

L'injection de dépendance ou comment découpler ses objets

Thibaud Dauce
thibaud@dauce.fr – @ThibaudDauce

26 novembre 2015

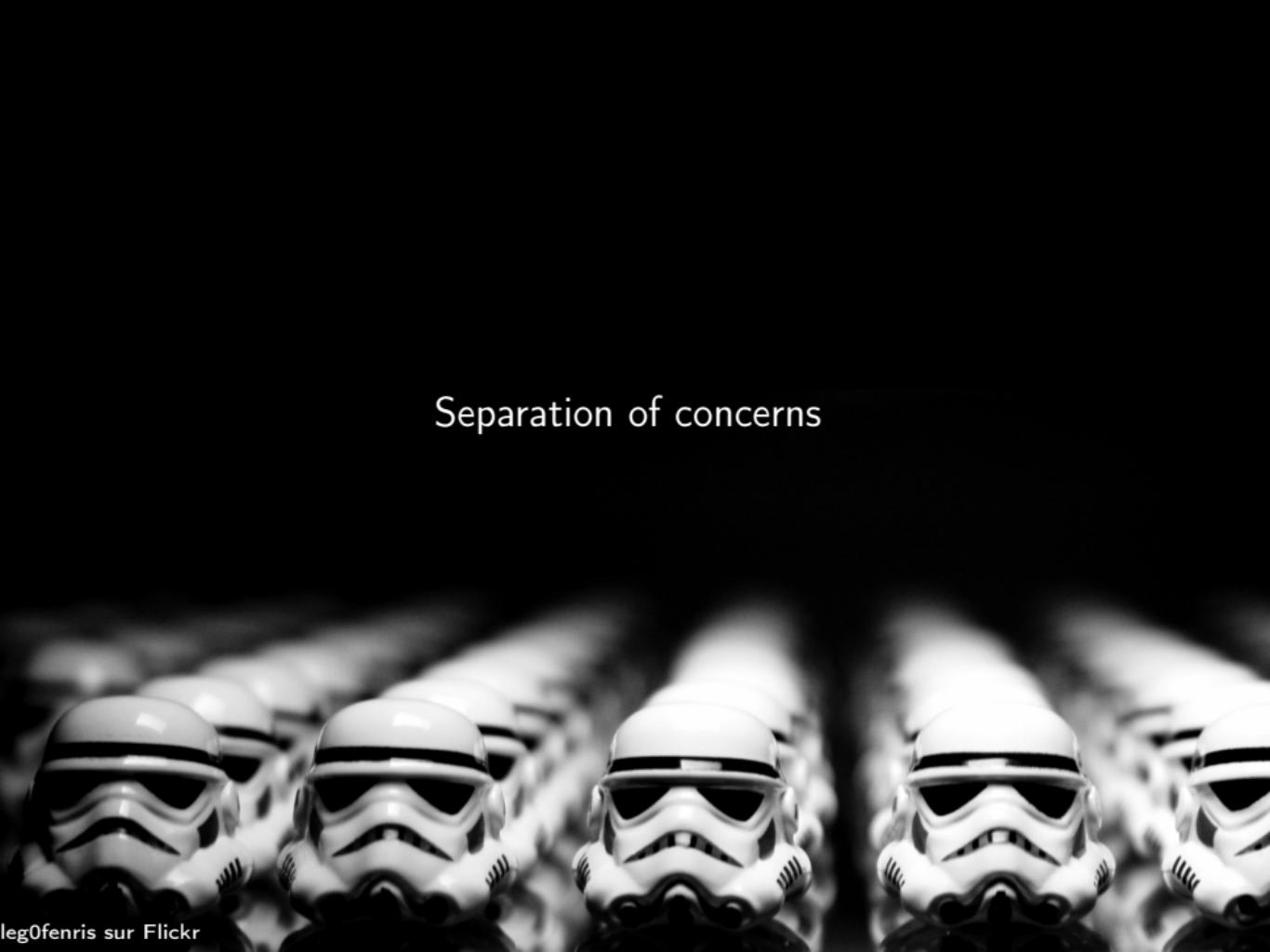
Application de recherche et de destruction de droides





```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function destroyDroid(Picture $picture) {
5         // Get all droids from file
6         $file = new File("yearbooks/droids.txt");
7         $allDroids = [];
8         foreach ($file->getLines() as $line) {
9             $allDroids[] = new Droid($line);
10        }
11
12        // Check droids
13        foreach ($allDroids as $droid) {
14            if ($droid->look === $picture) {
15                $droidFound = $droid;
16            }
17        }
18
19        // Destroy droid
20        if (isset($droidFound)) {
21            $deathStar = new DeathStar();
22            $deathStar->destroy($droidFound->getPlanet());
23        } else {
24            throw new DroidNotFoundException();
25        }
26    }
27 }
```





Separation of concerns

Analyse descendante

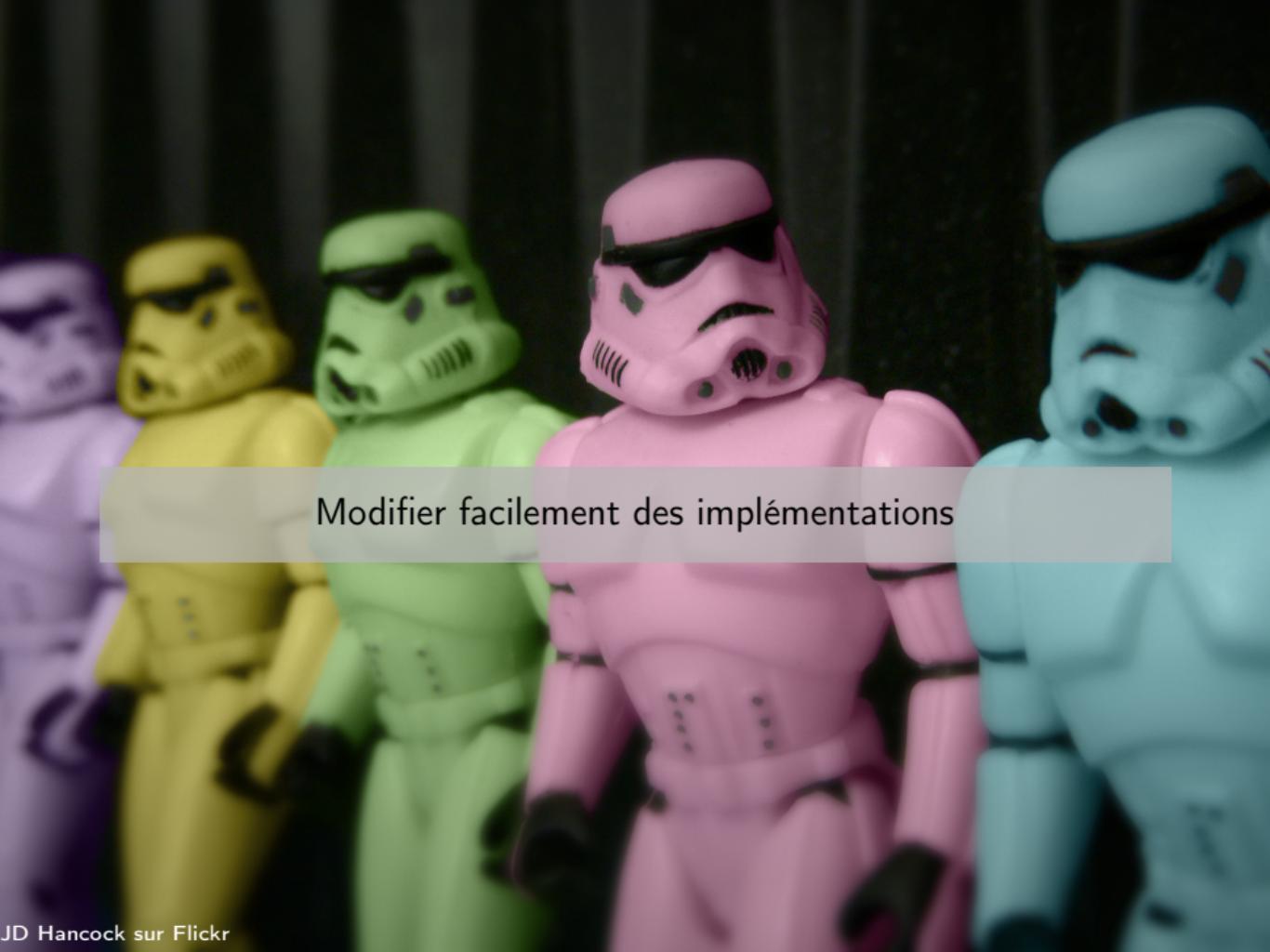
A Stormtrooper from Star Wars is shown from the side and back, wearing a white helmet and armor. He is working on a complex control panel or computer system in what appears to be a workshop or laboratory. The panel has numerous buttons, switches, and a small screen. In the background, there are shelves with various items and equipment. A small glowing orb sits on a stand to the left.

Tester unitairement son application

```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function destroyDroid(Picture $picture) {
5         // Get all droids from file
6         ...
7
8         // Check droids
9         foreach ($allDroids as $droid) {
10             if ($droid->theseArentTheDroidsYoureLookingFor()) {
11                 continue;
12             }
13
14             if ($droid->look === $picture) {
15                 $droidFound = $droid;
16             }
17         }
18
19         // Destroy droid
20         ...
21     }
22 }
```

```
1 <?php
2
3 class DroidRepository {
4     public function getAll(): array {
5         $file = new File("yearbooks/droids.txt");
6         $allDroids = [];
7         foreach ($file->getLines() as $line) {
8             $allDroids[] = new Droid($line);
9         }
10        return $allDroids;
11    }
12 }
13
14 class DroidFinder {
15     // return Maybe[Droid] ;-
16     public function findDroid(array $droids, Picture $picture): Droid {
17         // foreach droids check picture
18         return $droid;
19     }
20 }
21
22 class PlanetDestroyer {
23     public function destroy(Planet $planet) {
24         $deathStar = new DeathStar();
25         $deathStar->destroy($planet);
26     }
27 }
```

```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function destroyDroid(Picture $picture) {
5         // Get all droids from file
6         $droidRepository = new DroidRepository();
7         $droids = $droidRepository->getAll();
8
9         // Check droids
10        $droidFinder = new DroidFinder();
11        $droid = $droidFinder->findDroid($droids, $picture);
12
13        // Destroy droid
14        if (isset($droidFound)) {
15            $planetDestroyer = new PlanetDestroyer();
16            $planetDestroyer->destroy($droidFound->getPlanet());
17        } else {
18            throw new DroidNotFoundException();
19        }
20    }
21 }
```



Modifier facilement des implémentations

```
1 <?php
2
3 interface DroidRepository {
4     /**
5      * Return all the droids.
6      *
7      * @return Droid[]
8     */
9     public function getAll(): array;
10 }
11
12 final class FileDroidRepository implements DroidRepository {
13     public function getAll(): array {
14         // File implementation
15     }
16 }
17
18 final class MongoDroidRepository implements DroidRepository {
19     public function getAll(): array {
20         // Mongo implementation
21     }
22 }
```

```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function destroyDroid(Picture $picture) {
5         // Get all droids from Mongo
6         $droidRepository = new MongoDroidRepository();
7         $droids = $droidRepository->getAll();
8
9         // Check droids
10        ...
11
12         // Destroy droid
13        ...
14    }
15 }
```

```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function __construct(DroidRepository $droidRepository) {
5         $this->droidRepository = $droidRepository;
6     }
7
8     public function destroyDroid(Picture $picture) {
9         // Get all droids from file
10        $droids = $this->droidRepository->getAll();
11
12        // Check droids
13        ...
14
15        // Destroy droid
16        ...
17    }
18 }
```

A Stormtrooper toy from Star Wars is shown from the waist up, facing right. It is holding a paintbrush with dark paint on its bristles in its right hand, pointing it towards a chalkboard. The chalkboard has the words "HOME SWEET HOME" written in white chalk. A circular chalk drawing of a face is partially visible on the right side. A small black rectangular box with a thin white border is overlaid on the image, containing the text.

Pas toujours besoin d'interface

```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function __construct(
5         DroidRepository $droidRepository,
6         DroidFinder $droidFinder
7     ) {
8         $this->droidRepository = $droidRepository;
9         $this->droidFinder = $droidFinder;
10    }
11
12    public function destroyDroid(Picture $picture) {
13        // Get all droids from file
14        $droids = $this->droidRepository->getAll();
15
16        // Check droids
17        $droid = $this->droidFinder->findDroid($droids, $picture);
18
19        // Destroy droid
20        ...
21    }
22 }
```

```
1 <?php
2
3 class DroidSearchAndDestroy {
4     public function __construct(
5         DroidRepository $droidRepository,
6         DroidFinder $DroidFinder,
7         DeathStar $deathStar
8     ) {
9         $this->droidRepository = $droidRepository;
10        $this->droidFinder = $droidFinder;
11        $this->deathStar = $deathStar;
12    }
13
14    public function destroyDroid(Picture $picture) {
15        $droids = $this->droidRepository->getAll();
16
17        $droidFound = $this->droidFinder->findDroid($droids, $picture);
18
19        if (isset($droidFound)) {
20            $this->deathStar->destroy($droidFound->getPlanet());
21        } else {
22            throw new DroidNotFoundException();
23        }
24    }
25 }
```



```
1 <?php  
2  
3 $droidSearchAndDestroy = new DroidSearchAndDestroy(  
4     new MongoDroidRepository(  
5         new MongoConnection("user", "password")  
6     ),  
7     new DroidFinder(),  
8     new DeathStar(  
9         new Gun(  
10            new Trigger()  
11        ),  
12        new Reactor(  
13            new Engine(  
14                new FuelConsumer()  
15            )  
16        ),  
17        new Shield(  
18            new ForceField()  
19        )  
20    )  
21 );  
22  
23 $droidSearchAndDestroy->destroy($picture);
```



```
1 <?php
2
3 // Bind the interface to the implementation
4 $container->bind(DroidRepository::class, MongoDroidRepository::class);
5
6 // Bind the parameters
7 $container->bind(MongoConnection::class, function() {
8     return new MongoConnection("user", "password");
9 });
10
11 // Don't create a new object each time
12 $container->bindShared(DeathStar::class);
13
14 $droidSearchAndDestroy = $container->make(DroidSearchAndDestroy::class);
15 $droidSearchAndDestroy->destroy($picture);
```



thibaud@dauce.fr – @ThibaudDauce