⚙️ Configuration & Options Pattern — محاضرة عملية

> **كيفية إدارة الـ Settings بشكل احترافي وآمن**

📝 جدول المحتويات

🎯 مقدمة خفيفة

يعني إيه Configuration؟

Configuration = إعدادات بتغيّر حسب الـ Environment (تطوير/إنتاج/اختبار).

```
لما تشتغل تطوير:
- Database: localhost
- Email driver: log (بتطبع في اللوج)
- Debug: true (كشف أخطاء كاملة)

لما تشتغل في الإنتاج:
- Database: production server
- Email driver: SMTP (رسايل حقيقية)
- Debug: false (ما تظهر أخطاء للـ users)
```

ليه Options Pattern؟

القديم (السيء):

```php
// ❌ Hardcoded + مخيف
$dbHost = "localhost"; // معناه تغيير كود في كل environment!
$dbName = "mydb";
$dbUser = "root";
```

الحديث (الكويس):

```php
// ✅ Centralized + منظم + آمن
config('database.connections.mysql.host') // من .env أو config file
```

الأساسيات اللي لازم تعرفها 🧠

1) .env File (الـ Secrets)

ملف في الـ root فيه sensitive data بتاع المشروع:

```
# .env

APP_NAME=MyApp
APP_ENV=local
APP_DEBUG=true

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=mydb
DB_USERNAME=root
DB_PASSWORD=secret123

MAIL_DRIVER=log
MAIL_FROM=hello@example.com

API_KEY=abc123xyz...
```

مهم جداً:

- ❌ ما تحط .env في git
- ✅ احط .env.example في git (بدون القيم الحقيقية)
- ✅ كل developer يعمل نسخة بتاعته من .env

2) Config Files (التنظيم)

config files: مباشرة، استخدم .env بدل ما تقرأ من

```
// config/database.php
return [
    'default' => env('DB_CONNECTION', 'mysql'),
    'connections' => [
        'mysql' => [
            'driver' => 'mysql',
            'host' => env('DB_HOST', 'localhost'),
            'port' => env('DB_PORT', 3306),
            'database' => env('DB_DATABASE', 'mydb'),
            'username' => env('DB_USERNAME', 'root'),
            'password' => env('DB_PASSWORD', ''),
        ],
    ],
```

```
];

// في الكود:
config('database.connections.mysql.host') // ✅ آمن + منظم
```

3) Options Pattern (الـ Power)

Options Pattern = class بتأخذ كل الـ configuration وتعتبرها property:

```php
// app/Options/MailOptions.php
namespace App\Options;

class MailOptions {
    public function __construct(
        public string $driver = 'log',
        public string $from = 'hello@example.com',
        public int $timeout = 30,
        public bool $enabled = true,
    ) {}
}

// استخدام
$mailOptions = new MailOptions(
    driver: config('mail.driver'),
    from: config('mail.from.address'),
    timeout: config('mail.timeout'),
);

echo $mailOptions->driver; // 'log'
```

الفوائد:

- ✅ Type-safe (PHP الـ يتفحص types)
- ✅ المتاح options واضح جداً إيه الـ
- ✅ سهل للـ testing

4) Service Provider (الـ حط الـ Options في الـ Container)

```php
// app/Providers/AppServiceProvider.php

public function register() {
    $this->app->singleton(MailOptions::class, function () {
        return new MailOptions(
            driver: config('mail.driver'),
            from: config('mail.from.address'),
            timeout: config('mail.timeout'),
        );
    });
}
```

```php
// أي في controller أو service:
public function __construct(private MailOptions $options) {}

public function send() {
    if (!$this->options->enabled) return;
    // استخدم الـ options
}
```

💻 Options Pattern في Laravel

مثال عملي: Payment Configuration

```php
// config/payment.php
return [
    'default_provider' => env('PAYMENT_PROVIDER', 'stripe'),
    'stripe' => [
        'key' => env('STRIPE_KEY'),
        'secret' => env('STRIPE_SECRET'),
        'timeout' => env('STRIPE_TIMEOUT', 30),
    ],
    'paypal' => [
        'client_id' => env('PAYPAL_CLIENT_ID'),
        'secret' => env('PAYPAL_SECRET'),
        'timeout' => env('PAYPAL_TIMEOUT', 30),
    ],
    'retries' => env('PAYMENT_RETRIES', 3),
];
```

```php
// app/Options/PaymentOptions.php
namespace App\Options;

class PaymentOptions {
    public function __construct(
        public string $provider = 'stripe',
        public string $stripeKey = '',
        public string $stripeSecret = '',
        public string $paypalClientId = '',
        public string $paypalSecret = '',
        public int $retries = 3,
        public int $timeout = 30,
    ) {}

    public function getProviderSecret(): string {
        return match($this->provider) {
            'stripe' => $this->stripeSecret,
            'paypal' => $this->paypalSecret,
            default => throw new \Exception('Unknown provider'),
```

```
        };
    }
}
```

```
// app/Providers/PaymentServiceProvider.php
public function register() {
    $this->app->singleton(PaymentOptions::class, function ($app) {
        return new PaymentOptions(
            provider: $app['config']['payment.default_provider'],
            stripeKey: $app['config']['payment.stripe.key'],
            stripeSecret: $app['config']['payment.stripe.secret'],
            paypalClientId: $app['config']['payment.paypal.client_id'],
            paypalSecret: $app['config']['payment.paypal.secret'],
            retries: $app['config']['payment.retries'],
        );
    });
}
```

```
// app/Services/PaymentService.php
namespace App\Services;

use App\Options\PaymentOptions;

class PaymentService {
    public function __construct(private PaymentOptions $options) {}

    public function processPayment(float $amount, string $method): bool {
        $secret = $this->options->getProviderSecret();

        for ($i = 0; $i < $this->options->retries; $i++) {
            try {
                // استدعاء payment gateway
                $this->callGateway($amount, $method, $secret);
                return true;
            } catch (\Exception $e) {
                if ($i === $this->options->retries - 1) throw $e;
                sleep(1); // انتظر قبل المحاولة retry
            }
        }

        return false;
    }

    private function callGateway($amount, $method, $secret) {
        // Implementation
    }
}
```

```php
// في الـ Route أو Controller
Route::post('/pay', function (PaymentService $service) {
    $result = $service->processPayment(100, 'card');
    return response()->json(['success' => $result]);
});
```

مثال CURL:

```bash
# POST /pay
curl -X POST http://localhost:8000/api/pay \
  -H "Content-Type: application/json" \
  -d '{"amount": 100, "method": "card"}'

# Response:
# {
#   "success": true
# }
```

⚠ الأخطاء الشائعة

1) Hardcoding القيم

```php
// ❌ خطر جداً
class PaymentService {
    private $stripeKey = 'sk_test_123456'; // ❌ في الكود!

    public function charge() {
        // استخدم الـ key
    }
}

// ✅ صح
class PaymentService {
    public function __construct(private PaymentOptions $options) {}

    public function charge() {
        $key = $this->options->stripeSecret; // من الـ environment
    }
}
```

2) نسيان الـ Default Values

```php
// ❌ ما يكون موجود env variable لما الـ exception بتحصل
$timeout = config('payment.timeout'); // null!
```

```
// ✅ مع default
$timeout = config('payment.timeout', 30);
```

3) تسريب Secrets في الـ Logs

```
// ❌ خطر
\Log::info('Charging with key: ' . $this->options->stripeSecret);

// ✅ نقع الـ sensitive data
\Log::info('Charging with provider: ' . $this->options->provider);
```

4) عدم استخدام Type Hints

```
// ❌ مش واضح إيه الـ options
public function __construct($options) {}

// ✅ واضح جداً
public function __construct(PaymentOptions $options) {}
```

👔 Senior Corner

1) Configuration Caching (Performance)

```
# احفظ الـ config في cache في الإنتاج،
php artisan config:cache

# بدل تقرأها في كل request بتحصل من الـ cache
```

ملحوظة: لو عدّلت config, لازم تشغّل الـ command ده تاني.

2) Validation Configuration

```
// app/Options/PaymentOptions.php

public function validate(): self {
    if (empty($this->stripeSecret) && $this->provider === 'stripe') {
        throw new \InvalidArgumentException('Stripe secret is required');
    }

    if ($this->retries < 1 || $this->retries > 10) {
        throw new \RangeException('Retries must be between 1 and 10');
    }
```

```
    return $this;
}

// في الـــ provider
public function register() {
    $this->app->singleton(PaymentOptions::class, function ($app) {
        return (new PaymentOptions(...))->validate();
    });
}
```

3) Environment-Specific Configuration

```
// config/app.php
return [
    'env' => env('APP_ENV', 'production'),
    'debug' => env('APP_DEBUG', false),
    'log_level' => env('LOG_LEVEL', 'error'),
    'cache_config' => env('APP_ENV') === 'production',
];

// في الـــ provider
if (config('app.cache_config')) {
    $this->app->make('cache')->remember('payment_options', ..., fn() => new
PaymentOptions(...));
}
```

4) Macroable Configuration Class

```
class PaymentOptions {
    public static function fromEnv(): self {
        return new self(
            provider: env('PAYMENT_PROVIDER', 'stripe'),
            stripeSecret: env('STRIPE_SECRET'),
            // ... باقي الـــ properties
        );
    }
}

// استخدام
$options = PaymentOptions::fromEnv();
```

5) Configuration Inheritance

```
// Base configuration class
abstract class BaseOptions {
    protected array $defaults = [];
```

```
    public function get(string $key, $default = null) {
        return $this->{$key} ?? $this->defaults[$key] ?? $default;
    }
}

// Specific configuration
class PaymentOptions extends BaseOptions {
    protected array $defaults = [
        'retries' => 3,
        'timeout' => 30,
    ];
}
```

6) Testing مع Options

```
// tests/Feature/PaymentTest.php

public function test_payment_uses_correct_provider() {
    $options = new PaymentOptions(
        provider: 'stripe',
        stripeSecret: 'test_secret',
    );

    $service = new PaymentService($options);

    $this->assertTrue($service->processPayment(100, 'card'));
}

// Override في الـ test
public function setUp(): void {
    parent::setUp();

    $this->app->singleton(PaymentOptions::class, fn() =>
        new PaymentOptions(provider: 'test', retries: 1)
    );
}
```

🎓 خاتمة + تحدي

تحديك ليك:

اعمل **DatabaseOptions** class بـ:

- ✅ Connection type (mysql/postgres/sqlite)
- ✅ Host, Port, Database, Username, Password
- ✅ Connection pool settings
- ✅ Method `getConnectionString()` ترجع الـ DSN

- ✅ Validation للـ required fields

الحل:

```php
// app/Options/DatabaseOptions.php
namespace App\Options;

class DatabaseOptions {
    public function __construct(
        public string $connection = 'mysql',
        public string $host = 'localhost',
        public int $port = 3306,
        public string $database = '',
        public string $username = 'root',
        public string $password = '',
        public int $poolMin = 2,
        public int $poolMax = 10,
    ) {
        $this->validate();
    }

    public function validate(): void {
        if (empty($this->database)) {
            throw new \InvalidArgumentException('Database name is required');
        }
    }

    public function getConnectionString(): string {
        return match($this->connection) {
            'mysql' => "{$this->connection}://{$this->username}@{$this->host}:
{$this->port}/{$this->database}",
            'postgres' => "pgsql://{$this->username}@{$this->host}:{$this-
>port}/{$this->database}",
            default => throw new \Exception('Unsupported connection type'),
        };
    }
}
```

Checklist ليك:

- ✅ فهمت الفرق بين config files و .env
- ✅ عرفت ليه Options Pattern مهم
- ✅ عملت Options class لموضوع معين
- ✅ ربطت Options مع Service Provider
- ✅ فهمت كيفية استخدام Dependency Injection
- ✅ عرفت الأخطاء الشائعة

📋 Cheat Sheet

.env Variables الشهيرة

```
APP_NAME=MyApp
APP_ENV=local|production
APP_DEBUG=true|false

DB_CONNECTION=mysql|postgres|sqlite
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=mydb
DB_USERNAME=root
DB_PASSWORD=secret

MAIL_DRIVER=log|smtp|mailgun
MAIL_FROM_ADDRESS=hello@example.com

CACHE_DRIVER=file|redis|memcached
QUEUE_DRIVER=sync|redis|database
```

Config Functions

```
config('key')                    // اجلب قيمة
config('key', 'default')         // مع default
config(['key' => 'value'])       // حط قيمة
env('KEY', 'default')            // من .env
```

Options Pattern Steps

```
1. اعمل config file (config/payment.php)
2. اعمل Options class (app/Options/PaymentOptions.php)
3. Register في Service Provider
4. استخدم الـ Dependency Injection
5. Test مع different configurations
```

Best Practices

الممارسة	ليه؟
استخدم .env للـ secrets	أمان
استخدم config files للـ logic	منظم
استخدم Options Pattern	Type-safe + maintainable
احط default values	Prevent errors
cache config في production	Performance

الممارسة	ليه؟
validate configuration	Early error detection
ما تسجل secrets	Security

النهاية 🚀

الفكرة الأساسية:

> Configuration أمان = بشكل صحيح + flexibility + سهولة في الـ maintain.

يلا.. طبّق! 💪

□□ ♡ □□ □□□□□□ *Backend Instructor*