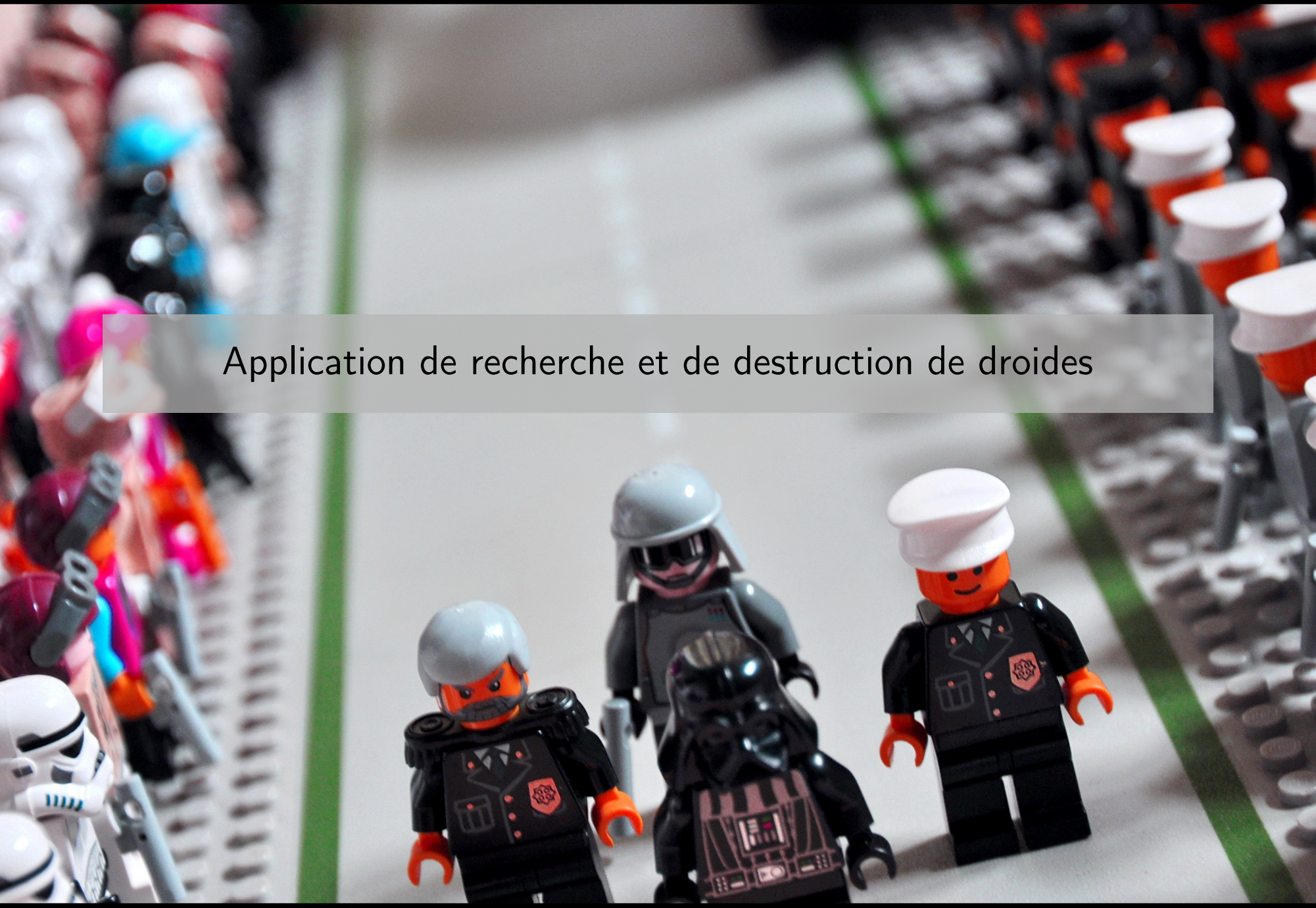
L'injection de dépendance ou comment découpler ses objets

Thibaud Dauce
thibaud@dauce.fr – @ThibaudDauce

26 novembre 2015



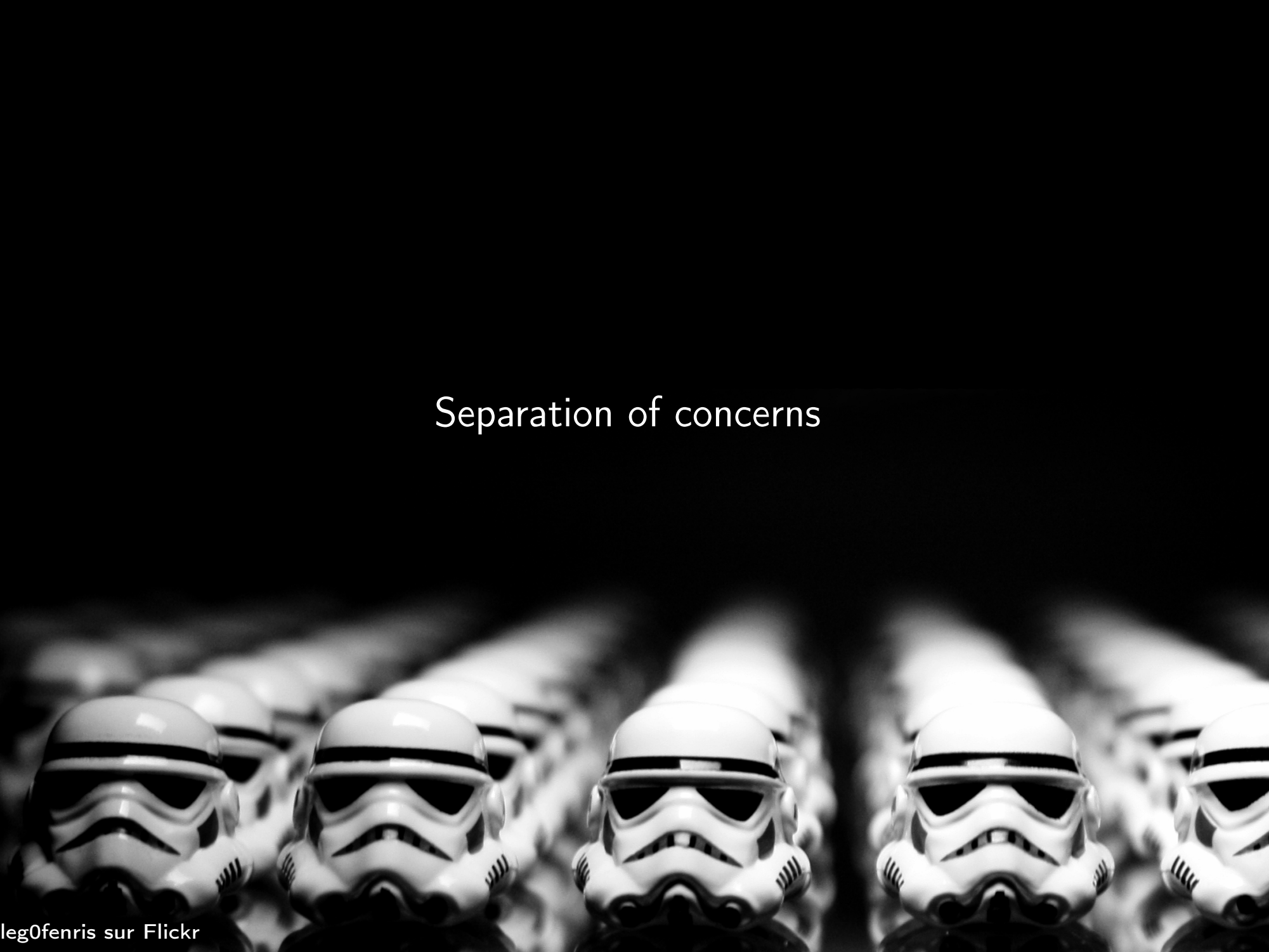
L'Ubuesque Boîte à Savon sur Flickr

daniel sur Flickr

```php
<?php

class DroidSearchAndDestroy {
  public function destroyDroid(Picture $picture) {
    // Get all droids from file
    $file = new File("yearbooks/droids.txt");
    $allDroids = [];
    foreach ($file->getLines() as $line) {
      $allDroids[] = new Droid($line);
    }

    // Check droids
    foreach ($allDroids as $droid) {
      if ($droid->look === $picture) {
        $droidFound = $droid;
      }
    }

    // Destroy droid
    if (isset($droidFound)) {
      $deathStar = new DeathStar();
      $deathStar->destroy($droidFound->getPlanet());
    } else {
      throw new DroidNotFoundException();
    }
  }
}
```

JD Hancock sur Flickr

Separation of concerns



leg0fenris sur Flickr

Analyse descendante

leg0fenris sur Flickr

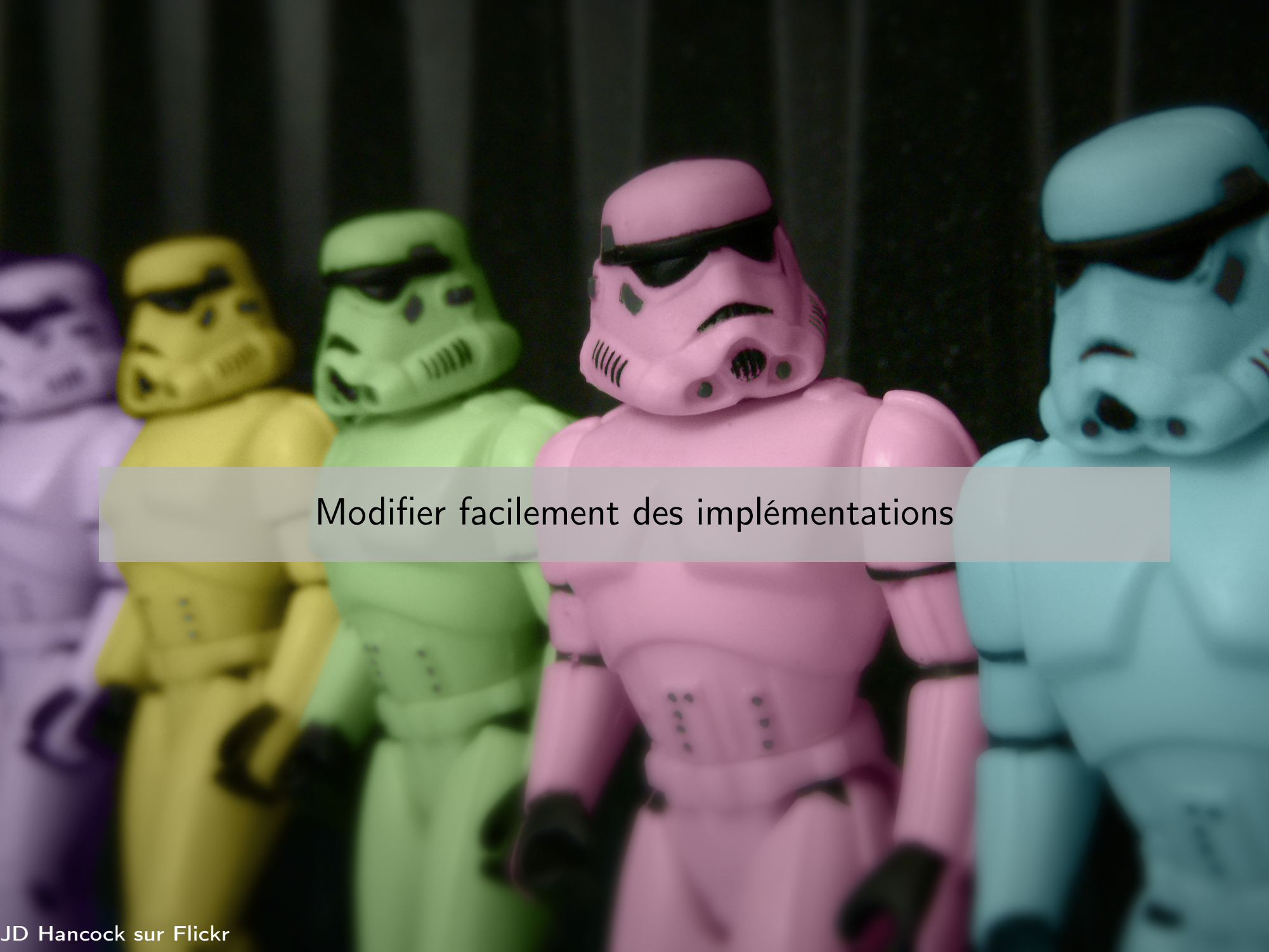
Tester unitairement son application

brett jordan sur Flickr

```php
<?php

class DroidSearchAndDestroy {
  public function destroyDroid(Picture $picture) {
    // Get all droids from file
    ...

    // Check droids
    foreach ($allDroids as $droid) {
      if ($droid->theseArentTheDroidsYoureLookingFor()) {
        continue;
      }

      if ($droid->look === $picture) {
        $droidFound = $droid;
      }
    }

    // Destroy droid
    ...
  }
}
```

```php
<?php

class DroidRepository {
  public function getAll(): array {
    $file = new File("yearbooks/droids.txt");
    $allDroids = [];
    foreach ($file->getLines() as $line) {
      $allDroids[] = new Droid($line);
    }
    return $allDroids;
  }
}

class DroidFinder {
  // return Maybe[Droid] ;-)
  public function findDroid(array $droids, Picture $picture): Droid {
    // foreach droids check picture
    return $droid;
  }
}

class PlanetDestroyer {
  public function destroy(Planet $planet) {
    $deathStar = new DeathStar();
    $deathStar->destroy($planet);
  }
}
```

```php
<?php

class DroidSearchAndDestroy {
  public function destroyDroid(Picture $picture) {
    // Get all droids from file
    $droidRepository = new DroidRepository();
    $droids = $droidRepository->getAll();

    // Check droids
    $droidFinder = new DroidFinder();
    $droid = $droidFinder->findDroid($droids, $picture);

    // Destroy droid
    if (isset($droidFound)) {
      $planetDestroyer = new PlanetDestroyer();
      $planetDestroyer->destroy($droidFound->getPlanet());
    } else {
      throw new DroidNotFoundException();
    }
  }
}
```



JD Hancock sur Flickr

```php
<?php

interface DroidRepository {
  /**
   * Return all the droids.
   *
   * @return Droid[]
   */
  public function getAll(): array;
}

final class FileDroidRepository implements DroidRepository {
  public function getAll(): array {
    // File implementation
  }
}

final class MongoDroidRepository implements DroidRepository {
  public function getAll(): array {
    // Mongo implementation
  }
}
```

```php
<?php

class DroidSearchAndDestroy {
  public function destroyDroid(Picture $picture) {
    // Get all droids from Mongo
    $droidRepository = new MongoDroidRepository();
    $droids = $droidRepository->getAll();

    // Check droids
    ...

    // Destroy droid
    ...
  }
}
```

```php
<?php

class DroidSearchAndDestroy {
  public function __construct(DroidRepository $droidRepository) {
    $this->droidRepository = $droidRepository;
  }

  public function destroyDroid(Picture $picture) {
    // Get all droids from file
    $droids = $this->droidRepository->getAll();

    // Check droids
    ...

    // Destroy droid
    ...
  }
}
```

Pas toujours besoin d'interface

Kristina Alexanderson sur Flickr

```php
<?php

class DroidSearchAndDestroy {
  public function __construct(
    DroidRepository $droidRepository,
    DroidFinder $droidFinder
  ) {
    $this->droidRepository = $droidRepository;
    $this->droidFinder = $droidFinder;
  }

  public function destroyDroid(Picture $picture) {
    // Get all droids from file
    $droids = $this->droidRepository->getAll();

    // Check droids
    $droid = $this->droidFinder->findDroid($droids, $picture);

    // Destroy droid
    ...
  }
}
```

```php
<?php

class DroidSearchAndDestroy {
  public function __construct(
    DroidRepository $droidRepository,
    DroidFinder $DroidFinder,
    DeathStar $deathStar
  ) {
    $this->droidRepository = $droidRepository;
    $this->droidFinder = $droidFinder;
    $this->deathStar = $deathStar;
  }

  public function destroyDroid(Picture $picture) {
    $droids = $this->droidRepository->getAll();

    $droidFound = $this->droidFinder->findDroid($droids, $picture);

    if (isset($droidFound)) {
      $this->deathStar->destroy($droidFound->getPlanet());
    } else {
      throw new DroidNotFoundException();
    }
  }
}
```

Kristina Alexanderson sur Flickr

```php
<?php

$droidSearchAndDestroy = new DroidSearchAndDestroy(
  new MongoDroidRepository(
    new MongoConnection("user", "password")
  ),
  new DroidFinder(),
  new DeathStar(
    new Gun(
      new Trigger()
    ),
    new Reactor(
      new Engine(
        new FuelConsumer()
      )
    ),
    new Shield(
      new ForceField()
    )
  )
);

$droidSearchAndDestroy->destroy($picture);
```

IoC Container

Walt Jabsco sur Flickr

```php
<?php

// Bind the interface to the implementation
$container->bind(DroidRepository::class, MongoDroidRepository::class);

// Bind the parameters
$container->bind(MongoConnection::class, function() {
  return new MongoConnection("user", "password");
});

// Don't create a new object each time
$container->bindShared(DeathStar::class);

$droidSearchAndDestroy = $container->make(DroidSearchAndDestroy::class);
$droidSearchAndDestroy->destroy($picture);
```

thibaud@dauce.fr – @ThibaudDauce

W_Minshull sur Flickr