# Programmation Web
# PHP5

Julien David

A101 - david@lipn.univ-paris13.fr

2016

# Plan

- Principes
- Syntaxe
- Base de données
- Sécurité

# Présentation Générale

## Langage PHP :

- Langage de script, interprété par le serveur,
- c'est aussi un langage objet.
- Produit du code HTML.
- Aucune mémoire entre deux "exécutions",
- Syntaxe proche du C/C++,

# Écrire du code PHP

## Code PHP

- Compris entre les balises `<?php` et `?>` à l'intérieur du HTML
- Les instructions terminent par **;**
- Commentaires :
  - // Commentaire uniligne
  - /* Commentaire multiligne */

# Produire du code HTML avec PHP

## L'instruction *echo*

- permet de décrire une chaîne de caractère contenant du code HTML.
- ce code sera inséré dans la page HTML envoyée au client.

### Fichier stocké sur le serveur

```
1  <!doctype html>
2  <html>
3    <head>
4      <meta charset="utf-8" />
5    </head>
6
7    <body>
8      <p>Je suis le paragraphe 1</p>
9      <?php echo "<p>Je suis le paragraphe 2</p>"; ?>
10     <p>Je suis le paragraphe 3</p>
11   </body>
12
13 </html>
```

### Fichier reçu par le client

```
1  <!doctype html>
2  <html>
3    <head>
4      <meta charset="utf-8" />
5    </head>
6
7    <body>
8      <p>Je suis le paragraphe 1</p>
9      <p>Je suis le paragraphe 2</p>
10     <p>Je suis le paragraphe 3</p>
11   </body>
12
13 </html>
```

# Inclure du code PHP dans du code HTML

## Deux méthodes :

1. écrire le code PHP directement dans le fichier HTML.
2. écrire le code PHP dans un fichier à part et inclure le code à l'aide de la fonction *require*.

**Dans les deux cas, le fichier doit avoir pour extention *php***

## Attention : commande *require*

- Cette commande ressemble au principe du *include* en *C/C++*.
- La différence est que le code chargé avec *require* est aussitôt executé.

# Variables

## Les variables

- Utilisation : on ajoute *$* devant le nom de la variable.
- le type d'une variable :
  - est implicite (pas besoin de déclarer une variable).
  - dépend de la valeur qu'on lui affecte (peut varier dans le temps).

# Variables

## Les types de variables

- Nombres Entiers,
- Nombres Flottants,
- Booléens,
- Chaînes de caractères.

# Chaînes de caractères

## La concaténation

Il est possible de concaténer des chaînes de caractères entre elles, ou avec des variables.
Opérateur : un point .

# Tableaux

## Les tableaux PHP sont des **map**

- un tableau associe une clé (entier ou chaîne de caractère) . . .
- . . . à une valeur de n'importe quel type.

# Structures de contrôle

## if, else if, else

Le principe est le même qu'en *C/C++*.

## Les quelques différences

- Connecteurs logiques : *and*, *or*
- Opérateur supplémentaire : *===* teste si la valeur et le type sont égaux.

# Boucles

## Boucles for et while

Le principe et la syntaxe sont les même qu'en *C*.

## La boucle *foreach*

Permet de parcourir tous les éléments d'un tableau.

# Fonctions

## Syntaxe

*function nomFonction(paramètres){ }*

# Classes

## Syntaxe

```
class nomClasse{
  private $variable1;
  public function getVariable1(){
    return $this->variable1;
  }
}
```

# Communication Client/Serveur

## Envoi d'informations au serveur

- Se fait via le protocole *http*.
- Le navigateur transmet le nom d'un paramètre et sa valeur.

# Communication Client/Serveur

## Méthode GET

Les paramètres sont envoyés via l'url.

## Syntaxe

- Entre l'url et le premier paramètre : ?
- Passage d'un paramètre : $nom = valeur$
- Si la valeur est une chaîne contenant des espaces : $espace \rightarrow +$
- Entre deux paramètres : &

# Communication Client/Serveur

## Méthode GET

Les paramètres sont envoyés via l'url.

## Côté serveur

Les paramètres sont stockés dans le tableau $_GET.

# Communication Client/Serveur

## Méthode GET

Les paramètres sont envoyés via l'url.

## Avantages

- Permet à l'utilisateur de modifier "facilement" sa requête en passant par l'url.

## Inconvénients

- URL moche et compliquée.
- Très mauvais du point de vue sécurité :
  - Les données apparaissent en clair, mots de passes compris.
  - permet de connaître les paramètres utilisés dans le code PHP.

# Communication Client/Serveur

## Méthode POST

Les données n'apparaissent pas dans l'URL.

## Côté serveur

Les paramètres sont stockés dans le tableau $_POST.

## Avantages

- l'url est simplifiée,
- les paramètres sont moins facilement accessibles.

# Communication Client/Serveur

## Sécurité

1. Il faut toujours tester si un paramètre est initialisé avant de l'utiliser
2. Le texte entré par l'utilisateur peut être du code. Il est alors interprété.

## Solutions

1. On utilise la commande *isset*
2. On traduit les caractères spéciaux du *html* en caractères simples.
   On utilise la commande *htmlentities*

# Variables SuperGlobales

## Stockage d'informations

Certaines informations ont besoin d'être conservée d'une exécution à une autre d'un code PHP.

## Stockage côté serveur

On utilise le tableau $_SESSION.

## Stockage côté client

On utilise le tableau $_COOKIE.

# Le tableau $_SESSION

## Principe

- variable globale,
- propre à chaque client,
- durée de vie : une session.

## Fonctionnement

- le script php commence par : *session_start()*
- pour vider le tableau (ex : en cas de déconnexion) : *session_destroy()*

# Le tableau $_SESSION

## Principe

- variable globale,
- propre à chaque client,
- durée de vie : une session (dépend de la configuration du serveur).

## Example d'utilisation

Savoir si l'utilisateur est connecté, connaître son identité, sont des informations qui **doivent** être stockée du côté serveur.

# Le tableau $_COOKIES

## Principe

- fichier stocké sur l'ordinateur du client,
- durée de vie : illimité.

## Fonctionnement

- pour créer un cookie, on utilise la commande : *setcookie*.

# Le tableau $_COOKIES

## Principe

- fichier stocké sur l'ordinateur du client,
- durée de vie : illimité.

## Avantage serveur

- permet de stocker de manière durable des données sur ses utilisateurs.
- ne prend pas d'espace mémoire sur le serveur.

## Inconvénient client

- utilise **votre** espace disque pour **vous ficher** !

## Inconvénient serveur

- certains clients interdisent les cookies.
- les cookies étant stockés en clair sur l'ordinateur, les données sont facilement modifiables et donc ne sont pas fiables.

# Base de données

## PHP et les bases de données

- il existe actuellement deux méthodes quasi-équivalentes pour accéder aux bases de données.
- dans ce cours, nous n'en verrons qu'une seule : la classe *PDO*.

# La classe PDO

## **P**HP **D**ata **O**bject

- les méthodes fournies sont compatibles avec différents SGBD.
- les requêtes peuvent être optimisées automatiquement.
- renvoie des exceptions en cas d'erreurs.

# La classe PDO : les méthodes

## Le constructeur

Permet d'établir la connexion avec la base de donnée. Contient 3 paramètres :

1. le Data Source Name (DSN) contenant :
   1. le SGBD
   2. le nom/l'adresse du serveur
   3. le nom de la base de donnée.
2. login
3. mot de passe

# La classe PDO : les méthodes

## La méthode *prepare*

- *Entrée* : une chaîne de caractère contenant une requête *SQL*.
- *Sortie* : un objet *PDO_STATEMENT*, une requête préparée.
- *Principe* : le SGDB analyse et optimise la requête.

## La méthode *exec*

- *Entrée* : une chaîne de caractère contenant une requête *SQL*.
- *Sortie* : le nombre de lignes affectées.
- *Principe* : utilisé surtout pour les INSERT et les UPDATE.

# La classe PDO_STATEMENT : les méthodes

## La méthode *execute*

- Méthode de la classe *PDO_STATEMENT*,
- *Entrée/Sortie* : rien
- *Principe* : le SGDB calcule et stocke le résultat de la requête.

# La classe PDO_STATEMENT : les méthodes

## La méthode *fetch*

- Méthode de la classe *PDO_STATEMENT*,
- *Entrée* : on précise sous quelle forme on désire obtenir le résultat
- *Sortie* : ligne suivante du résultat de la requête.
- *Principe* : permet de ne récupérer les résultats qu'un par un, de manière à ne pas surcharger le réseau, ou à avoir à stocker l'ensemble des résultats.

# La classe PDO_STATEMENT : les méthodes

## La méthode *fetch* : paramètres possibles

Obtenir la valeur de retour sous forme de tableau :

- PDO : :FETCH_ASSOC les clés sont des chaînes de caractères,
- PDO : :FETCH_NUM les clés sont des entiers,
- PDO : :FETCH_BOTH les deux types de clés fonctionnent (plus volumineux).

Obtenir la valeur de retour sous forme de classe :

- PDO : :FETCH_OBJ