# Développement WEB

## Présentation du cours
## Introduction au langage PHP

Luc Dartois

luc.dartois@u-pec.fr

# Objectifs du cours

## Objectifs du cours

- Comprendre le fonctionnement d'un site web simple,
- Savoir développer un site web relié à une base de données,
- Sensibiliser à la sécurité d'un site internet,
- Familiariser avec la structure MVC (Modèle, Vue, Contrôleur).

## Rythme

- 10 semaines de cours
- 4 Cours Magistraux sur les notions théoriques
- 1 séance de TD par semaine
- 2 séances de TP par semaine.

# Outils et ressources

## HTML, CSS, JavaScript

- Vos cours
- www.w3schools.com
- developer.mozilla.org/fr

## PHP

- Ce cours
- www.php.net

## SQL

- Vos cours de SGBD
- www.w3schools.com/sql/

# Modèle

Une application Web est une application avec un *serveur* fournissant des *clients*.

- Le serveur (site web) fournit des services.
- Les clients (utilisateurs) se connectent au serveur.

Elle se décompose en 3 couches principales, remplissant des rôles complémentaires.

# Trois couches

## Front end (ou devanture)

### Structure

Divise une page en blocs.
→ HTML

### Présentation

Définit l'esthétique de la page.
→ CSS

### Applicatif

Permet l'évolution de la page.
→ JavaScript

Le code du front end est donné par le serveur, mais interprété/exécuté par le navigateur du client !

# Trois couches

## Base de Données

### SGBD

Contient les données de l'application.
Pour nous, ce sera une base de données SQL.

Seule la back end a accès (lecture et écriture) à la base de données, ce qui permet de sécuriser les données.

## Back end (ou arrière-plan)

### PHP

Pour nous, ce sera du PHP. Le PHP sera interprété afin de générer une page HTML envoyée au client.

- Le back end fait le lien entre la Front end et la Base de données.
- Le code du back end est interprété par le serveur avant d'être fourni au client.
- Le client n'a donc pas accès au code, il est sécurisé.

# Protocole Web

- Un client lance une requête sur un serveur :
  `http://www.monsite.fr/bonjour.php`

# Protocole Web

- Un client lance une requête sur un serveur :
  `http://www.monsite.fr/bonjour.php`
- Le serveur (`www.monsite.fr`) interprète le fichier `bonjour.php`

# Protocole Web

- Un client lance une requête sur un serveur : `http://www.monsite.fr/bonjour.php`
- Le serveur (`www.monsite.fr`) interprète le fichier `bonjour.php`
- Il obtient du code HTML.

# Protocole Web

- Un client lance une requête sur un serveur : `http://www.monsite.fr/bonjour.php`
- Le serveur (`www.monsite.fr`) interprète le fichier `bonjour.php`
- Il obtient du code HTML.
- Il envoie ce code au client.

# Protocole Web

- Un client lance une requête sur un serveur : `http://www.monsite.fr/bonjour.php`
- Le serveur (`www.monsite.fr`) interprète le fichier `bonjour.php`
- Il obtient du code HTML.
- Il envoie ce code au client.
- Le navigateur interprète le code HTML et l'affiche.

# Protocole Web

- Un client lance une requête sur un serveur : `http://www.monsite.fr/bonjour.php`
- Le serveur (`www.monsite.fr`) interprète le fichier `bonjour.php`
- Il obtient du code HTML.
- Il envoie ce code au client.
- Le navigateur interprète le code HTML et l'affiche.

## URL, pour Uniform Resource Locator

La requête `http://www.monsite.fr/bonjour.php` est une **URL**. Ici, elle est composée de :

- Un protocole (comment) : http
- Un nom de domaine (où) : www.monsite.fr
- Un document (quoi) : bonjour.php

# Protocole Web

- Un client lance une requête sur un serveur : `http://www.monsite.fr/bonjour.php`
- Le serveur (`www.monsite.fr`) interprète le fichier `bonjour.php`
- Il obtient du code HTML.
- Il envoie ce code au client.
- Le navigateur interprète le code HTML et l'affiche.

## protocole://hostname:port/path?parameters

- Le protocole sera le plus souvent http ou https (voire ftp)
- hostname est le nom du domaine.
- port est optionnel, permet de préciser où on se connecte sur le serveur
- path décrit le chemin du fichier depuis la racine *du site*.
- parameters permet de donner des informations pour le script exécuté.

# Le langage PHP

Les clients reçoivent des pages HTML (majoritairement).
Les fichiers PHP doivent donc générer et fournir du code HTML.
En pratique, on intègre le code source PHP dans un document HTML grâce aux balises :

```
<?php
?>
```

Comme dans l'exemple bonjour.php !

# Le langage PHP

Les clients reçoivent des pages HTML (majoritairement).
Les fichiers PHP doivent donc générer et fournir du code HTML.
En pratique, on intègre le code source PHP dans un document HTML grâce aux balises :

```
<?php
?>
```

Comme dans l'exemple bonjour.php !

## Générer de l'HTML

Le code PHP génère de l'HTML en imprimant directement l'HTML que l'on souhaite intégrer, avec notamment la fonction `echo`.
On peut avoir plusieurs paires de balises PHP, qui partagent alors leurs variables, leurs fonctions et leur structure.

# Le langage PHP

Les clients reçoivent des pages HTML (majoritairement).
Les fichiers PHP doivent donc générer et fournir du code HTML.
En pratique, on intègre le code source PHP dans un document HTML grâce aux balises :

```
<?php
?>
```

Comme dans l'exemple bonjour.php !

## Interpréter le code

Pour exécuter votre code PHP, vous pouvez :

- Placer votre fichier dans `public_html` sur les machines virtuellesde l'IUT.
  → Vous pouvez accéder à votre fichier via :
  http://dwarves.iut-fbleau.fr/~login/nomFichier
- En terminal, via `php monfichier`.
  → Utile pour tester et récupérer les messages d'erreur.

# Le langage PHP (2)

Le PHP est un langage interprété côté serveur :

- Initialement signifiait Personal Home Page.
- Renommé en PHP : Hypertext Preprocessor.
- Reprend beaucoup de la syntaxe du C et du Perl.
- Supporte tous les standards du web.
- "PHP is used by 78.1% of all the websites whose server-side programming language we know." - W3Techs

# Les variables

Le PHP est un langage faiblement typé.
Les variables sont typées *implicitement* et *dynamiquement*. Il n'est pas nécessaire de les déclarer, ni de les initialiser !
Elles n'existent que le temps de l'interprétation. Une fois la page html calculée, elles cessent d'exister.

# Les variables

Le PHP est un langage faiblement typé.
Les variables sont typées *implicitement* et *dynamiquement*. Il n'est pas nécessaire de les déclarer, ni de les initialiser !
Elles n'existent que le temps de l'interprétation. Une fois la page html calculée, elles cessent d'exister.
Une variable possède tout de même un type, qui peut être :

| Type | Description | Valeur |
|---|---|---|
| NULL | Pointeur vide | NULL |
| boolean | Vrai ou faux | True |
| integer | Nombre entier | 12 |
| double | Nombre réel | 4.2 |
| string | Chaîne de caractères | "Bonjour" |
| array | Tableau | ["Pi",3,3.14] |

# Les variables

C'est l'utilisation d'une variable qui décide de son type. L'assignation se fait avec le signe =.

| Si j'écris : | alors |
|---|---|
| $var; | var est une variable de type NULL |
| $var=50; | var est maintenant un entier |
| $var+=5; | var vaut maintenant 50+5=55 |
| $var="oui" | var est maintenant une chaîne de caractères |
| $var=0x10 | var est un entier valant 16 (0x10 en héxa) |

→ Ce n'est pas parce que c'est possible que c'est nécessaire. Réutiliser des variables peut engendrer des erreurs, pour un gain minime.

# Les variables

Les fonctions d'information :

- `var_dump` affiche des informations structurées pour une variable,
- `print_r` affiche des informations lisibles.

# Les variables

Les fonctions d'information :

- `var_dump` affiche des informations structurées pour une variable,
- `print_r` affiche des informations lisibles.

D'autres fonctions :

| Fonction | Description |
|---|---|
| empty($var) | Renvoie vrai si la variable est vide |
| isset($var) | Renvoie vrai si la variable a été créée |
| unset($var) | Libère la variable |
| gettype($var) | Renvoie le type de la variable |
| settype($var,type) | Convertit la variable dans le type |
| is_string($var) | Renvoie vrai si la variable est un String |

# Les variables

Les fonctions d'information :

- `var_dump` affiche des informations structurées pour une variable,
- `print_r` affiche des informations lisibles.

D'autres fonctions :

| Fonction | Description |
|---|---|
| empty($var) | Renvoie vrai si la variable est vide |
| isset($var) | Renvoie vrai si la variable a été créée |
| unset($var) | Libère la variable |
| gettype($var) | Renvoie le type de la variable |
| settype($var,type) | Convertit la variable dans le type |
| is_string($var) | Renvoie vrai si la variable est un String |

"Pointeurs" :
Si $var vaut "toto", alors
${$var} désigne la variable $toto !

# Les chaînes de caractères

Les chaînes de caractères sont délimitées par des guillemets :
ex : ”Hello world !”
Ce qui est précédé de $ est évalué :
ex :

```
$var="world";
"Hello $var !" -> "Hello world !"
```

On protège certains caractères avec \ :
ex : ”Vous devez 3.50\$”
Ou l’ensemble de la chaîne avec des guillemets simples :
ex : ’Vous devez 3.50$’

# Les chaînes de caractères

Les chaînes de caractères sont délimitées par des guillemets :
ex : ”Hello world !”
Ce qui est précédé de $ est évalué :
ex :

```
$var="world";
"Hello $var !" -> "Hello world !"
```

On protège certains caractères avec \ :
ex : ”Vous devez 3.50\$”
Ou l'ensemble de la chaîne avec des guillemets simples :
ex : 'Vous devez 3.50$'
Afficher une chaîne l'ajoute dans le fichier html produit :

- Avec echo
- Avec printf, de la même façon qu'en C

# Les tableaux

Les tableaux sont des ensembles de couples (clé,valeur) qui peuvent être numérique ou alphabétique ! On utilise la fonction `array` pour créer un tableau :

- \$tab=array( cle1 => valeur1,
  cle2 => valeur2,
  ...
  cleN => valeurN);
- Si la clé n'est pas précisée, la valeur est stockée après la dernière clé entière.
- On ajoute une valeur avec :
  \$tab[]=NouvelleValeur;
- On réassigne une clé avec :
  \$tab[clé]=valeur;
- On supprime un couple avec :
  unset(\$tab[clé]);

# Les tableaux

Les tableaux sont des ensembles de couples (clé,valeur) qui peuvent être numérique ou alphabétique !

Si j'écris : $tab=array(
3,
5,
8=>"ok",
8.5,
"a"=>"aa",
9);

$tab[]=42;

# Les tableaux

Les tableaux sont des ensembles de couples (clé,valeur) qui peuvent être numérique ou alphabétique !

Si j'écris : $tab=array(
3,
5,
8=>"ok",
8.5,
"a"=>"aa",
9);

$tab[]=42;

J'obtient le tableau $tab :

| Clé | Valeur |
|---|---|
| 0 | 3 |
| 1 | 5 |
| 8 | ok |
| 9 | 8.5 |
| a | aa |
| 10 | 9 |
| 11 | 42 |

# Les commentaires

Du bon code étant du code commenté...
Il existe 3 façons de commenter du code PHP :

- // commente le reste de la ligne, à la C++
- # commente également la fin de la ligne, à la shell
- /* ... */ commente ce qu'il y a au milieu, que ce soit plusieurs lignes ou une partie de ligne.
  → Attention à ne pas les imbriquer ! Cela ne fonctionne pas.

# Les opérateurs

Les opérateurs arithmétiques et booléens sont les mêmes qu'en C (et que dans la plupart des langages):

| Opérateurs | arithmétiques |
|---|---|
| $i + $j | addition |
| $i - $j | soustraction |
| $i * $j | multiplication |
| $i / $j | division entière |
| $i % $j | reste de la division |
| $i++ | incrément |
| $i- - | décrément |

| Opérateurs | booléens |
|---|---|
| $a \|\| $b | OU logique |
| $a OR $b | OU logique |
| $a && $b | ET logique |
| $a AND $b | ET logique |
| $a XOR $b | OU exclusif |
| $a \| $b | OU bit à bit |
| $a & $b | ET bit à bit |
| $a ^ $b | XOR bit à bit |

# Les opérateurs

Les opérateurs arithmétiques et booléens sont les mêmes qu'en C (et que dans la plupart des langages):

| Opérateurs | relationnels |
|---|---|
| $a == $b | Égalité |
| $a <,<=$b | Infériorité (stricte) |
| $a >,>= $b | Supériorité (stricte) |
| $a != $b | Différence |
| $a <=> $b | Inférieur (-1), égal (0) et supérieur (1) |
| $a === $b | Égalité valeur et type |
| $a !== $b | Différence valeur ou type |

# Les opérateurs

Les opérateurs arithmétiques et booléens sont les mêmes qu'en C (et que dans la plupart des langages):

| Opérateurs | relationnels |
|---|---|
| $a == $b | Égalité |
| $a <,<=$b | Infériorité (stricte) |
| $a >,>= $b | Supériorité (stricte) |
| $a != $b | Différence |
| $a <=> $b | Inférieur (-1), égal (0) et supérieur (1) |
| $a === $b | Égalité valeur et type |
| $a !== $b | Différence valeur ou type |

La concaténation de chaînes de caractères se fait avec l'opérateur "." :
ex : echo "Type de \$*var*".gettype($var);
$var .=" j'ajoute à $var";

# Les conditions

## If-Then-Else

```
if(is_string($var))
{ echo $var; }
else{ echo "$var mal défini";}
```

# Les conditions

## If-Then-Else

```
if(is_string($var))
{ echo $var; }
else{ echo "$var mal défini";}
```

## Switch-Case

switch($n){
case 0 : echo ”aucun”;
case 1 :
case 2 : echo ”peu (1 ou 2)”;
default : echo ”beaucoup”;
}

# Les boucles

Les boucles `for` et `while` se construisent comme en C.

## Boucle while

```
while(expression){
...
}
```

## Boucles for

```
for( $i=1;$i<=10;$i++){
echo $i;
}
```

# Les boucles

Les boucles `for` et `while` se construisent comme en C.

## Boucle while

```
while(expression){
...
}
```

## Boucles for

```
for( $i=1;$i<=10;$i++){
echo $i;
}
```

## Boucle sur les tableaux

```
foreach($tab as $c=>$v){
}
```

Parcours le tableau $tab avec ($c,$v) prenant tour à tour les valeurs de chaque couple (clé,valeur) du tableau.

# Syntaxe alternative

Alternativement, les `if, switch, while, for` et `foreach` peuvent se passer des accolades { et } en utilisant les deux points (:) et `endif, endswitch,...`.

## Exemple

```
if(is_double($var)):
echo "<p> Nous avons un rééel ! </p>";
endif;
```

# Syntaxe alternative

Alternativement, les `if, switch, while, for` et `foreach` peuvent se passer des accolades { et } en utilisant les deux points (:) et `endif, endswitch,...`.

## Exemple

```
if(is_double($var)):
echo "<p> Nous avons un réél ! </p>";
endif;
```

## Re-exemple

```
<?php if(is_double($var)): ?>
<p>Nous avons un réél ! </p>
<?php endif; ?>
```

→ Clarté : on évite de refermer des accolades à travers de multiples bouts de codes PHP.

# Les fonctions

Les fonctions se déclarent avec le mot clé `function`
Le passage des arguments est un passage par valeur, càd que la fonction ne va pas modifier les variables passées en argument.

## Exemple

Une fonction simple faisant l'addition, sans modifier les variables :

```
function add($i,$j) {
$i+=$j;
return $i;
}
```

# Les fonctions

Les fonctions se déclarent avec le mot clé `function`
Comme en C, on peut passer des arguments par référence, en utilisant le symbole & :

## Exemple bis

```
function swap(&$x,&$y) {
$t=$x;
$x=$y;
$y=$t;
}
```

# Les fonctions

Les fonctions se déclarent avec le mot clé `function`
Le mot-clé `static` permet de garder la valeur d'une variable entre les différents appels d'une fonction.

## Exemple ter

```
function compteur() {
static $c=0; //$c vaut 0 au premier appel
$c++;        //$c vaut 1, et vaudra toujours 1 au second appel
return $c;
```

# Banque de fonctions

Il est courant de vouloir réutiliser les mêmes fonctions pour tous les fichiers PHP de notre site.
Il est utile, et même de bonne pratique, de regrouper les fonctions dans un fichier à part à la façon d'une bibliothèque.
Pour cela, on utilisera, en début de fichier :

## Include

```
<?php
include_once("fonctions.php");
?>
```

# Banque de fonctions

Il est courant de vouloir réutiliser les mêmes fonctions pour tous les fichiers PHP de notre site.
Il est utile, et même de bonne pratique, de regrouper les fonctions dans un fichier à part à la façon d'une bibliothèque.
Pour cela, on utilisera, en début de fichier :

## Include

```
<?php
include_once("fonctions.php");
?>
```

Cela peut également servir à inclure du code HTML :

## Include bis

```
<?php
include_once("header.php");
?>
```