Manuel Pijierro Sa - @mpijierro

s|ngular
it can be done.

Sobre mi

Manuel Pijierro Sa

mandev.es/cv/

s|ngular
it can be done.

Reference Leader Software Development at **S|Ngular**

@mpijierro

https://github.com/mpijierro

https://medium.com/@mpijierro

Sobre la charla

Talk is cheap. Show me the code

Torvalds, Linux (25-08-2000)

Sobre el código limpio

Correcto

Simple

Testable

Extensible

Sobre el código limpio

No existe una definición exacta

Tiene algo de subjetivo

Proceso de mejora contínua

Sobre el código limpio

"Code is like humor. When you have to explain it, it's bad"

Cory House, @housecor

"It is not enough for code to work"

Robert C. Martin, @unclebobmartin

Mantenibilidad

Mantenibilidad

$$\textbf{cost total} = \text{cost}_{\text{develop}} + \text{cost}_{\text{maintain}}$$

Edward Yurdon & **Larry L. Constantine**: Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design
https://www.win.tue.nl/~wstomv/quotes/structured-design.html

Mantenibilidad

$$\textbf{cost maintain} = cost_{understand} + cost_{change} + cost_{test} + cost_{deploy}$$

Edward Yurdon & **Larry L. Constantine**: Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design
https://www.win.tue.nl/~wstomv/quotes/structured-design.html

Mantenibilidad

Alta mantenibilidad

Mejor software

Cambios en menos tiempo y menos recursos

Software más barato

Software más competitivo

Proyectos y productos más competitivos

Empresas más solventes

Mantenibilidad

Objetivos

LEGIBLE

COMPRENSIBLE

Programación Orientada a Objetos

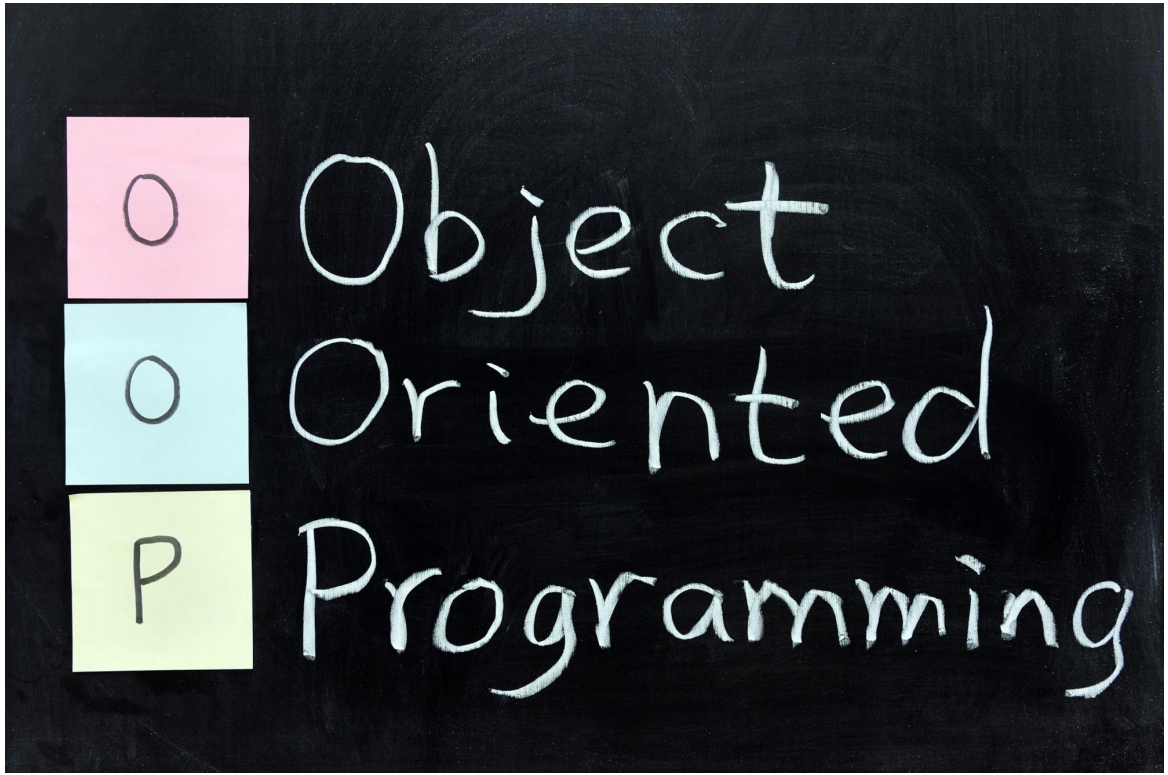


Imagen de https://medium.com/@richardeng/a-simple-explanation-of-oop-46a156581214

Programación Orientada a Objetos

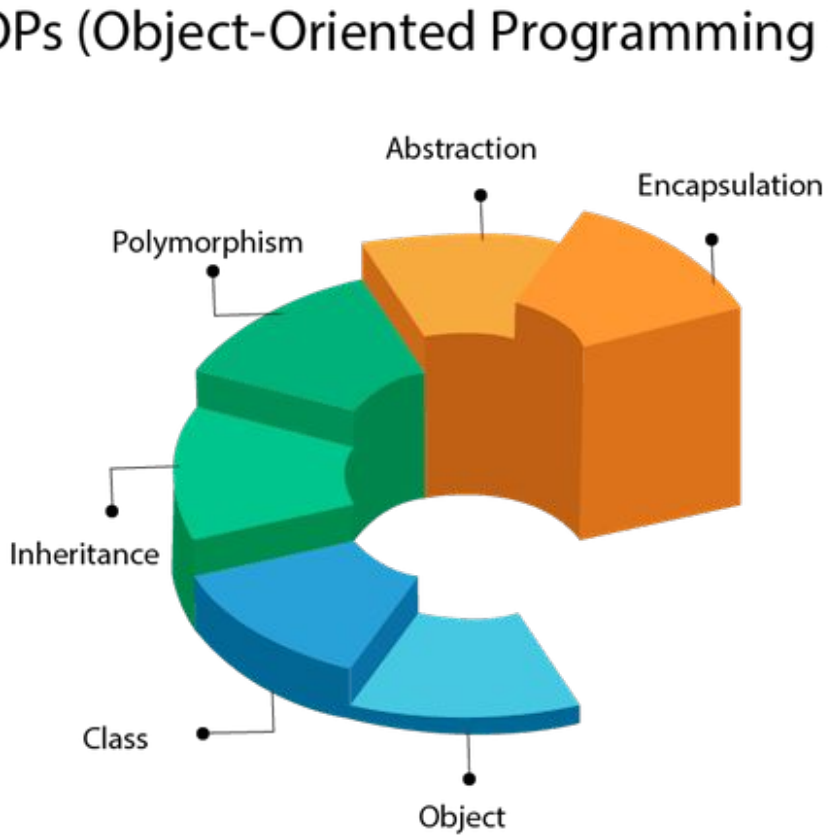


Imagen de https://www.javatpoint.com/java-oops-concepts

Programación Orientada a Objetos

Clases

Definición de características y comportamiento

modelan objetos del mudo real y abstractos

Programación Orientada a Objetos

Objetos

instancia de un clase

contienen sus atributos y comportamiento

siempre deben tener un **estado válido**

Programación Orientada a Objetos

Objetos

Sobre el estado

- debe permanecer oculto

```
class Foo{
    private $attribute1;
    private $attribute2;

    //....
}
```

Programación Orientada a Objetos

Objetos

Sobre el estado

- no revelar los detalles de implementación
- no revelar estructura interna

```
class Foo{

    [redacted]

    public function __construct(){
        //...
    }


    public function method_1 (){
        //...
    }

    public function method_2 (int $param){
        //...
    }

    public function method_3 (string $param){
        //.....
    }


}
```

Programación Orientada a Objetos

Objetos

Sobre el estado

- no usar “getters”

- porque revelan estructura interna y decisiones de diseño.
- porque tratamos a los objetos como estructura planas de datos y no como objetos .
- porque la información pierde valor semántico.

Programación Orientada a Objetos

Objetos

Sobre el estado

- no usar “getters”

Data thinking

```
Dog dog = new Dog();
dog.setBall(new Ball());
```

```
Dog dog = new Dog();
Ball ball = dog.getBall();
```

Object thinking

```
Dog dog = new Dog();
dog.take(new Ball());
Ball ball = dog.give();
```

https://www.yegor256.com/2014/09/16/getters-and-setters-are-evil.html

Programación Orientada a Objetos

Objetos

Sobre el estado

- no usar “getters”

Data thinking

```
class Product {

    private $id;
    private $name;
    private $price;
    private $date;

    public function getId():int{
        return $this->id;
    }

    public function getName(): string{
        return $this->name;
    }

    public function getPrice(): float{
        return $this->price;
    }

    public function getDate(): string{
        return $this->date;
    }

}
```

Object thinking

```
class Identification (){

    //...

}

class Money {

    //...patrón Money
}

class Product {

    private $id;
    private $name;
    private $price;
    private $date;

    public function id():Identification{
        return $this->id;
    }

    public function name(): string{
        return $this->name;
    }

    public function price(): Money{
        return new Money($this->price);
    }

    public function date(): Datetime{
        return new Datetime($this->date);
    }

}
```

https://www.yegor256.com/2014/09/16/getters-and-setters-are-evil.html

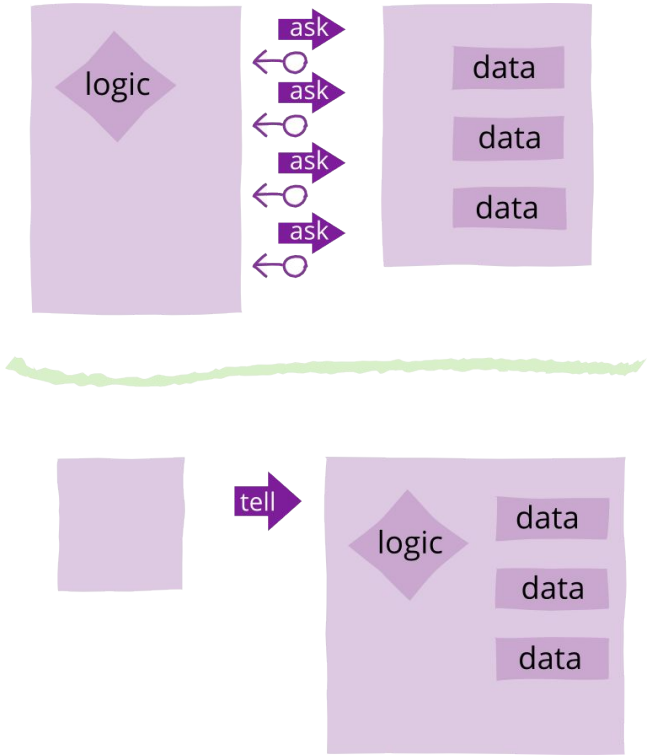
Programación Orientada a Objetos

Objetos

Sobre el estado

- Principio “Tell don’t ask”

“Dime que hago, no me preguntes”



https://martinfowler.com/bliki/TellDontAsk.html

Programación Orientada a Objetos

Objetos

Sobre el estado

- Principio “Tell don’t ask”
    - No volverse locos (*getter erradicator*)
    - Objetos intercambian información
    - Convertir *getters* en métodos semánticos que devuelvan tipos concretos

https://martinfowler.com/bliki/GetterEradicator.html

Tipos de objetos

Imagen de http://time.com/3931946/lego-sustainable-materials/

Tipos de objetos

- Servicios
- Entidades
- Value objects
- Domain events
- Event listener
- Dto's

Tipos de objetos

Servicios

- Objetos que realizan una tarea o devuelven información
- Se instancian una vez y se utilizan muchas veces

Tipos de objetos

Servicios

- Según el contexto
  - Servicios de aplicación
  - Servicios de dominio
  - Servicios de infraestructura

- Según el tipo de operación
  - **Command**: hacen una tarea
  - **Query**: obtienen información

Tipos de objetos

Entidades

- Objetos que representan un concepto del dominio
- Tienen identidad inmutable -› son identificables
- Tienen un alto significado semántico
- Son objetos inmutables
- Ej: Usuario, Producto, Pedido

Tipos de objetos

Value object

- Objetos que representan un concepto general
- No tienen identidad. Definidas por su valor
- Son tipos complejos que pueden tener comportamiento
- Son objetos inmutables
- Son conceptos medibles, cuantificables o descriptores
- Ej: Fechas, precio de un producto, peso, altura, edad

Tipos de objetos

Value object

- Les da sentido a tipos primitivos
- Los modela de acuerdo al dominio
- Le da un comportamiento propio

```php
class Age (){

    private $age;

    public function __construct (int $age){

        Assert::assertGreaterThanOrEqual($age,0);

        $this->age = $age;
    }

    public function get():int{
        //..
    }

    public function younger(){
        //
    }

    public function isMillenial(){
        //..
    }

    public function isMaduritoInteresante(){
        //..
    }

}
```

Tipos de objetos

Domain events

- Objetos que notifican a otros servicios que algo ha ocurrido en el modelo de escritura (command) dentro de nuestro dominio.

Tipos de objetos

Event Listener

- Objetos que realizan tareas secundarias relacionadas con una tarea principal.
- Suelen recibir como parámetro el evento al que están asociado.

Tipos de objetos

DTO's

- Data Transfer Object
- Objeto que lleva datos entre procesos
- Únicamente encapsula información
- No tienen comportamiento asociado -› modelos anémicos
- Inmutables

Creando objetos

Creando objetos

Creando servicios

Inyectar **dependencias** y **configuración** por **constructor**

Creando objetos

Creando servicios

```php
interface Logger
{
    public function log(string $message): void;
}

final class FileLogger implements Logger
{
    private Formatter $formatter;
    private string $logFilePath;

    public function __construct(Formatter $formatter, string $logFilePath)
    {
        $this->formatter = $formatter;
        $this->logFilePath = $logFilePath;
    }

    public function log(string $message): void
    {
        $formattedMessage = $this->formatter->format($message);

        //...

        file_put_contents($this->logFilePath, $formattedMessage, FILE_APPEND);

    }
}
```

Creando objetos

Creando servicios

Los **argumentos** del constructor deberían ser **obligatorios**

NO pasar argumentos con valor **null**

Creando objetos

Creando servicios

```php
interface Logger
{
    public function log(string $message): void;
}

final class FileLogger implements Logger
{
    private Formatter $formatter;

    public function __construct(Formatter $formatter = null)
    {
        $this->formatter = $formatter;
    }

    public function log(string $message): void
    {
        // Evitar estas comprobaciones...

        if ($this->formatter instanceof Logger){
            $this->formatter->format($message);
            //..
        }
    }
}
```

Creando objetos

Creando servicios

Únicamente usar inyección por constructor

⇩

NO utilizar *setters*

⇩

Servicios son **inmutables**

Creando objetos

Creando servicios

```php
class Logger{
    //..
}

final class BankImporter{

    private Logger $logger;

    public function __construct(){
        //...la dependencia de Logger no es un argumento en el constructor
    }

    public function setLogger (Logger $logger){
        $this->logger = $logger;
    }

    public function register ($amount){
        //..
        $this->loger->foo();

        //..
    }

}

$bankImporter = new Importer();
//¡excepción!...sin instancia de logger
//hay que conocer la clase e investigar...
$bankImporter->register(34);
```

Creando objetos

Creando servicios

Haz todas las **dependencias explícitas**

Convierte llamadas estáticas en dependencias de objetos

Creando objetos

Creando servicios

```php
final class DashboardController{

    public function __invoke(): Response{

        $recentPosts = [];

        if (Cache::has('recent_posts')) {
            $recentPosts = Cache::get('recent_posts');
        }
        // ...
    }
}
```

```php
final class DashboardController{

    private CacheProvider $cache;

    public function __construct(CacheProvider $cache){
        $this->cache = $cache;
    }

    public function __invoke(): Response {

        $recentPosts = [];

        if ($this->cache->has('recent_posts')) {
            $recentPosts = $this->cache->get('recent_posts');
        }
        // ...
    }
}
```

Creando objetos

Creando servicios

Dependencias y configuración por constructor

Datos relevantes para la **tarea** como **argumentos de métodos**

Creando objetos

Creando servicios

¿Parámetro de constructor o parámetro de método?

¿Podríamos ejecutar este servicio en un proceso batch sin requerir su instanciación una y otra vez?

Creando objetos

Creando servicios

```php
class Repository{
    public function save(){
        //...
    }
}

final class BankImporter{

    private Repository $repository;
    private $id;
    private $amount;

    public function __construct(Repository $repository, int $id, int amount){
        $this->repository = $repository;
        $this->id = $id;
        $this->amount = $amount;
    }

    public function save (){
        //..
        $this->repository->save($this->id, $this->amount);
        //..
    }

}

$bankImporter = new Importer(new Repository(), 1, 34);
$bankImporter->save();

$bankImporter = new Importer(new Repository(), 7, -4);
$bankImporter->save();

$bankImporter = new Importer(new Repository(), 19, 199);
$bankImporter->save();
```

```php
class Repository{
    public function save(){
        //...
    }
}

final class BankImporter{

    private Repository $repository;

    public function __construct(Repository $repository){
        $this->repository = $repository;
    }

    public function save (int $id, int $amount){
        //..
        $this->repository->save($id, $amount);
        //..
    }

}

$bankImporter = new Importer(new Repository());
$bankImporter->save(1,34);
$bankImporter->save(7,-4);
$bankImporter->save(19,199);
```

Creando objetos

Creando servicios

Dentro del constructor únicamente

- validar parámetros (aserciones)
- lanzar excepciones si un parámetro es inválido
- asignarlos a las propiedades

Creando objetos

Resumen

- Inyectar dependencias y configuración por constructor
- Argumentos del constructor obligatorios
- NO pasar argumentos con valor null
- Dependencias explícitas
- NO utilizar setters
- Convertir llamadas estáticas en dependencias de objetos
- Datos relevantes para la tarea como argumentos de métodos
- El constructor solo valida y asigna
- Servicios inmutables

Creando objetos

Inmutabilidad

- código predecible
- somos consciente del estado de nuestros objetos
- nos asegura que nuestro código no cambia
- si hay cambio de estado -› instanciamos un nuevo objeto

Usando objetos

Usando objetos

Tipos de operaciones de los **métodos**

COMMAND

QUERY

Command/Query Separation

https://martinfowler.com/bliki/CommandQuerySeparation.html

Usando objetos

Tipos de operaciones de los **métodos**

command →

query →

```php
final class Counter{

    private int $count = 0;

    public function increment(): void {
        $this->count++;
    }

    public function currentCount(): int {
        return $this->count;
    }
}
```

https://martinfowler.com/bliki/CommandQuerySeparation.html

Usando objetos

Obteniendo información con métodos *query*

- Únicamente retornan información
- No producen cambios en el estado observable del sistema
- Devuelven un único tipo de dato específico
- Idempotente
- No exponen el estado interno -› no *getters*
- Pueden encadenarse llamadas de métodos *query*

Usando objetos

Obteniendo información: sugerencia *naming* en métodos **query**

- Empiezan por **get**.....
  - Retornan lo buscado
  - Lanzan una excepción
- Empiezan por **find**.....
  - Retornan lo buscado
  - Retornan alternativa vacía: *empty list, null object*

```
class UserRepository{

    //...

    public function getById (int $id): User{
        $user = $this->orm->get($id);

        if ($user){
            return $user;
        }

        throw new ModelNotFoundException();

    }


    public function findOneBy (string $type) : Collection{

        return $this->orm->search($type)->all();;

    }

    //..
}
```

Usando objetos

Cambiando el estado con métodos ***command***

- No retornan nada: *return void*
- Limitar el alcance de un método command -› usar eventos
- No idempotente
- Si algo va mal -› lanza una excepción
- Puede usar *queries* para recoger información y ejecutar su tarea

Usando objetos

Cambiando el estado: sugerencia *naming* en métodos *command*

- Usar formas imperativas: “haz esto”, “haz aquello”
- Normalmente: verbo + objeto a tratar
- Indica que el cliente puede dar órdenes al objeto

Usando objetos

Usar **eventos** para realizar tareas secundarias

- Buscar cumplir el Principio de Responsabilidad Única

- Diferenciar tarea principal del método de tareas secundarias

    - ¿Tiene el nombre del método un *'And'* que indique tareas adicionales?
    - ¿Todas las líneas del método contribuyen a la tarea principal?
    - ¿Podría hacerse en un proceso background?

Usando objetos

Usar **eventos** para realizar tareas secundarias

```php
public function changeUserPassword( UserId $userId, string $plainTextPassword ): void {
    $user = $this->repository->getById($userId);
    $hashedPassword = ...;
    $user->changePassword($hashedPassword);
    $this->repository->save($user);
    $this->mailer->sendPasswordChangedEmail($userId);
}
```

¿Cómo limitamos el alcance de este método y cumplimos además el principio de responsabilidad única?

Usando objetos

Usar **eventos** para realizar tareas secundarias

1) Creamos un objeto que refleje el evento que ha ocurrido en el sistema

```php
final class UserPasswordChanged {

    private UserId $userId;

    public function __construct(UserId $userId) {
        $this->userId = $userId;
    }

    public function userId(): UserId {
        return $this->userId;
    }
}
```

Usando objetos

Usar **eventos** para realizar tareas secundarias

2) Sustituimos el código original por la generación del evento

```php
public function changeUserPassword( UserId $userId, string $plainTextPassword ): void {

    $user = $this->repository->getById($userId);
    $hashedPassword = ...;
    $user->changePassword($hashedPassword);
    $this->repository->save($user);

    /**
     * Después de cambiar la contraseña, lanzamos un evento `UserPasswordChanged`,
     * de manera que otros servicios puedan responder a él
     */

    $this->eventDispatcher->dispatch(
        new UserPasswordChanged($userId)
    );
}
```

Usando objetos

Usar **eventos** para realizar tareas secundarias

3) Creamos un listener que realice la tarea asociada al evento

```php
final class SendEmail {
    // ...
    public function whenUserPasswordChanged( UserPasswordChanged $event): void {

        $this->mailer->sendPasswordChangedEmail($event->userId());

    }
    //..
}
```

Usando objetos

Command/Query en **objetos**

COMMAND OBJECT

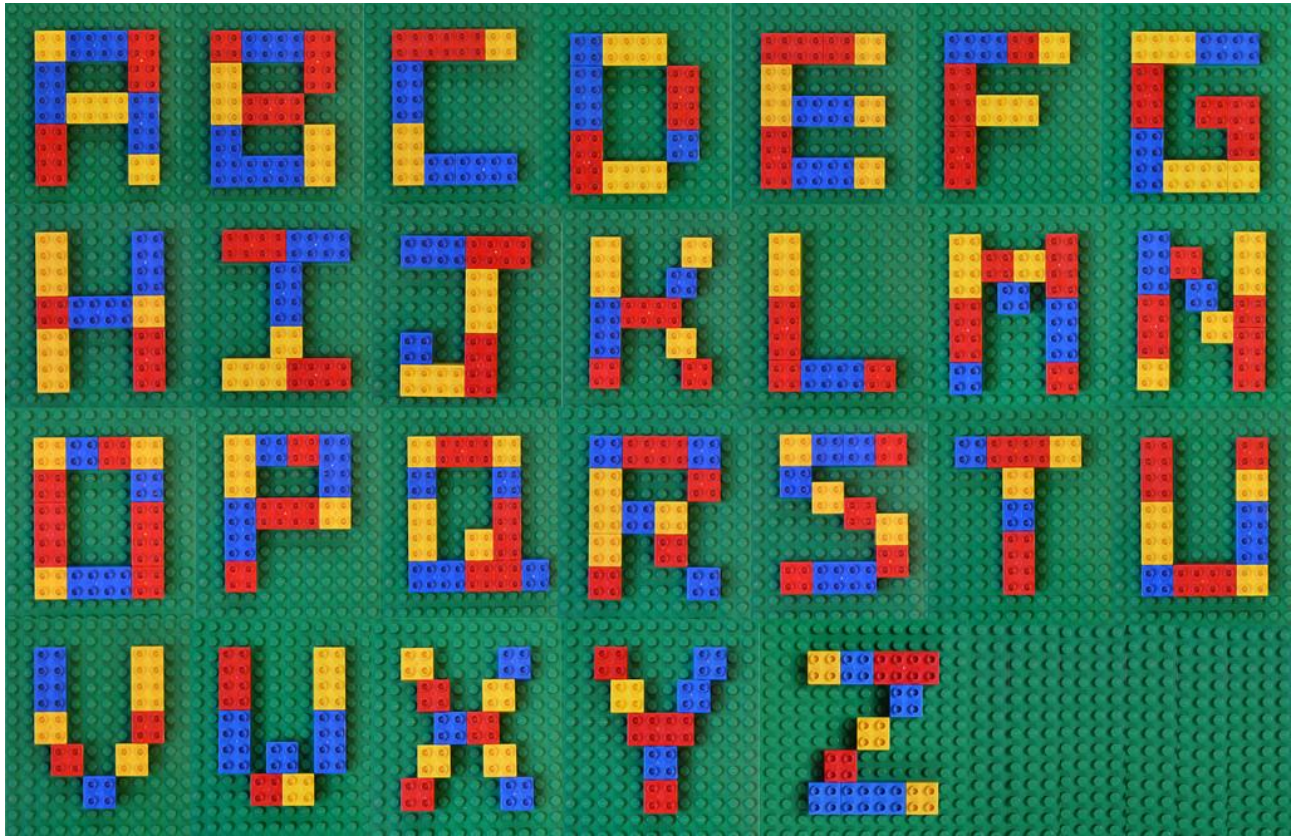
QUERY OBJECT

Command/Query Responsibility Segregation (CQRS)

s|ngular
it can be done.

Aspectos del código que mejoran y facilitan su legibilidad y comprensibilidad

Sobre los nombres



Sobre los nombres

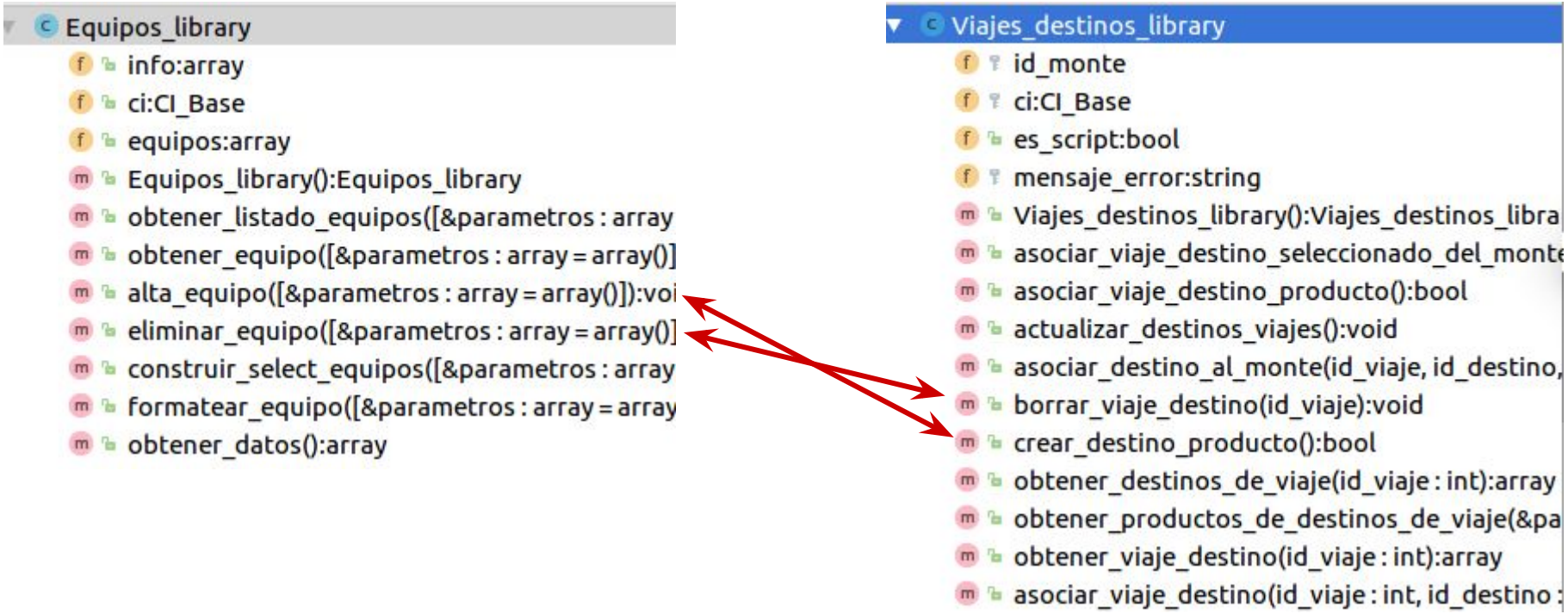
Evita **nombres** que no revelen la **intención**

```php
private function isAssoc(array $arr)
{
    if ([] === $arr) {
        return false;
    }

    return array_keys($arr) !== range( low: 0,  high: count($arr) - 1);
}
```

```php
function obtener_listado_productos(&$parametros = array())
{

    $this->info = array();

    $res = $this->ci->productos_model->obtener_listado_productos($parametros);

    if ($res->num_rows) {

        //...
        //....
    }
```

1) Sobre los nombres nombres

Evita usar varias palabras para un mismo concepto



1) Sobre los nombres

Una palabra por concepto usando **estándares** y **convenciones**

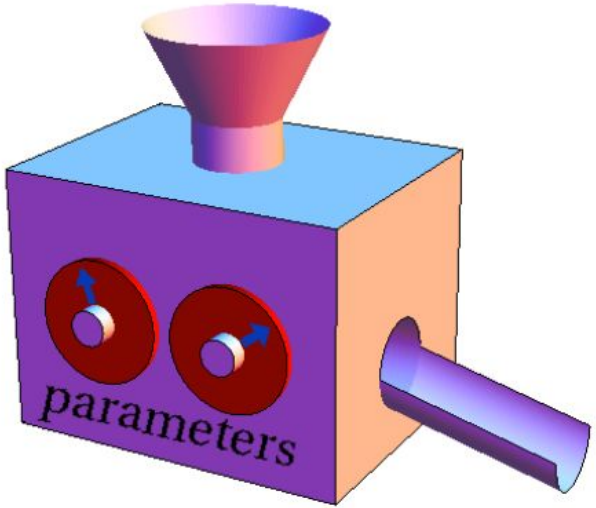
Nombres de variables

- begin / end
- locked / unlocked
- next / previous
- opened / closed
- up / down
- first / last
- min / max
- old / new
- source / targe

Nombre de funciones

- add / remove
- begin / end
- create / destroy
- first / last
- show / hide
- increment / decrement
- insert / delete
- lock / unlock
- get / set
- start / stop

Sobre los parámetros



Sobre los parámetros

Usa **operandos** y no opciones

```php
public function updateDeveloper ($name, $email,  $active = false){

    if ($active){
        $this->developer->save($name, $email);
    }
    else{
        $this->developer->save($name, '');
    }

}
```

Sobre los parámetros

Usa **operandos** y no opciones

```php
public function updateActiveDeveloper ($name, $email){
    $this->developer->save($name, $email);
}

public function updateUnactiveDeveloper ($name){
    $this->developer->save($name, '');
}
```

Sobre los parámetros

Objetos de dominio y **'data clumps'** (value objects)

```php
Class Gps
{
    private $latitude = '';
    private $longitude = '';

    public function __construct($latitude, $longitude)
    {

        $this->latitude = $latitude;
        $this->longitude = $longitude;
    }

    public function move ($latitude, $longitude){
        return new self($latitude, $longitude);
    }

}
```

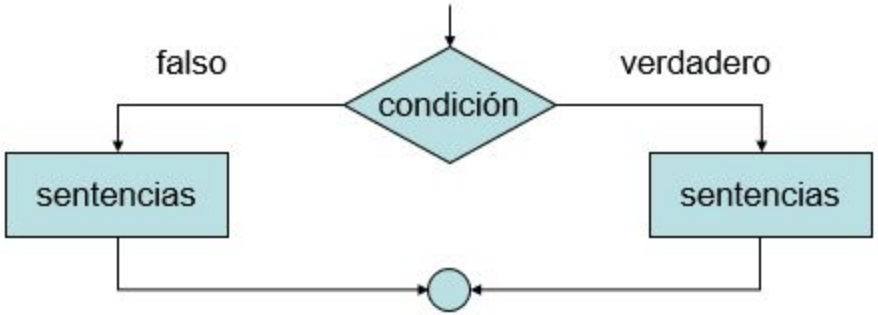
Sobre los parámetros

Objetos de dominio y **'data clumps'** (value objects)

```php
Class Position
{
    private $latitude = '';
    private $longitude = '';

    public function __construct(string $latitude, string $longitude)
    {
        $this->latitude = $latitude;
        $this->longitude = $longitude;
    }

    public function latitude():string
    {
        return $this->latitude;
    }

    public function longitude():string
    {
        return $this->longitude;
    }

    public function isNorth():bool
    {
        //...
    }

    //..
}
```

```php
Class Gps
{
    private $position;

    public function __construct(Position $position)
    {
        $this->position = $position;
    }

    public function move (Position $otherPosition): Gps{
        return new self($otherPosition);
    }

}
```

Sobre las condiciones



Sobre las condiciones

Utiliza **condicionales semánticos** fáciles de leer y comprender

```php
public function _invoke (){

    //....

    if ($foo == $var){
        //...
    }

    //...
}
```

```php
public function __invoke (){

    //....

    if ($this->hasLimit()){
        //...
    }

    //...
}


private function hasLimit(){
    return $this->foo == $this->var;
}
```

Sobre las condiciones

Utiliza **condicionales semánticos** fáciles de leer y comprender

```php
//...
//...
if ((($parametros['filtros']['id_cliente']) AND ($viaje['cliente_transporte']['id'] == $parametros['filtros']['id_cliente']))) {
    return true;
}
//...
//...
```

```php
//...
//...
$filtramosPorCliente = $parametros['filtros']['id_cliente'];
$clienteTieneTransporte = $viaje['cliente_transporte']['id'] == $parametros['filtros']['id_cliente'];

if ($filtramosPorCliente AND $clienteTieneTransporte) {
    return true;
}
//...
//...
```

Sobre las condiciones

Un solo nivel de indentación por método

```php
function register()
{
    if (!empty($_POST)) {
        $msg = '';
        if ($_POST['user_name']) {
            if ($_POST['user_password_new']) {
                if ($_POST['user_password_new'] === $_POST['user_password_repeat']) {
                    if (strlen($_POST['user_password_new']) > 5) {
                        if (strlen($_POST['user_name']) < 65 && strlen($_POST['user_name']) > 1) {
                            if (preg_match('/^[a-z\d]{2,64}$/i', $_POST['user_name'])) {
                                $user = read_user($_POST['user_name']);
                                if (!isset($user['user_name'])) {
                                    if ($_POST['user_email']) {
                                        if (strlen($_POST['user_email']) < 65) {
                                            if (filter_var($_POST['user_email'], FILTER_VALIDATE_EMAIL)) {
                                                create_user();
                                                $_SESSION['msg'] = 'You are now registered so please login';
                                                header('Location: ' . $_SERVER['PHP_SELF']);
                                                exit();
                                            } else $msg = 'You must provide a valid email address';
                                        } else $msg = 'Email must be less than 64 characters';
                                    } else $msg = 'Email cannot be empty';
                                } else $msg = 'Username already exists';
                            } else $msg = 'Username must be only a-z, A-Z, 0-9';
                        } else $msg = 'Username must be between 2 and 64 characters';
                    } else $msg = 'Password must be at least 6 characters';
                } else $msg = 'Passwords do not match';
            } else $msg = 'Empty Password';
        } else $msg = 'Empty Username';
        $_SESSION['msg'] = $msg;
    }
    return register_form();
}
```

Sobre las condiciones

Un solo nivel de indentación por método

```
function _obtener_url_categoria() {
    $cat_listado=$this->categoria_library->obtener_subtipo_seleccionado();
    if (!$cat_listado) {
        $cat_listado=$this->categoria_library->obtener_tipo_seleccionado();
        if (!$cat_listado) {
            $cat_listado=$this->categoria_library->obtener_subcategoria_seleccionada();
            if (!$cat_listado) {
                $cat_listado=$this->categoria_library->obtener_categoria_seleccionada();
                return $cat_listado['url'];
            } else {
                return $cat_listado['url'];
            }
        } else {
            return $cat_listado['url'];
        }
    } else {
        return $cat_listado['url'];
    }
}
```

Sobre las condiciones

Un solo nivel de indentación por método: **cláusulas de guarda**

```
function _obtener_url_categoria()
{
    if ($categoria = $this->categoria_library->obtener_subtipo_seleccionado())
        return $categoria['url'];
    }

    if ($categoria = $this->categoria_library->obtener_tipo_seleccionado()) {
        return $categoria['url'];
    }

    if ($categoria = $this->categoria_library->obtener_subcategoria_seleccionada()) {
        return $categoria['url'];
    }

    if ($categoria = $this->categoria_library->obtener_subcategoria_seleccionada()) {
        return $categoria['url'];
    }

    $categoria = $this->categoria_library->obtener_categoria_seleccionada();
    return $categoria['url'];
}
```

Sobre las condiciones

Un solo nivel de indentación por método

```
function do_login()
{
    if ( ! $this->dx_auth->is_logged_in()) {

        $val = $this->form_validation;

        if ($val->run() AND $this->dx_auth->login($val->set_value('username'), $val->set_value('password'), $val->set_value('remember'))) {

            $url_redireccion = $this->input->post('url');

            if ($url_redireccion == '') {
                $url_redireccion = '/';
            }

            redirect($url_redireccion, method: 'location');
        } else {

            if ($this->dx_auth->is_banned()) {
                $this->session->set_flashdata('error_login', "El usuario ".$val->set_value('username')." ha borrado su cuenta o está baneado.");

                return false;
            } else {
                $this->session->set_flashdata('error_login', "Nombre de usuario o contraseña incorrectos.");

                return false;
            }
        }
    } else {
        redirect( uri: '/', method: 'location');
    }
}
```

Sobre las condiciones

Un solo nivel de indentación por método

```php
function do_login()
{
    if ($this->usuarioEstaLogueado()) {
        $this->redireccionarA( url: '/');
    }

    if ($this->usuarioHaceLogin()) {
        $this->redireccionarA($this->urlRedireccion());
    }

    $this->configurarMensajeErrorLogin();

    return false;

}

private function redireccionarA(string $url){
    redirect($url, method: 'location');
}
```

```php
private function usuarioEstaLogueado(){
    return $this->dx_auth->is_logged_in();
}

private function usuarioHaceLogin (){

    $val = $this->form_validation;

    return ($val->run() AND $this->dx_auth->login($val->set_value('username'), $val->set_value('passwo
}

private function urlRedireccion (){
    $url_redireccion = $this->input->post('url');

    if ($url_redireccion == '') {
        $url_redireccion = '/';
    }

    return $url_redireccion;
}

private function configurarMensajeErrorLogin(){

    if ($this->usuarioEstaBaneado()) {
        $this->session->set_flashdata('error_login', "El usuario ".$val->set_value('username')." ha bo
    }
    else{
        $this->session->set_flashdata('error_login', "Nombre de usuario o contraseña incorrectos.");
    }
}

private function usuarioEstaBaneado(){
    return $this->dx_auth->is_banned()
}
```

Sobre las funciones

Aprovecha para contar una **historia** en cada **método público**...

```php
class StoreResourceHandler
{
    private $command;
    private $resource;
    //..
    public function __invoke(StoreResourceCommand $command)
    {

        $this->initializeCommand($command);

        $this->fillResource();

        $this->fillSource();

        $this->create();

        $this->attachAuthorTags();

        $this->attachEventTags();

        $this->attachTechnologyTags();

        $this->attachResourceTags();

    }
```

Sobre las funciones

...pero no cuentes historias de miedo

```php
function obtener_listado_montes(6$parametros = array())
{
    $lista = array();

    if ( ! isset($parametros['obtener_select_montes'])) {
        $parametros['obtener_select_montes'] = false;
    }

    $parametros['tipos_monte'] = $this->ci->input->post('tipos_monte');
    $parametros['energeticos'] = $this->ci->input->post('energeticos');
    $parametros['finalizados'] = $this->ci->input->post('finalizados');

    $res = $this->ci->montes_model->obtener_listado_montes($parametros);

    if ($res->num_rows) {

        $res = $res->result_array();

        foreach ($res AS $monte) {
            $this->formatear_datos_monte($monte);

            if ($monte['id_monte_gerencia']) {
                $monte['monte_gerencia'] = $this->ci->monte_gerencia_model->obtener_uno($monte['id_monte_gerencia']);
            }
            if (isset($parametros['obtener_tipo_monte'])) {
                $monte['tipo_monte'] = $this->obtener_tipo_monte($monte['id_monte']);
                $monte['es_proveedor'] = (isset($monte['tipo_monte'][ID_MONTE_PROVEEDOR])) ? true : false;
                $monte['es_parque'] = (isset($monte['tipo_monte'][ID_MONTE_TIPO_PARQUE])) ? true : false;
            }

            $monte['proveedores'] = $this->ci->montes_proveedores_library->obtener_proveedores_de_un_monte($monte['id_monte']);

            $lista[$monte['id_monte']] = $monte;
        }
    }

    if ($parametros['energeticos']) {
        foreach ($lista AS $id => $monte) {
            $lista[$id]['info_cultivo'] = $this->obtener_cultivo_energetico($monte['id_monte']);
            $lista[$id]['info_cultivo']['toneladas_extraidas'] = $this->ci->viajes_model->return_peso_viajes_energeticos($lista[$id]['info_cultivo']['id_cultivo']);
        }
    }

    $parametros['listado_montes'] = $lista;
    $this->info = $lista;

    if ($parametros['obtener_select_montes']) {
        $this->construir_select_montes($parametros);
    }

    return $this->info;
}
```

Sobre las funciones

Pequeñas, menos es más

```php
public function __invoke(UpdateSubscriptionAccessCommand $command)
{
    $this->initialize($command);

    $this->configureStatus();

    $this->obtainSubscription();

    $this->checkCompanyHasActiveSubscription();

    $this->obtainCompany();

    $this->updateStatusAccessKey();

}

private function initialize(UpdateSubscriptionAccessCommand $command)
{
    $this->command = $command;
}

private function configureStatus()
{
    $this->status = $this->accessKeyStatusFactory->buildStatusFromString($this->command->status());
}

private function obtainSubscription()
{
    $this->subscription = Subscription::findOrFail($this->command->subscriptionId());
}

private function checkCompanyHasActiveSubscription()
{
    if (! $this->subscription->isActive(Carbon::now())) {
        throw new SubscriptionIsNotActiveException();
    }
}

private function obtainCompany()
{
    $this->company = $this->subscription->company;
}
```

Sobre los principios

Evita las sorpresas: **Principio de la Mínima Sorpresa**

```php
private function associateUser(CreateCompanyCommand $command)
{
    $this->user = $this->company->users()->create([
        'username' => $command->getUsername(),
        'email' => $command->getUserEmail(),
        'name' => $command->getName(),
        'surname_first' => $command->getSurnameFirst(),
        'surname_second' => $command->getSurnameSecond(),
        'gender' => $command->getGender(),
        'password' => $command->getPassword(),
        'old_id' => null,
    ]);

    $this->user->assignRole(Role::ROLE_COMPANY);

    activity()->causedBy($this->user)->log(trans( key: 'activity.user_register'));

    $this->configUserVerification();

    event(new Registered($this->user));

}
```

Sobre los principios

Evita las sorpresas: **Principio de la Mínima Sorpresa**

```php
public function __invoke(CreateCompanyCommand $command)
{
    $this->createUser($command);

    $this->assignRole();

    $this->logActivity();

    $this->configUserVerification();

    $this->throwEvent();
}

private function createUser(CreateCompanyCommand $command)
{
    $this->user = $this->company->users()->create([
        'username' => $command->getUsername(),
        'email' => $command->getUserEmail(),
        'name' => $command->getName(),
        'surname_first' => $command->getSurnameFirst(),
        'surname_second' => $command->getSurnameSecond(),
        'gender' => $command->getGender(),
        'password' => $command->getPassword(),
        'old_id' => null,
        'uuid' => Uuid::generate()->string,
    ]);
}

private function assignRole (){
    $this->user->assignRole(Role::ROLE_COMPANY);
}

private function logActivity (){
    activity()->causedBy($this->user)->log(trans( key: 'activity.user_register'));
}


private function throwEvent (){
    event(new Registered($this->user));
}
```

Sobre los principios

Evita las sorpresas: **Principio de la Mínima Sorpresa**

```php
public function addFakeRegisterInFiles($slrFilePath, $fileName, $fakeRegisters) {
    shell_exec( cmd: 'rm -rf  '.$this->getPath().$fileName.'*');
    shell_exec( cmd: 'mkdir '.$this->getPath().$fileName);
    shell_exec( cmd: 'cp '.$this->getPath().'file[1-5].txt '.$this->getPath().$fileName);

    $file = 1;
    foreach ($fakeRegisters as $fake) {
        $line = $this->generateLineFakeRegister($fake).$this->addBreakLine();
        File::append($this->getPath().$fileName.'/file'.$file.'.txt', $line);
        if(file_exists ( filename: $this->getPath().$fileName.'/file'.($file+1).'.txt' )) {
            $file++;
        }
    }

    shell_exec( cmd: 'cat '.$this->getPath().$fileName.'/file[1-5].txt >> '.$slrFilePath);
    shell_exec( cmd: 'rm -rf  '.$this->getPath().$fileName.'/*');
}
```

Sobre los principios

Evita las sorpresas: **Principio de la Mínima Sorpresa**

```php
public function addFakeRegisterInFiles($slrFilePath, $fileName, $fakeRegisters) {

    $this->clearFolder($fileName);

    $this->createFolder($filename);

    $this->copyFiles($filename);

    $this->addFakeRegisterInFiles($fileName, $fakeRegisters);

    $this->writeFile($filename);

    $this->clearFolder($fileName);
}

private function clearFolder (string $filename){
    return shell_exec( cmd: 'rm -rf  '.$this->getPath().$fileName.'*');
}

private function createFolder (string $filename){
    return shell_exec( cmd: 'mkdir '.$this->getPath().$fileName);
}

private function copyFiles (string $filename){
    return shell_exec( cmd: 'cp '.$this->getPath().'file[1-5].txt '.$this->getPath().$fileName);
}

private function addFakeRegisters ($filename, $fakeRegisters){

    $file = 1;
    foreach ($fakeRegisters as $fake) {
        $line = $this->generateLineFakeRegister($fake).$this->addBreakLine();
        File::append($this->getPath().$fileName.'/file'.$file.'.txt', $line);
        if(file_exists ( filename: $this->getPath().$fileName.'/file'.($file+1).'.txt' )) {
            $file++;
        }
    }
}

private function writeFile (string $filename, string $slrFilePath){
    return shell_exec( cmd: 'cat '.$this->getPath().$fileName.'/file[1-5].txt >> '.$slrFilePath);
}
```

Sobre los principios

Evita las sorpresas: **Principio de la Mínima Sorpresa**

```php
interface Filesystem {

    public function clearFolder (string $path);
    public function cleateFolder (string $path);
    public function writeFile (string $path, string $filename);
    //..
}
```

```php
class FilesystemGateway implements Filesystem{
    public function clearFolder($path)
    {
        //...local
    }

    public function createFolder($path)
    {
        //...local
    }

    public function writeFile($path, $name)
    {
        //...local
    }
}

class S3Gateway implements Filesystem{
    public function clearFolder($path)
    {
        //...Amazon
    }

    public function createFolder($path)
    {
        //...Amazon
    }

    public function writeFile($path, $name)
    {
        //...Amazon
    }
}
```

Sobre los principios

Evita las sorpresas: **Principio de la Mínima Sorpresa**

```php
class Foo {

    private $filesystem;

    public function __construct(Filesystem $filesystem)
    {
        $this->filesystem = $filesystem;
    }

    public function addFakeRegisterInFiles($slrFilePath, $fileName, $fakeRegisters) {

        $this->clearFolder($fileName);

        $this->createFolder($filename);

        $this->copyFiles($filename);

        $this->addFakeRegisterInFiles($fileName, $fakeRegisters);

        $this->writeFile($filename);

        $this->clearFolder($fileName);
    }

    private function crearFolder (string $path){
        $this->filesystem->clearFolder($path);
    }
    //..
}
```

Sobre los comentarios

El **código** es el único que siempre dice la **verdad**

```php
/**
 * @return mixed
 */
public function getCreatedAt()
{

    if (empty($this->createdAt)) {
        return Carbon::now();
    }

    return $this->createdAt;
}
```

```php
/*
 * Construir un select con el listado de matriculas de un viaje
 *    Entrada
 *        $parametros['listado_matriculas']
 *        $parametros['nombre_options']
 */

function es_post_viaje_conductor()
{

    if ($_POST['viaje_conductor']) {
        return true;
    }

    return false;
}
```

Sobre los comentarios

El **código** es el único que siempre dice la **verdad**

```php
foreach ($res AS $producto) {

    // Obtener viajes desde este monte-producto para ver las toneladas que se llevan extraídas.
    $data['id_monte_producto'] = $producto['id_registro'];
    $producto['tn_extraidas'] = $this->ci->viajes_model->obtener_toneladas_extraidas($producto['id_registro']);

    // Formatear datos del monte-producto
    $this->formatear_datos_producto_monte($producto);

    // Obtener datos del producto origen
    $producto['producto_origen'] = array();
    if ($producto['id_producto']) {
        $producto['producto_origen'] = $this->ci->productos_library->obtener_producto($producto);
    }

    // Obtener datos del destino
    if ($producto['id_destino_producto']) {
        $datos = $this->ci->destinos_library->obtener_datos_destino_producto($producto);
        $producto['datos_destino'] = $datos['datos_destino'];
        $producto['producto_destino'] = $datos['producto_destino'];
    }
```

Sobre los comentarios

El **código** es el único que siempre dice la **verdad**

```php
foreach ($res AS $producto) {
    $this->obtenerViajes();
    $this->obtenerMonte();
    $this->obtenerProducto();
    $this->obtenerDestino();
```

Simetría

El código debe parecerse en cualquier parte del proyecto

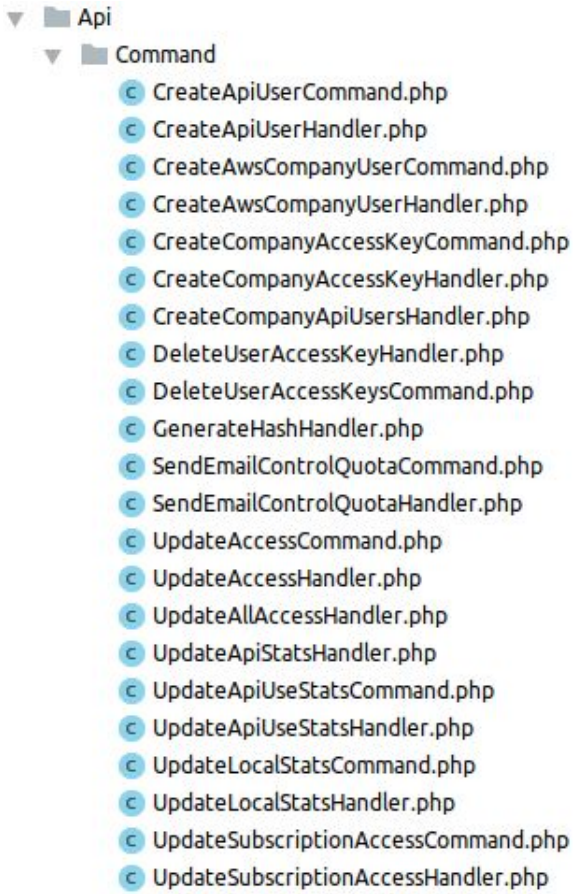
Simetría sintáctica

Simetría semántica

Simetría sistémica

Simetría

Simetría **sistémica**



Simetría

Simetría **sistémica**

```php
public function __invoke(UpdateApiUseStatsCommand $command)
{
    $this->initialize($command);
    $this->checkDayHasNotBeenProccessed();
    $this->downloadLogs();
    $this->processLogs();
    $this->beginTransaction();
    $this->resetStatsByDay();
    $this->writeStatsInLocal();
    $this->endTransaction();
    $this->writeStatsInApi();
    $this->markProcessAsSuccessful();
}
```

```php
public function __invoke(CreateAwsCompanyUserCommand $command)
{
    $this->initialize($command);
    $this->generateUsername();
    $this->checkUsernameExists();
    $this->createApiUserInAws();
    $this->generateApiUserInSlr();
    $this->attachToGroup();
}
```

```php
public function __invoke(UpdateSubscriptionAccessCommand $command)
{
    $this->initialize($command);
    $this->configureStatus();
    $this->obtainSubscription();
    $this->checkCompanyHasActiveSubscription();
    $this->obtainCompany();
    $this->updateStatusAccessKey();
}
```

Simetría

Simetría en métodos

```
[scope] function methodName(type name, ...): void|[return-type]
{
    [pre-conditions checks]

    [failure scenarios]

    [happy path]

    [post-condition checks]

    [return void|specific-return-type]
}
```

https://leanpub.com/object-design

Simetría

Simetría en métodos: **Pre-conditions check**

- validamos todos los parámetros
- generan excepciones *InvalidArgumentException* cuando algo va mal
- podemos usar aserciones
- argumentos como *value objects* o *entities* ahorran validaciones

```
if ([alguna pre-condición no se cumple]) {
    throw new InvalidArgumentException(...);
}
```

```
public function foo (int $value){

    Assertion::inArray($value, [...]);
    //...
}
```

Simetría

Simetría en métodos: **Pre-conditions check**

```php
// before:
public function sendConfirmationEmail(string $emailAddress): void
{
    Assertion::email($emailAddress);
    // ...
}

// after:
final class EmailAddress
{
    private string $emailAddress;

    public function __construct(string $emailAddress)
    {
        $this->emailAddress = $emailAddress;
    }
}

public function sendConfirmationEmail(
    EmailAddress $emailAddress
): void {
    // no need to validate $emailAddress anymore
}
```

Simetría

Simetría en métodos: **failure scenarios**

- se corresponden con errores funcionales normalmente generados por condiciones externas
- generan excepciones de tipo *runtime*

```php
public function getRowById(int $id): array
{
    /**
     * Esto podría lanzar una excepción InvalidArgumentException
     */
    Assertion::greaterThan($id, 0, 'ID should be greater than 0');

    $record = $this->db->find($id);
    /**
     * Esto es nuestro 'failure scenario':
     * no encontramos el registro así que RuntimeException
     */
    if ($record === null) {
        throw new RuntimeException(sprintf('Could not find record with ID "%d"',$id);
    }

    return $record;
}
```

Simetría

Simetría en métodos: **happy path**

https://leanpub.com/object-design

Simetría

Simetría en métodos: *post-condition checks*

- comprobaciones y validaciones antes de hacer el *return*

```php
public function someCrazyComplicatedCalculation(): int
{
    // ...
    $result = ...;

    /**
     * Podríamos comprobar el resultado antes de devolverlo
     * */

    Assertion::greaterThan(0, $result);

    return $result;
}
```

Simetría

Simetría en métodos: *return*

- se realiza el *return* del método
- solo los métodos *query* deberían devolver algo
- mejor devolver tipos de datos que primitivas
- devolver únicamente un tipo de dato específico por método

De tu parte queda...

Entrenar

De tu parte queda...

Entrenar

Concentrarse

De tu parte queda...

Entrenar

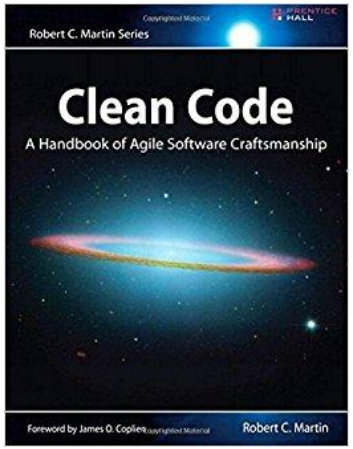
Concentrarse

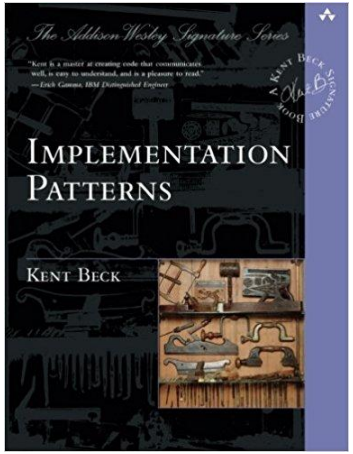
Ser consciente

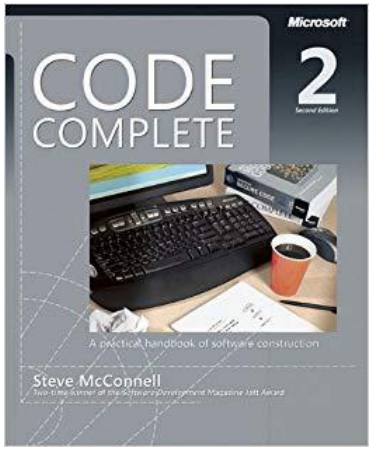
De tu parte queda...

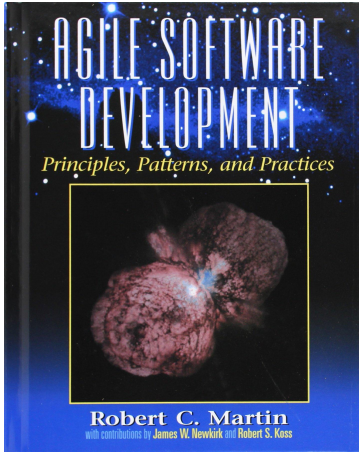
Entrenar
Concentrarse
Ser consciente
Ser responsable

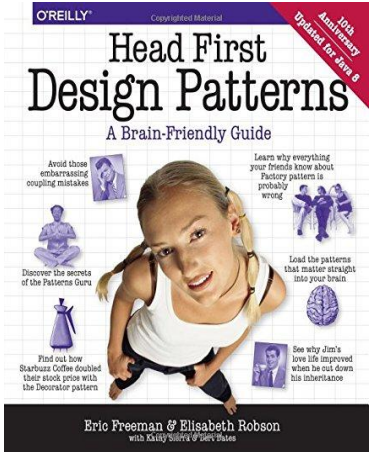
Bibliografía recomendada



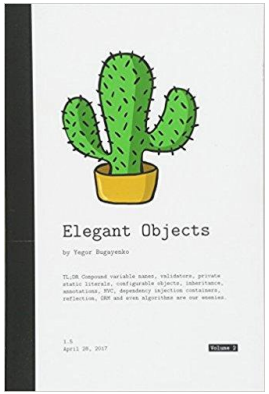


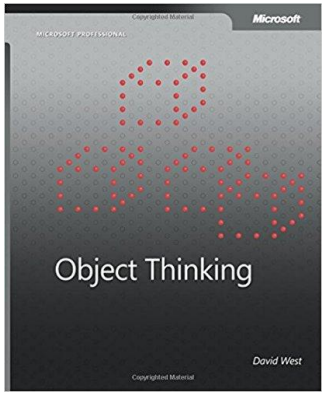


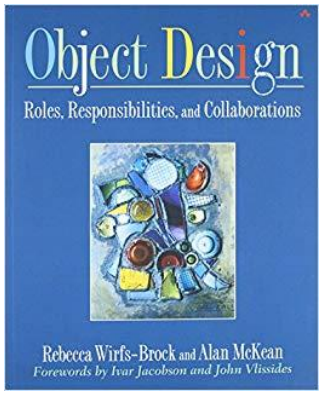




Bibliografía recomendada ♥







Ahora te toca a ti

¿preguntas? ¿comentarios? ¿sugerencias?

Fin

Gracias por vuestra atención.