PHP
Programmation Orientée Objet

erick@rxlabz.com

POO

La Fabrick

POO

- Principes
  - Classes
  - Héritage
  - Abstraction
  - Interface
  - Composition

La Fabrick

La Fabrick

Court métrage (procédural)

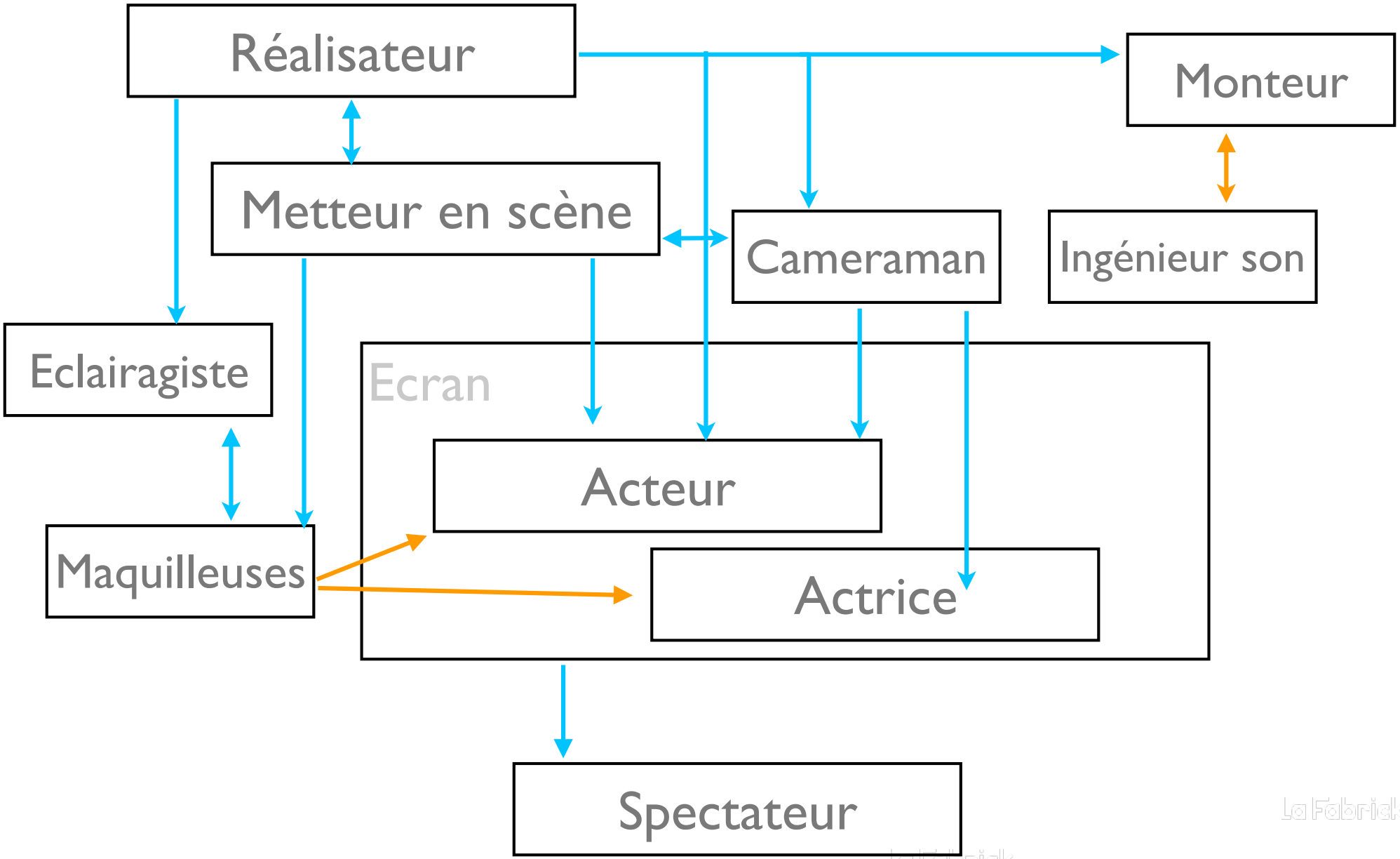
Une seule et même entité exécute l'ensemble des taches nécessaires.

- Ecrire scénario » ecrireScenario()
- Ecrire dialogue » ecrireDialogue()
- apprendre dialogue » apprendreDialogue()
- construire décor » construireDecor()
- régler l'éclairage » regleEclairage()
- allumer la caméra » allumeCamera()
- allumer le micro » allumeMicro()
- jouer scène » joueScene()
- monterFilm » monterFilm()
- ...

La Fabrick

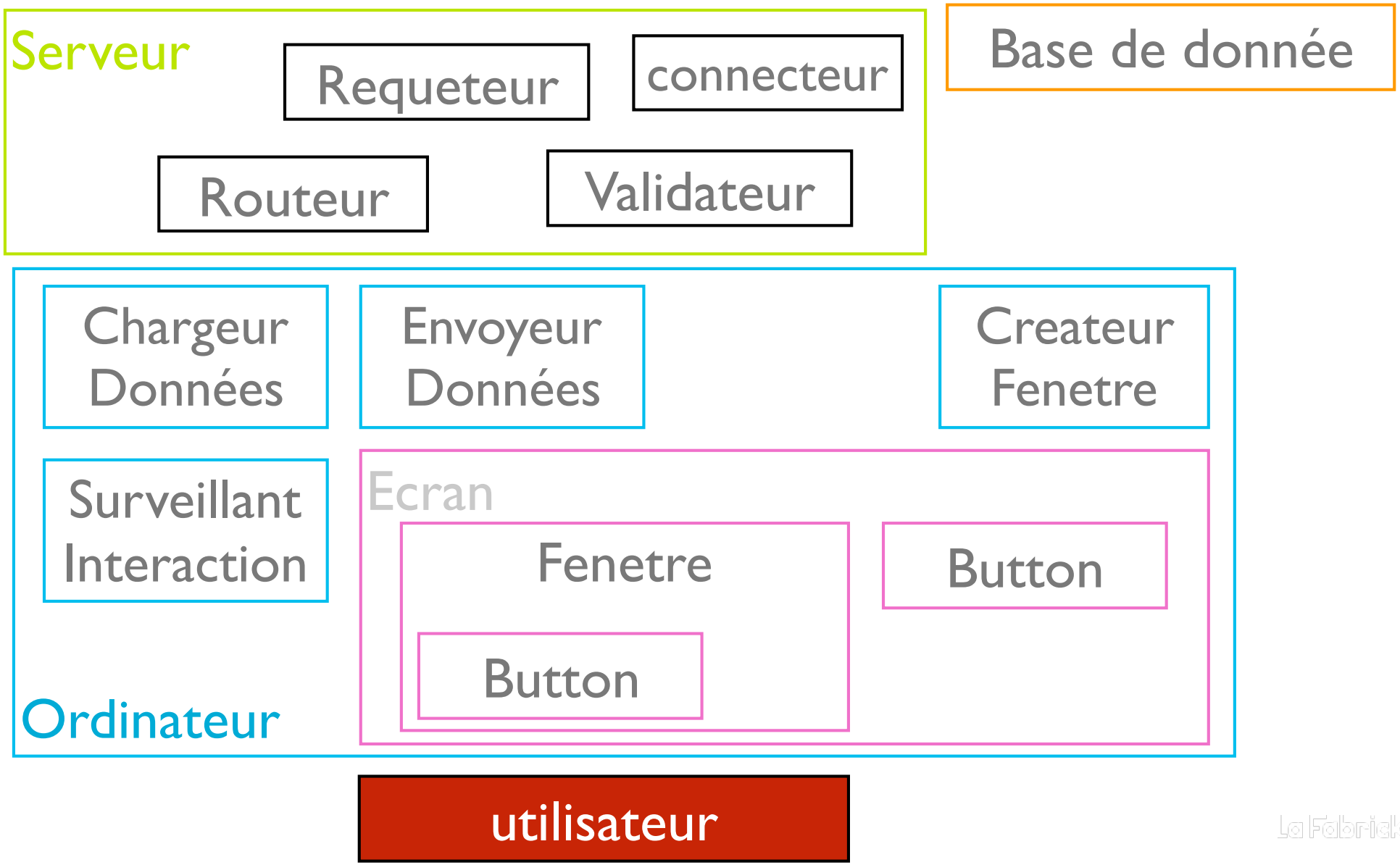
La Fabrick

Hollywood (POO)



La Fabrick

Application web



La Fabrick

POO

- Class
  - Constantes
  - Variables / Propriétés / Attributs
  - Accesseurs
  - Fonctions / Méthodes

La Fabrick

La Fabrick

Propriétés

```
class Rectangle {

    public $longueur;

    public $largeur;

}
```

La Fabrick

La Fabrick

Portée / Accès

- public : accessible pour tout le monde
- private : accessible seulement dans la classe
- protected : accessible depuis la classe et ses sous-classes
- static : accessible directement via la Classe (sans besoin d'instance)
- final : ne peut être étendu (classe) ou overridé (fonction)

La Fabrick

La Fabrick

Constantes

- class Cercle {

  const PI = 3.14;

  ...

  }

  Utilisation

  - function perimetre(){ return 2 * self::PI * $this->rayon ; }
  - $perimetre = Cercle::PI ;

La Fabrick

La Fabrick

Méthodes & arguments

- les méthodes d'une classe peuvent nécessiter des arguments :
  - function chargeGalerie($url)
- Les arguments peuvent être facultatifs
  - function chargeGalerie($url = 'galerie.xml')

La Fabrick

La Fabrick

Héritage

- L'héritage permet de créer une classe sur la base d'une autre, on appelle cela "étendre une classe"

- La classe qui en étend une autre, est appelée "sous-classe". Elle récupère toutes les propriétés et méthodes de la "super-classe"

La Fabrick

La Fabrick

Héritage

```
class Galerie extends Sprite {

    public function Galerie(){
        super();
    }

}
```

La Fabrick

La Fabrick

Héritage & override

- L'override de méthode permet de redéfinir dans une sous classe la définition d'une méthode de sa super-classe.
  - override public function showListe():void
    {
        ... // définition du comportement spécifique à la sous-classe

La Fabrick

La Fabrick

Héritage

- Le mot clé super permet de faire référence aux méthodes d'une "super-classe"

- // dans le constructeur
  public function GaleriePhoto():void{
  super(); // référence au constructeur de Galerie
  }

- // dans le méthodes public et protected
  override public function showListe():void{
  super.showListe();//référence à la méthode showList de Galerie
  ... // définition des opération propres à GaleriePhoto
  }

La Fabrick
La Fabrick

Interface

- Les interfaces définissent un ensemble de méthodes implémentées par un ensemble de classes.
  - Elles permettent d'écrire un code en fonction d'un «contrat de fonctionnalités», et non pas en fonction d'une classe particulière.

La Fabrick

La Fabrick

Interfaces

- interface IChargeur
  {

    function charge(url : String):void;
    function decharge():void;
  }
- Implémentation

```
public class ChargeurImage implements IChargeur {
    public function charge( url:String ):void{...}
    public function decharge():void{...}
}
```

La Fabrick

La Fabrick

POO principes

- Encapsulation
- Abstraction
- Polymorphisme

La Fabrick

La Fabrick

La Fabrick

Encapsulation

- L'encapsulation est en quelque sorte un (le) principe de constitution des objets :
  le rassemblement des caractèristiques et fonctionnalités qui constitue une entité distincte
  - exemple :
    - composants
    - Objet de données (Value Object)

La Fabrick
La Fabrick
La Fabrick

Encapsulation

- En principe, l'utilisation d'un objet ne nécessitent pas la connaissance et encore moins la modification du fonctionnement interne.

  L'objet propose des méthodes publiques, mais l'implémentation interne doit rester inconnue pour l'extérieur.

  - exemple : get / set

La Fabrick

La Fabrick

La Fabrick

Polymorphisme

- Le polymorphisme est la possibilité de prendre plusieurs formes.
  - En utilisant des interfaces ou des classes abstraites, on laisse la possibilité à notre code de prendre “différentes formes“, différentes classes concrètes au final.

La Fabrick
La Fabrick
La Fabrick

Interface

- Program to an 'interface',
  not an 'implementation'
  - » vos classes seront beaucoup plus souples si elles “dépendent“ d’interfaces, et non pas d’autres classes «concrètes»

La Fabrick
La Fabrick
La Fabrick