

Vers des architectures plus évoluées

Quels exemples de classes donneriez vous ?????

3

MVC de base et optimisation

4

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, notamment en Web !

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

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

5

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. (Et pourquoi de ce cours)

6

Architecture MVC

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

```

graph LR
    U[utilisateur] <--> CU[Couche utilisateur]
    CU <--> CM[Couche métier]
    CM <--> CP[Couche persistance]
    CU --- S1[Serveur 1]
    CM --- S1
    CP --- S2[Serveur 2]
  
```

The diagram illustrates a 3-tier architecture. At the top, three rectangular boxes represent different layers: "Couche utilisateur" (User Layer) in white, "Couche métier" (Business Layer) in light blue, and "Couche persistance" (Persistence Layer) in light orange. Double-headed arrows connect the User Layer to the Business Layer, and the Business Layer to the Persistence Layer. Below these layers, the text "Serveur 1" is centered under the User and Business layers, and "Serveur 2" is centered under the Persistence layer. This indicates that the User and Business layers run on the same server (Serveur 1), while the Persistence layer runs on a separate server (Serveur 2).

Le MVC prend place dans la première couche

7

MVC, architecture

Architecture Vue en Module Objet 3

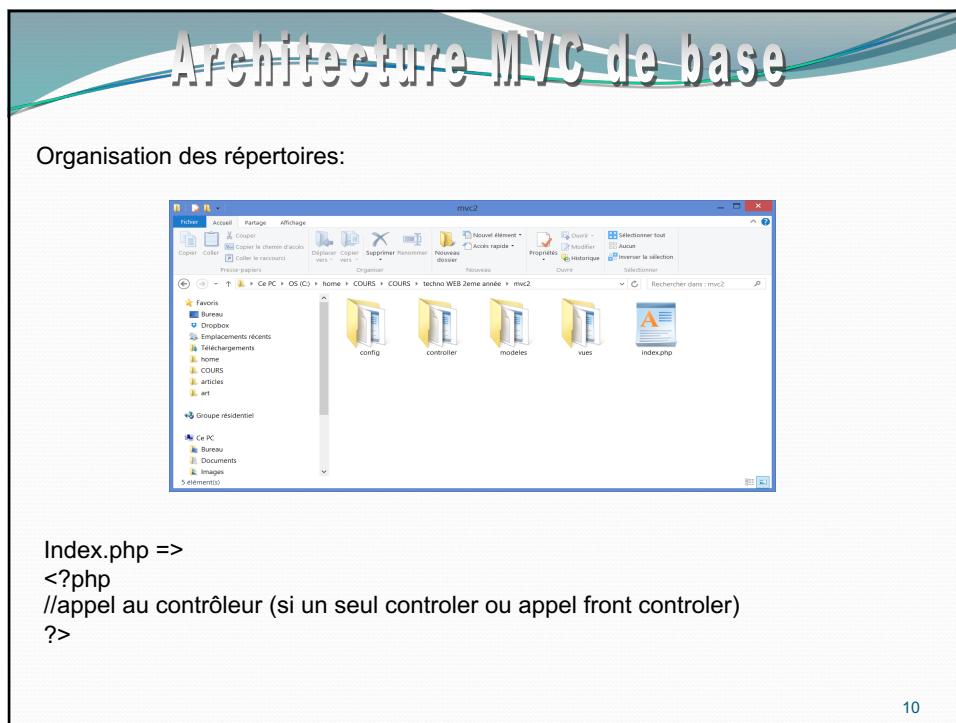
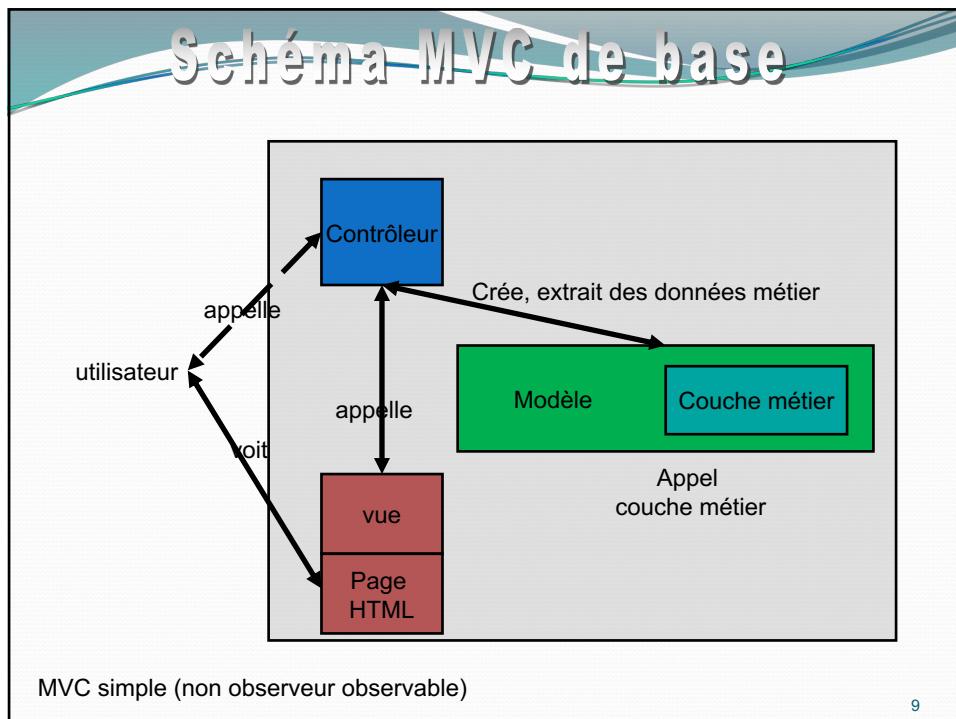
```

classDiagram
    actor Utilisateur
    class Vue {
        FXML
        JavaFX
    }
    class Model {
        Metier
    }
    class Contrroleur

    Utilisateur --> Vue : affiche
    Utilisateur --> Model : utilise
    Vue --> Model : observe
    Model --> Vue : notifie
    Vue --> Contrroleur : transfere les événements
    Contrroleur --> Model : demande la modification
    Model --> Contrroleur : cr閐 ou modifie
  
```

The diagram shows the MVC (Model-View-Controller) architecture. It features three main components: "Vue" (View), "Modèle" (Model), and "Contrôleur" (Controller). An external "Utilisateur" (User) actor interacts with the View. The View contains "FXML" and "JavaFX" components. The View observes the Model, and the Model notifies the View. The View transfers events to the Controller, and the Controller requests changes from the Model. The Model also sends changes back to the Controller.

8



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: donnée par la vue
=>indique le traitement à faire
3. Création d'un modèle, gestion des données
4. Redirection vers la vue en donnant les données
(=>page réponse ou une page d'erreur)

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

11

Contrôleur dans MVC

Les **actions** ?

```

graph LR
    User((user)) --> Action1((action1))
    User --> Action2((action2))
    User --> ActionN((action n))
  
```

<?php
Class ControllerUser {
 Si Action1 -> méthode 1
 ...
 Si Action n -> méthode n
?>

Ex: connecter, ajouterLivre, supprimerLivre,
Action NULL (ou pas d'action ->en général appel page principale)

=> Toujours commencer par effectuer une analyse !!!

12

Contrôleur dans MVC

Les **actions** ? Comment les donner ?

Sur un lien, GET

```
<a href=http://...?action=etre_attentif>
```

Dans un formulaire GET ou POST

```
<form action=http://...?action=etre_attentif>
```

Ou

```
<input type=hidden name=action...>
```

13

Contrôleur dans MVC

```
<?php
try{
$action=$_GET['action'];
switch($action) {
    case action1:
        action1();
        break;
    ...
    //mauvaise action
    default:
        $dVueEreur[] = "Erreur d'appel
php";
        require
        (_DIR_.'.../vues/vuephp1.php');
        // ajout du code de la vue ici
        break;
} }
```

14

Contrôleur dans MVC

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

    $dVue = array (
        //données
    );
    //utilisation d'une vue pour afficher
    require ('vue/vue.php');
}
?>
```

Contrôleurs toujours identiques, pas besoin du pattern observeur/observable
(possible avec closure et classes anonymes...)

=> Facile à faire depuis une analyse !

15

Modèle et MVC

Modèle

Couche qui manipule les données
peut valider les données
peut être simple: retourne une chaîne ? Une liste de chaîne
Souvent fait appel à DAL pour extraire des collections d'objets
-> appel aux classes ***Gateway (cours PDO et DAL)

Modèle = classe, pas d'HTML ou autre

Session, cookies ? Idéalement gestion par un modèle

Modèles et MVC

Modèle

Remarques:

Parfois, il n'y a pas de modèle

Pas de SQL dans un modèle, ni de parseur, etc.

Une méthode d'un modèle peut faire plusieurs choses :
demander une collection d'objets depuis une BD, mettre
à jour une session, etc.

17

Vues et MVC

**Vue
(html+php)**

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

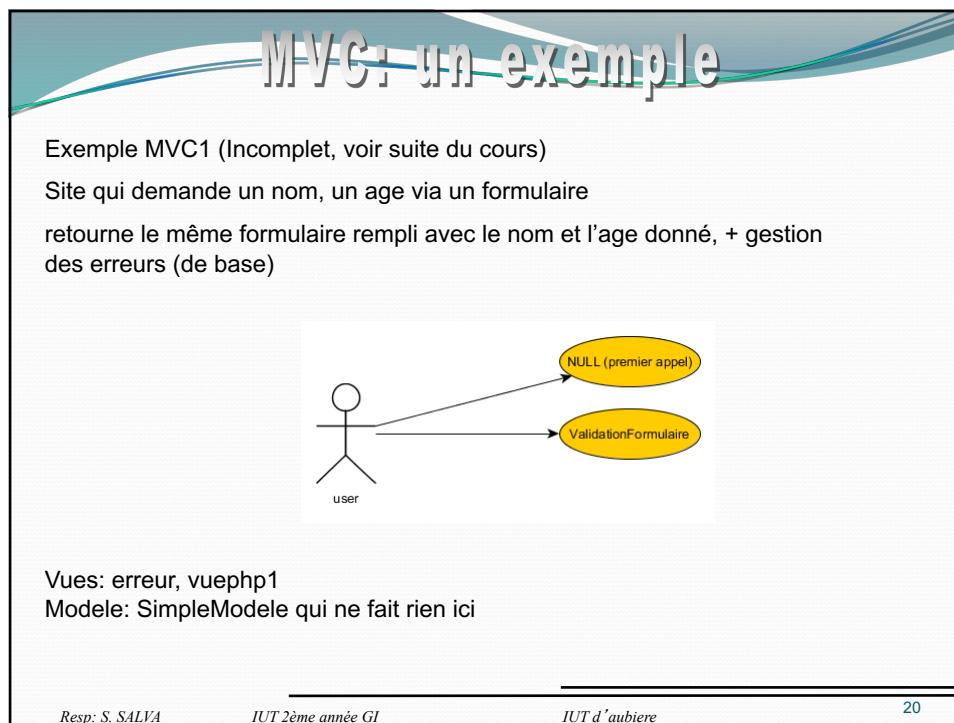
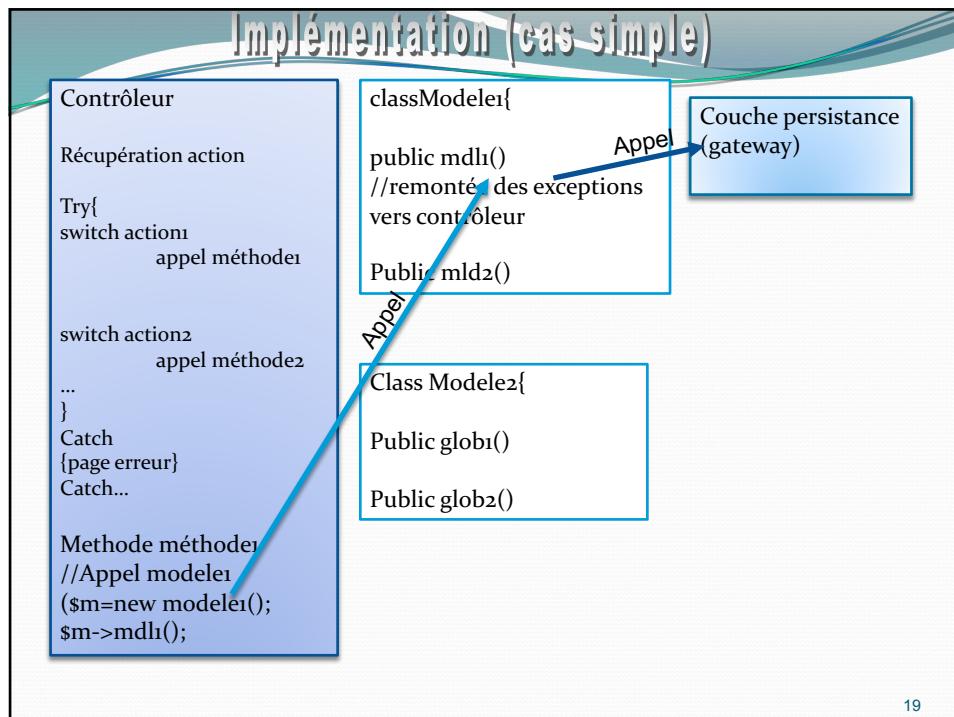
1. Récupère des variables produites par contrôleur ou modèle (tableaux, variables, etc.)
2. Crée la page avec ces données et echo et boucles

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

Pas de : \$_GET, \$_POST, \$_SESSION, etc. car ces données ne sont pas validées !

Les boutons (ou autres) de la vue font appels uniquement à des contrôleurs en leur donnant des paramètres et une action !

18



MVC: un exemple

Vuephp d'erreur (minimaliste)

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

<h1>ERREUR !!!!</h1>

<?php
if (isset($dataVueErreur)) {

foreach ($dataVueErreur as $value){
    echo $value;
}
?>

</body> </html>
```

données obtenues par la variable \$VueErreur

Resp: S. SALVA IUT 2ème année GI IUT d'aubiere 21

MVC: un exemple

'vuephp1.php'

```
<!DOCTYPE html>
<html lang="fr">
...<?php
// on vérifie les données provenant du modèle
if (isset($dataVue)) //tableau contenant les informations à afficher
{?>
<center>

<?php

if (isset($dataVueErreur) && count($dataVueErreur)>0) {
    echo "<h2>ERREUR !!!!</h2>";
    foreach ($dataVueErreur as $value){
        echo $value;
    }
?>
<h2>Personne - formulaire</h2><hr>
<!-- affichage des données provenant du modèle, juste pour exemple ici-->
<?= $dataVue['data'] ?>
```

22

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></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

Resp: S. SALVA *IUT 2ème année GI* *IUT d'aubiere* **23**

MVC: un exemple

Index.php ->contrôleur:

```

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

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

try{
$action=$_REQUEST['action']; // on peut aussi utiliser $_POST, $_GET bien sûr

switch($action) {

```

24

MVC: un exemple

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

//mauvaise action
default:
    $dataVueErreur[] = "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

Appel vue erreur

Appel vue erreur

25

MVC: un exemple

```
function Reinit() {

$dataVue = array (
    'nom' => "",
    'age' => 0,
);

require (__DIR__.'../../vues/vuephp1.php');
```

Appel Vue formulaire

26

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

Appel Vue formulaire

27

MVC : un exemple

ValidationFormulaire=> valide les paramètres reçus

```

class Validation {

static function val_form($nom,$age,&$dataVueEreur) {

if (!isset($nom)||$nom=="") {
    $dataVueEreur[] ="pas de nom";
    throw new Exception('pas d\'action');
}
if (!isset($age)||$age=="") {
    $dataVueEreur[] ="pas d'age ";
    throw new Exception('pas d\'age');
}
} } ?>

```

Mais on peut aussi retourner un booleen

28

Optimisation de l'architecture précédente

Critique précédente architecture :

- Pas tout objet !
- Un require par classe...
- Des url (de vues) en dur dans le code

=> 2eme architecture

- -> utilisation du fichier de config (ou d'une classe !)
- Autoloader
- **Code MVC v2 dispo sur <http://berlin.iut.local/~progweb/> et <https://github.com/sasa27/Basic-PHP-MVC>**

29



Vers des architectures plus évoluées

Comment charger une classe ?

- Require
- **Mieux : autoloader**
- `spl_autoload_register` — Enregistre une fonction en tant qu'implémentation de `__autoload()`->charge des classes
 - <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-0/fr/>, <http://www.php-fig.org/psr-4/>

30

Optimisation de l'architecture précédente

Index.php

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

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

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

31

Optimisation de l'architecture précédente

config.php (exemple simple, il y a plein de façons de faire)

Avec des variables globales
 Variables récupérables dans des fonctions avec mot clé *global*
 Voir <http://php.net/manual/fr/language.variables.scope.php>

```
<?php
//préfixe
$rep=__DIR__.'../';

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

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

32

Vers des architectures plus évoluées

Exemple simple de classe de Chargement (de classes)

```

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();
        if(!spl_autoload_register(array(self::$_instance, '_autoload'),
false)) {
            throw RuntimeException(sprintf("%s : Could not start
the autoload', __CLASS__));
        }
        ....
    }
}

private static function _autoload($class){
    global $rep; // récupère var. globales
    $filename = $class.'.php';
    $dir
    =array('modeles','./','config','contrôleur');
    foreach ($dir as $d){
        $file=$rep.$d.$filename;
        //echo $file;
        if (file_exists($file))
        {
            include $file;
        }
    }
}

```

A chaque *new* -> on lance *_autoload()*

33

Optimisation de l'architecture précédente

Classe Controller

```

<?php

class controller {

    function __construct(){
        //code du contrôleur ici
        global $rep,$vues;

        // on démarre ou reprend la session
        session_start();
        ...
        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() {
        ...
    }
}

```

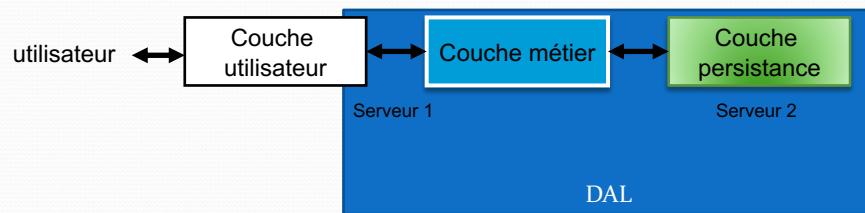
34

Lien avec Data access layer

35

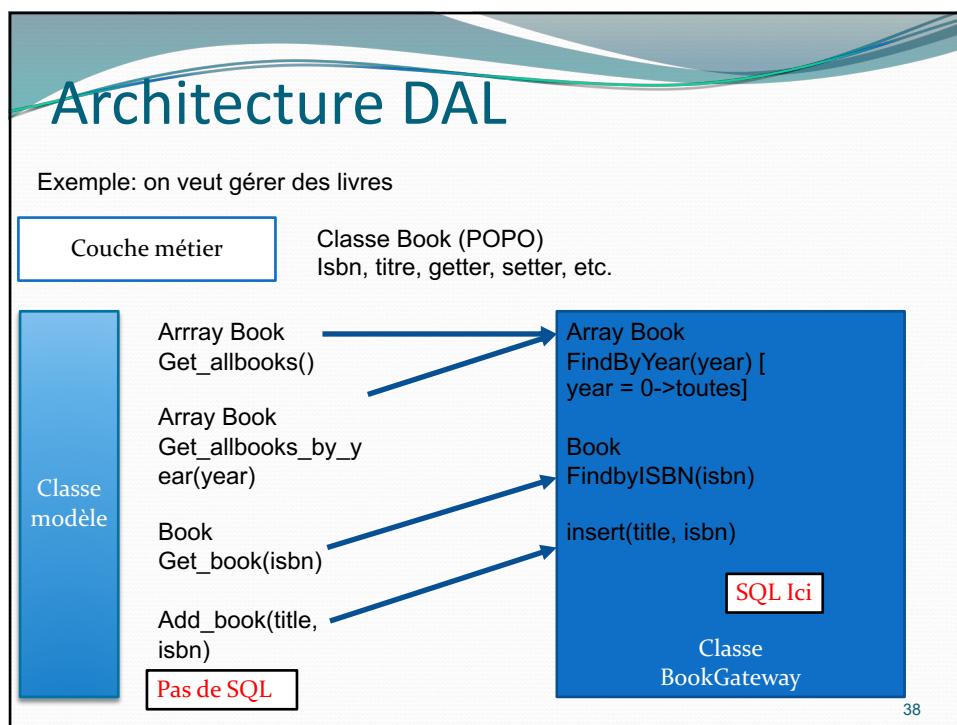
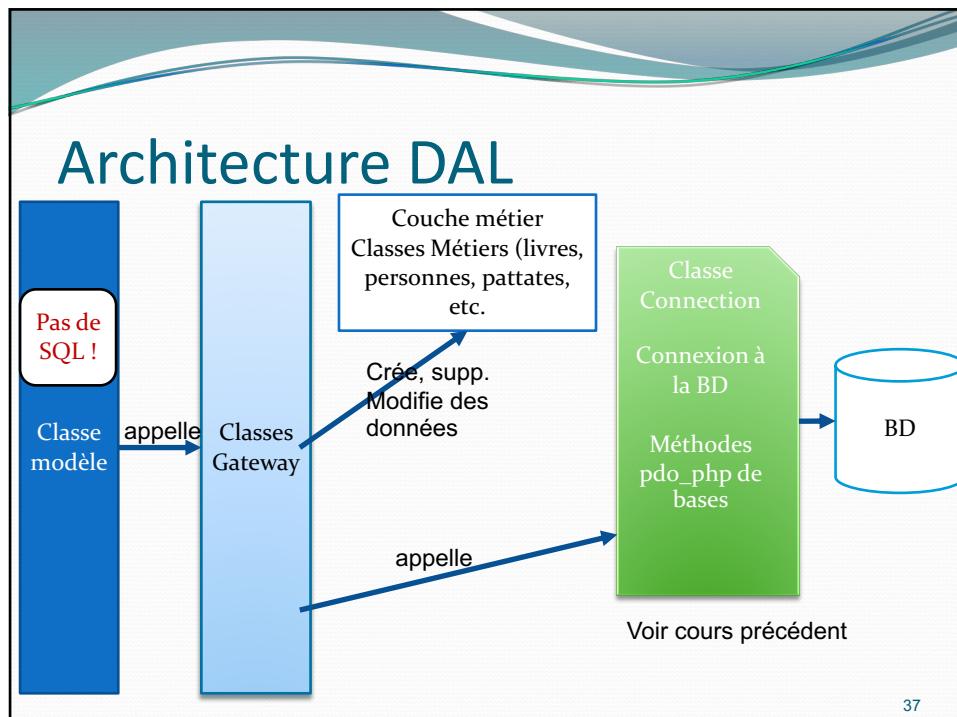
Présentation DAL

- Couche d'accès aux données (DAL)



- Couche généralement composée de plusieurs classes
 - Permet de gérer les données à la sauce Objet
- on y trouve des patrons : singleton, fabrique, stratégie, etc.

36



Architecture DAL

Exemple: on veut gérer des livres

Classe modèle

```
Public function get_allbooks($year){  
  
    $b = new BookGateway(...);  
    $Tab_de_Books=$b->FindByYear($year);  
    // puis renvoyer vers le contrôleur (return) ou autre...  
}  
  
Ou avec méthodes statiques quand c'est possible
```

39

Architecture DAL

Exemple: on veut gérer des instances de Books

Classe
BookG
ateway

```
Public function FindByYear($year){  
    //préparation + execution requête sql  
    If $year>>0 {  
        $query=' select * from Books where year=:year';  
        $this->$con->executeQuery($query, array( ':year'  
=> array($year,PDO::PARAM_STR),  
            ));  
  
        $results=$this->con->getResults();  
        Foreach ($results as $row)  
            $Tab_de_Books[]=new  
Book($row['isbn'], $row['titre'], $row['year']);  
        Return $Tab_de_Books;  
    }  
    Else{...}
```

Création des
objets,
On retourne les
objets

40

Architecture DAL

Composée du code en PDO

Rôle:
préparer les requêtes, injecter les paramètres, exécuter la requête
=> Découpler l'accès bas niveau de la BD du reste du projet

Non rôle:
ne traite pas les erreurs
Pas d'affichage

!! Ne pas retourner d'objet statement (objet retourné par l'appel prepare)
!! Ne pas faire de fetchall (ou autre) or de cette classe
Chacun son rôle

41

Architecture DAL

Gestion des erreurs (PDOException)

Cas 1 : le plus simple

Pas de *try catch* dans la classe Connection
Les erreurs sont automatiquement remontées dans le Contrôleur

C'est le rôle du Contrôleur de récupérer les erreurs, de les gérer et d'appeler la bonne vue qui affiche les erreurs

Dans contrôleur

```
Try{ $action=... switch($action)
Catch (PDOException $e)
    {$dataVueEreur[] = $e->getMessage()
    require ($vue['vue']);}
```

42

Architecture DAL

Classe
Connection

Connexion à
la BD

Méthodes
pdo.php de
bases

Gestion des erreurs

Cas 2 :

Class Connection : pas de *try ...catch*

Les classes *Gateway récupèrent les exceptions PDO
retournent des exceptions non PDO

**Catch (PDOException \$e) {throw new Exception
('message');}**

=> découpler PDO du reste
=>le contrôleur n'est plus lié à PDO

ou ajoute des messages d'erreur dans tableau d'erreur
tableau d'erreur = var globale

43

Gestion des acteurs !

44

Des acteurs ...

- Plusieurs acteurs peuvent être utilisateurs d'une application Web:

- Visiteur



- Utilisateur



- Administrateur

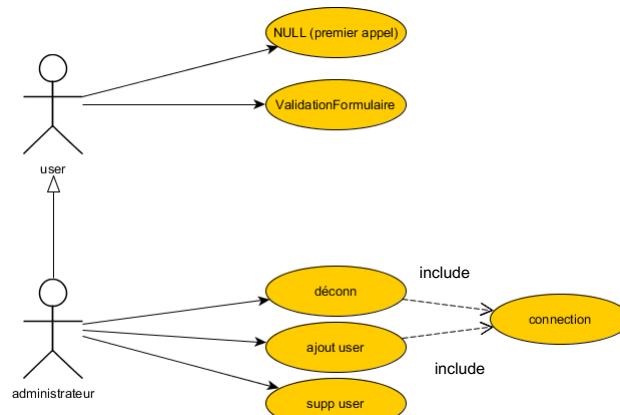


-

45

Des acteurs

- En UML, exemple:



46

Des acteurs

- En UML, exemple:

```

classDiagram
    actor User
    actor Admin
    User --> Admin : user
    User->> ValidationFormulaire : ValidationFormulaire
    User->> ContrroleurUser : NULL (premier appel)
    Admin--> ValidationFormulaire : déconn
    Admin--> ContrroleurAdmin : ajout user
    Admin--> ContrroleurAdmin : supp user
    Admin--> ContrroleurAdmin : connection
    Note over ValidationFormulaire: include
    Note over connection: include
  
```

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

Contrôleur User
2 actions

Contrôleur Admin
4 actions

47

Des acteurs

- Comment lier les contrôleurs

```

classDiagram
    class ContrroleurAdmin
    class ContrroleurUser
    ContrroleurAdmin <|-- ContrroleurUser
  
```

Contrôleur User
2 actions

Contrôleur Admin
4 actions

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

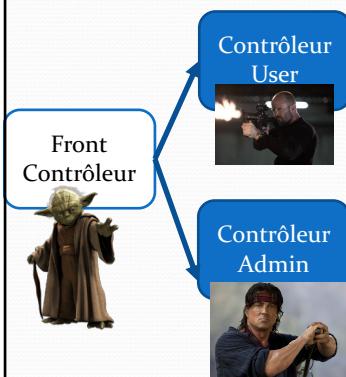
On perd en souplesse



48

Des acteurs

- Comment lier les contrôleurs



Solution 2 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

49

Front controller

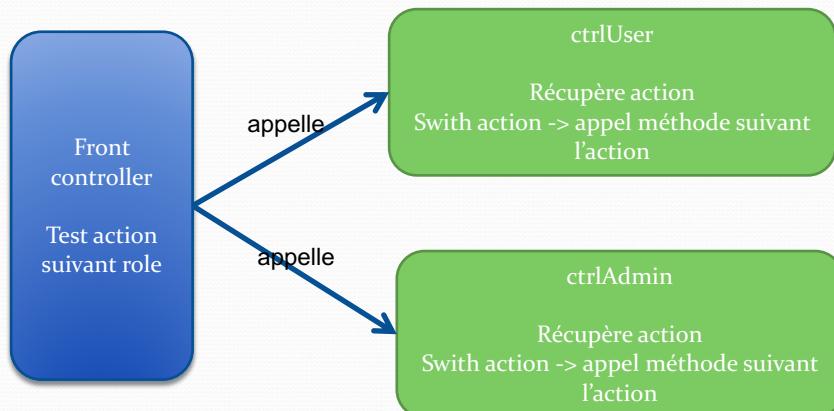
- C'est un patron d'architecture
- Wikipedia:

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."

50

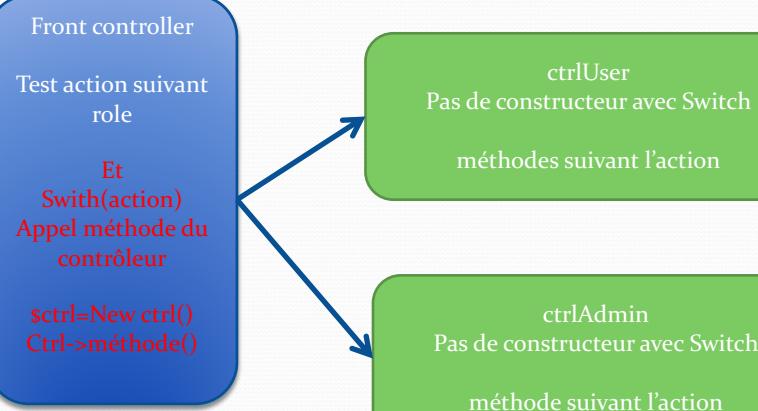
Front controller (du cours)



51

Front controller

Dans les Frameworks: c'est un peu différent



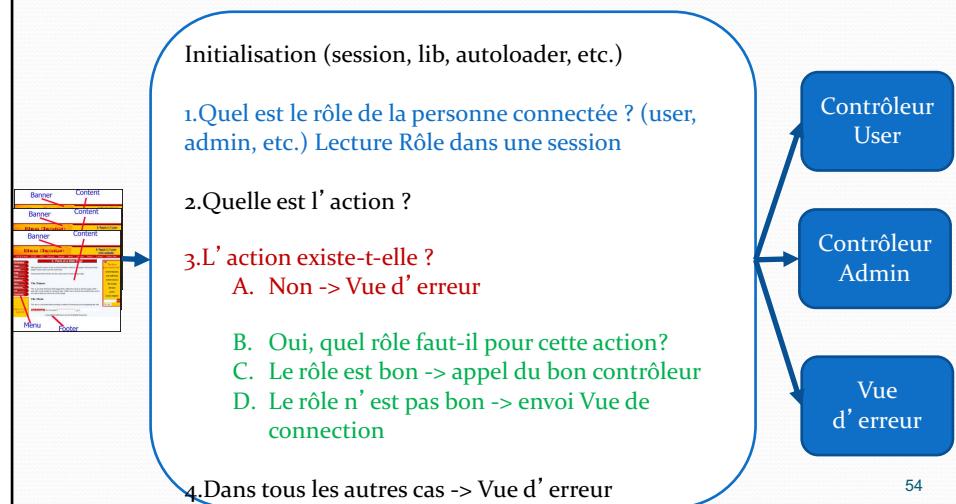
52

Front controller

- Front controller = classe appelée dans index.php (new FrontController());
- Reprenez toutes les vues
 - Chaque lien ou action doit faire appel à index.php
- Code principal du Front controller dans son constructeur

Front controller

- Idée générale de fonctionnement



54

Front controller

Quel est le rôle de la personne connectée ? (user, admin, etc.) Lecture du rôle dans une session

Exemple:

- Classe MdlAdmin
 - Méthode connection()
 - Méthode déconnexion()
 - Méthode isAdmin()

Front controller

Classe MdlAdmin

- Méthode connection(\$login, \$mdp)
 1. Nettoyage (\$login, \$mdp)
 2. Appel DAL pour vérifier si \$login+\$mdp dans la BD
 3. Ajout Rôle dans session

```
$_SESSION['role']='admin';
$_SESSION['login']=$login;
```
- On peut sécuriser en cryptant données de session, etc.
 - <http://php.net/manual/fr/session.security.php>
- Méthode déconnexion()
- Méthode isAdmin()

Front controller

- Classe MdlAdmin

- Méthode déconnexion()

```
session_unset();
session_destroy();
$_SESSION = array();
```

- Méthode isAdmin()

```
{ //teste rôle dans la session, retourne instance d'objet ou booleen
if isset $_SESSION['login'] && isset $_SESSION['role']) {
    $login=Nettoyer::nettoyer_string($_SESSION['login']);
    $role=Nettoyer::nettoyer_string($_SESSION['role']);
    return new Admin($login,$role)
} else return null;
}
```

Front controller

- Algorithme simplifié (2 acteurs Admin, User):

```
$listeAction_Admin = array('deconnecter','supprimer', 'ajouter');
//appel modèle admin pour vérifier si utilisateur est connecté
Try{
    $admin = mdlAdmin.isAdmin(); (ici il suffit de tester si la session existe ou pas)
```

Récupération de l'action
Si action dans listeAction_Admin
 si \$admin=null
 require Page_Authentification // ou appel ctrlUser avec action connecter
 sinon
 new CtrlAdmin()

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

Front controller

- Algorithme simplifié (2 acteurs Admin, User):

Attention:

toujours tester si \$admin==null au début de CtrAdmin

Si null ->page d'erreur

appel du Ctrl Admin directement doit retourner une page d'erreur !!!

59

Front Controller

```

try{ $String_actor="";
$listeActions=array(
    'Admin' => array ('action1','action2'),
    'SuperAdmin' => array ('action3','action4'));

$action=$_REQUEST['action'];
//methode isGoodAction -> retourne " ou acteur

$string_actor=isGoodAction($action);
$mdlclass="Mdl".$string_actor;
$mdlActor= new $mdlclass();
$actor=$mdlActor->isActor();

if ($string_actor!=""){
    if ($actor==null)
        require('pageAuth'.$string_actor);
    else {
        $ctrlclass= "Ctrl".$string_actor;
        new $ctrlclass();
    }
}
else {new CtrlUser();}
}

catch(Exception $e) {require ('pageErreur');} catch(Error $e2) { $error=$e2->getMessage(); require ('pageErreur');}

```

60

Front controller

- Autres fonctionnalités du front controller:
 - Vérification de toutes les actions

61

Front controller

- Autres fonctionnalités du front (mode avancé)

URL rewriting, routing (Au lieu d'utiliser des paramètres GET/POST ou de formulaire):

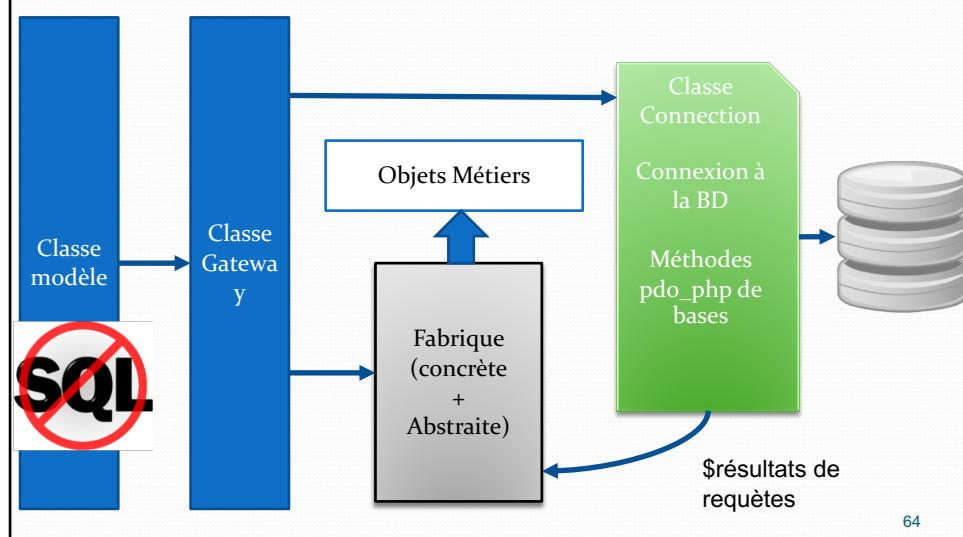
 - Action dans l'URL ex: .../user1/action1/param1/param2
 - front controller découpe l'URL récupère l'action et envoie au Contrôleur qui doit faire l'action
 - -> routage !!!
 - <http://www.phpaddiction.com/tags/axial/url-routing-with-php-part-one/>
 - <https://github.com/bencrowder/router>

62

Pour aller (un peu) plus loin!

63

Architecture DAL, optimisation



64

Architecture DAL, optimisation

Avantage:

Fabrique seule responsable de la création / distribution de l'objet,

Elle peut faire :

- préparer l'encodage (UTF8, ...)
- Réaliser des opérations de log
- etc.

65

Architecture DAL, optimisation

Exemple:

```

class DBFactory_Books{
    public static function create ($resultats, $param){
        if ( ! Isset($resultats)){
            throw new Exception ('Aucune donnée!');
        }

        switch ($param){
            case 'mysql':
                mysqlBooks($resultats);
                break;
            case 'xml':
                XMLBooks($resultats);
                break;
            default:
                throw new Exception ('Type de base inconnu');
        }
    }
}

```

66

Architecture DAL, optimisation

Exemple:

```

Objets Métiers
↑
Fabrique (concrète)

```

```

mysqlBooks($resultats) {
    $Tab_Books=array(); $i=0;
    Foreach ($results as $row)
        $Tab_de_Books[]=$row=new Book($row['isbn'], $row['titre'], $row['year']);
    Return $Tab_Books;
}

```

67

Séparation HTML PHP

Moteur de template : séparer code HTML et PHP

Pourquoi ? Souvent 2 métiers (postes)

```

graph TD
    subgraph Template_Engine [Moteur template]
        direction TB
        TB[Variables boucles]
        B[variables]
    end
    HTML[HTML] --- TE[Template_Engine]
    PHP[PHP] --- TE

```

Soit à la main (PHP est un moteur de template à la base)

Soit outils
Exemple Moteur : twig (<https://twig.symfony.com/>)

Resp: S. SALVA

IUT 2ème année GI

IUT d'aubiere

68

Séparation HTML PHP

Exemples:

```

    graph LR
      HTML[HTML] -- "Variables boucles" --> TWIG[Twig]
      TWIG -- "variables" --> PHP[PHP]
  
```

Template Index.html

```

<html><body>
<strong>{{ name }}</strong> <br />

{% if users|length > 0 %}<ul>
{% for user in users %}
    <li>{{ user.username }}</li>
{% endfor %}
{% endif %}

</body></html>
  
```

PHP code

```

<?php
require_once __DIR__ . '/vendor/autoload.php';
$loader = new Twig_Loader_Filesystem('templates'); // Dossier contenant les templates
$twig = new Twig_Environment($loader, array( 'cache' => false ));

echo $twig->render('index.html', array(
    'name' => 'Salva',
    'users'=> array(
        '1'=> array(
            'username'=>'toto'),
        '2'=> array(
            'username'=>'sasa')
    )));
  
```

69

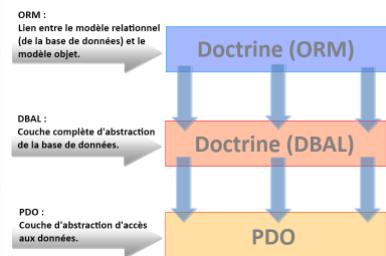
Architecture DAL, optimisation

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

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

70

Architecture DAL, optimisation



Tutoriel : <http://openclassrooms.com/courses/apprendre-a-utiliser-doctrine>

Résumé dans la suite:

71

Architecture DAL, optimisation



Modifier le fichier ~/config/global.php
Configuration

```

<?php
// Adaptez bien sûr le DSN à votre cas.
define('CFG_DB_DSN', 'mysql://root@localhost/db_doctrine_test');

define('LIB_DIR', dirname(__FILE__).'/..lib/');
define('CFG_DIR', dirname(__FILE__). '/');
define('WEB_DIR', dirname(__FILE__).'/..web/');
define('HTML_DIR', dirname(__FILE__).'/..html');

require_once(LIB_DIR.'vendor/doctrine/Doctrine.php');
spl_autoload_register(array('Doctrine_Core', 'autoload'));

//necessaire pour trouver les modèles
spl_autoload_register(array('Doctrine_Core', 'modelsAutoload'));

```

72

Architecture DAL, optimisation



Modifier le fichier ~/config/global.php
Configuration

```
$manager = Doctrine_Manager::getInstance();
$conn   = Doctrine_Manager::connection(CFG_DB_DSN);

$manager->setAttribute(Doctrine_Core::ATTR_VALIDATE,
Doctrine_Core::VALIDATE_ALL);
$manager->setAttribute(Doctrine_Core::ATTR_AUTO_ACCESSOR_OVERRIDE, true);
$manager->setAttribute(Doctrine_Core::ATTR_AUTOLOAD_TABLE_CLASSES, true);
$manager->setAttribute(Doctrine_Core::ATTR_MODEL_LOADING,

Doctrine_Core::MODEL_LOADING_CONSERVATIVE);
Doctrine_Core::loadModels(LIB_DIR.'models/');
```

73

Architecture DAL, optimisation



Déclaration des Modèles en Yaml
(puis génération de code en PHP)

ou écriture des classes en PHP
(1 classe/modèle + 1 classe pour la table)

fichier ~/config/schema.yml :

User:

tableName: member //nom table si différent du nom de la classe
columns:

login:
type: string(255)
unique: true
notnull: true

password:
type: string(255)
notnull: true

74

Architecture DAL, optimisation



Déclaration des Modèles en Yaml
(génération de code en PHP)

ou écriture des classes en PHP
(1 classe/modèle + 1 classe pour la table)

fichier ~/config/schema.yml :

```

Article:
  columns:
    title:
      type: string(255)
      notnull: true
    content:
      type: string
      notnull: true
    user_id:
      type: integer
      relations:
        User: //nom relation
        class: User
        local: user_id
        foreign: id
        alias: User # Inutile, puisque l'alias par
        défaut est le nom de la classe.
        foreignAlias: Articles
        ...
  
```

75

Architecture DAL, optimisation



Création des tables, des modèles:

```

<?php
require(dirname(__FILE__).'/../config/global.php');

// Si elle existe, supprimez la base existante.
Doctrine_Core::dropDatabases();

// Création de la base (uniquement si elle n'EXISTE PAS)
Doctrine_Core::createDatabases();

// Création des fichiers de modèle à partir du schema.yml
// Si vous n'utilisez pas le Yaml, n'exécutez pas cette ligne !
Doctrine_Core::generateModelsFromYaml(CFG_DIR.'schema.yml', LIB_DIR.'models',
array('generateTableClasses' => true));

// Création des tables
Doctrine_Core::createTablesFromModels(LIB_DIR.'models');
  
```

76

Architecture DAL, optimisation



Création instance d'article et sauvegarde:

```
<?php
$article = new Article();

<?php
// plusieurs possibilités d'accès
$article->title = 'Mon titre';
$article['title'] = 'Mon titre';
$article->set('title', 'Mon titre');
$article->content = "Contenu de l'article...";
$article['content'] = "Contenu de l'article...";
```

//sauvegarde en BD
\$article->save();

77

Architecture DAL, optimisation



Récupérer une instance d'objet

```
<?php
$table = Doctrine_Core::getTable('Article');

// Récupérer un article avec son id, par exemple $id = 1
$article = $table->find($id);

// tout récupérer
$articles = $table->findAll();

// Récupérer un article avec une autre propriété
$article = $table->findOneByTitle('Mon titre');
```

78

Architecture DAL, optimisation



Récupérer une instance d'objet

```
$article = $table->findOneByTitle('Mon titre');
$article = $table->findOneByTitleAndContent('Mon titre', "Contenu de l'article...");
$article = $table->findOneByUser($user);
```

Magie !

Ces méthodes n'ont pas été écrites !

`findOneByTitle` revient à

Appel méthode Doctrine `findOneBy`
`Title, 'Mon titre'` est interprété comme un filtre

Équivalent à

`findBy(array('Title' => 'Mon titre'), // Critere)`

Voir aussi `findBy`

79

Architecture DAL, optimisation

Langage DQL

SELECT

```
<?php
$q = Doctrine_Query::create()
->from('Article a')
->where('a.title = :title AND a.publishable =
:publishable', array(
':title' => $title,
':publishable' => $publishable
))
->andWhere('a.content LIKE :keyword',
'%'.$keyword.'%');

$resultats = $q->execute();
```

UPDATE

```
<?php
$q = Doctrine_Query::create()
->update('Article a')
->where('a.title = ?', 'mon titre')
->orWhere('a.id = ?', 1)
// Soit :
->set('content', 'un super texte');
// ou soit :
->set('content', '?', $content);
// ou encore
->set(array('content' => $content))

$number = $q->execute();
```

80

Ce qui n'a pas été vu

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

81

Ce qui n'a pas été vu

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

Services Web (Rest, SOAP)

instances d'objets sur serveurs Web

génériques, utilisables sur Web, Mobile, desktop !

Marre du SQL ?

Faites du NoSQL

dépôt où 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

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

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

Jquery (à connaître, de base)

WinJs pour Windows 8, 10

createJs,phaser ->jeux en JS

BabylonJS->(opengl (webgl)) dans navigateur, façon easy

Aframe -> VR

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



83

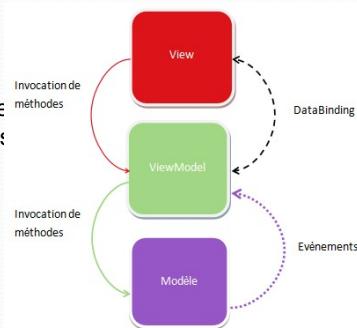
Ce qui n'a pas été vu

Angular, React
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. Il contient la data spécifique à la gestion de l'écran et les méthodes de réponses aux interactions utilisateurs. Il 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 qu'on n'a pas été vu

Faire des tests !

Test unitaires (de classes): phpunit

Test d'intégration: selenium, CasperJs

The slide features three icons: a blue square with a white 'P' for PHPUnit, a grey rounded square with a green checkmark and a white 'Se' for Selenium, and a white ghost-like character with a smile and 'js' for CasperJS.

85

D'autres patterns

Pattern Observer avec classes anonymes et **SplObserver**

Pattern CommandBus

But: séparer les aspects techniques lourds des contrôleurs
Ex: acheter un podcast -> demander login, utiliser carte de crédit, assigner podcast, etc.

Objet Command = classe métier contenant les données provenant d'une requête, puis exécuter par un Objet Commandhandler (découplage)

Intégré dans les frameworks laravel, symfony, etc.

Ex:
<http://php-and-symfony.matthiasnoback.nl/2015/01/a-wave-of-command-buses/>

Implémentation: simpleBus: <https://github.com/SimpleBus>

Resp: S. SALVA *IUT 2ème année GI* *IUT d'aubiere* *86*