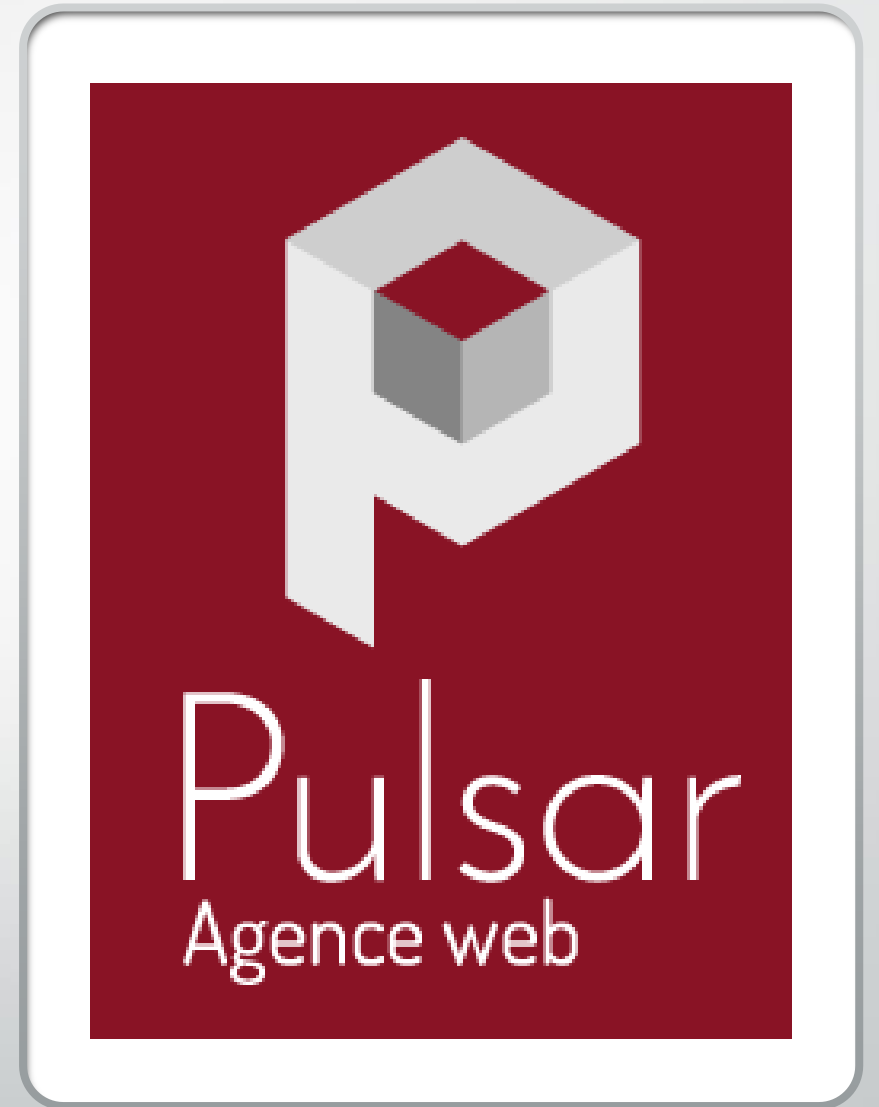


# Formation : projet web en MVC psr-4 d'un crud

*(create, read, update, delete)*

[www.pulsar-informatique.com](http://www.pulsar-informatique.com)



**FORMATEUR** : Joachim Thibout  
développeur **php symfony joomla! zend**

joachim@pulsar-informatique.com

Site perso :

<http://dune-empereur-jeu.fr>

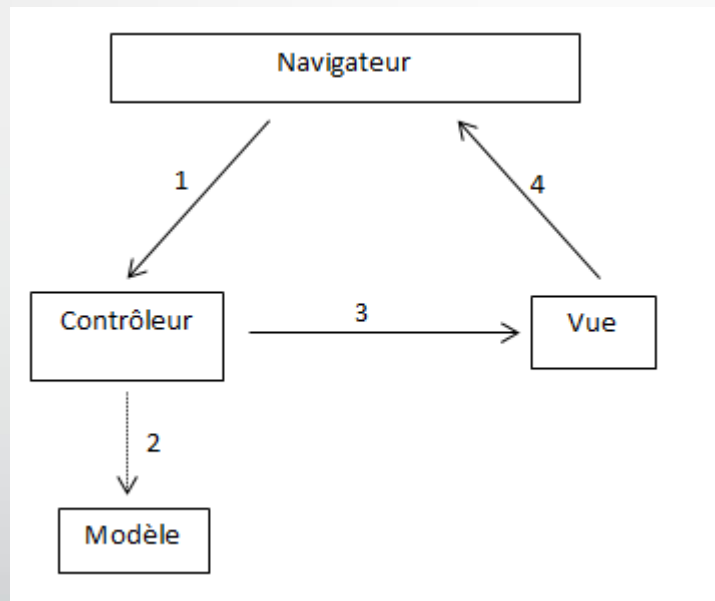
# PLAN DE LA FORMATION

1. Trouver et télécharger un MVC sur le web
2. Récupérer les cdn jQuery et bootstrap et préparer ses propres fichiers (ajout aussi d'une class Utils, disponible à la racine du site)
3. Faire le plan du site désiré ( en faisant le menu par exemple pour toutes les vues du site)
4. Création de la Base de donnée – pas à pas : on crée une table, puis on crée la page concernée
5. Faire le CRUD de l'élément créé en base de donnée
6. Faire les liaisons entre les différentes tables ( 1 to N ) (N to N)
7. Créer les filtres de recherche
8. Gérer les droits utilisateurs
9. Ajouter des filtres javascript / du css et pourquoi pas un peu d'HTML5

# 1. Trouver et télécharger un MVC sur le web

**Modèle-Vue-Contrôleur** est un motif d'architecture. Le motif est composé de trois types de modules ayant trois responsabilités différentes: les modèles, les vues et les contrôleurs.

- Un **modèle** contient les données à afficher.
- Une **vue** contient la présentation de l'interface graphique.
- Un **contrôleur** contient la logique concernant les actions effectuées par l'utilisateur (bref, il fait office de passe-plat : il récupère les données –SQL, XML, TEXT, JSON- et les balance dans la vue)



Rechercher sur le web un mvc PHP : on en trouve facilement sur GITHUB :

<https://github.com/jimmiw/php-mvc-base>

# MOD REWRITE ACTIVATION

<http://askubuntu.com/questions/48362/how-to-enable-mod-rewrite-in-apache>

[https://itx-technologies.com/blog/28-installer-et-activer-le-module-mod\\_rewrite-sur-apache-et-ubuntu](https://itx-technologies.com/blog/28-installer-et-activer-le-module-mod_rewrite-sur-apache-et-ubuntu)

## 2. Récupérer les cdn jQuery et bootstrap

Préparer le layout de son site web en y ajoutant le CSS et le jQuery. On peut aussi rajouter le normalize.css (<https://eonasdan.github.io/bootstrap-datetimepicker>)

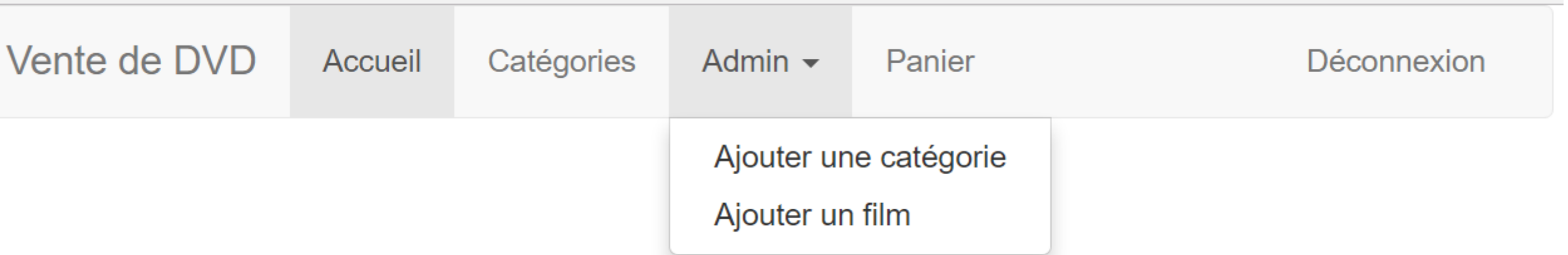
Faire aussi un include du menu dans le layout

```
<script type="text/javascript" src="//code.jquery.com/jquery-2.1.1.min.js"></script>
<script type="text/javascript" src="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/moment.js/2.9.0/moment-with-locales.js"></script>
<script src="//cdn.rawgit.com/Eonasdan/bootstrap-datetimepicker/e8bddc60e73c1ec2475f827be36e1957af72e2ea/src/js/bootstrap-datetimepicker.js"></script>

<link rel="stylesheet" type="text/css" media="screen"
href="//maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css" />
<link rel="stylesheet" href="//maxcdn.bootstrapcdn.com/font-awesome/4.3.0/css/font-awesome.min.css">
<link href="//cdn.rawgit.com/Eonasdan/bootstrap-datetimepicker/e8bddc60e73c1ec2475f827be36e1957af72e2ea/build/css/bootstrap-datetimepicker.css"
rel="stylesheet">
<link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/normalize/5.0.0/normalize.css">
```

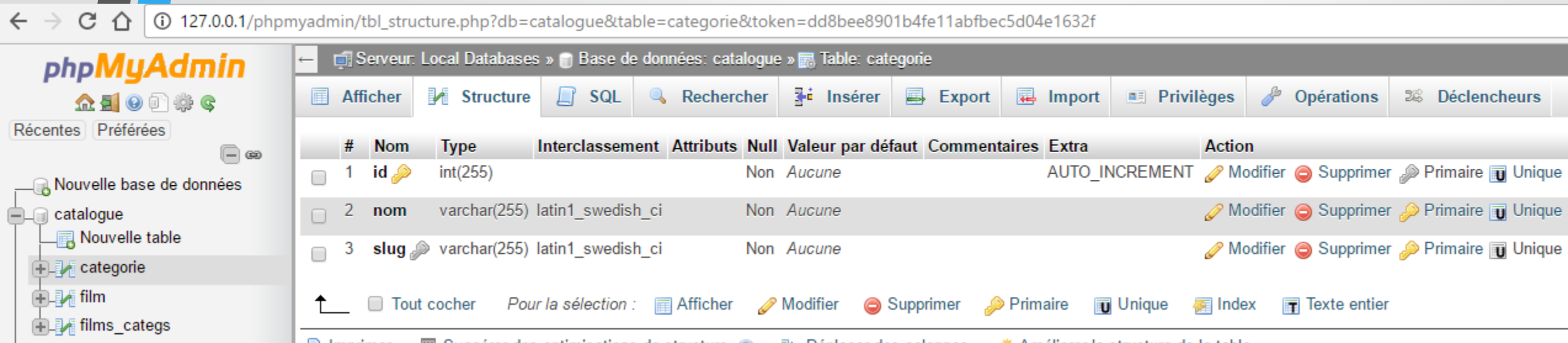
```
<div class="container">
    <?php include('menu.phtml') ;?>
    <?php echo $this->content() ; ?>
</div>
```

### 3. Faire le plan du site désiré (Créer le menu)



Pour créer un menu de ce type, chercher sur le web des exemples de menu bootstrap

## 4. Créer la base de donnée en relation avec le menu créé



The screenshot shows the phpMyAdmin interface. The left sidebar displays the database structure: 'catalogue' is selected, and 'Nouvelle table' is highlighted. The main area shows the 'Table: categorie' structure. The table has three columns: 'id' (int(255), primary key, auto-increment), 'nom' (varchar(255), latin1\_swedish\_ci, primary key), and 'slug' (varchar(255), latin1\_swedish\_ci, primary key). The 'id' column is marked as 'AUTO\_INCREMENT'. The 'nom' and 'slug' columns are marked as 'Unique'.

#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra	Action
1	id	int(255)			Non	Aucune		AUTO_INCREMENT	Modifier Supprimer Primaire Unique
2	nom	varchar(255) latin1_swedish_ci			Non	Aucune			Modifier Supprimer Primaire Unique
3	slug	varchar(255) latin1_swedish_ci			Non	Aucune			Modifier Supprimer Primaire Unique

Below the table structure, there are checkboxes for 'Tout cocher' and a selection menu with options: 'Afficher', 'Modifier', 'Supprimer', 'Primaire', 'Unique', 'Index', and 'Texte entier'.

Créer la table categorie, puis définir les champs. On est pas obligé à ce stade de créer la table film et films\_categs











## 5. Faire le CRUD de la table créée précédemment

Vente de DVD Accueil Catégories Réalisateur Admin ▾ Inscription Connexion

Ajouter une catégorie  
Ajouter un film

### Liste des catégories

Drames	
Historique	
Horreur	
Humour	
Policier	
Romance	
Sci-fi	
Sentimental	

← AJOUT de catégorie

→ SUPPRESSION de catégorie

LISTE de catégorie

← EDITION de catégorie

Ce type de contenu ne nécessite pas de vue détail (cREAD<sub>ud</sub>)

On commence d'abord par la vue formulaire de création des catégories

Faire la vue du formulaire dans addcateg.phtml:

```
<form method="POST" action="add/categorie" class="form-horizontal">
  <div class="form-group">
    <label for="categorie">Catégorie:</label><input type="text"
class="form-control" value="" id="categorie" name="categorie">
  </div>
  <button type="submit" id="ajax" class="btn btn-default">Submit</button>
</form>
```

## Faire l'action addcategAction du controleur

```
public function addcategAction()  
{  
    if(isset($_POST['categorie'])){  
        $nom = $_POST['categorie'];  
        $alias = Utils::generateSlug($nom);  
  
        $params = [  
            'nom' => $nom,  
            'slug' => $alias  
        ];  
  
        $cat = new Categorie();  
        $this->view->insertReturn = $cat->save($params);  
    }  
}
```

Mais s'occuper donc du Model pour pouvoir insérer dans la base de donnée

# On liste maintenant les catégories

Récupérer dans le controlleur les catégories dans une actions « categories »: les passer à la vue et les afficher dans cette nouvelle vue

```
public function categoriesAction()  
{  
    $cats = new Categorie();  
  
    $categs = $cats->fetchAll('nom asc');  
    $this->view->categs = $categs;  
  
}
```

# Vue catégories

```
<h1>Liste des catégories</h1>
<?php
echo '<ul class="list-group">';
foreach ($this->categs as $categ) {
echo '
    <li class="list-group-item"><a href="categorie?id=' . $categ->id . '">' . $categ->nom . '</a>
<a href="#" class="pull-right"><span class="glyphicon glyphicon-trash"></span></a>
    </li>';
}
echo '</ul>';
```

## S'occuper de l'update de catégorie :

Il faut pouvoir éditer la catégorie créée : pour ça, à partir de la vue liste, on fait un lien sur le formulaire d'update de la catégorie en passant l'ID de la catégorie dans le lien. Pour un comportement différent, on peut créer une vue différente. Usuellement, on utilise la même vue pour l'édition et la création

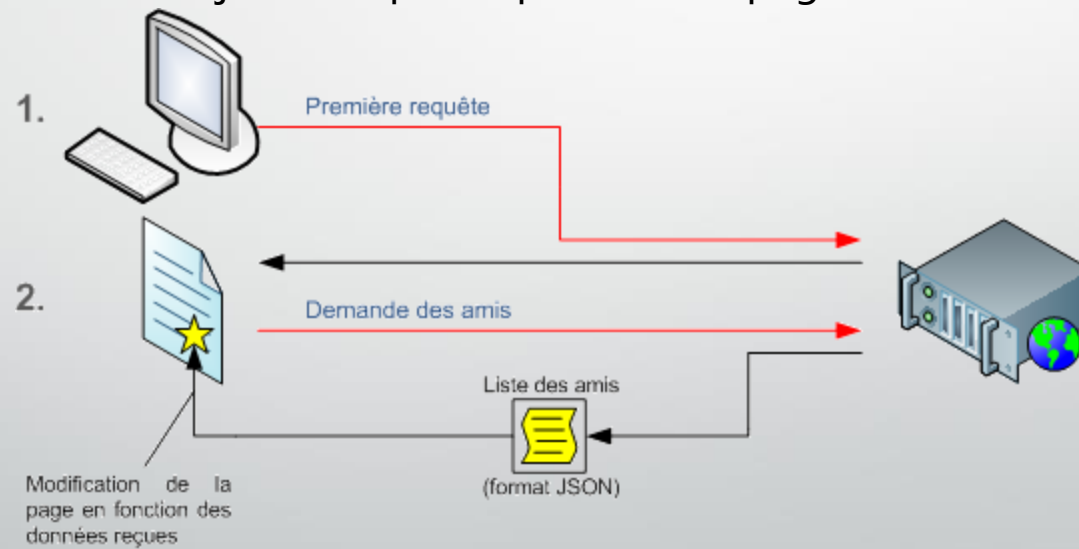
```
echo '<li class="list-group-item"><a href="categorie-update?id='.$categ-  
>id.'">'.$categ->nom.'</a>';
```

On gère dans le controller le paramètre id passé dans l'url

```
public function categorieAction()  
{  
    $id = $_GET['id'];  
    $cats = new Categorie();  
    $categ = $cats->fetchOne($id);  
  
    $this->view->categ = $categ;  
}
```

# Un petit mot sur l'Ajax

La première requête est la même qu'une requête normale – ça on ne sait rien y faire. La différence va résider dans le fait que quand l'utilisateur cliquera sur un lien – ou un autre élément cliquable – la page ne se rechargera pas et le navigateur enverra une requête au serveur, lequel renverra les données demandées dans un format léger – comme le format JSON. Dans ce cas, le serveur n'aura renvoyé qu'un minimum de données ce qui est beaucoup plus léger et donc plus rapide. Le navigateur, par le biais de JavaScript, peut alors mettre à jour une petite partie de la page avec les données reçues du serveur.



On remplit donc le formulaire d'édition et on en profite pour faire un peu d'AJAX

```
<script type="text/javascript">
    $('#ajax').click(function() {
        var cat = $('#categorie').val();
        var id = $('#id').val();
        $.ajax({
            method: "GET",
            url: "updatecategorie?cat="+cat+'&id='+id,
        })
        .done(function( msg ) {
            if (msg == 1){
                $('.hide').attr('class', 'alert alert-success');
                $('.alert-success').html('Changement effectué');
            }else{
                $('.hide').attr('class', 'alert alert-warning');
                $('.alert-warning').html('Problème détecté');
            }
        })
    });
</script>
```



## Et voici l'Action updatecategorie du controleur ajax

```
public function categorieupdateAction() {  
    $nom    = $_GET['cat'];  
    $id      = $_GET['id'];  
    $alias  = Utils::generateSlug($nom);  
  
    $params = [  
        'id' => $id,  
        'nom' => $nom,  
        'slug' => $alias  
    ];  
    $cat = new Categorie();  
    $return = $cat->save($params);  
    echo $return;  
    die();  
}
```

<http://dune-empereur-jeu.fr>

et aussi

<http://wedgeanthill.com>



# On peut maintenant s'occuper de la suppression de la catégorie

A partir de la vue liste des catégories, mettre un lien à l'icone de suppression

```
echo '<a onclick="return confirm(\'êtes-vous sûr de vouloir supprimer cette catégorie ?\')"  
href="?iddelete='.$cat->id.'" class="pull-right"><span class="glyphicon glyphicon-  
trash"></span></a>';
```

On gère dans le controller et on gère dans la vue

```
$delete = $this->_getParam('iddelete');  
if($delete != '')  
    $this->view->delete = $cats->delete($delete);
```

```
<?php if($this->delete>0) {  
    ?>  
    <div class="alert alert-success"><strong>Félicitations !</strong> La catégorie a été supprimée  
avec succès</div>  
    <?php  
} elseif ($this->insertReturn===false) {  
    ?>  
    <div class="alert alert-danger"><strong>ERREUR !</strong> La catégorie n'a pas pu être  
supprimée</div>  
    <?php  
}
```

# S'occuper des films

Afficher

Structure

SQL

Rechercher

Insérer

Export

Import

Privèges

	#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/>	1	id	int(11)			Non	Aucune		AUTO_INCREMENT
<input type="checkbox"/>	2	titre	varchar(255)	latin1_swedish_ci		Non	Aucune		
<input type="checkbox"/>	3	slug	varchar(255)	latin1_swedish_ci		Non	Aucune		
<input type="checkbox"/>	4	image	varchar(255)	latin1_swedish_ci		Non	Aucune		
<input type="checkbox"/>	5	realisateur	int(11)			Non	Aucune		
<input type="checkbox"/>	6	description	text	latin1_swedish_ci		Non	Aucune		
<input type="checkbox"/>	7	date_sortie	date			Oui	NULL		

On refait tout pareil que les catégories –mis à part l'update en ajax : on fait une action addfilm avec la vue qui lui est liée : cette même action gèrera la vue création/édition et son traitement



# Vue formulaire des films

Dans cette vue formulaire, il faut d'abord passer la liste des catégories qu'on peut cocher à partir du controller

```
$cat = new Catégorie();  
$this->view->cats = $cat->fetchAll();
```

Dans la vue formulaire, il faut mettre tous les champs créés en base. On fait de l'upload, il faut donc mettre cet attribut dans le formulaire

```
<form method="POST" enctype="multipart/form-data" action="<?php echo WEB_ROOT; ?>/add/film"  
class="form-horizontal">  
<input type="hidden" name="MAX_FILE_SIZE" value="200000" />
```

En ce qui concerne les catégories et le calendrier :

```
<div class="form-group">  
  <label for="categorie">Catégorie:</label>  
  <?php  
    foreach ($this->cats as $cat) {  
      echo '<div class="checkbox"><label for="'. $cat->id. '">  
        <input type="checkbox" name="categ[]" id="'. $cat->id. '" value="'. $cat->id. '" />'. $cat->nom  
        . '</label></div>';  
    }  
  ?>  
</div>  
  
  <script type="text/javascript">  
    $(function () {  
      $('#datetimepicker1').datetimepicker( {  
        locale: 'fr',  
        format: 'DD/MM/YYYY'  
      });  
    });  
  </script>
```

# On retourne dans le controller de création de film pour gérer les données

```
$titre          = $this->_getParam('titre');
$categ          = $this->_getParam('categ');
$description     = $this->_getParam('description');
$date_sortie    = $this->_getParam('date_sortie');

$filmCateg      = new FilmCateg();
if($titre!='' && count($categ>0) && $description!='' && $date_sortie!=''){

    $alias = Uutils::generateSlug($titre);
    if($_FILES['image']['name']!='') {
        $titleImg = md5(uniqid(rand(), true));
        $extension_upload = strtolower(substr(strrchr($_FILES['image']['name'], '.'), 1));
        $titleImg = $titleImg . '.' . $extension_upload;

if (!isset($_FILES['image']) OR $_FILES['image']['error'] > 0) return FALSE;
//Test: extension

$extensions_valides = array('jpg', 'jpeg', 'gif', 'png');

if (!in_array($extension_upload, $extensions_valides)) return FALSE;
//Test: taille limite
if ($_FILES['image']['size'] > 1500000) return FALSE;
//Déplacement
move_uploaded_file($_FILES['image']['tmp_name'],'images/film/'. $titleImg);

    } else
        $titleImg="default.jpg";
```

```
$date = explode('/', $date_sortie);  
$dateFormatBase = $date[2].'-'. $date[1].'-'. $date[0];
```

```
$params = [  
    'titre' => $titre,  
    'image' => $titleImg,  
    'slug' => $alias,  
    'description' => $description,  
    'date_sortie' => $dateFormatBase,  
    'realisateur' => 0  
];
```

```
$insertReturn = $filmModel->save($params);  
if($insertReturn>0){
```

```
    foreach ($categ as $cate){  
        $params=[  
            'id_film' => $insertReturn,  
            'id_categorie' => $cate  
        ];  
        $filmCateg->save($params);  
    }
```

```
$this->view->reussite = $insertReturn;//s'occuper du message de félicitation  
dans la vue formulaire de création de film
```

# Update du film

Dans la liste des films, faire un lien qui redirige sur la même vue formulaire que l'insertion : on passe en paramètre un id, on récupère les données dans le controller et on renvoie à la vue les données précédemment rentrées

```
if($iParam!='') { //mettre à la fin du controller pour récupérer les nouvelles données
    $this->view->film = $filmModel->fetchOne($iParam);
    $this->view->categs = $cat->categorieFilm($iParam, PDO::FETCH_ASSOC);
}
```

```
<div class="form-group">
    <label for="categorie">Catégorie:</label>
    <?php
        foreach ($this->cats as $cat){
            $attr='';
            if(count($this->categs)==0)$this->categs= array();
            foreach( $this->categs as $categ)
                if($categ['id'] == $cat->id)$attr='checked="checked"';
            echo '<div class="checkbox"><label for="'. $cat->id.'">
                <input '. $attr.' type="checkbox" name="categ[]" id="'. $cat->id.'" value="'. $cat->id.'"
            />'. $cat->nom
                .'</label></div>';
        }
    ?>
</div>
```



# POUR FAIRE L'UPDATE dans le controller

```
if($idParam!='')  
    $filmUpdate = $filmModel->fetchOne($idParam);
```

Rajouter un elseif dans l'insertion d'image

```
elseif($idParam!='')  
    $titleImg= $filmUpdate->image;
```

Et aussi rajouter cette ligne lors de l'insertion

```
if($idParam!='') $params['id'] = $idParam;
```

Pour mettre à jour les catégories, il faut d'abord  
supprimer les liens existants

```
if($idParam!='') $filmCateg->deleteCategFilm($idParam);
```

# On peut maintenant s'occuper de la suppression des films

À partir de la liste, créer un lien de suppression. Faire la suppression dans le controller AVANT le fetchAll

```
$delete = $this->_getParam('iddelete');  
if($delete != '')  
    $this->view->delete = $modelFilm->delete($delete);
```

On renvoie un résultat : penser à faire le message sur la vue. LA GESTION DES MESSAGES D'ERREUR/DE CONFIRMATION se fait normalement à travers les sessions : on mets dans le controller le message qu'on veut dans une session. Dans la vue, on affiche toujours cette session, et si elle est remplie, on la supprime. Ca pourrait faire l'objet d'un développement ultérieur 😊

# AVANT de s'occuper du panier, créer les utilisateurs

On s'occupe d'abord du formulaire d'inscription, puis de la connexion.  
Penser à hasher le password pour le rendre illisible en base.

Dans le controller du login, faire ensuite la connexion : on crée une session USER qui contient les informations du profil de l'utilisateur 😊

```
$login          = $this->_getParam('login');  
$password       = $this->_getParam('password');  
  
if($login!='' && $password!=''){  
    $user = new User();  
    $loginAct = $user->login($login, $password);  
  
    if($loginAct->id>0){  
        $_SESSION['user']['profil'] = $loginAct;  
        $this->view->login=1;  
    }else  
        $this->view->login=false;  
}
```

# DERNIERE VUE DU FILM : LA VUE DETAIL

Dans cette vue, on pourrait justement avoir accès au panier ! – seulement pour les inscrits

```
<p><button class="panier btn btn-default btn-sm" role="button">  
    <span class="glyphicon glyphicon-shopping-cart"></span> <strong>Ajouter au panier</strong>  
</button></p>
```

On doit donc prévoir de l'ajax pour ajouter l'article au panier. Penser à mettre un hidden id du film pour pouvoir le récupérer en js (plus propre que de le récupérer dans l'url)

```
<input type="hidden" id="id" value="<?php echo $this->id;?>" />
```

Faire un click sur le bouton, récupérer l'id et mettre en ajax

```
$.ajax({  
    method: "GET",  
    url: "ajaxpanier?add="+id  
})  
    .done
```

Mettre un message de confirmation dans le done

```
$('.hide').attr('class', 'alert alert-success');  
$('.alert-success').html('Ajout au panier effectué');
```

## Action ajaxpanier

```
$add = $this->_getParam('add');  
$film = $modelFilm->fetchOne($add);  
$_SESSION['user']['panier'][$film->id] =  
[  
    'film' => $film  
]
```

# Le panier !

Cette vue liste simplement le contenu de la session 'panier'. On peut ensuite générer un pdf, en utilisant la lib « fpdf » : il faut vérifier qu'elle marche bien, puis ensuite l'override pour lui permettre d'afficher un tableau dans un PDF :

<http://www.fpdf.org/en/script/script70.php>

<https://github.com/Althenar/pracownia/tree/master/inc/fpdf>

## MAIS VOYONS D'ABORD NOTRE PANIER

# Un p'tit coup de pub (subtil) ?

- <https://www.facebook.com/duneJeuPointAndClick/>
- <http://dune-empereur-jeu.fr>
- <http://wedgeanthill.com>

## Développe Ta Fourmilière



Ecitoninae

**Cunégonde**

Attaque : 3

Fécondité : 2

Robuste : 1

Nourriture : 527

Population : 2

Bois : 12

Pierre : 11

Humus : 0

Cuivre : 0

Champignon : 0

Morts : 165

Soldat : 0

Récupérer Des Graines

0

Rentrer La Nourriture Dans La Colonie

Faire Des P'tites Larves

0

Mettre Les Larves En Couveuse

### Améliorer La Colonie !

☀ ETE (☆☆)

150 Population

30 Bois

40 Pierre

800 Nourriture

🍂 AUTOMNE (☆☆☆)

350 Population

60 Bois

60 Pierre

1800 Nourriture

40 humus

🍷 HIVER (☆☆☆☆)

500 Population

100 Bois

100 Pierre

2500 Nourriture

100 Humus

70 Champignon

100 Soldat

# PANIER

```
if($_SESSION['user']['profil']->id=='')
    header('Location: main');
$this->view->panier =
($_SESSION['user']['panier'])?$_SESSION['user']['panier']:array();
```

```
<h1>Mon panier</h1>
<table class="table table-striped" width="647">
    <thead>        <tr>                <th>Film</th>                <th>Quantité</th>
                <th>Modifier la quantité</th>
    </tr>        </thead>        <tbody>
<?php
foreach ($this->panier as $key =>$panier){

    echo '<tr class="row_'.$panier['film']->id.'">
        <td><a href="film?id='.$panier['film']->id.'">'.$panier['film']->titre.'</a> </td>
        <td><span class="quantity_'.$panier['film']->id.'">'.$panier['quantite'].'</span></td>
        <td>
            <button id="remove_'.$panier['film']->id.'" class="btn btn-default btn-sm glyphicon glyphicon-minus"></button>
            <button data-id="'.$panier['film']->id.'" class="btn btn-default btn-sm glyphicon glyphicon-plus"></button>
        </td>
    </tr>';
}
echo '</tbody></table>';
```



# Le javascript du panier

L'idée de mettre les attributs data-id dynamique est immensément géniale:

C'est ainsi qu'on récupère en js dans une liste d'élément, un élément choisi

```
$( '.glyphicon-plus' ).click( function() {  
    var id = $( this ).data( 'id' );  
    $.ajax({  
        method: "GET",  
        url: "ajaxpanier?add="+id  
    })  
    .done( function( msg ) {  
        $( '.quantity_'+id ).html( parseInt( $( '.quantity_'+id ).html(), 10 ) + 1 );  
        $( '.nbPanier' ).html( parseInt( $( '.nbPanier' ).html(), 10 ) + 1 );  
    });  
});
```

IL FAUT METTRE UNE NOTION DE QUANTITE DANS NOTRE SESSION PANIER. FAISONS D'ABORD LE MINUS, ET ON S'EN OCCUPE ! On bouchonne l'ajax pour le moment.

Pour le minus, j'ai conservé les ID dynamique juste pour faire une sélection en jQuery d'une div « commençant par »

Ce n'est pas la bonne méthode. Il faut faire comme vue précédemment. Mais c'est quand même intéressant.

```
$('button[id^="remove_"]').click(function() {  
    var id = $(this).attr('id');  
    var id = id.replace('remove_', '');
```

# Ajaxpanier et quantité

```
$add = $this->_getParam('add');
$remove = $this->_getParam('remove');

$modelFilm = new Film();
$film = $modelFilm->fetchOne(($add)?$add:$remove);

$quantity = ($_SESSION['user']['panier'][$film->id])?$_SESSION['user']['panier'][$film->id]['quantite']:0;

if($add!='')
    $_SESSION['user']['panier'][$film->id] = [
        'quantite' => $quantity+1,
        'film'      => $film
    ];

if($remove!='')
    $_SESSION['user']['panier'][$film->id] = [
        'quantite' => $quantity-1,
        'film'      => $film
    ];

if($_SESSION['user']['panier'][$film->id]['quantite']==0)
    unset($_SESSION['user']['panier'][$film->id]);
die();
```

## 7. Filtre de recherche :

Pour la recherche par nom de film, on fait juste un formulaire qui redirige sur l'action de l'index. On fait une requête de recherche dans le controller

```
$searchTitre = $this->_getParam('titre');
```

```
$films = $modelFilm->fetchAll('titre asc');  
if($searchTitre!='')  
    $films = $modelFilm->searchByTitre($searchTitre);
```

# RECHERCHE PAR CATEGORIE

```
$this->view->allCateg = $categFilm->fetchAll('nom asc');
```

```
<div class="col-md-3"><h4>Recherche par catégorie</h4>
<ul>
    <?php foreach($this->allCateg as $categ){
        echo '<li class="list-group-item">
<input type="checkbox" value="'. $categ->id. '"/>
    <a href="#"><label for="check_'. $categ->id. '">'. $categ->nom. '</label></a></li>';
    }?>
</ul>
</div>
<div class="col-md-9">
    <table class="table table-striped" width="647">
```

# Le jQuery qui va avec :

```
<script>
    $('input[type="checkbox"]').click(function() {
        checkedArray = new Array;
        $('input[type="checkbox"]:checked').each(function() {
            checkedArray.push($(this).val());
        });
        $.ajax({
            method: "POST",
            url: "searchajaxparcategorie",
            data: {categ: checkedArray}
        })
        .done(function( msg ) {
            $('.col-md-9').html('');
            $('.col-md-9').html(msg);
        });
    });
</script>
```

Usuellement, l'ajax renvoie les données en json : on déparse le retour de l'ajax et on reconstruit le DOM en js. C'est la manière optimale de fonctionnement.

ICI, on est plus fainéant. On renvoie de l'HTML brut qu'on se contente d'afficher

## 8. Gérer les droits utilisateurs

Le ROLE du user doit être testé avant chaque vue admin.

ATTENTION : LE MENU NE GERE QUE  
L’AFFICHAGE, PENSEZ A FAIRE LE IF DANS LES  
ACTIONS DU CONTROLLER !

# La petite touche en plus de l'étoile sur le menu panier

Je fournis le DOM, vous fournissez le js !

```
<?php } if($_SESSION['user']){
    ?><li><a class="pull-left menu-panier" href="<?php echo WEB_ROOT; ?>/panier">Panier</a>
<?php
    if(count($_SESSION['user']['panier'])>0){
        $total = 0;
        foreach($_SESSION['user']['panier'] as $panier){
            $total+=$panier['quantite'];
        }
        echo '<span class="pull-left fa fa-certificate"></span>
        <span class="nbPanier">'.$total.'</span>';
    }
} ?></li>
```



# La Génération du PDF

La librairie FPDF a bien été installée dans le dossier lib.

On crée un lien dans le panier qui redirige sur la création du PDF : on passe en donnée le panier, on construit le tableau, et on le passe à l'objet PDF qui gère la chose.

En passant, on oublie pas de vider la session PANIER

```
<?php
$html='
<table>
    <tr>
        <td>Film</td>
        <td>Quantité</td>
    </tr>';
foreach ($this->panier as $key =>$panier){
    $html.='<tr>
        <td>' . $panier['film']->titre . ' </td>
        <td>' . $panier['quantite'] . ' </td>
    </tr>';
}
$html.='</table>';

$pdf=new PDF_HTML_Table();
$pdf->AddPage();
$pdf->SetFont('Arial','',10);
$pdf->SetMargins(0, 0);
$pdf->WriteHTML(utf8_decode("<h1>Facture envoyée</h1><br>$html<br>Merci de votre
visite !"));
$pdf->Output();
```

# IDEE POUR AMELIORER LE PROJET

Faire une pagination de la liste des films, bien sûr !

# MERCI POUR TOUT !!!! Des questions ?

joachim@pulsar-informatique.com

Merci pour tout ! Vous êtes ma classe d'élève préférée !  
(enfin, vous êtes aussi la seule 😊 - pour le moment)

Ca a été une semaine très agréable, on a bien bossé ensemble, à fond, et ça restera pour moi une expérience très riche, dans bien des domaines 😊

Bossez bien, lâchez rien, et je vous souhaite à tous une réussite professionnelle ( et pas que d'ailleurs) !

Allez, je retourne dans mon 60, le char à bœuf a une roue embourbée, j'attrape le premier mouton qui passe et je taille la route pour rentrer ! ++