Université Lumière Lyon 2 — Institut de la communication

Programmation Web backend

L3 Informatique

Année 2023-2024

Jérôme Darmont

https://eric.univ-lyon2.fr/jdarmont/

Actualité du cours

https://eric.univ-lyon2.fr/jdarmont/?page_id=445

https://eric.univ-lyon2.fr/jdarmont/?feed=rss2

https://social.sciences.re/@darmont #l3progweb

Plan du cours

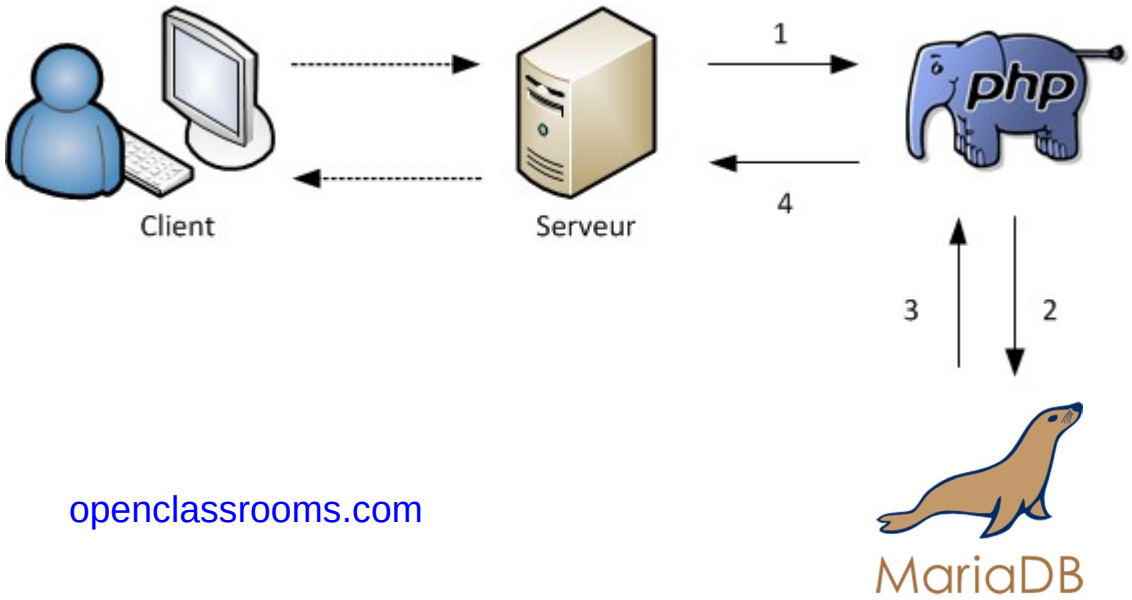
- Objectifs du cours
- PHP objet
- Gabarits
- Architecture MVC
- Formulaires de saisie
- Sessions
- Interface PHP-base de données

Création de sites web dynamiques

- Pages web générées à la demande
- À l'aide d'un langage de programmation

openclassrooms.com

Interfaçage avec une base de données



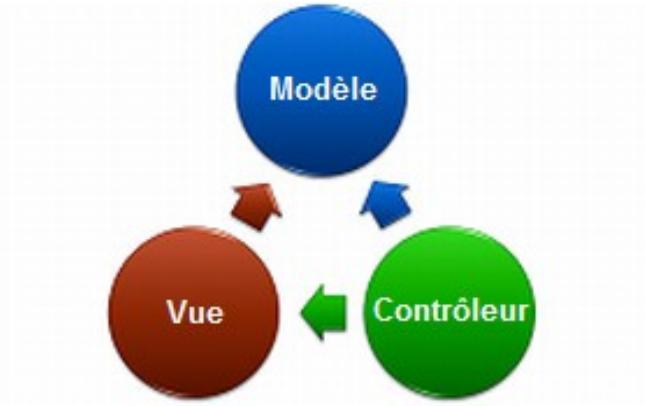
openclassrooms.com

Une multitude de langages

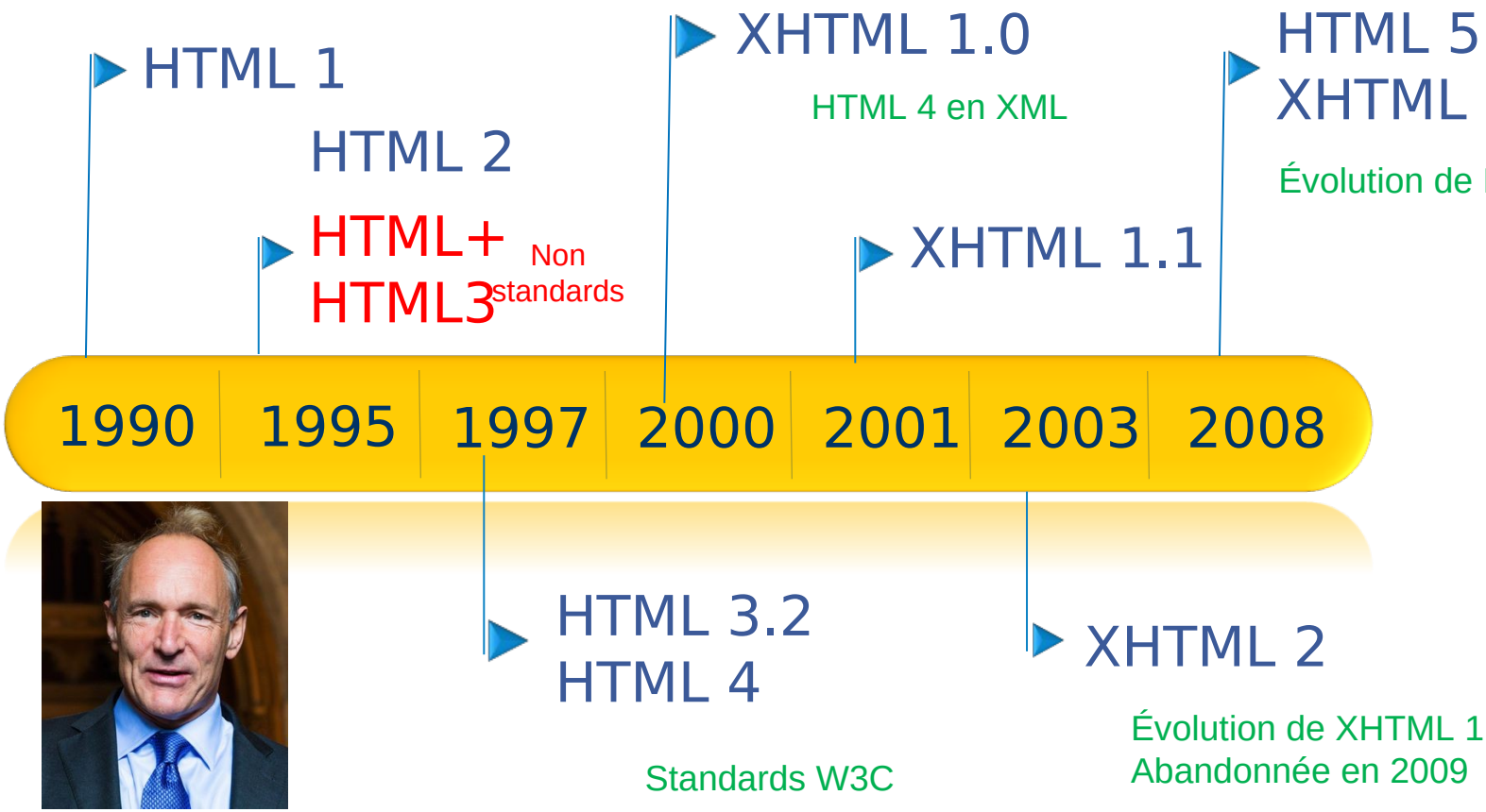
Tâches	Langage
Structuration et contenu statique	HTML
Présentation	CSS
Contenu dynamique	PHP ou autre
Opérations sur la base de données	SQL

Principe de séparation du code

- Pourquoi ?
  - Maintenance facilitée
    - Code plus lisible
    - La modification d'une tâche n'affecte pas les autres
  - Réutilisation de code
  - Métiers différents
- Comment ?
  - Architecture MVC
  - Gabarits (*templates*)

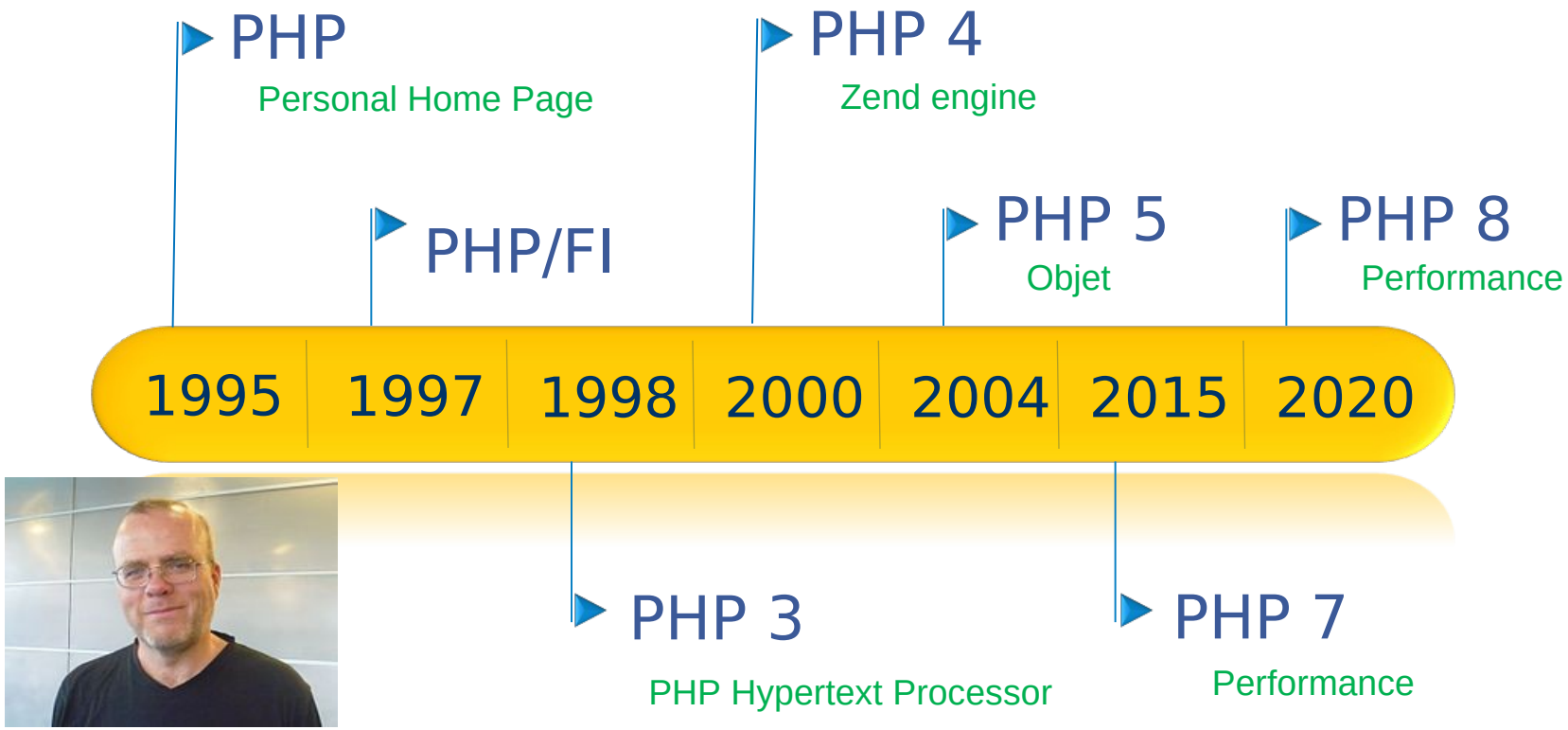


Des versions qui coexistent (1/2)



Tim Berners-Lee

Des versions qui coexistent (2/2)



Rasmus Lerdorf

Nécessité de standardisation

- Par le World Wide Web Consortium
  - Validateur de code HTML
  - Validateur de code CSS
- Par les concepteurs de PHP
  - *PHP Standards Recommendations (PSR)*
  - La majorité des bibliothèques devient objet

Récapitulatif des objectifs

- Programmation Web backend moderne
  - Programmation orientée objet
  - Architecture de sites web séparant les différents langages utilisés
- Bonnes pratiques
  - Code standardisé
  - Code validé

N° de la question : 160

Sondage express

A) Je n’ai jamais étudié la conception de sites web.

B) Je connais les langages HTML et CSS.

C) Je connais le langage PHP.

D) Je connais le langage PHP orienté objet.

E) Je connais les architectures MVC.

Répondre sur https://toreply.univ-lille.fr

Question n° 160

Plan du cours

- ✓ Objectifs du cours
- PHP objet
- Gabarits
- Architecture MVC
- Formulaires de saisie
- Sessions
- Interface PHP-base de données

Programmation impérative

- Basée sur la logique du traitement
- Utilise beaucoup les structures de contrôle
  - Boucles
  - Tests
- Importance des structures de données
- Lisibilité et maintenabilité souvent inversement proportionnelle à la taille du code

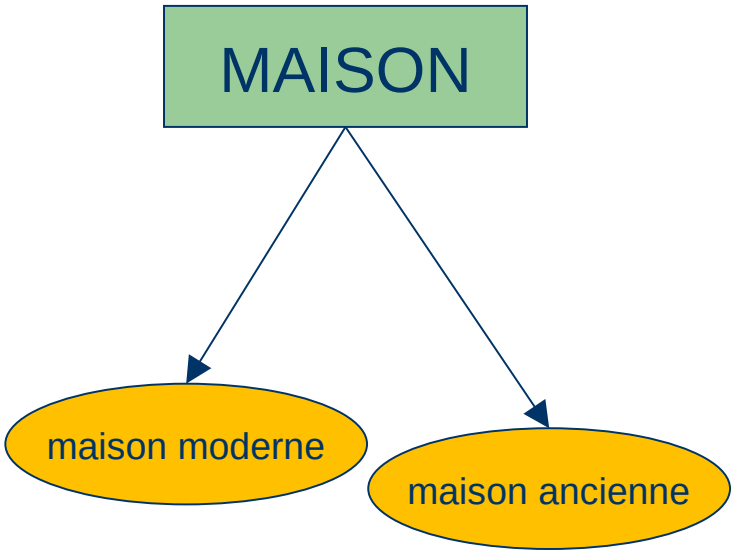
Programmation orientée objet

- Basée sur la définition des données
- Modularité
  - Plus facile à développer et maintenir
- Abstraction
  - Permet de créer des types indéfinis dans le langage
  - Plus facile à réutiliser
- Spécialisation
  - Évite la duplication du code d'objets similaires

Notions de classe et d'objet

Concept de maison

Classe



Objets de la classe Maison

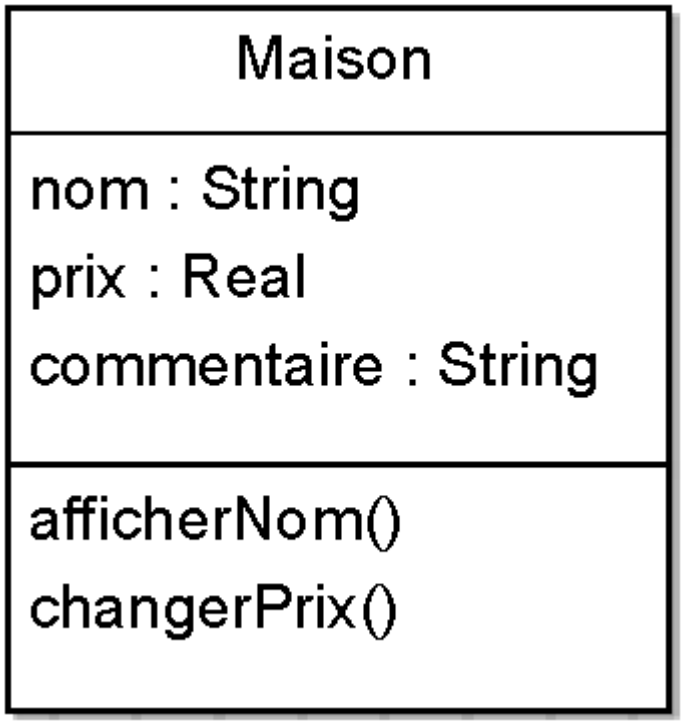
Encapsulation

- Rassembler données (attributs) et traitements (méthodes) dans une classe
- Seules les méthodes permettent de lire/écrire les attributs d'un objet
- Protection des objets

Exemple de classe

Attributs

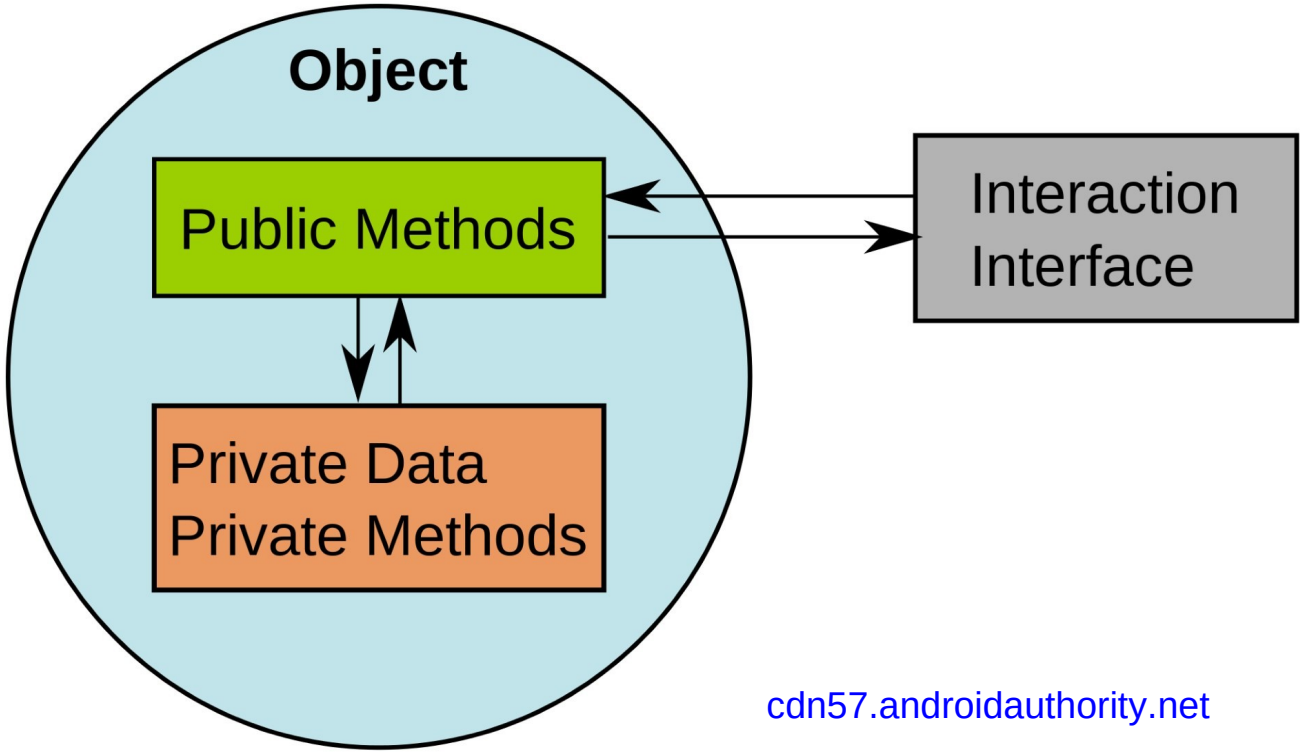
Méthodes



Visibilité des attributs et méthodes

Public	Visible par toute méthode, même extérieure à la classe (notion d'interface) Un attribut ne devrait jamais être public !
Privé	Visible uniquement par les méthodes de la classe
Protégé	Visible uniquement par les méthodes de la classe et de ses sous-classes (on y reviendra)

Interface



cdn57.androidauthority.net

La classe Maison en PHP

```php
<?php
        class Maison {
                // Attributs (NB : ceci est un commentaire)
                private $nom;
                protected $prix;
                public $commentaire;        // À éviter absolument !

                // Méthodes
                public function afficherNom() {
                        return $this->nom;          // $this est l'objet courant
                }
                public function changerPrix($nouveau_prix) {
                        $this->prix = $nouveau_prix;
                }
        }
?>
```

Instanciation

```php
<?php
        // Création d'objets
        $maison_moderne = new Maison();
        $maison_ancienne = new Maison();

        // Accès à un attribut
        $maison_moderne->commentaire = "Trop chère !";
        // Possible car commentaire est un attribut public (dangereux)

        // Appel de méthodes
        $nom_maison = maison_ancienne->afficherNom();
        $maison_moderne->changerPrix(350000);
?>
```

Constructeur

Méthode qui permet d'initialiser un objet

```
<?php
        class Maison { // suite

                // Constructeur
                function __construct($nom) {
                        $this->nom = $nom;
                        $this->prix = 200000;
                        $this->commentaire = "No comment";
                }
        }
?>
```

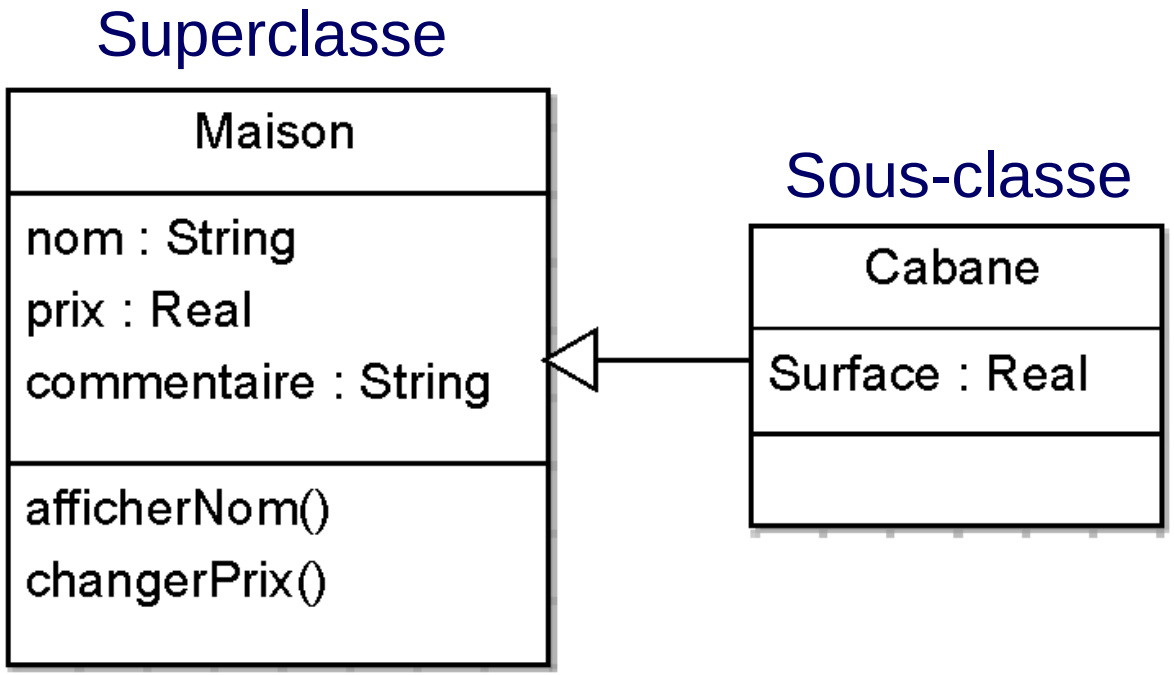
Appel au constructeur

C’est implicite !

```
<?php
        // Création d’objets
        $maison_moderne = new Maison("Maison de style");

        $maison_ancienne = new Maison("Maison rénovée");
?>
```

Héritage



- Cabane EST UNe Maison
- Cabane hérite des attributs et des méthodes de Maison

La sous-classe Cabane en PHP

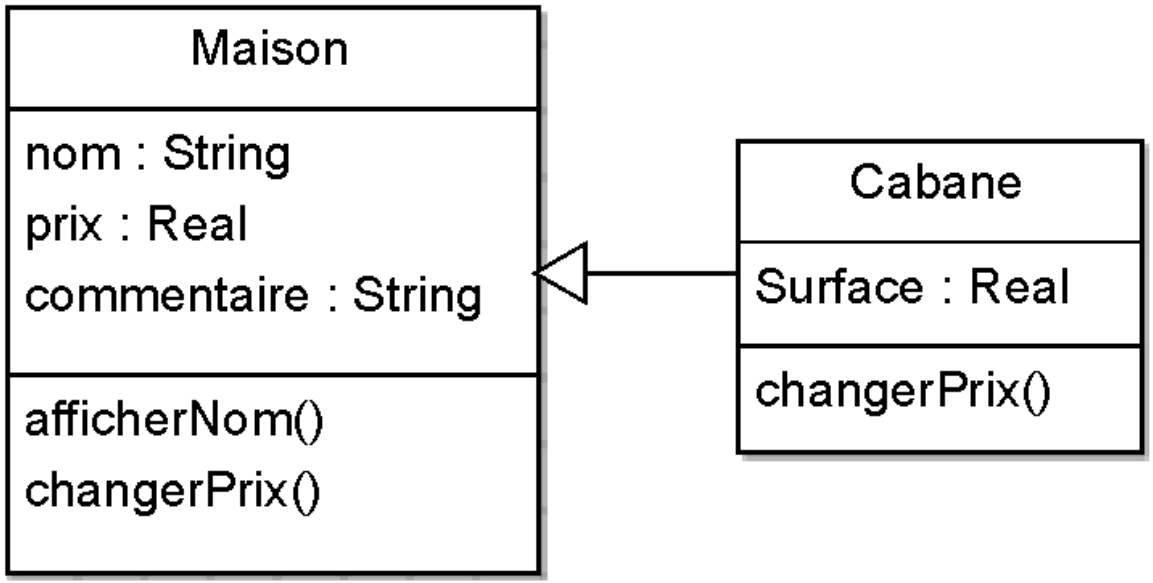
```php
<?php
        class Cabane extends Maison {
                // Attribut spécifique
                private $surface;
        }

        // Instanciation
        $cabanon = new Cabane("Ma cabane au Canada");

        // Appel des méthodes héritées
        $cabanon->changerPrix(5000);
?>
```

Polymorphisme

Redéfinition des méthodes héritées dans la sous-classe



Méthode surchargée

Surcharge dans Cabane

```php
<?php
        class Cabane extends Maison {
                // Attribut spécifique
                private $surface;
                // Constructeur surchargé
                function __construct($nom, $surface) {
                        parent::__construct($nom);
                        $this->surface = $surface;
                }
                // Méthode surchargée
                public function changerPrix($prix, $surface) {
                        parent::changerPrix($prix);
                        $this->surface = $surface;  // Pas très approprié
                }                                   // C'est juste pour l'exemple
        }
?>
```

N° de la question : 98

Sondage express

J’ai compris :

A) les classes, les attributs, les méthodes.

B) les constructeurs.

C) l’héritage.

D) le polymorphisme.

E) on verra en TD !

Répondre sur https://toreply.univ-lille.fr

Question n° 98

Variables et types

- Variables : préfixées par le caractère $
- PHP ne nécessite pas de déclaration explicite du type de variable (⚠).
- Types de données :

	Ex. d'affectation
– Nombres entiers : int, integer	$i = 1;
– Nombres réels : real, double, float	$pi = 3.14;
– Chaînes de caractères : string	$ch = "oui";

- Conversion de type : "cast" comme en C
  - Ex. `$ipi = (int) $pi; // $ipi est égal à 3`

Tableaux

- Scalaires ou associatifs
  - Création par assignation des valeurs
  - Ex. $tab_scalaire[0] = "Chaîne 0"; // Indiçage à partir de 0
    $tab_scalaire[1] = "Chaîne 1";
    $tab_associatif["Dupont"] = 30;

- Fonctions associées :
  - Initialisation :
    Ex. $notes_scal = array(10, 12.5, 15, 8);
    $notes_assoc = array("Valeriia" => 16, "Vadim" => 12);
  - Nombre d'éléments :
    Ex. $n = count($notes_scal)

Tableaux multidimensionnels

- Possibilité de mélanger indices scalaires et associatifs
  - Ex.

```
$matrice_scal[0][0] = 2;
$matrice_mixte["Dupont"][0] = 30;

$matrice2 = array(
                array(1, 0, 0),
                array(0, 1, 0),
                array(0, 0, 1)     );
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Constantes

- Variables d'environnement
  - Ex. $_SERVER["PHP_SELF"]
    $_SERVER["SERVER_NAME"]
    $_SERVER["HTTP_REFERER"]
    $_SERVER["REMOTE_ADDR"]

- Constantes définies par l'utilisateur
  - Ex. define("MA_CHAINE", "Valeur de MA_CHAINE");
    define("PI", 3.14159265);

Opérateurs arithmétiques

- Opérateurs d'affectation — Exemples
  - Affectation simple : `$a = 2;`
  - Affectation multiple : `$a = $b = 2;`
  - Affectation + opération : `$a += 2;` `// a = a + 2`
  - Pré/post incrémentation/décrémentation :
    `++$a;` `--$a;`
    `$a++;` `$a--;`
  - Affectation conditionnelle : `$max = ($a > $b) ? $a : $b;`

- Opérateurs arithmétiques : + - * / % (modulo)

Opérateurs de chaînes

- Concaténation de chaînes de caractères : **.**
  - Ex. `$ch1 = $ch2 . $ch3;`
    `$ch1 .= $ch4;` `// $ch1 = $ch1 . $ch4;`

- Caractères spéciaux dans les chaînes (échappement)
  - Antislash : `\\`
  - Dollar : `\$`
  - Guillemets : `\"`

Opérateurs logiques

- Opérateurs logiques
  - ET : and ou &&
  - OU : or ou ||
  - OU exclusif : xor
  - NON : !
- Opérateurs de comparaison
  - Égalité/Différence : == != === pour les booléens
  - Inférieur/Supérieur : < >
  - Inférieur ou égal/Supérieur ou égal : <= >=

Tests (1/2)

if (condition) {instructions}
[elseif (condition) {instructions}]
[else {instructions}]

- Ex.

```
if ($a > $b)
        $res = "A > B"; // Une seule instruction

if ($a > $b) { // Plusieurs instructions
        $res = "A > B";
        $b = $a;
}
```

Tests (2/2)

```
if ($a > $b) {
        $res = "A > B";
}
else {
        $res = "A <= B";
}
```

```
if ($a > $b)
        $res = "A > B";
elseif  ($a < $b)
        $res = "A < B";
else
        $res = "A = B";
```

```
if (un_booleen === true)
        $res = "Vrai !";
```

Sélection par cas (1/2)

switch(variable) {cas}

- Ex.

```
switch($i) {
    case 0:
        $res = "i = 0";
    break;

    case 1:
        $res = "i = 1";
    break;

    case 2:
        $res = "i = 2";
    break;
}
```

Sélection par cas (2/2)

```
switch($ch) {
        case "a":
                $res = "A";
        break;
        case "bb":
                $res = "BB";
        break;
        case "ccc":
                $res = "CCC";
        break;
        default:
                $res = "Autre cas que a, bb ou ccc";
}
```

Boucles (1/2)

- Tant que : while(condition) {instructions}
  - Ex.
```
$i = 1; $res = "";
while ($i <= 10) {
        $res .= $i;
        $i++;
}
```

- Répéter tant que : do {instructions} while(condition)
  - Ex.
```
$i = 1; $res = "";
do {
        $res .= $i++;
} while ($i <= 10);
```

Boucles (2/2)

- Pour : for (initialisation; condition; incrémentation) {
  instructions
  }

  - Ex.
```
$res = "";
for ($i = 1; $i <= 10; $i++)
    $res .= $i;
```

Parcours de tableau scalaire

- Boucle "pour tout élément"

```
foreach (tableau as valeur) {
        instructions
}
```

  - Ex.

```
$tab = array ("Rouge", "Vert", "Bleu");
$res = "Valeurs :";
foreach ($tab as $val) {
        $res .= " $val";
}
```

Parcours de tableau associatif

```
foreach (tableau as clé => valeur) {
                    instructions
}
```

- Ex.
```
$tab = array(   "Rouge" => "#FF0000",
                "Vert" => "#00FF00",
                "Bleu" => "#0000FF"     );
$res = "Clés/Valeurs :";
foreach ($tab as $cle => $val) {
        $res .= " ($cle, $val)";
}
```

Inclusion de fichiers externes

- Fonction require() : Provoque une erreur fatale si le fichier requis manque (interruption du script)
- Fonction include() : Provoque seulement un avertissement (*warning*) si le fichier requis manque
- Évaluation des fichiers inclus en mode HTML
- Aide à séparer le code PHP et HTML

- Exemples (paramètre des fonctions : une URL)
  - require("biblio.class.php");
  - include("une_page_web.html");
  - include("http://serveur.fr/pg.html");

PHP Standards Recommendations (PSR)

- Recommandations pour améliorer l'interopérabilité des applications PHP
- Tendent à devenir des standards
- Parmi celles qui sont validées :
  - PSR-1 : Basic Coding Standard
  - PSR-2 : Coding Style Guide

L’essentiel de PSR-1

Balises PHP	<?php ?> ou <?= ?>
Encodage des caractères	UTF-8 without BOM
Nom de classes	En StudlyCaps
Constantes	En MAJUSCULES
Noms de fonctions/méthodes	En camelCase()

L’essentiel de PSR-2 (1/2)

Code PHP	En PSR-1
Sauts de ligne	Unix
Longueur des lignes de code	80 caractères maxi de préférence
Instructions	Une seule par ligne
Indentation	4 espaces (pas de tabulation)
Blocs de code	Retour à la ligne après { } seule sur une ligne

L’essentiel de PSR-2 (2/2)

Visibilité des attributs/méthodes	Doit être déclarée
Instructions/constantes PHP	En minuscules
Paramètres de fonctions/méthodes	Pas d’espace avant la virgule Une espace après la virgule
Appel de fonctions/méthodes	Pas d’espace avant les ()
Structures de contrôle	Une espace après l’instruction Entre les (), pas d’espace après (ni avant), mais une espace après)

N° de la question : 821

Sondage express

Comment affiche-t-on des données sur une page web ?

A) Avec une instruction du type echo "blablabla";

B) Avec une instruction du type echo " blablabla ";

C) Avec une instruction du type print(" blablabla ");

D) Avec une instruction du type print_r($tableau);

E) Grâce aux gabarits

Répondre sur https://toreply.univ-lille.fr

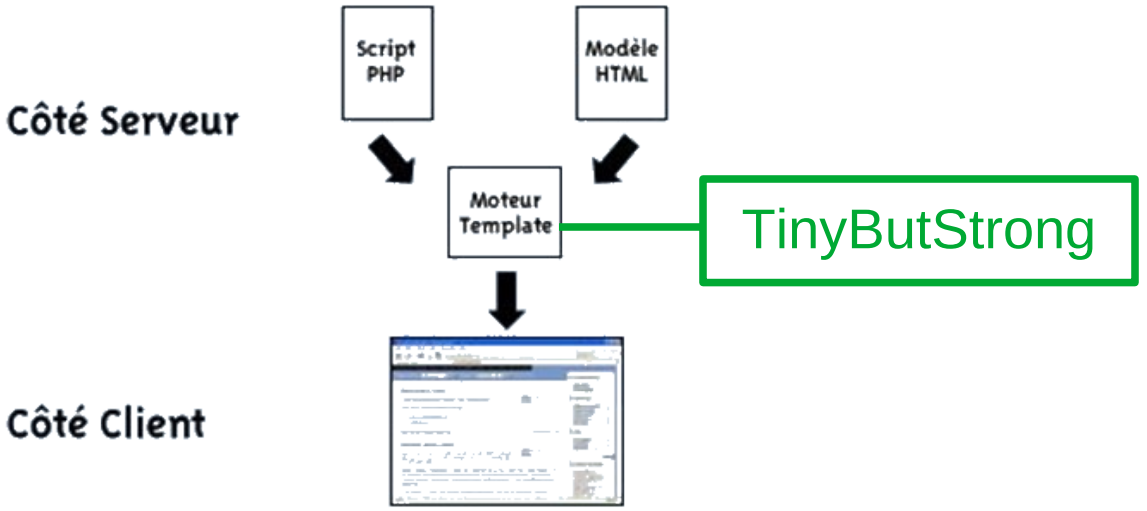
Question n° 821

Plan du cours

- ✓ Objectifs du cours
- ✓ PHP objet
- Gabarits
- Architecture MVC
- Formulaires de saisie
- Sessions
- Interface PHP-base de données

Objectifs

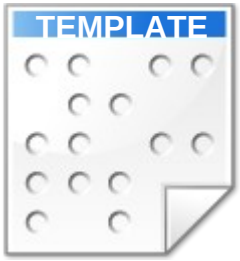
- Séparation du code PHP et HTML
- Partage de gabarits entre plusieurs applications web



phpcodeur.net

Exemple de gabarit

```
<!-- fichier HTML5 gab1.tpl.html -->
< !DOCTYPE html>
<html lang="fr">
        <head>
                <meta charset="utf-8" />
                <title>Gabarit simple</title>
        </head>
        <body>
                <ul>
                        <li>[onshow.nom]</li>       <!-- paramètres -->
                        <li>[onshow.prenom]</li>    <!-- du gabarit -->
                </ul>
        </body>
</html>
```



Affectation de variables à un gabarit

```
// Dans un fichier PHP

// Création d'un objet TinyButStrong (bibliothèque en PHP objet)
require("tbs_class.php");
$tbs = new clsTinyButStrong;

// Préparation des données
$nom = "Jérôme";
$prenom = "Darmont";

// Mise en œuvre du gabarit
$tbs->LoadTemplate("gab1.tpl.html");          // Chargement du gabarit
$tbs->Show();                  // Affectation des valeurs des variables PHP
                               // aux paramètres du gabarit
```

Exemple de gabarit avec répétition

```
<html lang="fr"> <!-- fichier gab2.tpl.html -->
        <-- il faudrait insérer la section <head> </head> ici -->
        <body>
                <table>
                        <caption>[onshow.legende]</caption>
                                <tr>
                                        <th>Nom</th>
                                        <th>Prénom</th>
                                </tr>
                                <tr>
                                        <td>[nom.val;block=tr]</td>
                                        <td>[prenom.val;block=tr]</td>
                                </tr>
                </table>
        </body>
</html>
```

Mise en œuvre d'un bloc de répétition

```
// Création d'un objet TinyButStrong
require("tbs_class.php");
$tbs = new clsTinyButStrong;

// Préparation des données
$legende = "Liste des personnes";
$tabNoms = array("Metzler", "Scuturici", "Cugliari");
$tabPrenoms = array("Guillaume", "Mihaela", "Jairo");

// Mise en œuvre du gabarit
$tbs->LoadTemplate("gab2.tpl.html");
$tbs->MergeBlock("nom", $tabNoms);
$tbs->MergeBlock("prenom", $tabPrenoms);
$tbs->Show();
```

N° de la question : 89

Sondage express

Pourquoi utilise-t-on des gabarits
pour afficher des données dynamiques ?

A) Aucune idée

B) Pour se compliquer la vie

C) Pour séparer le code HTML et le code PHP

Répondre sur https://toreply.univ-lille.fr

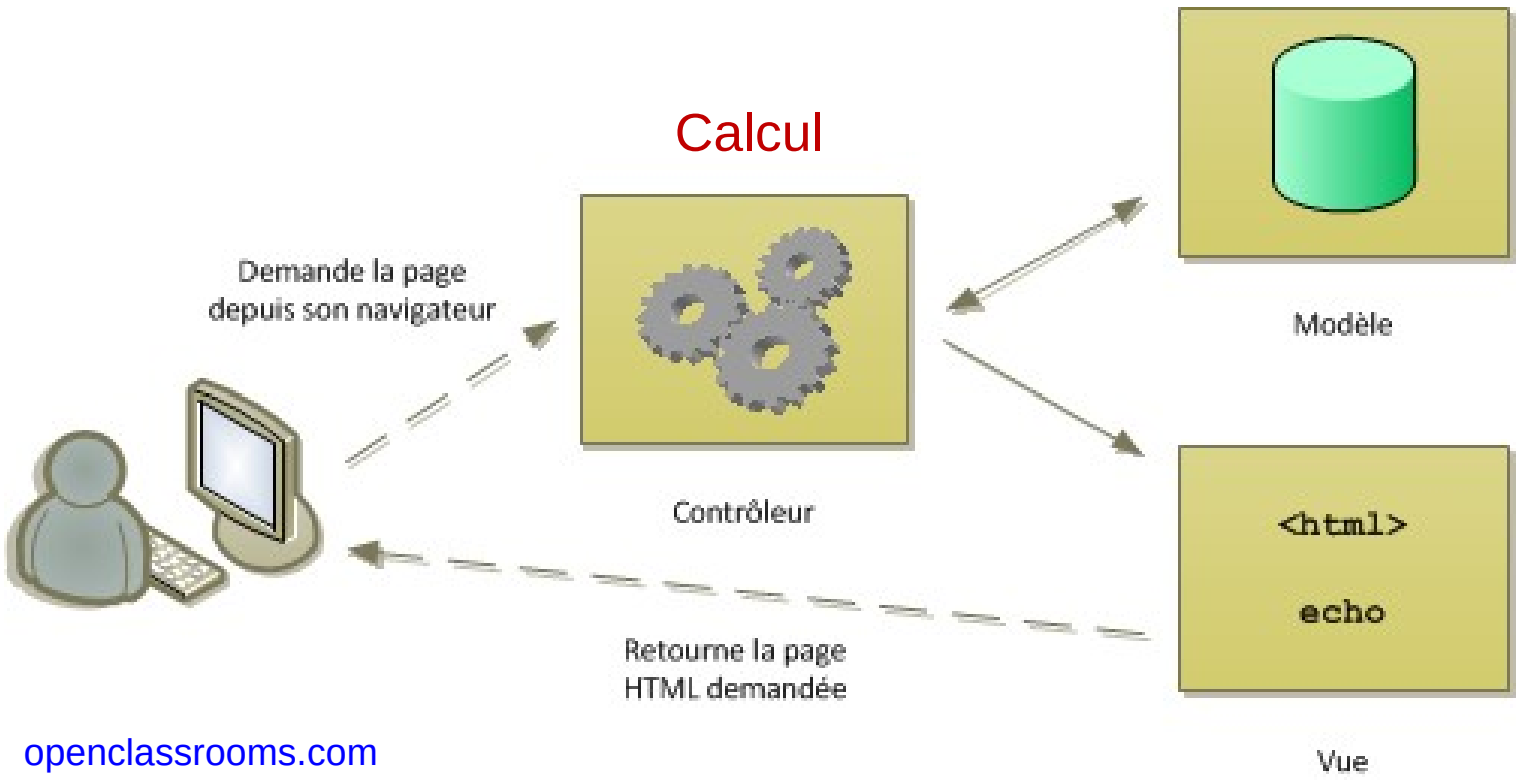
Question n° 89

Plan du cours

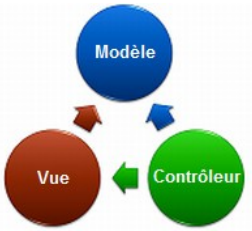
- ✓ Objectifs du cours
- ✓ PHP objet
- ✓ Gabarits
- Architecture MVC
- Formulaires de saisie
- Sessions
- Interface PHP-base de données

Trygve Reenskaug
1978

Principe de MVC

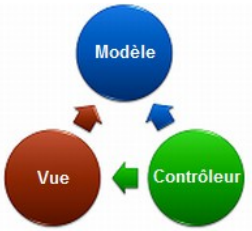


openclassrooms.com



Modèle

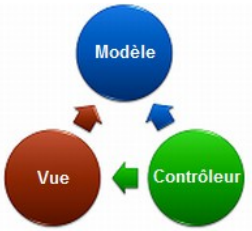
- Gestion des données
  - Base de données ou fichiers
- Ensemble de classes et de méthodes
  - Mises à jour (ajout/modification/suppression)
  - Interrogation
- Langages
  - PHP
  - SQL



Vue

- Présentation des résultats
- Interaction avec l'utilisateur·trice

- Langages
  - HTML uniquement
  - Utilisation de gabarits



Contrôleur

- Analyse des requêtes de l'utilisateur·trice
- Interrogation ou mise à jour du modèle
- Modification de la vue

- Langage
  - PHP

N° de la question : 987

Sondage express

Combien de fichiers au minimum faut-il pour mettre en œuvre une architecture MVC ?

A) 1 fichier

B) 2 fichiers

C) 3 fichiers

Répondre sur https://toreply.univ-lille.fr

Question n° 987

Plan du cours

- ✓ Objectifs du cours
- ✓ PHP objet
- ✓ Gabarits
- ✓ Architecture MVC
- Formulaires de saisie
- Sessions
- Interface PHP-base de données

Saisie de données dans une page web

- Définition d'un formulaire : <form> </form>
  - Attribut action : URL de la page PHP à exécuter après validation du formulaire
  - Attribut method : méthode de transmission des données (valeurs possibles : get et post)
  - Attribut enctype : type d'encodage (par défaut application/x-www-form-urlencoded ou multipart/form-data pour envoyer des fichiers)

Méthodes de transmission des données

- Différence entre les méthodes get et post
  - get : apparition des valeurs saisies en paramètres de l'URL de la page action
  - post : . valeurs saisies cachées
    . quantité de données possible plus importante
- Exemple
  <form action="ajout_etu.php" method="post">…</form>
- Structure d'un formulaire : ensemble de zones de saisie (groupes de champs)
- Groupe de champs : <fieldset> </fieldset>

Champs (1/5)

- Saisie dans un formulaire : <input />
  - Attribut obligatoire : name, nom de la variable
- Champ texte : <input type="text" size="" />
  - Ex. <input type="text" name="nom" size="30" />
- Suggestion de valeurs : <datalist>…</datalist>
  - Ex.

```
<datalist id="prop_noms">
        <option value="Dupond">
        <option value="Durand">
        <option value="Martin">
</datalist>
<input type="text" name="nom" list="prop_noms" />
```

Champs (2/5)

- Vérification syntaxique : expressions régulières
  - Ex. <input type="email" pattern="[^ @]*@[^ @]*" />
- Champ mot de passe : <input type="password" />
  - Ex. <input type="password" name="passwd" size="8" />
- Champ caché : <input type="hidden" value="" />
  - Ex. <input type="hidden" name="numetu" value="10" />
- Fichier : <input type="file" />
  - Ex. <input type="file" name="Fichier_téléchargé" />

Champs (3/5)

- Bouton radio : <input type="radio" value="" />
  - Ex.
    Homme : <input type="radio" name="genre" value="H" />
    Femme : <input type="radio" name="genre" value="F" />

- Case à cocher : <input type="checkbox" />
  - Ex.
    choix 1 : <input type="checkbox" name="choix[]" value="choix1" />
    choix 2 : <input type="checkbox"

Champs (4/5)

- Boutons de commande
  - Attribut type = submit | reset : validation ou réinitialisation du formulaire
  - Attribut value : légende du bouton
  - Ex. <input type="submit" name="Valider" value="Valider" />
    <input type="reset" name="Annuler" value="Annuler" />

- Zone de texte long : <textarea> </textarea>
  - Attribut name : nom de la zone de texte
  - Attributs rows et cols : nombre de lignes / colonnes
  - Ex. <textarea name="texte" rows="10" cols="60">
    </textarea>

Champs (5/5)

- Liste déroulante : <select> </select>
  - Attribut name : nom de la variable choix
  - Élément <option> </option> : objet de la liste
  - Attribut selected de <option> : choix par défaut

  - Ex.
```
<select name="annee">
    <option>L3</option>
    <option selected="selected">M1</option>
    <option>M2</option>
</select>
```

Accessibilité

- Description de champ : <label> </label>
  - Ex.
```
<label for="id_nom">Nom</label>
<input type="text" id="id_nom" name="nom" />
```

- Légende de zone de saisie : <legend> </legend>
  - Ex.
```
<fieldset>
    <legend>État civil de l'étudiant</legend>
    ...
</fieldset>
```

Exemple complet de formulaire (1/2)

```
<form action="ajout_etu.php" method="post">
        <fieldset>
                <legend>Choix de formation</legend>
                <p>
                        <label for="idnom">Nom</label>
                        <input type="text" id="idnom" name="nom" size="50" />
                </p>
                <p>
                        <label for="idage">Âge</label>
                        <input type="text" id="idage" name="age" size="2" />
                </p>
```

Exemple complet de formulaire (2/2)

```
                <p>
                        <label for="idan">Année</label>
                                <select id="idan" name="annee">
                                        <option>L3</option>
                                        <option selected="selected">M1</option>
                                        <option>M2</option>
                                </select>
                </p>
                <input type="hidden" name="action" value="ajout" />
                <p>
                        <input type="reset" value="Annuler" />
                        <input type="submit" value="Valider" />
                </p>
        </fieldset>
</form>
```

Exploitation des données d'un formulaire

- Dans la page cible (Ex. ajout_etu.php du champ action)
  - Tableaux associatifs $_GET[ ] et $_POST[ ]
  - Permettent d'accéder aux valeurs transmises par les méthodes get et post des formulaires, respectivement
- Exemple :

```
$nom    = $_POST["nom"];
$age    = $_POST["age"];
$annee = $_POST["annee"];
```

- Cas particuliers :
  - Cases à cocher (valeur "on" si cochée)
  - Fichiers

Téléchargement de fichier (1/2)

- Étape 1 : formulaire dans une page HTML
  - Ex.

```
<form action="telechargement.php" method="post"
                                enctype="multipart/form-data">
<fieldset>
    <legend>Téléchargement de fichier<legend>
    <input type="hidden" name="MAX_FILE_SIZE" value="50000" />
    Fichier : <input name="monfichier" type="file" />
    <input type="submit" value="Télécharger" />
</fieldset>
</form>
```

- Étape 2 : traitement à l'aide de PHP

Téléchargement de fichier (2/2)

- Variables disponibles dans la page cible
  - $_FILES["monfichier"]["name"] : nom original du fichier
  - $_FILES["monfichier"]["type"] : type du fichier
  - $_FILES["monfichier"]["size"] : taille du fichier
  - $_FILES["monfichier"]["tmp_name"] : nom temporaire du fichier sur la machine serveur

- Exemple de code dans la page telechargement.php

```
$destination = "/home/jd/public_html/" . $_FILES["monfichier"]["name"];
$res = move_uploaded_file($_FILES["monfichier"]["tmp_name"],
                          $destination);
if ($res) $etat = "Fichier téléchargé avec succès";
else $etat = $_FILES["monfichier"]["error"];
```

N° de la question : 790

Sondage express

Si un formulaire envoie des données textuelles et un fichier image, quels tableaux systèmes PHP seront disponibles pour les récupérer ?

A) $_GET[ ]

B) $_POST[ ]

C) $_FILES[ ]

Répondre sur https://toreply.univ-lille.fr

Question n° 790

Plan du cours

- ✓ Objectifs du cours
- ✓ PHP objet
- ✓ Gabarits
- ✓ Architecture MVC
- ✓ Formulaires de saisie
- Sessions
- Interface PHP-base de données

Sessions

- Objectif : Stockage de variables lors de la navigation sur plusieurs pages web successives

- Utilisations courantes :
  - Identification des visiteurs d'un site par login et mot de passe stockés dans une base de données
  - Gestion du profil des utilisateurs
  - …

Fonctions de session

- session_start(); // Indique un environnement session
  // Doit être dans toutes les pages
  // Doit précéder tout entête HTML

- session_id(); // Indique l'identifiant de la session
- session_name(); // Indique le nom de la session

- session_destroy(); // Détruit la session

Variables de session

- Tableau associatif de variables (ou de tableaux) $_SESSION[ ]
  - Ex.
```
$_SESSION["nomEtu"] = "Darmont";
$_SESSION["tabNotes"] = array(18, 19, 19.5, 20);
```
- Supprimer une variable : unset()
  - Ex.
```
unset($_SESSION["tabNotes"]);
```
- Tester l'existence d'une variable : isset()
  - Ex.
```
if (isset($_SESSION["nomEtu"]))
        // On peut travailler avec la variable de session
else
        // Il faut probablement initialiser la variable de session
```

Exemple de session (vue)

```
<!DOCTYPE html> <!-- gab-session.tpl.html -->
<html lang="fr">
    <head>
        <meta charset="utf-8" />
        <title>Exemple de session</title>
    </head>
    <body>
        <ul>
            <li>Identifiant de session : [onshow.id]</li>
            <li>Nom de session : [onshow.session]</li>
            <li>Variable de session nom : [onshow.nom]</li>
            <li>Variable de session prénom : [onshow.prenom]</li>
        </ul>
        <a href="controleur-session.php">Page suivante</a>
    </body>
</html>
```

Exemple de session (modèle)

```php
<?php // modele-session.inc.php

class sessionData{
    function get_id() {
        return session_id();
    }
    function get_name() {
        return session_name();
    }
    function test_var($var_session) {
        if (isset($var_session)) return $var_session;
        else return "indéfinie";
    }
}
?>
```

Exemple de session (contrôleur 1/3)

```
<?php // controleur-session.php

// Mode session activé
session_start();

// Inclusions
require("modele-session.inc.php");
require("tbs_class.php");

// Créations d'objets
$data = new sessionData;
$tbs = new clsTinyButStrong;

// Initialisation de l'étape 1
if (!isset($_SESSION["etape"]))
        $_SESSION["etape"] = 1;
```

Exemple de session (contrôleur 2/3)

```
// Moteur de l'application
switch ($_SESSION["etape"]) {
    case 1: // Initialisation des variables session
        $_SESSION["nom"] = "Darmont";
        $_SESSION["prenom"] = "Jérôme";
        $_SESSION["etape"] = 2;
    break;
    case 2: // Suppression de la variable prenom
        unset($_SESSION["prenom"]);
        $_SESSION["etape"] = 3;
    break;
    case 3: // Suppression de la session
        session_destroy();
    break;
}
```

Exemple de session (contrôleur 3/3)

```
// Affichage du résultat

$id = $data->get_id();
$session = $data->get_name();
$nom = $data->test_var($_SESSION["nom"]);
$prenom = $data->test_var($_SESSION["prenom"]);

$tbs->LoadTemplate("gab-session.tpl.html");
$tbs->Show();

?>
```

N° de la question : 808

Sondage express

Mine de rien, nous venons de concevoir notre première application web MVC !

Répondre sur https://toreply.univ-lille.fr

Question n° 808

Plan du cours

- ✓ Objectifs du cours
- ✓ PHP objet
- ✓ Gabarits
- ✓ Architecture MVC
- ✓ Formulaires de saisie
- ✓ Sessions
- Interface PHP-base de données

Généralités


Michael Widenius

- Principe : imbrication de requêtes SQL dans PHP + formulaires HTML pour les mises à jour

- SGBD utilisé : MariaDB
  - SGBD relationnel multi-utilisateurs rapide et open-source
  - 1995 : MySQL AB (v1-v5.0)
  - 2008 : rachat de MySQL AB par Sun Microsystems (v5.1)
  - 2010 : rachat de Sun Microsystems par Oracle (v5.5-v8.0)
  - 2010 : MariaDB *fork* libre de MySQL (v5.2-v10.3)
    Désormais utilisé dans de nombreuses distributions Linux
  - 2023 : MariaDB v11.2.1

PHP Data Objects (PDO)

- Accès unifié à différents SGBD
  - Remplace les API précédentes (ex. API PHP-MySQL), obsolètes et vouées à disparaître
- Préparation des requêtes
  - Performance lors d'exécutions multiples
  - Sécurité (notamment contre les injections de code)
- Gestion des exceptions (erreurs)

Connexion à une BD

- Connexion

  $pdo = new PDO($id_serveur_BD, $login, $mot_de_passe);
  - $id_serveur_BD = "pilote:host=serveur;dbname=nomBD";
  - Ex.
    $c = new PDO("mysql:host=localhost;dbname=darmont", "darmont", "x");

- Gestion des erreurs

  try { instructions } catch () { traitementDesErreurs }
  - Ex.

```
try {
        // Connexion
} catch (PDOException $erreur) {
        $message = $erreur->getMessage() ;
}
```

Exécution d’une requête

- Préparation de la requête
  $res = $pdo->prepare(requêteSQL)
  - Ex. $res = $c->prepare("select nom, prenom from etudiant");
- Exécution de la requête
  - Ex. $res->execute();
- Accès au résultat de la requête (ligne par ligne)
  - Ex. foreach($res as $ligne)
    $nom_complet[ ] = $ligne["nom"] . " " . $ligne["prenom"];

Résultat intégral

- Chargement complet du résultat de la requête en mémoire
  - Rapide, utile quand on utilise des gabarits (*templates*)
  - À éviter lorsque le résultat est volumineux
- Exemple

```
$res = $c->prepare("select nom, prenom from etudiant");
$res->execute();
$data = $res->fetchAll(PDO::FETCH_ASSOC);
foreach($data as $ligne)
        $nom_complet[ ] = $ligne["nom"] . "  " . $ligne["prenom"];
```

Requêtes paramétrées

```
// Exemple avec paramètres anonymes
$res = $c->prepare("select nom, prenom from etudiant
                    where datenaiss > ? and note >= ?");

$res->execute(["2018-01-01", 10]);
// Les paramètres doivent être passés dans l'ordre
```

Requêtes paramétrées multiples (1/2)

```
// Exemple avec paramètres nommés
$res = $c->prepare("insert into ETUDIANT
                    values (:id, :nom, :prenom, NULL, NULL, NULL)");

// Liaison des paramètres
$res->bindParam(":id", $id);
$res->bindParam(":nom", $nom);
$res->bindParam(":prenom", $prenom);
```

Requêtes paramétrées multiples (2/2)

```
// 1re insertion
$id = 800;
$nom = "Bentayeb";
$prenom = "Fadila";
$res->execute();

// 2e insertion
$id = 810;
$nom = "Harbi";
$prenom = "Nouria";
$res->execute();
```

Exemple complet

```
try {
        // Connexion
        $c = new PDO("mysql:host=localhost;dbname=darmont", "darmont", "x");
        $c->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        // Requête d'interrogation
        $res = c->prepare("select nom, prenom from etudiant");
        $res->execute();
        foreach($res as $ligne)
                $nom_complet[ ] = $ligne["nom"] . "  " . $ligne["prenom"];
        $n = $res->rowCount() . " résultat(s)";
        // NB : rowCount() fonctionne aussi avec les requêtes de mise à jour

} catch (PDOException $erreur) { // Gestion des erreurs
        $message = $erreur->getMessage();
}
```

Mise à jour d'une base de données

```
<?php // Suite de l'exemple ajout_etu.php (transparents 73-75)
$c = new PDO("mysql:host=localhost;dbname=darmont",
                                        "darmont", "x");
$resultat = $c->prepare("insert into etudiant values (?, ?, ?)");

$resultat->execute([    $_POST["nom"],
                        $_POST["age"],
                        $_POST["annee"]     ]);
if ($resultat->rowCount() > 0)
        $message = "Etudiant.e ajouté.e";
?>
```

Cas pratique MVC

- Affichage des données d'une table

codemat	libelle	coef
STA	Statistique	0.4
INF	Informatique	0.4
ECO	Économétrie	0.2

Identifiants de connexion

```php
<?php    // fichier connect.inc.php

// Pour des raisons de sécurité, on ne définit les identifiants de connexion
// à la base de données qu'une seule fois, dans ce fichier.

$host = "mysql02.univ-lyon2.fr";      // Adresse du serveur MariaDB
$dbname = "php_darmont";              // Nom de la base de données
$login = "php_darmont";               // Nom de l'utilisateur
$password = "jeneleddiraipas";        // Mot de passe

?>
```

Modèle

```php
<?php    // fichier modele.class.php
class Modele {
        private $id_connexion;
        private function connexion() {
                require("connect.inc.php");
                $this->id_connexion = new PDO("mysql:host=$host;dbname=$dbname",
                                                $login, $password);
        }
        public function lireMatieres() {
                $this->connexion();
                $res = $this->id_connexion->prepare("select * from matiere");
                $res->execute();
                return $res;
        }
}
?>
```

Vue

```
<!DOCTYPE html>
<html lang="fr"> <!-- fichier vue.tpl.html -->
        <head>
                <meta charset="utf-8" />
                <title>Cas pratique MVC</title>
        </head>
        <body>
                <table>
                        <caption>Liste des matières</caption>
                        <tr> <th>codemat</th> <th>libelle</th> <th>coef</th> </tr>
                        <tr>
                                <td>[codemat.val;block=tr]</td>
                                <td>[libelle.val;block=tr]</td>
                                <td>[coef.val;block=tr]</td>
                        </tr>
                </table>
        </body>
</html>
```

Contrôleur (1/3)

```
<?php // fichier controleur-bd.class.php

class Affichage {

        private $tbs;

        function __construct($tbs) {
                $this->tbs = $tbs;
        }
```

Contrôleur (2/3)

```
        public function afficherMatieres($data) {
                $i = 0;
                $tabCode = array() ;
                $tabLibe = array();
                $tabCoef = array();
                foreach($data as $ligne) {
                        $tabCode[$i++] = $ligne["codemat"];
                        $tabLibe[$i++] = $ligne["libelle"];
                        $tabCoef[$i++] = $ligne["coef"];
                }
                $this->tbs->LoadTemplate("vue.tpl.html");
                $this->tbs->MergeBlock("codemat", $tabCode);
                $this->tbs->MergeBlock("libelle", $tabLibe);
                $this->tbs->MergeBlock("coef", $tabCoef);
                $this->tbs->Show();
        }
}
?>
```

Contrôleur (3/3)

```php
<?php   // fichier controleur-bd.php

        require("tbs_class.php");
        require("modele.class.php");
        require("controleur-bd.class.php");

        $tbs = new clsTinyButStrong;
        $mod = new Modele();
        $aff = new Affichage($tbs);

        $data = $mod->lireMatieres();
        $aff->afficherMatieres($data);
?>
```

N° de la question : 691

Sondage express

Quelle sont les instructions de connexion à MariaDB et de gestion d'erreur ?

A) mysql_connect() or die()

B) mysqli_connect() or die()

C) try { new PDO } catch { }

D) Autre chose, il y a forcément un piège !

Répondre sur https://toreply.univ-lille.fr

Question n° 691

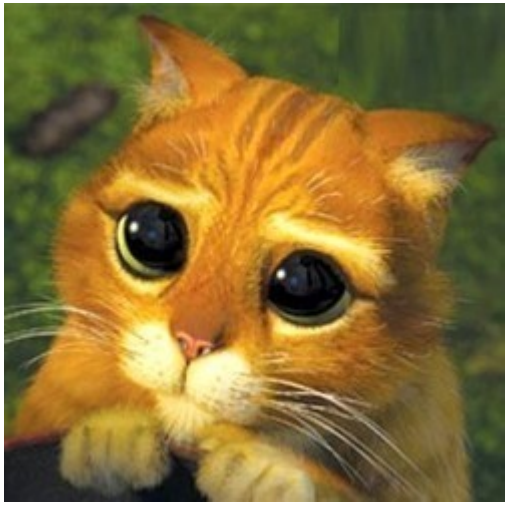
Plan du cours

- ✓ Objectifs du cours
- ✓ PHP objet
- ✓ Gabarits
- ✓ Architecture MVC
- ✓ Formulaires de saisie
- ✓ Sessions
- ✓ Interface PHP-base de données

N° de la question : 969

Sondage express

Votre appréciation sur ce cours magistral ?



Répondre sur https://toreply.univ-lille.fr

Question n° 969