

# PROGRAMMATION ORIENTÉE OBJET

# Objectifs de formation

- ★ Développement en POO
- ★ Utilisation de PDO
- ★ Etudes des design-pattern (MVC, SINGLETON,...)

# Développement en POO

## Qu'est-ce qu'un objet?

### Définition simplifiée

C'est une boîte dans laquelle il y a des informations stockées et qui a des boutons sur lesquels on appuie pour obtenir ses informations.

### Dans la vraie vie

C'est une super-variable php qui contient des variables simples (attributs) et des fonctions (méthodes) qui permettent de récupérer ses attributs.

# Développement en POO

Prenons un exemple concret...

Article = Objet

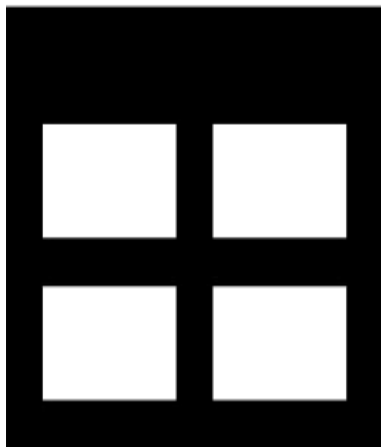


Titre
Auteur
Date de création
Visuel
Corps de texte

# Développement en POO

Prenons un exemple concret...

Patron = Classe



titre
auteur
date
image
message

# Développement en POO

1 classe = 1 fichier

# Développement en POO

Le code est pensé en “Framework”

# Développement en POO

Dans mon code, pas de redirection vers des scripts PHP. Tout se passe sur index.php



# Développement en POO

Prenons un exemple concret...

Classe

Attributs	Méthodes
\$titre	donneMoiTitre()
\$auteur	donneMoiAuteur()
\$date	donneMoiDate()
\$image	donneMoiImage()
\$message	donneMoiMessage()

# Développement en POO

## Classe

```
<?php
```

```
class Article {  
    private $_titre;  
    private $_auteur;  
    private $_date;  
    private $_image;  
    private $_message;  
  
    public function donneMoiTitre() { }  
    public function donneMoiAuteur() { }  
    public function donneMoiDate() { }  
    public function donneMoiImage() { }  
    public function donneMoiMessage() { }  
}
```

```
?>
```

# Développement en POO

## Création de notre 1er objet

```
<?php

class Article {
    private $_titre = "J'aime les objets";
    private $_auteur = "Philippe Stark";
    ...

    public function donneMoiTitre() {
        return $this->_titre;
    }
    ...
}

$article = new Article();

?>
```

# Développement en POO

## Quelques essais...

```
class Article {  
    $_titre = "J'aime les objets";  
}  
---  
class Article {  
    public $_titre = "J'aime les objets";  
}  
$article = new Article();  
print $article->_titre;  
---  
class Article {  
    private $_titre = "J'aime les objets";  
}  
$article = new Article();  
print $article->_titre;
```

```
class Article {  
    private $_titre = 'J'aime les objets';  
    public function donneMoiTitre() {  
        print 'Je veux pas';  
    }  
}  
$article = new Article();  
$article->donneMoiTitre();  
---  
class Article {  
    private $_titre = 'J'aime les objets';  
    public function donneMoiTitre() {  
        return $this->_titre;  
    }  
}  
$article = new Article();  
print $article->donneMoiTitre();
```

# Développement en POO

## Setters et Getters

```
class Article {  
    private $_titre = 'J'aime les objets';  
  
    public function getTitre() {  
        return $this->_titre;  
    }  
  
    public function setTitre($new_titre) {  
        $this->_titre = $new_titre;  
    }  
}  
$article = new Article();  
$article->setTitre('J'aime beaucoup les objets :-)');  
print $article->getTitre();
```

# Développement en POO

## Setters

```
public function setTitre($titre) {  
    if(is_numeric($titre)) {  
        print '<p class="warning">Le titre de l\'article ne peut pas  
être uniquement un nombre entier</p>';  
    }  
    else {  
        $this->_titre = $titre;  
    }  
}  
$article = new Article();  
$article->setTitre('J\'aime beaucoup les objets :-)');  
print '<h2>'.$article->getTitre().'</h2>';
```

# Développement en POO

## Constructeur

Méthode magique

```
class Article {  
    private $_titre;  
  
    public function __construct($post) {  
        $this->setTitre($post['titre']);  
    }  
  
    public function getTitre() {  
        return $this->_titre;  
    }  
  
    public function setTitre($new_titre) {  
        $this->_titre = $new_titre;  
    }  
}  
$article = new Article($_POST);
```

# Développement en POO

## Hydratation

```
public function __construct(array $data) {
    $this->hydrate($data);
}

public function hydrate(array $data) {
    foreach ($data as $key => $value) {
        $method = 'set'.ucfirst($key);
        if (method_exists($this, $method)) {
            $this->$method($value);
        }
    }
}
```



# Développement en POO

API PDO:

<http://www.php.net/manual/fr/book.pdo.php>

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 1

```
try {  
    $DB = new PDO('mysql:host=localhost;dbname=aston', 'root', '');  
}  
catch(Exception $e) {  
    die('Erreur : ' . $e->getMessage());  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 1

```
class ArticleManager {  
    private $_db;  
    public function __construct($db) {  
        $this->setDb($db);  
    }  
  
    //  SETTERS  
  
    public function setDb(PDO $db) {  
        $this->_db = $db;  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 2

```
class ArticleManager {  
    ...  
    public function addArticle(Article $article) {  
  
        $ADD_ARTICLE = $this->_db->prepare('INSERT INTO article SET titre=:  
titre, auteur=:auteur, date=NOW(), image=:image, message=:message');  
        $ADD_ARTICLE->execute(array(  
            ':titre' => $article->getTitre(),  
            ...  
        ));  
        $ADD_ARTICLE->closeCursor();  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 2

```
class ArticleManager {  
    ...  
    public function addArticle(Article $article) {  
  
        $ADD_ARTICLE = $this->_db->prepare('INSERT INTO article SET titre=:  
titre, auteur=:auteur, date=NOW(), image=:image, message=:message');  
        $ADD_ARTICLE->bindParam(':titre', $article->getTitre());  
        ...  
        $ADD_ARTICLE->execute();  
        $ADD_ARTICLE->closeCursor();  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 3

```
class ArticleManager {  
    ...  
    public function getAllArticle() {  
  
        $LIST = $this->_db->query('SELECT *, DATE_FORMAT(date, \'%d/%m/%y\')  
AS date FROM article ORDER BY date DESC');  
  
        while ($list_fetch = $LIST->fetchAll()) {  
            $article[] = $list_fetch;  
        }  
        return $article;  
        $LIST->closeCursor();  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 3

```
class ArticleManager {  
    ...  
    public function getAllArticle() {  
  
        $LIST = $this->_db->query('SELECT *, DATE_FORMAT(date, \'%d/%m/%y\')  
AS date FROM article ORDER BY date DESC');  
  
        while ($list_fetch = $LIST->fetch(PDO::FETCH_ASSOC)) {  
            $article[] = new Article($list_fetch);  
        }  
        return $article;  
        $LIST->closeCursor();  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 4

```
class ArticleManager {  
    ...  
    public function getArticle($article_id) {  
  
        $ARTICLE = $this->_db->query('SELECT *, DATE_FORMAT(date, \'%d/%m/%  
y\') AS date FROM article WHERE id='.$article_id);  
  
        $article_fetch = $ARTICLE->fetch(PDO::FETCH_ASSOC);  
        $ARTICLE->closeCursor();  
        return new Article($article_fetch);  
    }  
}
```



# Développement en POO

## Connexion à la BDD : Les Managers - Partie 4

```
class ArticleManager {  
    ...  
    public function getArticle($article_id) {  
  
        $ARTICLE = $this->_db->prepare('SELECT *, DATE_FORMAT(date, \'%d/%  
m/%y\') AS date FROM article WHERE id=:id');  
        $ARTICLE->bindValue(':id', $article_id);           $ARTICLE->  
        >execute();  
        $article = $ARTICLE->fetch(PDO::FETCH_ASSOC);  
        $ARTICLE->closeCursor();  
        return new Article($article);  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 5

```
class ArticleManager {  
    ...  
    public function updateArticle(Article $article) {  
  
        $UP_ARTICLE = $this->_db->prepare('UPDATE article SET titre=:titre,  
auteur=:auteur, image=:image, message=:message WHERE id=:id');  
        $UP_ARTICLE->bindValue(':id', $article->getId());  
        ...  
        $UP_ARTICLE->execute();  
        $UP_ARTICLE->closeCursor();  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 5

```
class ArticleManager {  
    ...  
    public function updateArticle(Article $article) {  
  
        $UP_ARTICLE = $this->_db->prepare('UPDATE article SET titre=:titre,  
auteur=:auteur, image=:image, message=:message WHERE id=:id');  
        $UP_ARTICLE->bindValue(':id', $article->getId());  
        ...  
        $UP_ARTICLE->execute();  
        $UP_ARTICLE->closeCursor();  
    }  
}
```

# Développement en POO

## Connexion à la BDD : Les Managers - Partie 6

```
class ArticleManager {  
    ...  
    public function deleteArticle($article_id) {  
        $article_id = (int) $article_id;  
        $this->_db->exec('DELETE FROM article WHERE  
id= '.$article_id);  
    }  
}
```

# Développement en POO

## Le modèle MVC

**Modèle**

**Vue**

**Contrôleur**

<http://book.cakephp.org/2.0/fr/contents.html>

# Développement en POO

## Le modèle MVC

### Contrôleur :

- Implémente les actions.
- Autant de classes que nécessaire, regroupés par entités logiques (Articles, utilisateurs...)
- Et aussi pour les erreurs (404, 403, ...).
- Chaque contrôleur a ses fonctions (Lister, ajouter, modifier, supprimer,...)

# Développement en POO

## Le modèle MVC

### Modèle :

- Gère les échanges avec la BDD.
- Une classe par entité dans la BDD (utilisateur, article, produit, catégorie, etc..).

# Développement en POO

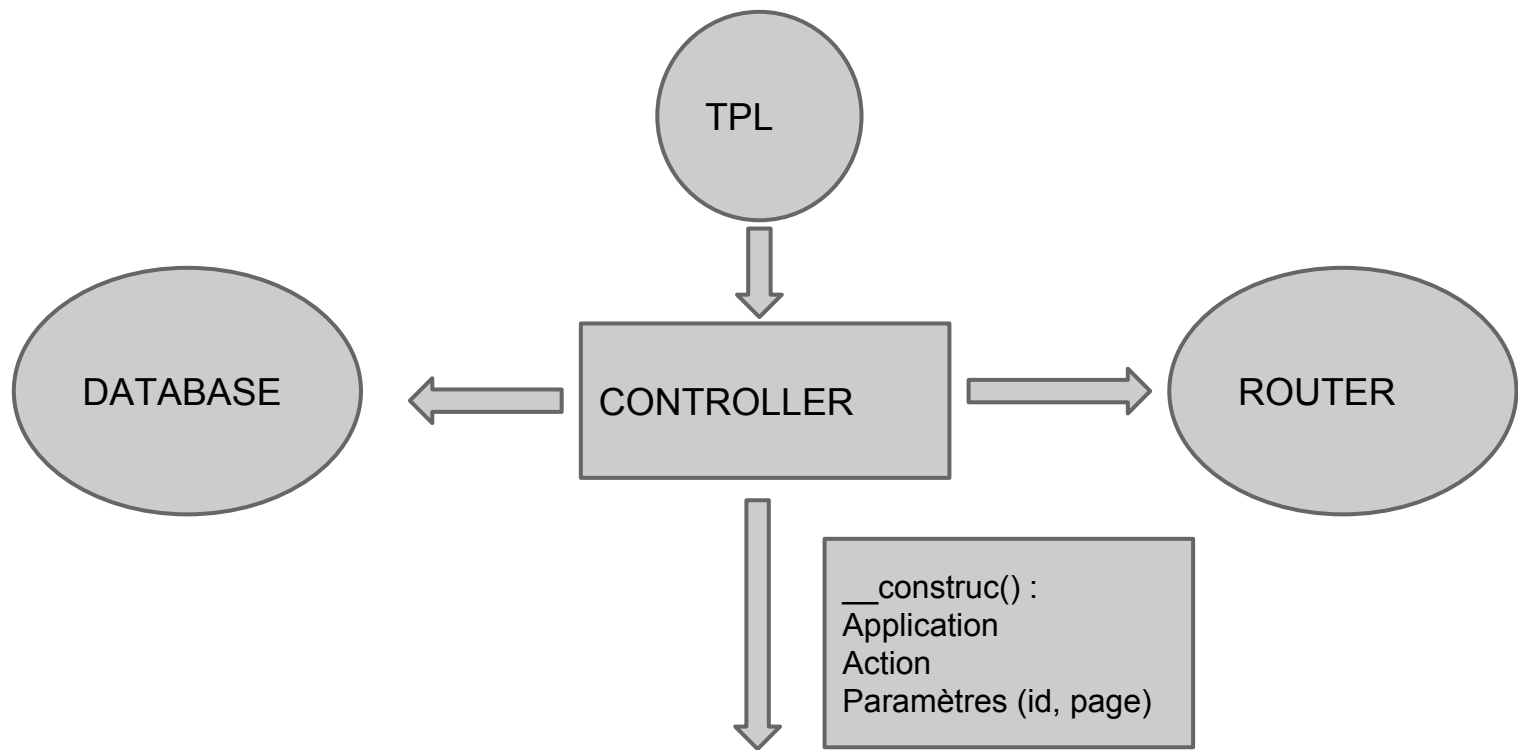
## Le modèle MVC

### Vue :

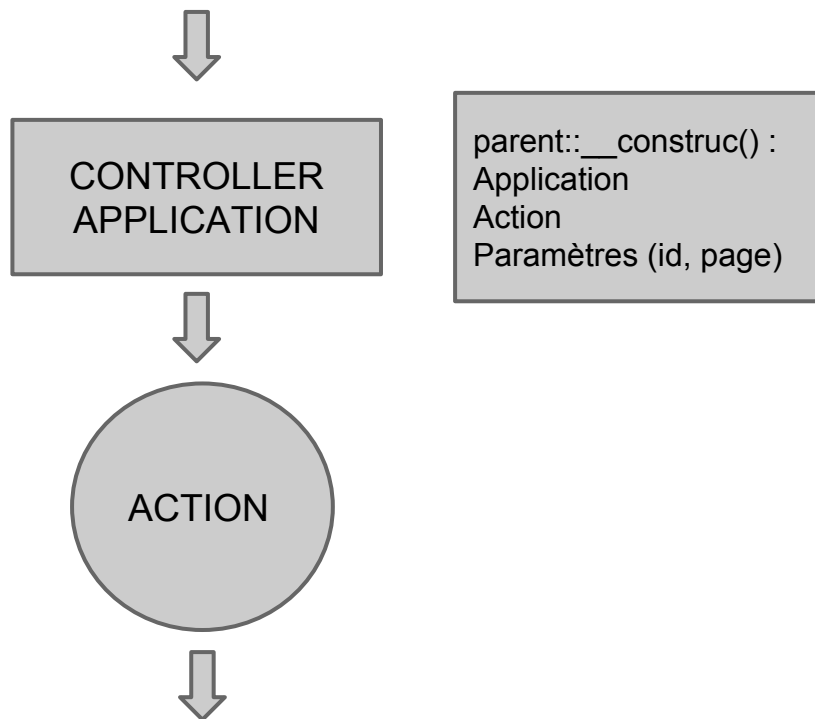
- Affichages HTML.
- Injecte des parties variables (provenant du modèle par exemple).
- On peut utilisé un moteur de gabarit (Smarty, Twig).



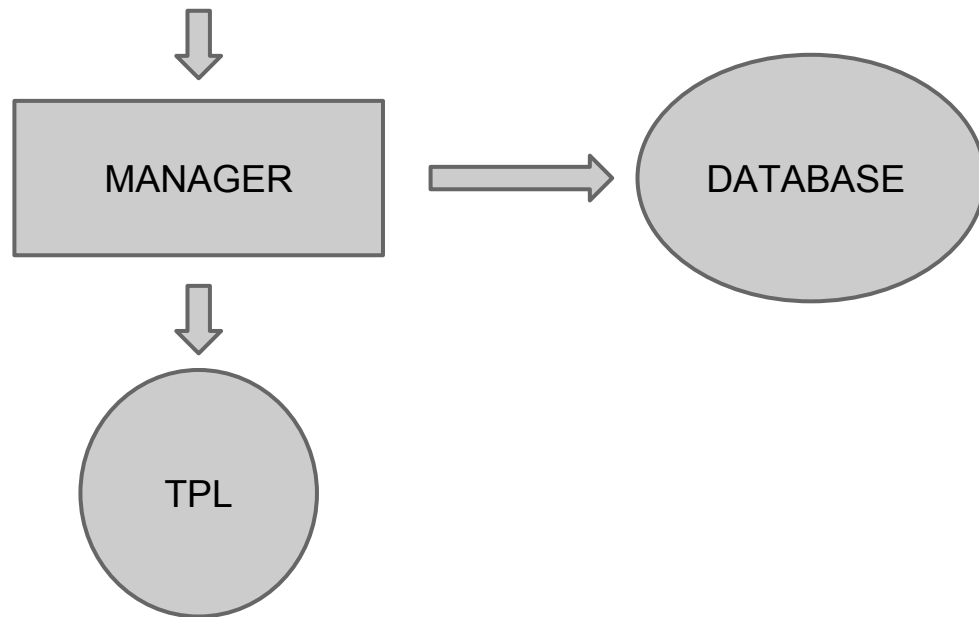
# Notre framework



# Notre framework



# Notre framework



# Développement en POO

Framework MVC simples:

KissMVC

<http://kissmvc.com/>

SimpleMVC

<http://simplemvc.berejeb.com/>

# Design patterns - Introduction

Un design pattern, traduit en français par motif de conception ou patron de conception est un terme de génie logiciel qui désigne une solution standard d'architecture logicielle correspondant à un problème donné.

# Design patterns - Introduction

Les mêmes problèmes de conceptions reviennent souvent. On peut les régler d'une façon standardisée.

- ★ Ne pas réinventer la roue.

Inutile de chercher pendant des heures une solution adéquate.

- ★ Adopter une solution optimale.

Il s'agit d'une solution éprouvée, certainement meilleure qu'une autre que vous pourriez imaginer.

- ★ Fournir un vocabulaire commun aux développeurs.

Utiliser le nom d'un design pattern explique sans ambiguïté de quoi il s'agit.

# Design patterns - Introduction

<http://design-patterns.fr>

# Design patterns - MVC

## Problématique :

Un code pompeux, mal organisé et difficilement maintenable.

## Solution :

Une organisation structurée en fichiers simples et en trois fonctionnalités :  
Modèle, vue et contrôleur .



# Design patterns - Singleton

Il s'agit sûrement de la pattern la plus connue.

## Problématique :

S'assurer qu'il existe une seule instance d'un objet donné pour toute l'application.

## Solution :

Une méthode statique pour contrôler l'instanciation. Rendre ce processus d'instanciation l'unique solution possible pour la classe en question.

# Design patterns - Singleton

```
class Article {  
    private static $_instance = null;  
    public static function getInstance() {  
        if (is_null(self :: $_instance)) {  
            self :: $_instance = new Article();  
        }  
        return self :: $_instance;  
    }  
    public function __clone() {  
        print 'vous ne pouvez cloner un objet Article';  
    }  
}
```

# Design patterns - Factory

## Problématique :

Beaucoup de classes dépendantes de sous-classes.

## Solution :

Définit une interface pour la création d'un objet en déléguant à ses sous-classes le choix des classes à instancier.

# Design patterns - Factory

```
class DBFactory {
    public static function create ($connectionString){
        if (($driverEndPos = strpos ($connectionString, ':')) === false){
            throw new Exception ('Mauvaise chaine de connexion');
        }

        switch (substr ($connectionString, 0, $driverEndPos)){
            case 'mysql':
                $db = new DBMySQL ($connectionString1);
                break;
            case 'oracle':
                $db = new DBOracle ($connectionString1);
                break;
            default:
                throw new Exception ('Type de base inconnu');
        }
        return $db;
    }
}
```

