribed_events(): array {
    $slug = rocket_get_constant( constant_name: 'WP_ROCKET_SLUG', default: 'wp_rocket_se

    return [
        'update_option_' . $slug                => [
            [ 'clean_used_css_and_cache', 9, 2 ],
            [ 'maybe_set_processing_transient', 50, 2 ],
            [ 'maybe_unlock_preload', 9, 2 ],
            [ 'maybe_delete_transient', 10, 2 ],
        ],
        'switch_theme'                          => 'truncate_used_css',
        'permalink_structure_changed'           => 'truncate_used_css',
        'rocket_domain_options_changed'         => 'truncate_used_css',
        'wp_trash_post'                         => 'delete_used_css_on_update_or_
        'delete_post'                           => 'delete_used_css_on_update_or_
        'clean_post_cache'                      => 'delete_used_css_on_update_or_
        'wp_update_comment_count'               => 'delete_used_css_on_update_or_
        'edit_term'                             => 'delete_term_used_css',
        'pre_delete_term'                       => 'delete_term_used_css',
        'admin_notices'                         => [
            [ 'display_no_table_notice' ],
            [ 'notice_write_permissions' ],
        ],
        'rocket_before_add_field_to_settings'   => [
            [ 'set_optimize_css_delivery_value', 10, 1 ],
            [ 'set_optimize_css_delivery_method_value', 10, 1 ],
        ],
        'wp_rocket_upgrade'                     => [
            [ 'set_option_on_update', 14, 2 ],
            [ 'update_safelist_items', 15, 2 ],
            [ 'delete_used_css', 16, 2 ],
            [ 'cancel_pending_jobs_as', 16, 2 ],
            [ 'drop_resources_table', 18, 2 ],
        ],
```

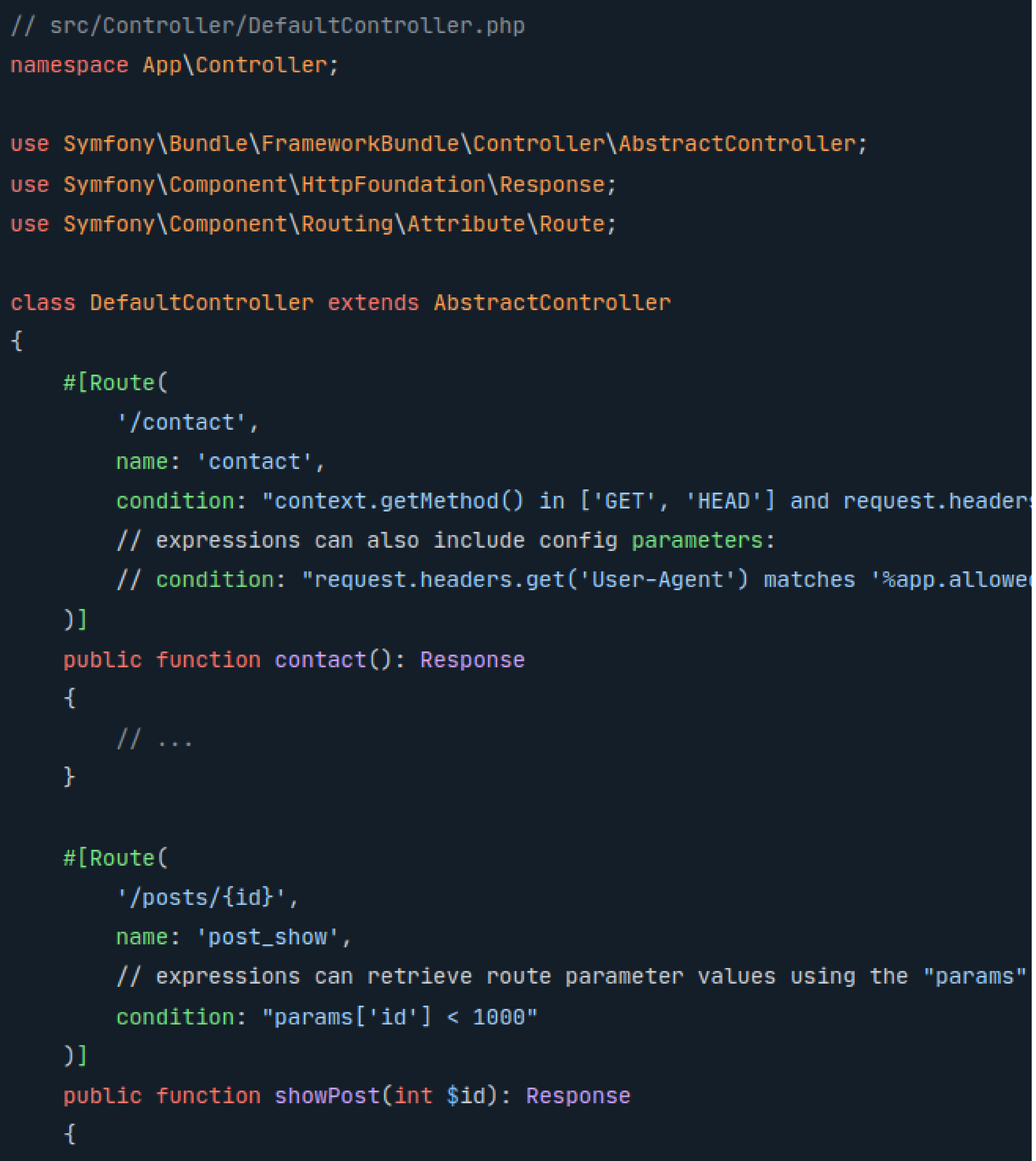
Yet another issue:

- Big block
- Multiple syntaxes
- Really verbose



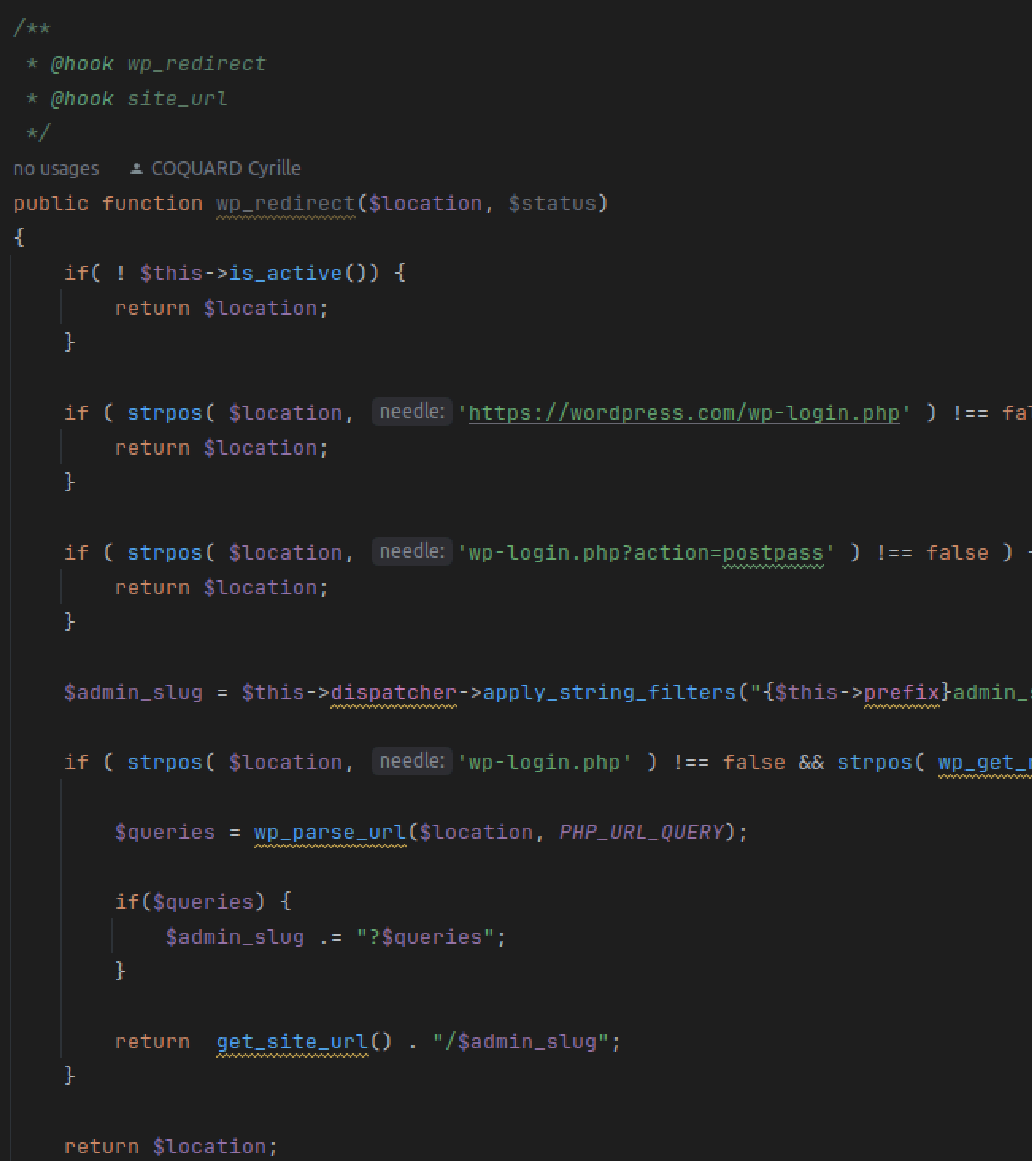
```
use ...

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "api" middleware group. Make something great!
|
*/

Route::middleware( middleware: 'auth:sanctum')->get( uri: '/user', function (Request $request) {
    return $request->user();
});

Route::get( uri: '/{product}/latest', [\App\Http\Controllers\CheckLastVersion::class, 'check']);
Route::get( uri: '/{product}/changelog', [\App\Http\Controllers\ListChangeLog::class, 'list']);
Route::post( uri: '/{product}/upload', [\App\Http\Controllers\UploadVersion::class, 'upload'])->
Route::get( uri: '/{product}/{version}', [\App\Http\Controllers\FetchZip::class, 'fetch'])->midd
Route::post( uri: '/', [\App\Http\Controllers\CreateProduct::class, 'create'])->middleware( middl
Route::post( uri: '/{product}/licence', [\App\Http\Controllers\CreateLicence::class, 'create'])-
Route::post( uri: '/{product}/licence/cancel', [\App\Http\Controllers\CancelLicence::class, 'can
```

Looking somewhere else



```php
// src/Controller/DefaultController.php
namespace App\Controller;

use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

class DefaultController extends AbstractController
{
    #[Route(
        '/contact',
        name: 'contact',
        condition: "context.getMethod() in ['GET', 'HEAD'] and request.header
        // expressions can also include config parameters:
        // condition: "request.headers.get('User-Agent') matches '%app.allowe
    )]
    public function contact(): Response
    {
        // ...
    }

    #[Route(
        '/posts/{id}',
        name: 'post_show',
        // expressions can retrieve route parameter values using the "params"
        condition: "params['id'] < 1000"
    )]
    public function showPost(int $id): Response
    {
```

Looking somewhere else again



```
/**
 * @hook wp_redirect
 * @hook site_url
 */
no usages   COQUARD Cyrille
public function wp_redirect($location, $status)
{
    if( ! $this->is_active()) {
        return $location;
    }

    if ( strpos( $location,  needle: 'https://wordpress.com/wp-login.php' ) !== fa
        return $location;
    }

    if ( strpos( $location,  needle: 'wp-login.php?action=postpass' ) !== false )
        return $location;
    }

    $admin_slug = $this->dispatcher->apply_string_filters("{$this->prefix}admin_

    if ( strpos( $location,  needle: 'wp-login.php' ) !== false && strpos( wp_get_

        $queries = wp_parse_url($location, PHP_URL_QUERY);

        if($queries) {
            $admin_slug .= "?$queries";
        }

        return  get_site_url() . "/$admin_slug";
    }

    return $location;
```

The final solution:

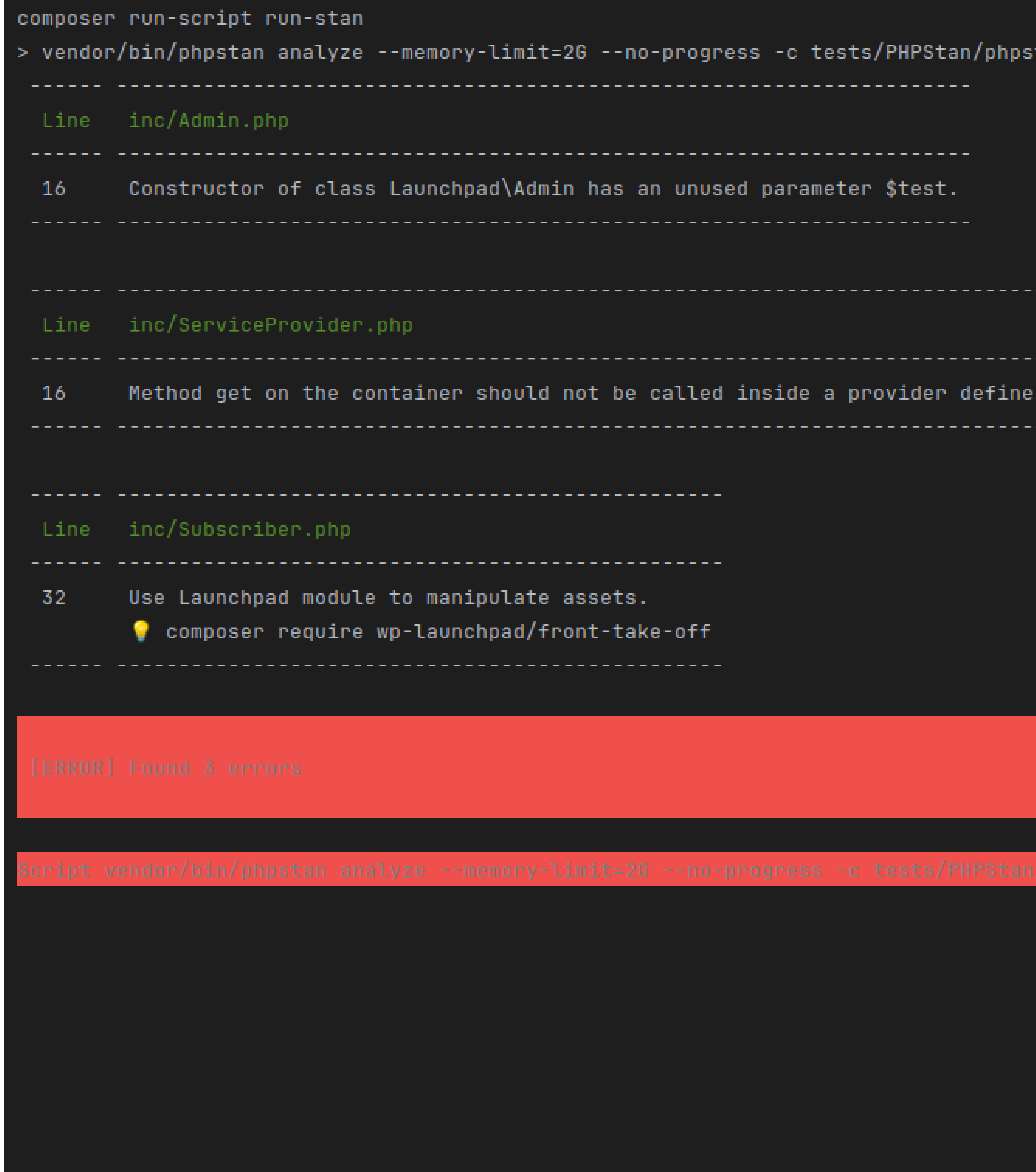
- One syntax
- Few elements
- Easy to find

Advancing in the smog:

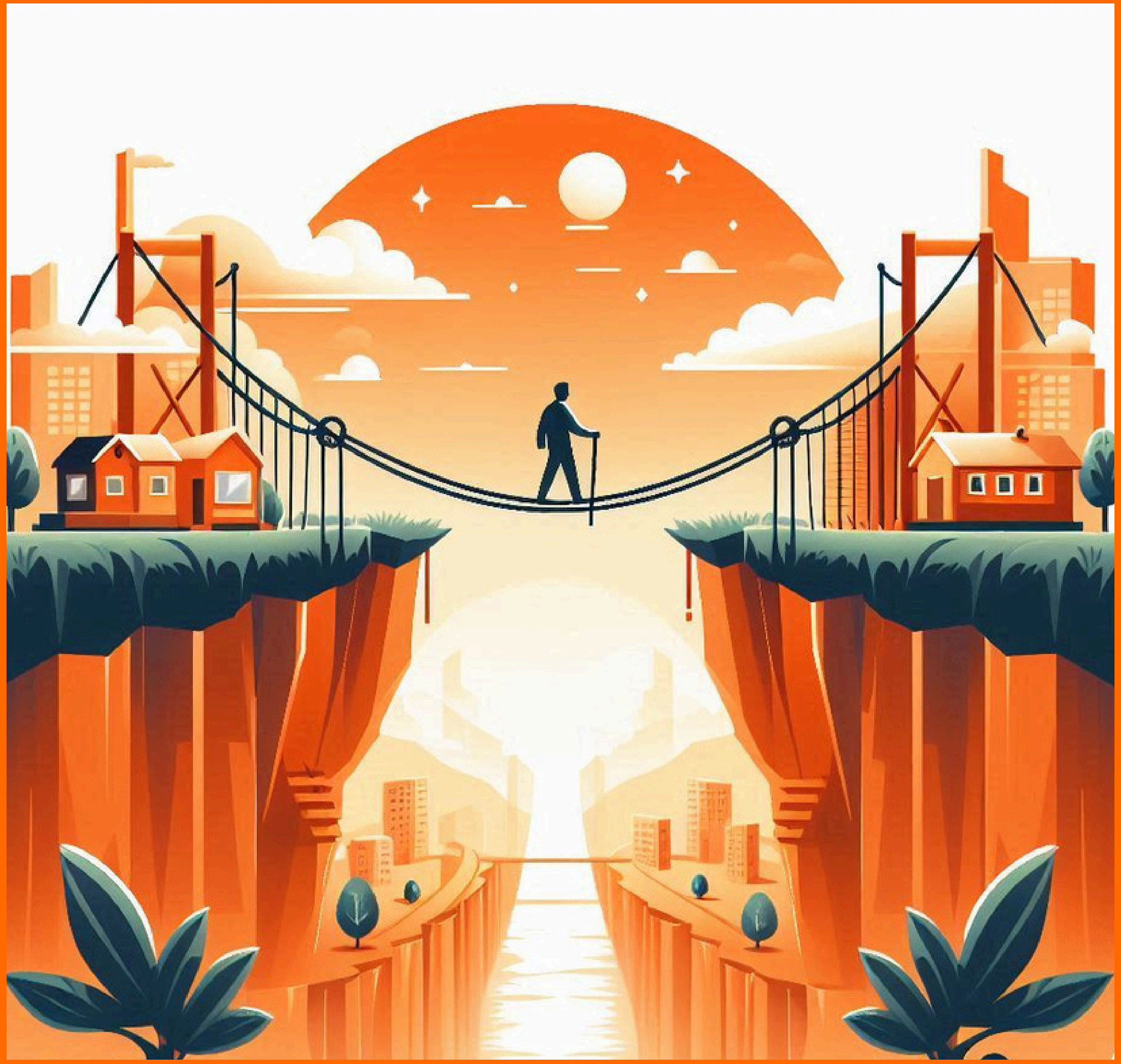
- Code the framework way
- Time consuming
- No focus on what matters

It is **OK** to **fail:**

- Code your way
- Run PHPStan
- Get hints



```
composer run-script run-stan
> vendor/bin/phpstan analyze --memory-limit=2G --no-progress -c tests/PHPStan/phps
 ------ ---------------------------------------------------------------------
  Line   inc/Admin.php
 ------ ---------------------------------------------------------------------
  16     Constructor of class Launchpad\Admin has an unused parameter $test.
 ------ ---------------------------------------------------------------------


 ------ -----------------------------------------------------------------------
  Line   inc/ServiceProvider.php
 ------ -----------------------------------------------------------------------
  16     Method get on the container should not be called inside a provider define
 ------ -----------------------------------------------------------------------


 ------ ----------------------------------------------------
  Line   inc/Subscriber.php
 ------ ----------------------------------------------------
  32     Use Launchpad module to manipulate assets.
         💡 composer require wp-launchpad/front-take-off
 ------ ----------------------------------------------------


 [ERROR] Found 3 errors


Script vendor/bin/phpstan analyze --memory-limit=2G --no-progress -c tests/PHPStan
```



I invented nothing:

- I share same way I learn
- One community is essential

Don’t hesitate to contribute !

COQUARD Cyrille
Software engineer
at WP Media

Launchpad repo