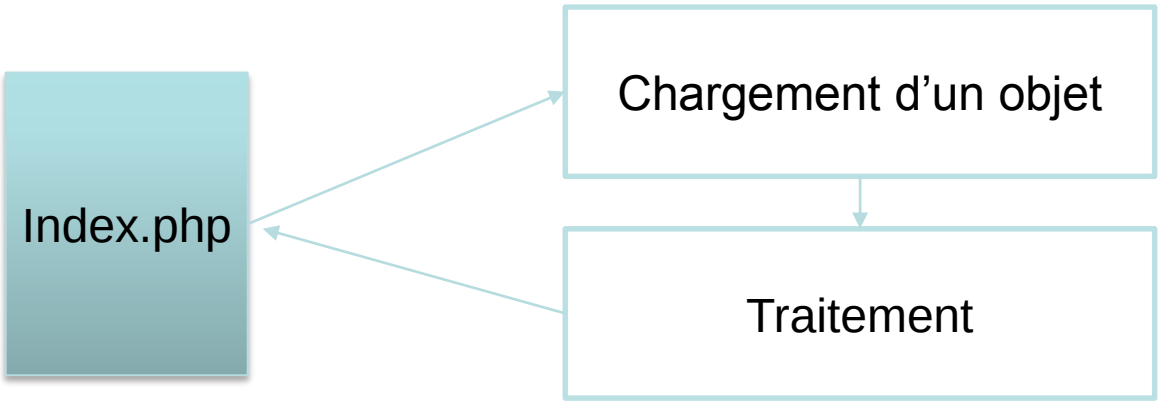
PHP

Architectures avancées

Vers des architectures plus évoluées

- Séparation du traitement des données
- Utilisation de plusieurs classes à partir du CGI initial
- Modèle Service to Workers
- Modèle vue/contrôleur



Évite la redondance

Facilite la réutilisation

Vers des modèles plus évolués

Quels exemples de classes donneriez vous ?????

Comment charger une classe ?

- Require
- Mieux : autoloader
  - *http://php.net/manual/fr/language.oop5.autoload.php, que vous pouvez faire vous-même*
  - *Autoloader PSR-0, PSR-4 si utilisation de Namespaces, http://www.php-fig.org/psr/psr-0/fr/, http://www.php-fig.org/psr/psr-4/*

MVC DE BASE ET OPTIMISATION

Vers des modèles plus évolués

A ce stade, si l'on vous demande d'implanter une appli Web, vous devriez faire:

- Des pages d'erreurs (au moins une)
- Des classes : BD? Validation, authentification?, autoloader

1. Toujours valider les données utilisées: données de formulaire, de session, de cookies etc.,

   - Fonction Filter_var() PHP *http://php.net/manual/fr/book.filter.php*
   - Par exp. régulière;
   - Par isset, etc.

Vers des modèles plus évolués

2. Nettoyage :
   - Fonction Filter_var() avec filtres (FILTER_SANITIZE_EMAIL) :*http://php.net/manual/fr/filter.filters.sanitize.php*
   - libs comme *HTML_Purifier*

Critique de cette approche:

- Tout est mélangé (traitement données, affichage, validation, etc.)

- Difficile à maintenir

Architecture via pattern MVC

Modèle vue contrôleur

- séparer l'affichage du traitement
- Définition de rôles (qui fait l'affichage, etc.)
- utilisation de plusieurs fichiers et classes (traitement, affichage, etc.)

Programmation extrêmement répandue en Web !

ex: Servlet/JSP (struts, spring) en Java

ex: php.mvc, symfony, zend,Yii, etc. en php

Architecture via pattern MVC

Modèle vue contrôleur

Plusieurs possibilités:

1. Développement d'un MVC complet qui est proche du problème traité (peu fréquent)

2. Utilisation d'un framework MVC

   - Nécessité de comprendre, de se former mais ensuite rapidité d'implantation, garantie qualité de certaines parties (DAL, etc.)

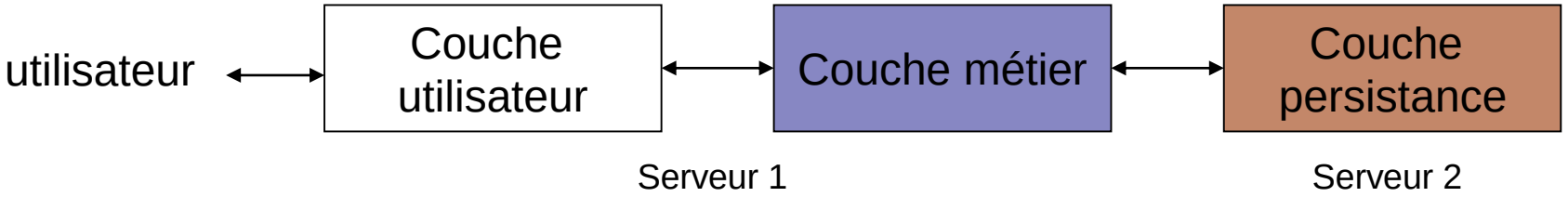
   - Il faut comprendre le MVC -> retour à 1. dans la suite
   - Et pourquoi de ce cours

Architecture MVC

Pourquoi utiliser des MVC

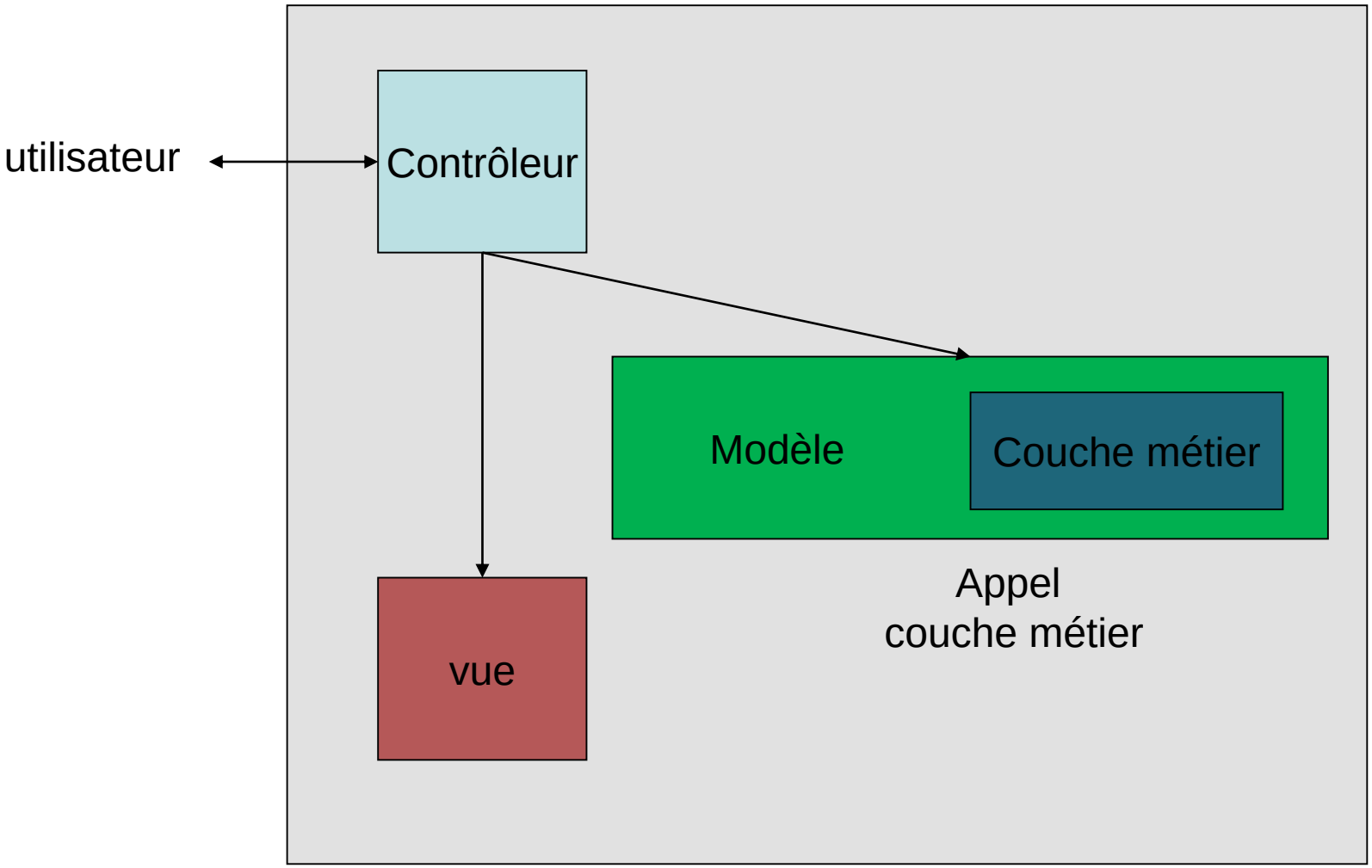
- Pour casser les pieds des étudiants…
- Pour obtenir une architecture lisible, facilement modifiable et interchangeable

Aujourd'hui, une application web, est au moins basée sur une architecture 3 tiers



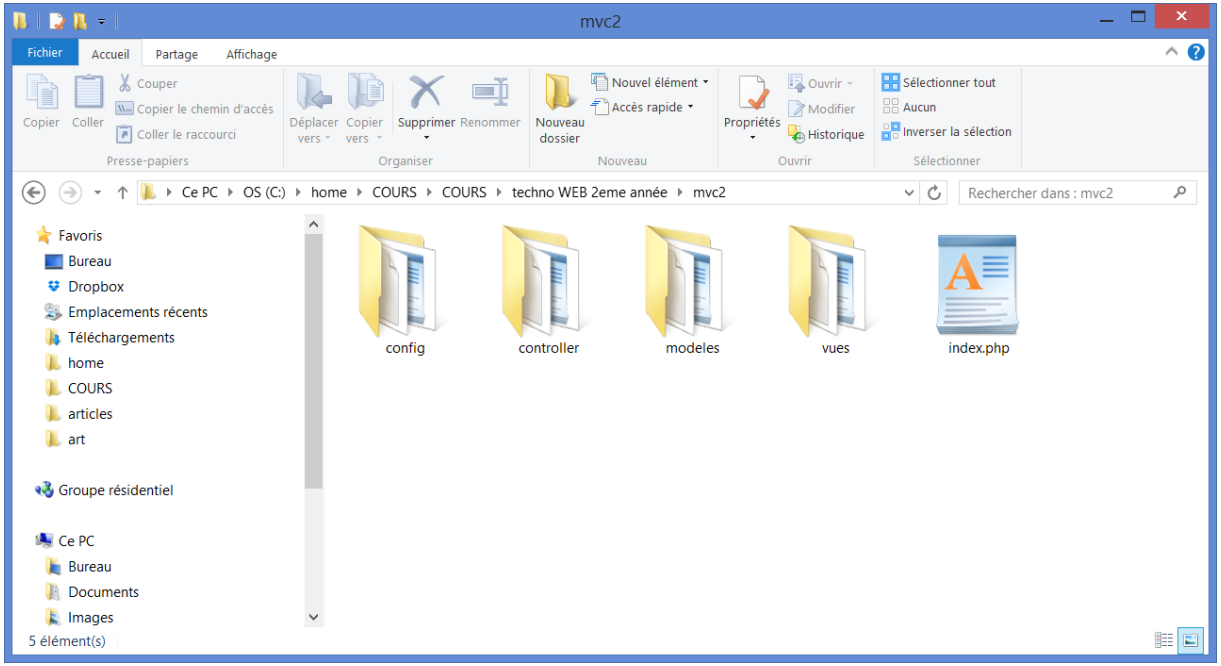
Le MVC prend place dans la première couche

Architecture MVC de base



Architecture MVC de base

Organisation des répertoires:



Index.php =>

```
<?php
//si contrôleur pas une classe, cas simpliste
  header('Location: controller/controller.php');
?>
```

Contrôleur dans MVC

Contrôleur

C'est toujours le contrôleur qui est appelé

Son rôle:

1. Contrôler l'intégrité des données reçues (Validation)

   (peut aussi être fait par modèle)

2. Lecture d'une action:

   Action: indique le traitement (demande d'infos bd, ….)

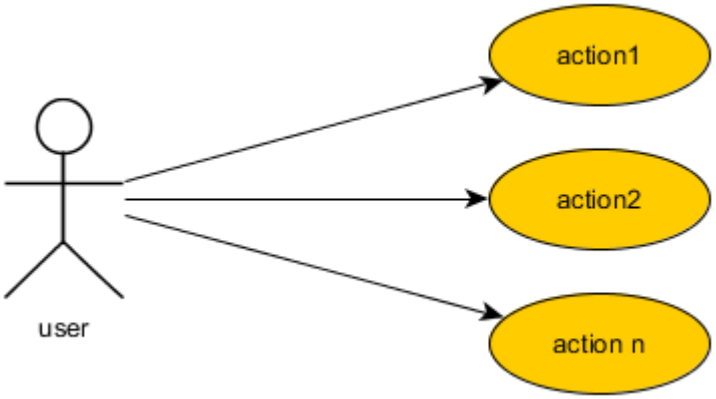
3. Contrôleur crée un modèle, gestion des données (C.R.U.D.)

4. Redirection vers la vue en donnant les données (=>page réponse ou une page d'erreur)

Contrôleur gère les erreurs (classiques + exceptions !!!)

Contrôleur dans MVC

Les actions ?




```
<?php
Class ControllerUser {

Action1 -> execution 1
…
Action n -> execution  n
?>
```

Ex: connecter, ajouterLivre, supprimerLivre, NULL (pas d'action en général 1er appel)

=> Toujours commencer par effectuer une analyse !!!

Contrôleur dans MVC

Les actions ?

Comment les coder ?

Sur un lien:
<a href=http://…?action=etre_attentif

Dans un formulaire
<form action=http://…?action=etre_attentif

Ou

<input type=hidden name=action…>

Contrôleur dans MVC

```php
<?php

//chargement biblio

//chargement config
include_once(__DIR__.'/../config/Config.php');

//debut

//on initialise un tableau d'erreur
$dVueEreur = array ();

try{
$action=$_REQUEST['action'];

switch($action) {

//pas d'action, on réinitialise 1er appel
case action1:
        action1();
        break;
```

Contrôleur dans MVC

```
//mauvaise action
default:
$dVueEreur[] = "Erreur d'appel php";
require (__DIR__.'/../vues/vuephp1.php');
break;
}

} catch (PDOException $e)
{
        //si erreur BD, pas le cas ici
        $dVueEreur[] = "Erreur inattendue!!! ";
        require (vuephp_erreur);

}
catch (Exception $e2)
        {
        $dVueEreur[] = "Erreur inattendue!!! ";
        require (vuephp_erreur);
        }
//fin
exit(0);
```

Contrôleur dans MVC

```
function action1() {
//traitement
$mdl = new modele();
$mdl->traitement();

$dVue = array (
        //données
        );
        require (vuephp);
}
?>
```

Contrôleurs toujours identiques, pas observeur/observable
=> Facile à faire depuis une analyse !

Modèle et MVC

Modèle

Couche qui manipule les données

peut valider les données

utilise une classe BD « pour se connecter à la base »

stocke des infos

fait appel à la base pour extraire des informations, retourne une chaîne ? Une liste de chaîne ou un objet ?

Voir plus loin (DAL, ORM? Etc.)

Modèle = classe, pas d'HTML ou autre

Session, cookies ? Idéalement gestion par un modèle

Modèles et MVC

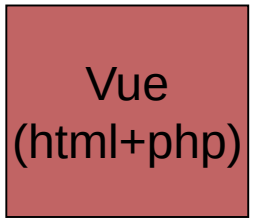
Modèle

Parfois, il n'y a pas de modèle

Parfois, c'est une simple chaîne (String)

Et d'autres fois c'est un objet qui se connecte à une DAL (Data Access Layer= couches métiers)

Vues et MVC



Vue : page html+php qui produit la page HTML (php est utilisé en moteur de template)

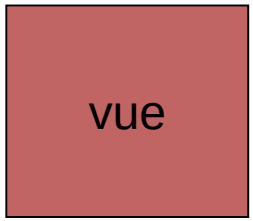
1. Récupère des données produites depuis le contrôleur ou le modèle
2. Crée la page avec ces données

Attention: ne se connecte pas à la base ou ne modifie pas des données => c'est le modèle qui le fait !!!

On évite: $_GET, $_POST, $_SESSION, etc. car ces données ne sont pas validées !

Les boutons (ou autres) de la vue font appels uniquement au contrôleur en lui donnant des paramètres et une action !

Vues et MVC



La vue est décomposée en 6 parties : header, entete, menu, menu contextuel, contenu, footer

Les boutons (ou autres) de la vue font appels uniquement au contrôleur en lui donnant des paramètres et une action !

La vue affiche, (utiliser <?= ?> echo short tag si possible), pas de decl. de fonction

Lien contrôleur - modèle

Contrôleur

Récupération action

Try{
Test action1
appel méthode1

Test action2
appel méthode2
…
}
Catch
{page erreur}
Catch…

Methode 1
Appel modele 1
(new modele1)

Modele1

Méthode globale 1
throws

Méthode globale 2
throws

Modele2

Méthode globale 1
throws

Méthode globale 2
throws

Couche persistance

Factory d'objets métiers

Couche métier

ORMs

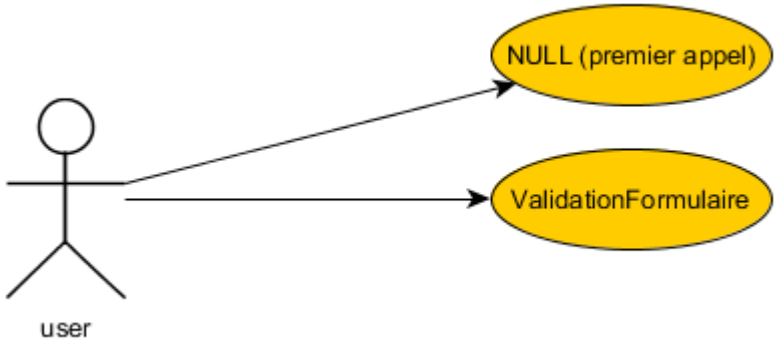
Pas de panique !
On voit plus loin

MVC: un exemple

Récupérable en entier là!!!!!

Site qui demande un nom, un age via un formulaire

retourne le même formulaire rempli avec le nom et l'age donné, + gestion des erreurs (de base)



Vues: erreur, vuephp1
Modele: SimpleModele qui ne fait rien ici

MVC: un exemple

Vuephp d'erreur

```
<!DOCTYPE html>
<html lang="fr">
<body>

<h1>ERREUR !!!!!</h1>
<?php
if (isset($dVueErreur)) {
foreach ($dVueErreur as $value){
    echo $value;
}
}
?>



</body> </html>
```

données obtenues par la variable $VueErreur

MVC: un exemple

‘vuephp1.php'

```
<!DOCTYPE html>
<html lang="fr">
…<?php

// on vérifie les données provenant du modèle -> pas obligatoire si site bien
administré
if (isset($dVue))
{?>
<center>

<?php
if (isset($dataVueEreur) && count($dataVueEreur)>0) {
echo "<h2>ERREUR !!!!!</h2>";
foreach ($dVueEreur as $value){
    echo $value;
}}
?>
<h2>Personne - formulaire</h2>
<hr>
<!-- affichage de données provenant du modèle -->
<?= $dataVue['data']  ?>
```

MVC: un exemple

```
<form method="post" >
<table> <tr>
<td>Nom</td>
<td><input name="txtNom" value="<?= $dataVue['nom']  ?>" type="text" size="20"></td>
 </tr>
<tr><td>Age</td>
<td><input name="txtAge" value="<?= $dataVue['age'] ?>" type="text" size="3"></td>
</tr><tr></table>
<table> <tr>
<td><input type="submit" value="Envoyer"></td>

<!-- action !!!!!!!!!! -->
<input type="hidden" name="action" value="validationFormulaire">
</form></center>

<?php }
else {
print ("erreur !!<br>");
print ("utilisation anormale de la vuephp");
} ?>

</body> </html>
```

Donne ici l'action du contrôleur

MVC: un exemple

Squelette du contrôleur:

```
<?php
//chargement biblio
require_once(__DIR__.'/Validation.php');
require_once(__DIR__.'/../modeles/Simplemodel.php');
//chargement config
include_once(__DIR__.'/../config/Config.php');

//debut
//on initialise un tableau d'erreur
$dataVueEreur = array ();

try{
$action=$_REQUEST['action'];

switch($action) {
```

MVC: un exemple

```
//pas d'action, on réinitialise 1er appel
case NULL:
        Reinit();
        break;
case "validationFormulaire":
        ValidationFormulaire();
        break;

//mauvaise action
default:
$dataVueEreur[] =   "Erreur d'appel php";
require (__DIR__.'/../vues/VueErreur.php');
break;
}
catch (Exception $e2){…}

//fin
exit(0);
```

On traite l'action dans une méthode à part (qui peut être dans un autre fichier

Redirection vers page erreur

Redirection vers page erreur

MVC: un exemple

```
function Reinit() {
$dataVue = array (
            'nom' => "",
            'age' => 0,
            );
            require (__DIR__.'/../vues/vuephp1.php');
}
```

Redirection vers page formulaire

MVC: un exemple

```
function ValidationFormulaire() {
//si exception, ca remonte !!!
$nom=$_POST['txtNom']; // txtNom = nom du champ texte dans le formulaire
$age=$_POST['txtAge'];
Validation::val_form($nom,$age,$dataVueEreur);

$model = new Simplemodel();
$data=$model->get_data();

$dataVue = array (
            'nom' => $nom,
            'age' => $age,
      'data' => $data,
            );
            require (__DIR__.'/../vues/vuephp1.php');
}

?>
```

ICI
Validation simple
MAIS aussi
Filtrage éventuellement
Ici ou dans modèle

Redirection vers page formulaire

MVC: un exemple

ValidationFormulaire=> valide les paramètres recus

```
class Validation {

static function val_form($nom,$age,&$dataVueEreur) {
$b=TRUE;
if (!isset($nom)||$nom=="") {
        $dataVueEreur[] ="pas de nom";
        $b=FALSE;
        throw new Exception('pas d\'action');
}
if (!isset($age)||$age=="") {
        $dataVueEreur[] ="pas d'age ";
        $b=FALSE;
}
return $b;
}} ?>
```

Mais on peut aussi faire partout: throw new Exception(‘blabla');

Optimisation de l'architecture précédente

Critique précédente architecture :

- Pas tout objet !
- Pas d'autoloader, un require par classe…
- Des url (de vues) en dur dans le code -> utilisation du fichier de config (ou d'une classe !)
- 2eme architecture

Exemple précédent revisité et dispo ici !!!!

Optimisation de l'architecture précédente

Index.php

```
<?php
//chargement config
require_once(__DIR__.'/config/config.php');

//chargement autoloader pour autochargement des classes
require_once(__DIR__.'/config/Autoload.php');
Autoload::charger();

//chargement controleur
$cont = new controller();
?>
```

Optimisation de l'architecture précédente

config.php

```
<?php
//préfixe
$rep=__DIR__.'/../';
// liste des modules à inclure
$dConfig['includes']= array('controller/Validation.php');

//BD
$base="sasa";
$login="";
$mdp="";

//Vues
$vues['erreur']='vues/erreur.php';
$vues['vuephp1']='vues/vuephp1.php';
```

Optimisation de l'architecture précédente

Classe Autoloader (exemple très simple, non psr0, psr4 !)

Singleton !

```
class Autoload
{
        private static $_instance = null;
        public static function charger()
    {
        if(null !== self::$_instance) {
            throw new RuntimeException(sprintf('%s is already started', __CLASS__));
        }
        self::$_instance = new self();
```

Optimisation de l'architecture précédente

```
…
…

private static function _autoload($class)
    {
        global $rep;  $filename = $class.'.php';
        $dir =array('modeles/','./','config/','controleur/');

foreach ($dir as $d){
        $file=$rep.$d.$filename;
             if (file_exists($file)){
            include $file;      } }

    }
}
```

Nom des répertoires

Optimisation de l'architecture précédente

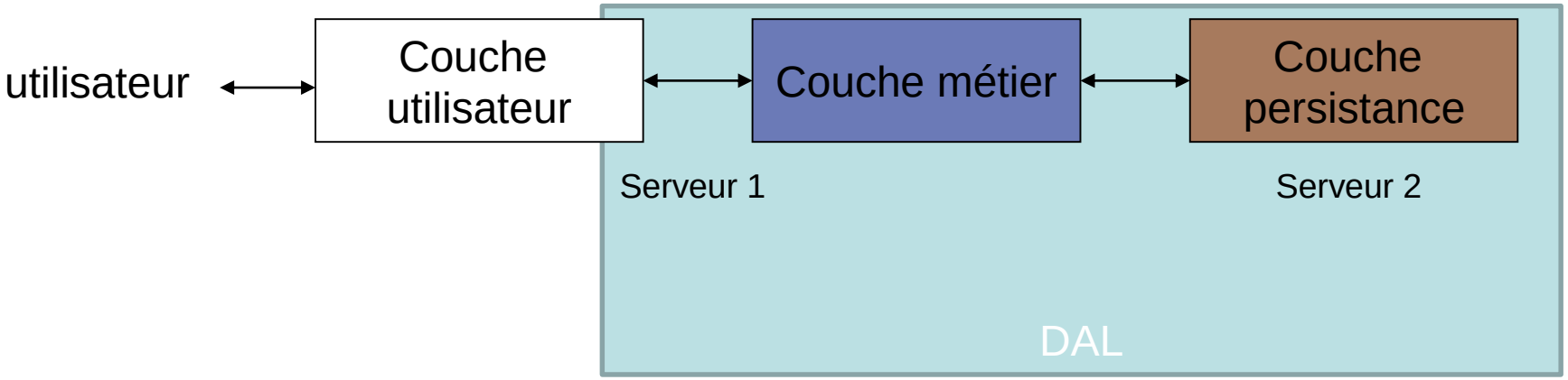
Classe Controleur

```
<?php

class controller {

function __construct() {

// on démarre ou reprend la session
session_start();

//debut
…
try{
$action=$_REQUEST['action'];
switch($action) {

//pas d'action, on réinitialise 1er
appel
…}
} catch (PDOException $e)
{
        //si erreur BD, pas le cas ici
        $dataVueEreur[] =   "Erreur
inattendue!!! ";
        require ($rep.$vues['erreur']);
}

catch (Exception $e2)
        {
        $dataVueEreur[] =   "Erreur
inattendue!!! ";
        require ($rep.$vues['erreur']);
        }
}//fin constructeur

function Reinit() {
…
        require ($rep.$vues['vuephp1']);}

function ValidationFormulaire() {

}//fin class.?
```

DATA ACCESS LAYER

Présentation DAL

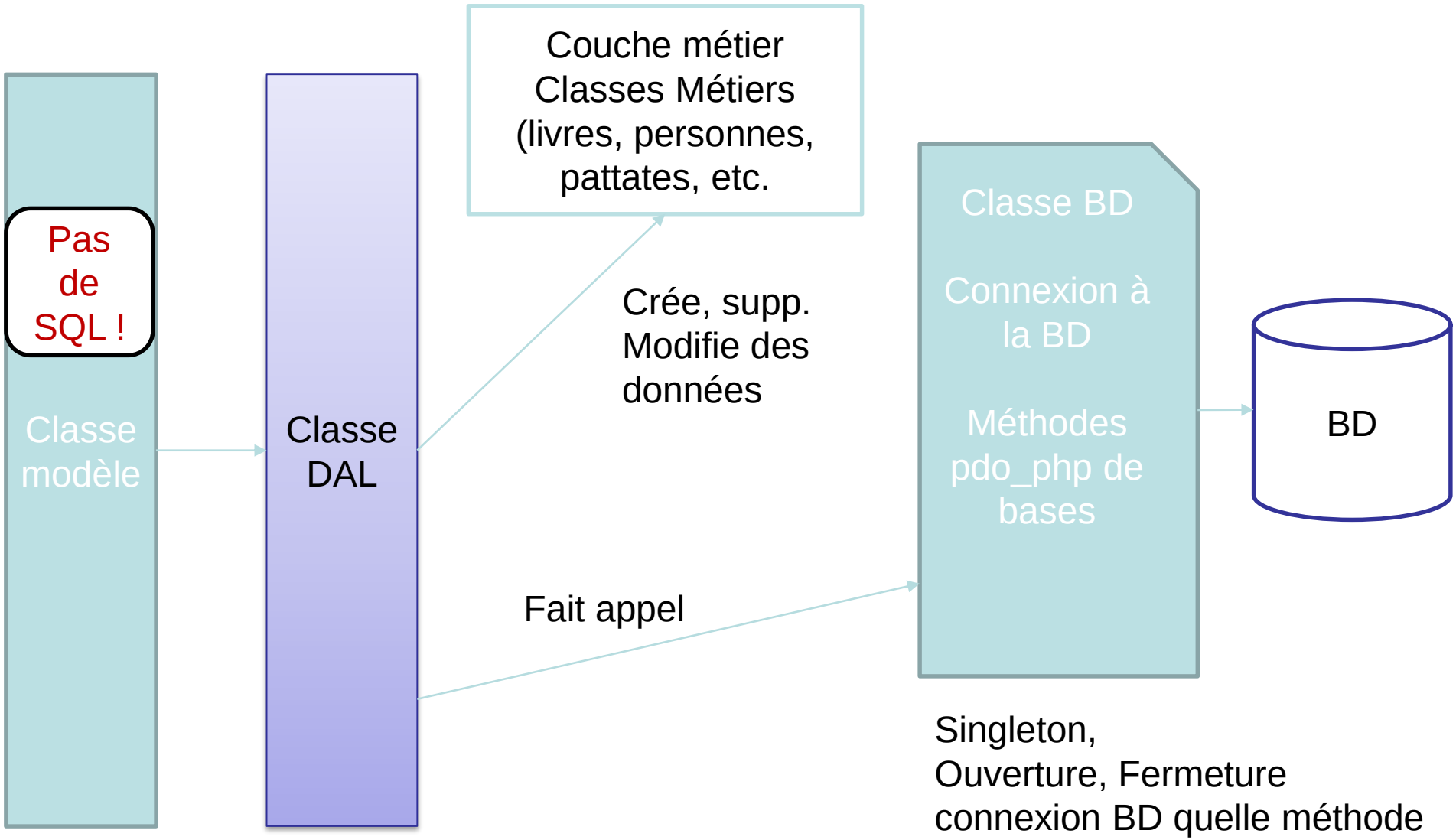
- Couche d'accès aux données (DAL)



- Couche généralement composée de plusieurs classes
  - Permet de gérer les données à la sauce Objet
  - Pas de mysql_* n'importe ou, n'importe quand

- on y trouve des patrons : singleton, factory, voire stratégie

Architecture DAL



Architecture DAL

Classe BD

Connexion à la BD

Méthodes pdo_php de bases

Singleton,

Ouverture connexion BD quelle méthode ?

Fermeture connexion BD quelle méthode ?

Query($sql) -> $résultats

Update($sql)

Bien faire la gestion des erreurs !!!
Pdo renvoie des Exceptions, vous pouvez aussi en ajouter !

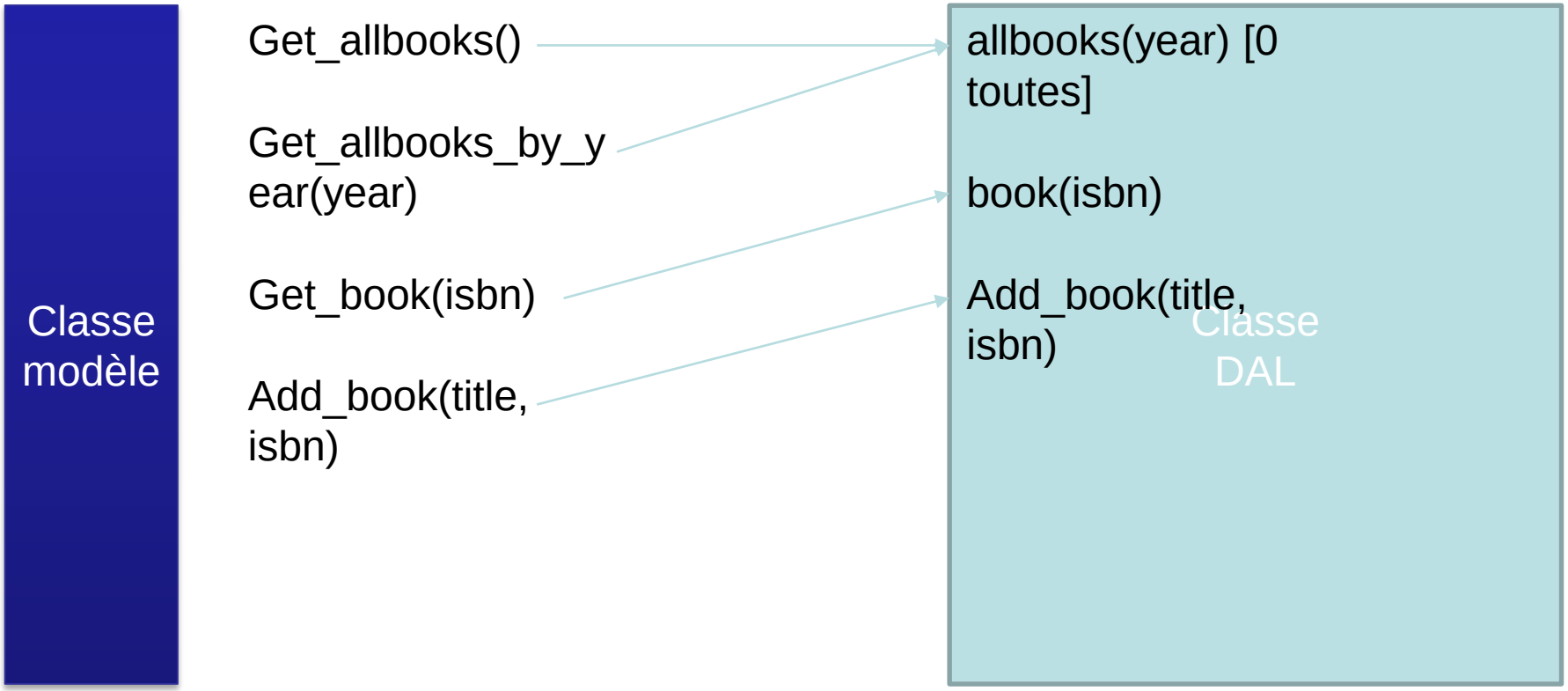
Quiz : exceptions reçues ou ?

Architecture DAL

Exemple: on veut gérer des livres

Couche métier

Classe Livre (POPO)
Isbn, titre, getter, setter, etc.



Architecture DAL

Exemple: on veut gérer des livres

Classe modèle

```
Public function get_allbooks($year){
$dal = new DAL();
$dal->allbooks($year);}
```

Ou faire une classe DAL avec méthodes statique…

Architecture DAL

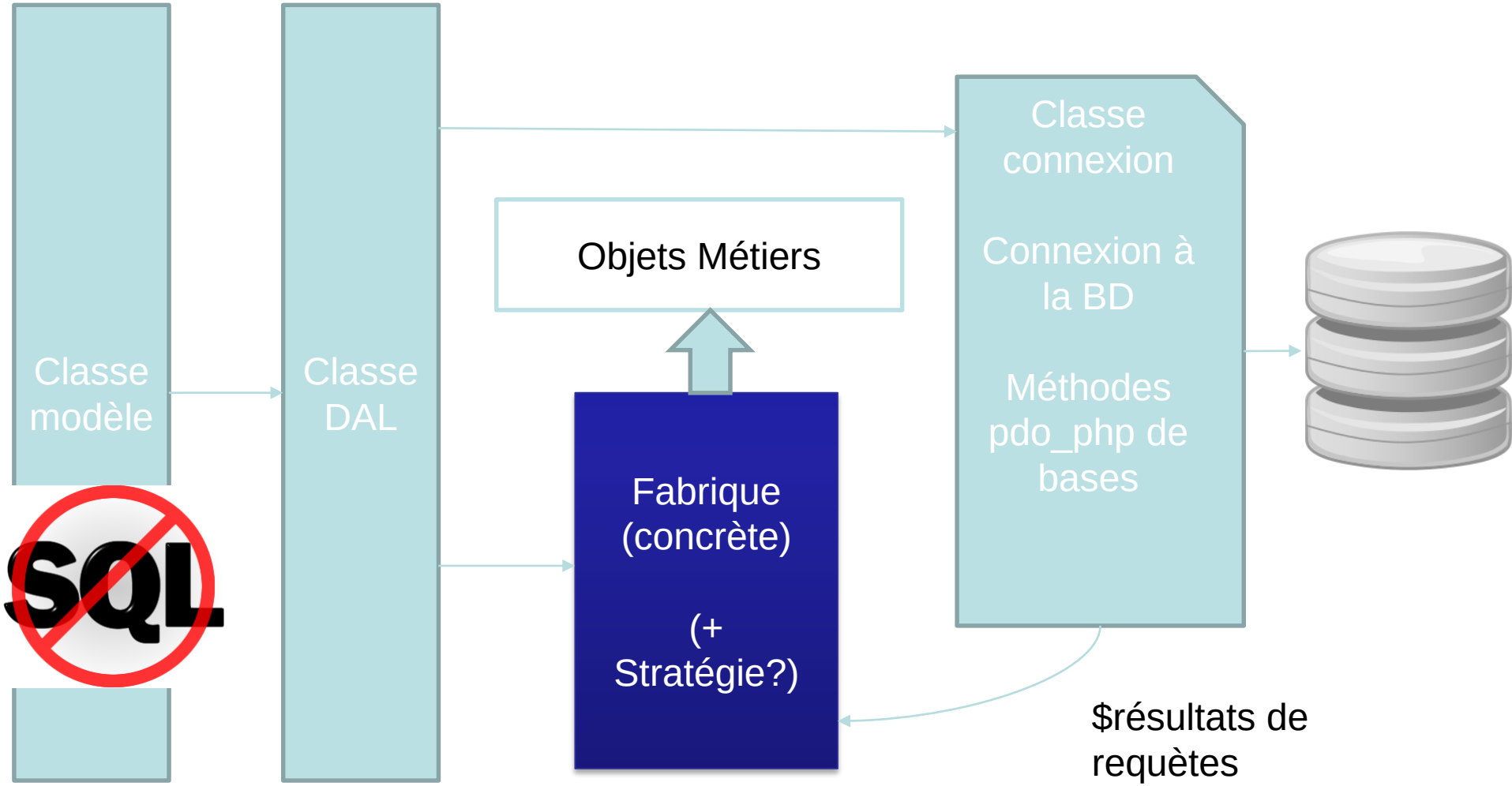
Exemple: on veut gérer des livres

Classe DAL

```
Public function allbooks($year){
//préparation requète sql
If $year<>0
        $req_sql=« select * from Books where
        year='$year' »;
…
//Appel classe BD
$resultats=BD::gestInstance()->query($req_sql);

//création de livres
While ( $ligne = $resultats->fetch() )
        Livre l = new Livre ($ligne->titre, $ligne->isbn);
….
}
```

Exemple classe Connexion (http://www.apprendre-php.com/tutoriels/tutoriel-47-classe-singleton-d-accs-aux-sgbd-intgrant-

Architecture DAL, optimisation



Architecture DAL, optimisation

Faire une DAL au moins une fois pour comprendre c'est bien
Mais à chaque fois, c'est lourd !

Pas de SQL !

Classe modèle

doctrine

propel
smart, easy object persistence

ORM
(object relational mapping)

Doctrine, propel

On donne la couche métier, relations entre classes et tables et c'est tout (gen. de code)

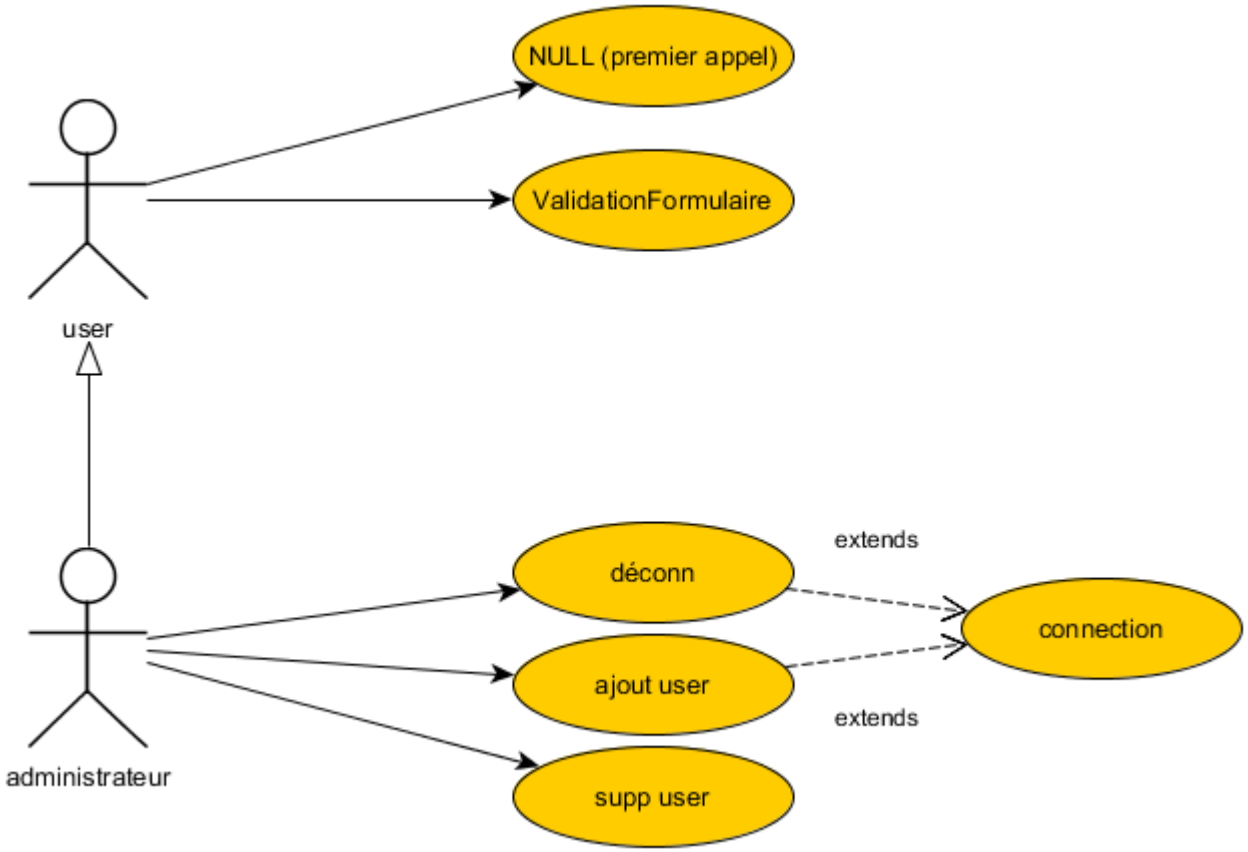
GESTION DES ACTEURS !

Des acteurs … quel cinéma…

- Plusieurs acteurs peuvent être utilisateurs d'une application Web:
  - Visiteur
  - Utilisateur
  - Administrateur
  - ….

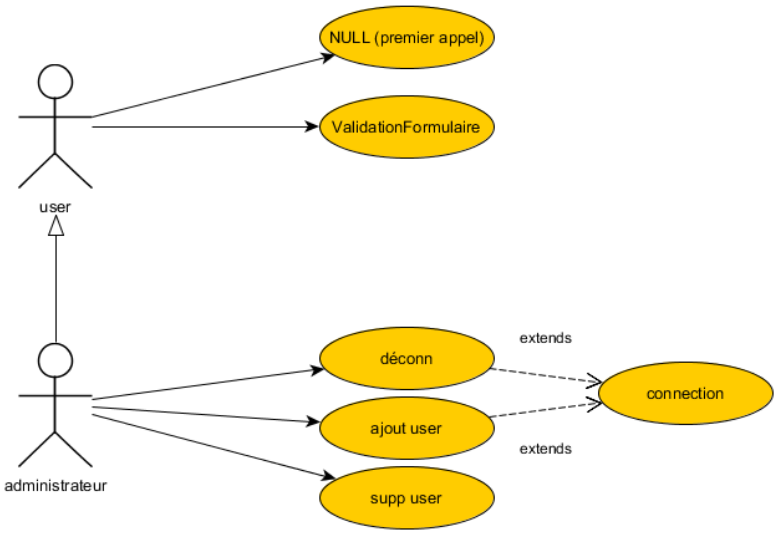
Des acteurs

- En UML, exemple:



Des acteurs

- En UML, exemple:



Contrôleur User

2 actions

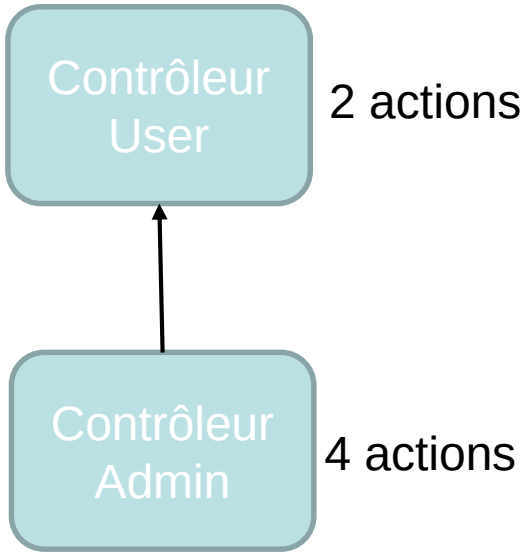
Contrôleur Admin

4 actions

+:
on fait l'analyse, et on a l'architecture !
C'est clair, propre, maintenable, pas 50 questions à se poser

Des acteurs

- Comment lier les contrôleurs

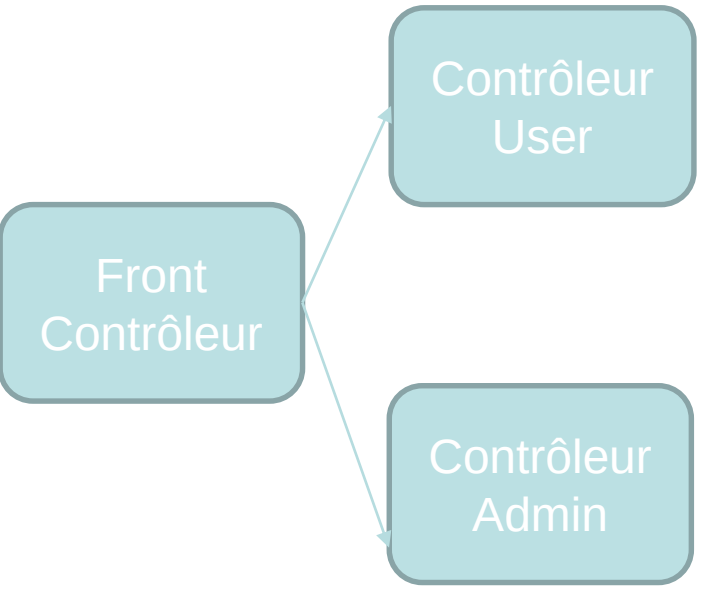


Solution 1
Héritage
Dans chaque Vue, on fait appel à l'un ou l'autre

+: solution simple
-: mais complexe à mettre en œuvre car chaque lien ou formulaire doit faire appel au bon contrôleur
On perd en souplesse

Des acteurs

- Comment lier les contrôleurs



Solution 2
Utilisation d'un front contrôleur (a.k.a. dispatcheur)

Rôle: choisit le bon contrôleur suivant les droits et l'action demandée

-: ajout d'un contrôleur supplémentaire
+: toujours le même, partout
+: gère les droits

Front controller

- C’est un patron de conception

- Wikipedia:

The **Front Controller Pattern** is a software design pattern listed in several pattern catalogs.

The pattern relates to the design of web applications. It "provides a centralized entry point for handling requests."

Front controller

- Idée générale de fonctionnement

Initialisation (session, lib, autoloader, etc.)

1. Quel est le rôle de la personne connectée ? (user, admin, etc.) Lecture dans une session
2. Quelle est l'action ?
3. L'action existe-t-elle ?
   1. Non -> Vue d'erreur
4. Oui, quel rôle faut-il ?
   1. Le role est bon -> appel du contrôleur
   2. Le role n'est pas bon -> envoi Vue de connection
5. Dans tous les autres cas -> Vue d'erreur

Contrôleur User

Contrôleur Admin

Vue d'erreur

Front controller

- Algorithme simplifié (2 acteurs Admin, User):

```
//appel modèle admin pour vérifier si utilisateur est connecté
Try{
Admin a = mdlAdmin.isAdmin(); (ici il suffit de tester si la session existe ou pas)
// ou utiliser objet session de php directement
Récupération de l'action
Si action dans listeAction_Admin
        si a=null
                require Page Authentification
        sinon
                require Ctlr Admin

Sinon require Ctrl User
//on peut aussi valider l'action ici au lieu de le faire dans Ctrl User
}
Catch {Page_Erreur}
```

Front controller

- Algorithme simplifié (2 acteurs Admin, User):

Attention:
toujours tester si a==null au début de Ctrl Admin

(appel du Ctrl Admin directement -> page d’erreur !!!)

Front controller

- Autres fonctionnalités du front:
  - Vérification de toutes les actions
  - URL rewritting: (Au lieu d'utiliser des paramètres GET/POST ou de formulaire):
    - Action dans l'URL ex: …/user1/action1/param1/param2
    - front controller découpe l'URL récupère l'action et envoie au contrôleur qui doit faire l'action
    - -> (routage !!!)

Ce qui n'a pas été vu

(Et donc ce que vous devriez voir)

Frameworks
Simples:

code igniter (https://ellislab.com/codeigniter),
Yii (http://www.yiiframework.com/)

Plus complexes:

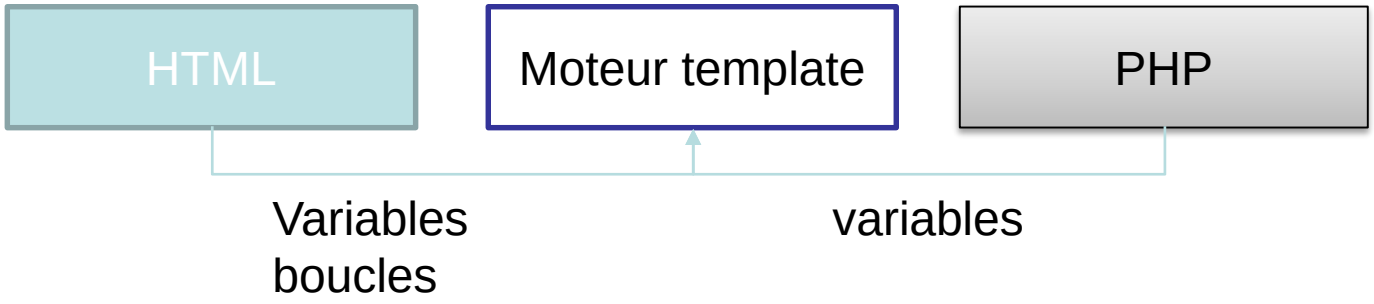
zend (http://framework.zend.com/),
symfony2 (http://symfony.com/)

Ces frameworks intègrent des ORM soit maison soit doctrine, propel

Ce qui n'a pas été vu

(Et donc ce que vous devriez voir)

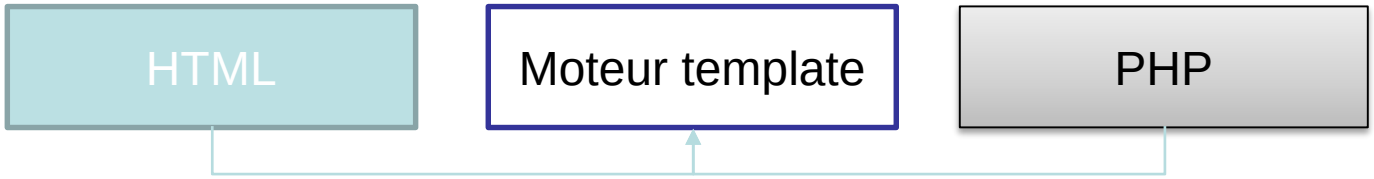
Moteur de template : séparer code HTML et PHP



Exemple Moteur : smarty (http://www.smarty.net/)

Ce qui n'a pas été vu

(Et donc ce que vous devriez voir)



Variables
boucles

variables

```
- <strong>{$ma_variable}</strong> <br />
- {$une_variable} <br />
- {$une_autre_variable}
```

```
<?php
require("tpl/smarty.class.php");

$tpl = new Smarty();

$tpl->assign("ma_variable","Je suis une variable");

$tpl->assign(array(
        "une_variable" => "Je suis une variable",
        "une_autre_variable" => "Je suis une belle
variable"
        // ...
        ));

$tpl->display("index.html");
?>
```

Ce qui n'a pas été vu

(Et donc ce que vous devriez voir)

Oubliez les ORM, les bases de données ???????

Services Web (Rest, SOAP)
- instances d'objets sur serveurs Web
- remplaçent ORM
- génériques, utilisables sur Web, Mobile, desktop !

Marre du SQL ?
Faites du NoSQL
- dépôt ou l'on met de tout comme on veut
- MongoDb, ElasticSearch

Marre de l'hébergement (d'ailleurs c'est quoi ?)
- Faites du Cloud, des VMs,

Ce qui n'a pas été vu

(Et donc ce que vous devriez voir)

Le vaste Monde du JS (Javascript), qui bouge tous les 6 mois

Tout est mis à la sauce JS ! (windows, mobile, jeux)

Jquery (à connaitre, de base)
WinJs pour Windows 8
createJs ->jeux en JS
BabylonJS (opengl (webgl)) dans navigateur, façon easy

NodeJs (application réseaux serveur en asynchrone) http://nodejs.org/
envie de faire son serveur Web plus rapide qu'Apache ?

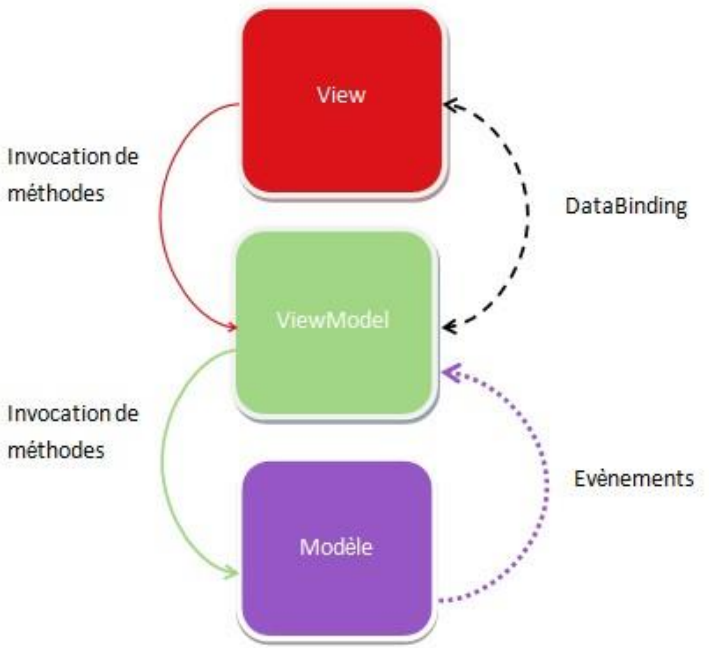
Ce qui n'a pas été vu

Angular, Backbone, React
en quelquesorte, faire du MVVM (adieu MVC)

•La vue est couplée aux données via du DataBinding et invoque les méthodes du ViewModel.

•Le ViewModel invoque les méthodes du modèle. contient la data spécifique à la gestion de l'écran e méthodes de réponses aux interactions utilisateurs contient également une référence vers un ou des modèles.

•Le modèle contient la data et les méthodes de manipulation de cette dernière (calculs, appels de services, …).



Ce qui n'a pas été vu

Faire des tests !

Test unitaires (de classes): phpunit

Test d’intégration: selenium, CasperJs

fin