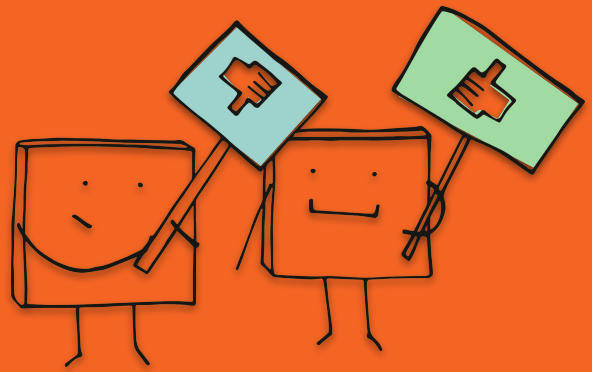
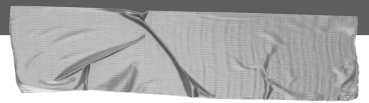
Validación de solicitudes HTTP: reflexión



Santiago Fonzo
Instituto Superior Zona Oeste - ISPI N° 9045
Programación II
Sebastián Bruselario
16 de septiembre de 2025

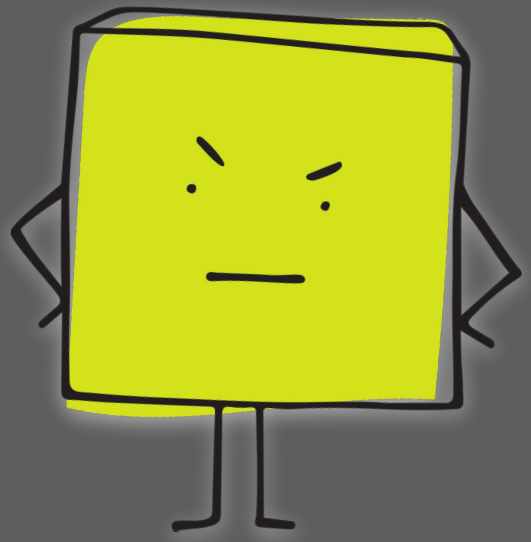
Solicitudes HTTP con Body

- POST - PUT - PATCH
- Estructuras de datos variable
- Tipos de datos variables



```
{
  “nombre”:  “Santiago”,
  “edad”:  22,
  “curso”: {
        “nombre”: “programación II”,
        “profesor”: “S. Bruselario”
  },
  “hobbies”: [
        “tocar la batería”,
        “ver series”
  ]
}
```

400 - Bad Request



API - Endpoints

- Información con una estructura y tipos de datos particulares
- Validación temprana
  - Control
  - Evitar procesamiento innecesario
  - Eficiencia de recursos

Tiene valor?

```php
<?php

$body = [
        “name” => ”Santiago”
] ;

var_dump(isset($body[“name”]));
// bool(true)

var_dump(isset($body[“city”]));
// bool(false)
```

1. isset

Determina si una o más variables están declaradas y son diferente de null.

```
isset(mixed $var, mixed ...$vars): bool
```

➔ **Simple**
Verificación del valor de una variable en una sola llamada.

➔ **Limitado**
Solo determina si hay o no un valor.

➔ **Poco escalable**
Si la estructura es compleja se tienen condiciones complejas.

Tiene valor y tipo correcto?

```php
<?php

$body = [“name” => ”Santiago”] ;

var_dump(is_string($body[“name”])); // bool(true)

var_dump(is_int($body[“name”])); // bool(false)

var_dump(is_string($body[“city”])); // bool(false)
```

2. is_xtype

Determina si una variable es del tipo *xtype*.

```
is_int(mixed $value): bool
```

➔ **Simple**
Verificación del valor y validación del tipo de dato de una variable en única llamada.

➔ **Sustituye a isset()**
Puede manejar valores nulos. Pero genera una *Warning*.

➔ **Poco escalable**
Si la estructura es compleja se tienen condiciones complejas.

Tiene valor, tipo y formato correcto?

```php
<?php

$body = [“email” => ”Santiago”] ;

var_dump(filter_var(
        $body[“email”],
        FILTER_VALIDATE_EMAIL)
); // bool(false)

var_dump(is_int($body[“name”])); // bool(false)
```

3. filter_var

Filtra una variable con el filtro que se indique.

```
filter_var(mixed $value, int $filter =
FILTER_DEFAULT, array|int $options = 0):
mixed
```

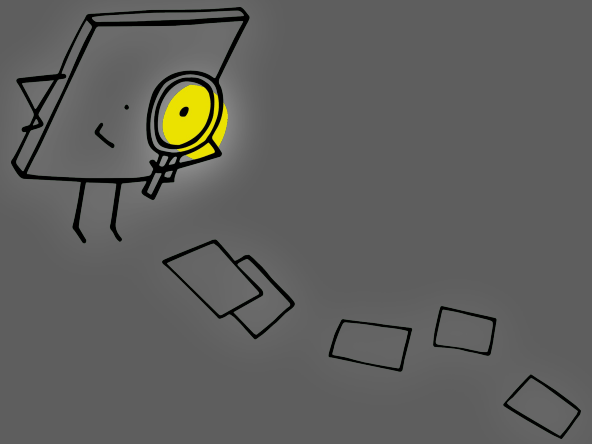
➔ **Práctico**
Muchos filtros están definidos por defecto.

➔ **Versátil**
Puede validar valor, tipo, formato y personalizarse con un callback.

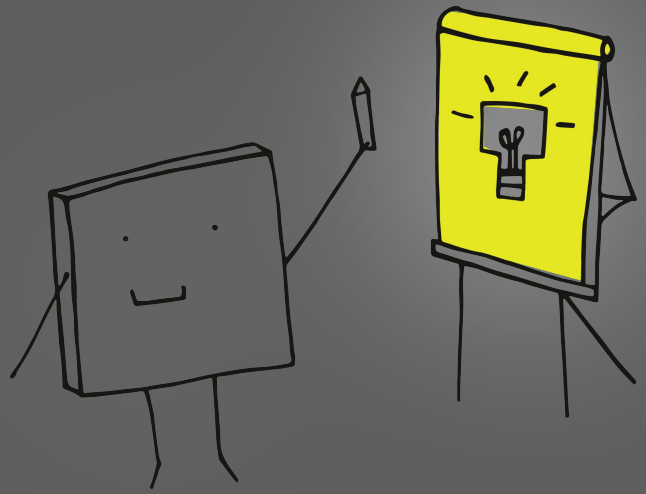
➔ **Poco escalable**
Verifico un formato (por ejemplo de email) en una sola llamada.

¿Entonces?

- Cada endpoint con su propia validación
  - Errores
  - Poco mantenible
- ¿Abstraer a una única función?
  - Array como parámetro
- ¿Cómo puedo hacerlo agnóstico para que sea reutilizable?



Reflexión y atributos

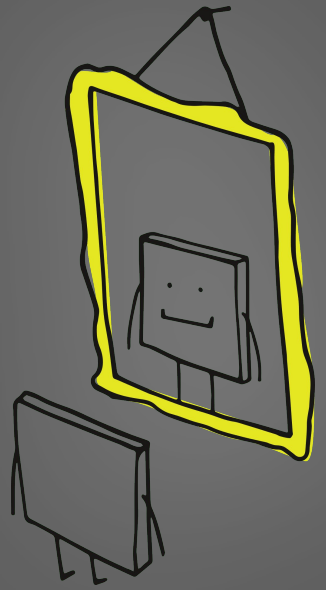


Reflexión y atributos

- Reflexión
  - PHP 5.0
  - Obtener información de una clase en tiempo de ejecución
- Atributos
  - PHP 8.0
  - Metadata: datos sobre los datos

Reflexión

- $reflectionClassInstance = new ReflectionClass($class);
- $props = $reflectionClassInstance->getProperties($filter);
- foreach ($props as $property)
  - ->getAttributes();
  - ->getName();
  - ->getType;



Atributos

- Metadata
- Son clases con la metadata *Attribute*
  - Propiedades
  - Métodos

Middleware

- Centraliza procesamiento
- Se ejecuta antes que el procesamiento del endpoint
- Recibe una clase con metadata que modela el cuerpo de la solicitud esperada

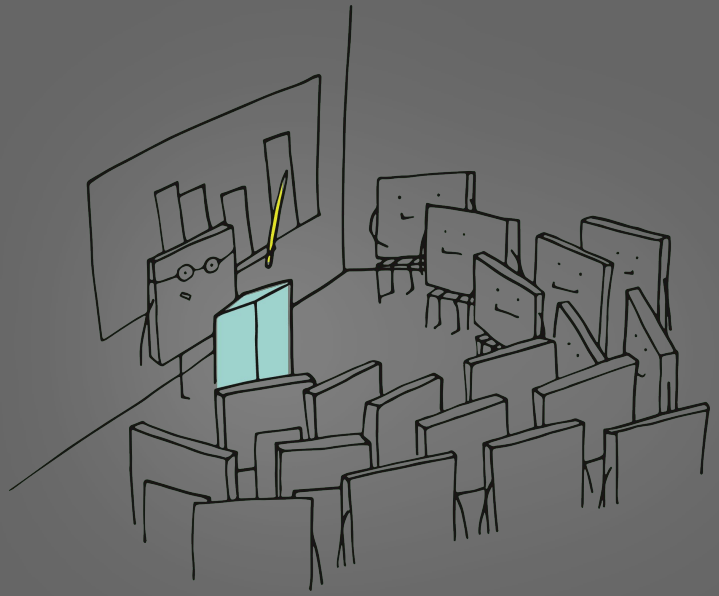
Resumiendo

- Reusable
- Lógica centralizada
- Fácilmente extensible

- La reflexión cuesta*
- Implementación compleja

* Cache

Implementación



Gracias.