Et si on repensait les ORMs ?

→ Baptiste Langlade

→ Architecte chez Efalia

→ Lyon

→ ~95 packages Open Source

→ 10+ ans XP

Domain Driven Design

Une `Voiture` a une `CarteGrise`

```php
class Voiture
{
    public function __construct(
        private Id $id,
        private CarteGrise $carteGrise,
    ) {}
}

class CarteGrise
{
    public function __construct(
        private string $immatriculation,
        private string $proprietaire,
        private string $adresse,
    ) {}
}
```

```php
class Voiture
{
    private int $id;

    public function __construct(
        private CarteGrise $carteGrise,
    ) {}
}

class CarteGrise
{
    private int $id;

    public function __construct(
        private string $immatriculation,
        private string $proprietaire,
        private string $adresse,
    ) {}
}
```

```php
class Voiture
{
    private int $id;

    public function __construct(
        private CarteGrise $carteGrise,
    ) {}
}

class CarteGrise
{
    private int $id;

    public function __construct(
        private string $immatriculation,
        private string $proprietaire,
        private string $adresse,
    ) {}
}
```

```php
class Voiture
{
    private int $id;
    private CarteGrise $carteGrise;

    public function __construct(
        string $immatriculation,
        string $proprietaire,
        string $adresse,
    ) {
        $this->carteGrise = new CarteGrise($this, $immatriculation, $proprietaire, $adresse);
    }
}

class CarteGrise
{
    private int $id;

    public function __construct(
        Voiture $voiture,
        private string $immatriculation,
        private string $proprietaire,
        private string $adresse,
    ) {}
}
```

```php
use Doctrine\ORM\EntityManagerInterface;

function (EntityManagerInterface $manager) {
    $entities = $manager
        ->getRepository(Voiture::class)
        ->findAll();
}
```

```php
use Doctrine\ORM\EntityManagerInterface;

function (EntityManagerInterface $manager) {
    $repository = $manager->getRepository(Voiture::class);
    $count = $repository->count();

    for ($offset = 0; $offset < $count; $offset += 100) {
        $entities = $repository->findBy(
            limit: 100,
            offset: $offset,
        );
    }
}
```

```php
use Doctrine\ORM\EntityManagerInterface;

function (EntityManagerInterface $manager) {
    $repository = $manager->getRepository(Voiture::class);
    $count = $repository->count();

    for ($offset = 0; $offset < $count; $offset += 100) {
        $manager->clear();
        $entities = $repository->findBy(
            limit: 100,
            offset: $offset,
        );
    }
}
```

Arrive Formal !

```
composer require formal/orm
```

```php
use Formal\ORM\Id;

final readonly class Voiture
{
    /** @param Id<self> $id */
    public function __construct(
        private Id $id,
        private CarteGrise $carteGrise,
    ) {}
}

final readonly class CarteGrise
{
    public function __construct(
        private string $immatriculation,
        private string $proprietaire,
        private string $adresse,
    ) {}
}
```

```php
$carteGrise = new CarteGrise('aa-123-bb', 'John Doe', 'Somewhereville');
$voiture1 = new Voiture(
    Id::new(Voiture::class),
    $carteGrise,
);
$voiture2 = new Voiture(
    Id::new(Voiture::class),
    $carteGrise,
);
```

```php
$repository = $manager->repository(Voiture::class);
$manager->transactional(
    static function() use ($repository) {
        $voiture1 = ...;
        $voiture2 = ...;
        $repository->put($voiture1);
        $repository->put($voiture2);

        return Either::right(new SideEffect);
    },
);
```

```php
$repository = $manager->repository(Voiture::class);
$manager->transactional(
    static function() use ($repository) {
        $voiture = ...;
        $voiture = $voiture->changerAdresse('nouvelle adresse');
        $repository->put($voiture);

        return Either::right(new SideEffect);
    },
);
```

```php
$manager
    ->repository(Voiture::class)
    ->all()
    ->foreach(static fn(Voiture $voiture) => doSomething($voiture));
```

```php
$manager
    ->repository(Voiture::class)
    ->all()
    ->drop(1_000)
    ->take(100)
    ->foreach(static fn(Voiture $voiture) => doSomething($voiture));
```

Sureté

No SQL

```php
use Formal\ORM\Specification\Entity;
use Innmind\Specification\Property;
use Innmind\Specification\Sign;

$manager
    ->repository(Voiture::class)
    ->matching(
        Entity::of('carteGrise', Property::of(
            'immatriculation',
            Sign::equality,
            'aa-123-bb',
        )),
    )
    ->foreach(static fn(Voiture $voiture) => doSomething($voiture));
```

Stockage

→ SQL (Mysql, MariaDB et PostgreSQL)

→ SQL (Mysql, MariaDB et PostgreSQL)

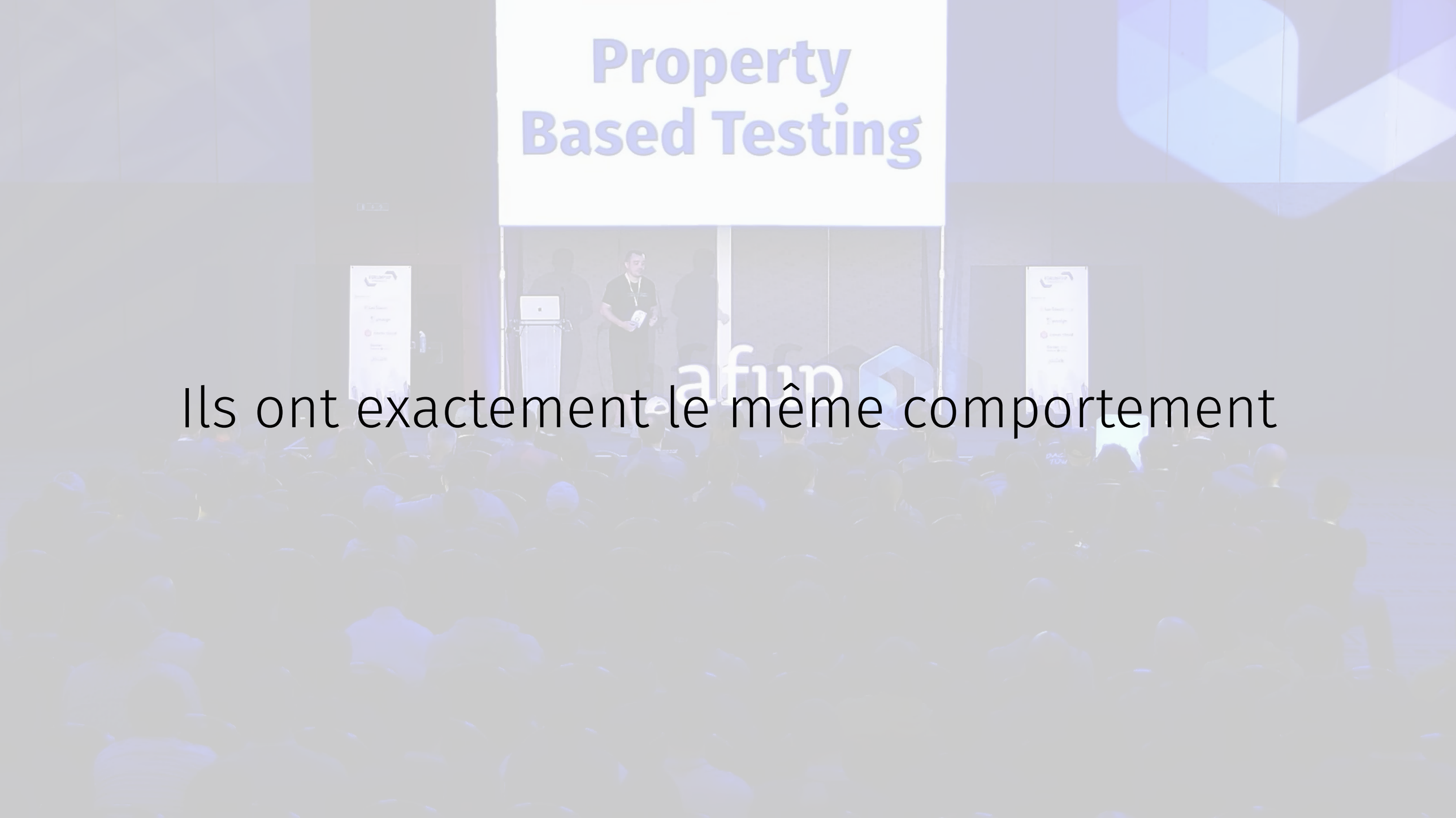
→ Filesystem

→ concret

- → SQL (Mysql, MariaDB et PostgreSQL)
- → Filesystem
  - → concret
  - → S3

- → SQL (Mysql, MariaDB et PostgreSQL)
- → Filesystem
  - → concret
  - → S3
  - → en mémoire

- → SQL (Mysql, MariaDB et PostgreSQL)
- → Filesystem
  - → concret
  - → en mémoire
  - → S3
- → Elasticsearch



Ils ont exactement le même comportement

Ecosystème Innmind

→ Génération de fichier

→ Body requête/réponse HTTP

→ Input de processus

→ Envoi de messages AMQP

→ Asynchrone

Performance

~40% plus rapide que Doctrine

Welcome to the Formal ORM

Et plus

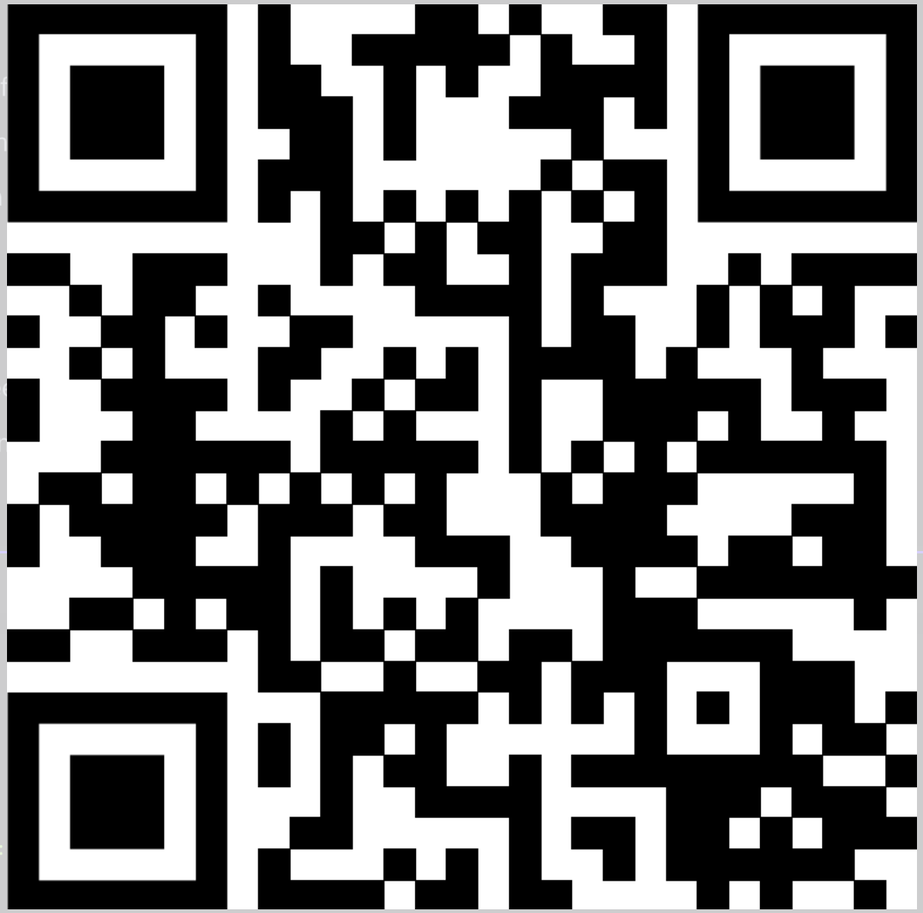
This ORM focuses on simplifying data manipulation.

This is achieved by:

- using immutable objects
- each aggregate *owning* the objects it ref
- using monads to fetch aggregates (from
- using the specification pattern to match

This allows:

- simpler app design (as it can be pure)
- memory efficiency (the ORM doesn't ke
- long living processes (since there is no
- to work asynchronously

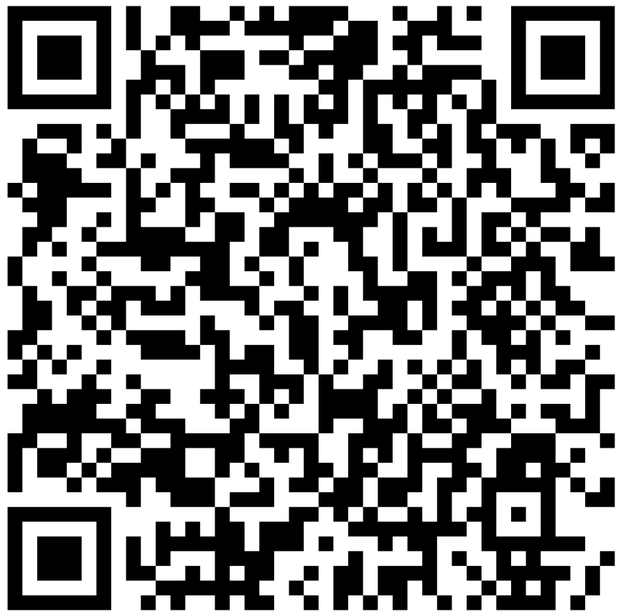


Sneak peak

```php
use Formal\ORM\{
    Manager,
    Sort,
};
use Formal\AccessLayer\Connection\PDO;
use Innmind\Url\Url;

$manager = Manager::sql(
    PDO::of(Url::of('mysql://user:pwd@host:
);
$_ = $manager
    ->repository(YourAggregate::class)
    ->all()
    ->sort('someProperty', Sort::asc)
    ->drop(150)
    ->take(50)
```

Questions



baptiste_forum20.24

Twitter @Baptouuuu

Github @Baptouuuu/talks