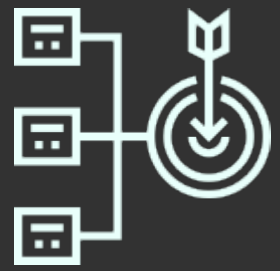
PHP
PREMIÈRE PARTIE

OBJECTIFS DU COURS

Première partie du cours sur PHP :

- Voir les bases du langage : syntaxe, points particuliers
- Apprendre à récupérer les données issues d'un formulaire
- Apprendre à utiliser une session php
- Ecriture de fonctions



La présentation part du principe que vous savez programmer dans un langage à syntaxe C (comme Java par exemple).

INTRODUCTION

PHP ? C'EST QUOI ?

- A l'origine, un langage de « script » : « petits » programmes
- Exécution « côté serveur » : pour nous, c'est Apache qui s'en charge
- Permet, par exemple, de :
  - traiter les données des formulaires
  - communiquer avec une base de données
  - créer des sites dynamiques, c'est-à-dire où les pages sont « calculées » par le serveur avant envoi au client

PETIT HISTORIQUE

Source principale : https://www.php.net/manual/fr/history.php.php

- 1994 : créé par Rasmus Lerdorf pour son usage personnel. But : communiquer avec des bases de données, création de pages web dynamiques (ex : livres d'or)
- 1997 : repris, réécrit par Zeev Suraski et Andi Gutmans (Zend) :
- 1998 : PHP 3.0
- 2000 : PHP 4.0, Zend Engine
- 2005 : PHP 5.0, Zend Engine 2.0
- 2015 : PHP 7.0
- PHP: Hypertext Preprocessor

ARCHITECTURE

- On dispose d'un serveur (service) web, comme Apache
- On lui ajoute un module pour l'exécution de PHP

- Lorsqu'une ressource est demandée au serveur, en http :
  - Si c'est un fichier avec une extension .php, il est traité (« interprété ») par le module PHP.
  - Si ce script effectue des sorties (« affichages »), elles sont envoyées au client.

ARCHITECTURE

Important : il faut donc accéder au fichier en http, pour que le serveur(service) web le traite. Pas d'accès direct en « double clic » via un gestionnaire de fichiers.

Autrement dit, concrètement :

L'url dans la barre d'adresse de votre navigateur doit commencer par un nom de serveur, une adresse ip, ou localhost.

PHP : BASES DU LANGAGE

OU ÉCRIRE LE CODE ?

Dans des fichiers textes avec extension php, avec des balises `<?php ... ... ?>` :

<?php

... mon code...

?>

Exemple :

```
<?php
        echo "Hello 3iL World";
 ?>
```

echo : une instruction pour « afficher », c'est-à-dire pour nous placer dans le flux envoyé au client

OU ÉCRIRE LE CODE ?

- Fichier avec des éléments HTML et des blocs de php

```
<body>
      <h1> <?php "My very joli grand titre" ?> </h1>
</body>
```

OÙ ÉCRIRE LE CODE ?

- Avec du HTML : on peut « intriquer »

Pratique…

… mais lisibilité délicate

Rq: extension .php, toujours.

01_exemple.php

```
<body>
 <?php
    $estContent = true;
    if ($estContent)
    {
 ?>
    <p> Chuis content </p>
 <?php
    }
    else
    {
 ?>
    <p> Chuis pas content </p>
 <?php
    }
 ?>
</body>
```

OÙ ÉCRIRE LE CODE ?

- Dans un script PHP pur
- Une bonne pratique : on ne ferme pas la balise

```php
<?php

    $texte = "Yop yop";

    echo $texte;
```

Pas de « main » ou autre fonction servant de point d'entrée, exécution dans l'ordre des instructions présentes dans le script.

SYNTAXE GÉNÉRALE : COMME JAVA, C, ETC.

- On retrouve, par exemple :
  - Le point-virgule à la fin de chaque instruction
  - Les boucles classiques : `for`, `do...while`, et `while`
  - Les conditionnelles : `if... else...`
  - Les commentaires avec `/* ... */` et `//...`
  - La sensibilité à la casse (majuscules et minuscules ne sont pas interchangeable… la plupart du temps)

INCLUSIONS : INCLUDE ET REQUIRE

- Inclure un fichier dans un autre :
  - Pour « factoriser », ne pas réécrire du code
  - Utiliser des bibliothèques de fonctions, des classes
- Peut « même » servir avec des fichiers ne contenant pas de php, par exemple inclusion sur toutes les pages d'un site d'une barre de navigation (pur HTML).
- Deux mots clés :
  - `include` : si le fichier à inclure n'est pas trouvé, l'exécution se poursuit sans notification
  - `require` : déclenche une erreur si le fichier n'est pas trouvé
- Variante :
  - `include_once` et `require_once` : assure qu'une inclusion n'est pas faite 2 fois.

INCLUSIONS : INCLUDE ET REQUIRE

page01.php (extrait)

```
<!doctype HTML>
<html>
  <head>
        .........
  </head>

  <body class='container'>
    <header class='hautDePage'>
      <strong>Bandeau du site, à venir</strong>
    </header>

    <?php include "inc_menu.php" ?>

    <main class='contenuPrincipal'>
              .......
    </main>

        ...
  </body>
</html>
```

Le fichier inlcus est copié ici.

inc_menu.php (extraits)

```
<!-- menu : navbar, avec booststrap, tiré presque brut de la doc -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
 <a class="navbar-brand" href="#">*Logo*</a>
  .....
 <div class="collapse navbar-collapse" id="navbarNav">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" href="page01.php">Page 01</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="page02.php">Page 02</a>
    </li>
    ..........
</ul>
 </div>
</nav>
```

Exemples dans 02b_inclusion.zip, version améliorée dans 02b_inclusion bis.zip

VARIABLES ET TYPES

- Les noms de variables sont préfixés par $
- PHP est faiblement typé :
  - pas de déclaration explicite de variables avec un type spécifié
  - possibilité d'affecter un contenu d'un certain type dans une variable qui contenait un contenu d'un autre type
  - mais, à l'inverse, possibilité de préciser les types des paramètres des fonctions (voir plus loin)

TYPES DE DONNÉES

- Typage faible ne signifie pas « pas de typage » : comment interpréter les octets contenus dans les variables ? Texte ? Nombre à virgule flottante ?...

- 4 types simples (exhaustif) :
  - integer
  - float (double)
  - boolean
  - string

TYPES DE DONNÉES

- 4 types complexes (exhaustif) :
  - Pour nous :
    - array
    - object
  - Pas pour nous dans l'immédiat:
    - callable (ou callback)
    - iterable (PHP 7.1) : tableau ou objet qui implémente l'interface Traversable

- 2 types spéciaux :
  - null, type qui ne peut avoir qu'une seule valeur : NULL (insensible à la casse)
  - resource : référence vers une ressource externe

TYPES NUMÉRIQUES : EXEMPLE

03_types_numeriques.php

```php
<?php
    echo '<strong>Défintion d\'une var numérique : $nombre</strong><br>';
    $unNombre = 15;
    echo gettype($unNombre);
    echo "<br>";
    echo $unNombre;
    echo "<br><br>";

    echo '<strong>Après division par 2, stockage dans $nombre</strong><br>';
    $unNombre = $unNombre / 2;
    echo gettype($unNombre);
    echo "<br>";
    echo $unNombre;
    echo "<br>";
```

TYPES NUMÉRIQUES : OPÉRATEURS

- Opérateurs arithmétiques comme en Java ou C : `+, *, /, -, %`
- `$a` puissance `$b` : `$a ** $b`
- Opérateurs abrégés : `++, --, +=, -=, *=, /=`

- Opérateurs de comparaison comme en Java ou C : `<, >, >=, <=, ==, !=`
- Opérateur « spaceship » (php 7) : `<=>` (lien vers la doc)
- Opérateurs de comparaison avec vérification du type :
  - `===` : valeurs égales et même types
  - `!==` : valeurs différentes ou types différents

TYPE CHAINE DE CARACTÈRES

- Notation entre simples ou doubles quotes :
  - Équivalent pour du texte pur (pas de distinction caractère unique / chaîne de caractères)
  - Mais : dans des doubles quotes, les variables sont remplacées par leur valeur.
- Opérateur de concaténation : .
- Opérateur + : addition numérique, peut s'exécuter si les chaines opérandes peuvent être converties en chaines.

TYPE CHAINE : EXEMPLE

04_type_chaine.php

```php
<?php
    $nombre = 75;
    $texte1 = "Il y a ";
    $texte2 = " pourcents de dormeurs dans la salle.";

    // concaténation : opérateur point,  nombre converti automatiquement
    $phrase = $texte1.$nombre.$texte2;

    echo $phrase;
    echo "<br><br>";
    echo "Il y a $nombre pourcents de dormeurs ici (doubles quotes). <br>";
    echo 'Il y a $nombre pourcents de dormeurs ici (simples quotes). <br>';

    $texte3 = "50";
    $texte4 = "75.5";
    echo $texte3 + $texte4;
    echo "<br>";
    echo gettype($texte3 + $texte4);
```

TYPE BOOLÉEN ET OPÉRATEURS LOGIQUES

- Pour le type lui-même : similaire à ce que vous connaissez.
- Opérateurs logiques :
  - `and, or, xor`
  - `&&, ||, !`
- Deux syntaxes, niveaux de priorités différents : and, or, xor sont moins prioritaires, cf la doc officielle
  https://www.php.net/manual/fr/language.operators.precedence.php

TYPE ARRAY

- Tableau : « tableau associatif », ensemble de paires clé/valeur.
- Mot clé `array`, ou syntaxe courte : `[]`
- Ajout en fin : `$monTab[]=15;` (ou `array_push`)

- Remarque, pour voir le contenu d'une variable : fonction `var_dump()`

TYPE ARRAY

05_type_array.php

```php
<?php
    $tableau1 = array(
                        "nom" => "Rambo",
                        "prenom" => "Arthur",
                        "profession" => "clown poète",
                        "age" => 156                    // cases hétérogènes
                    );

    $tableau2 = [
                    "nom" => "Raimbaud",
                    "prenom" => "John",
                    "profession" => "clown triste",
                    "age" => 66
                ];
    // se poursuit page suivante
```

TYPE ARRAY

05_type_array.php

```php
var_dump($tableau1);
var_dump($tableau2);

// tableau vide :
$tableau3 = [];

// ajout en fin de tableau, avec un indice automatique en guise de clé
$tableau3[] = "yop";
$tableau3[] = 15;
$tableau3[] = $tableau1; // un tableau dans un tableau
$tableau3["triste"] = $tableau2;
$tableau3[] = "coincoin";

var_dump($tableau3);
```

TYPE ARRAY : PARCOURS

- Parcours avec un foreach (« Pour chaque »), 2 versions :

05_type_array_suite.php

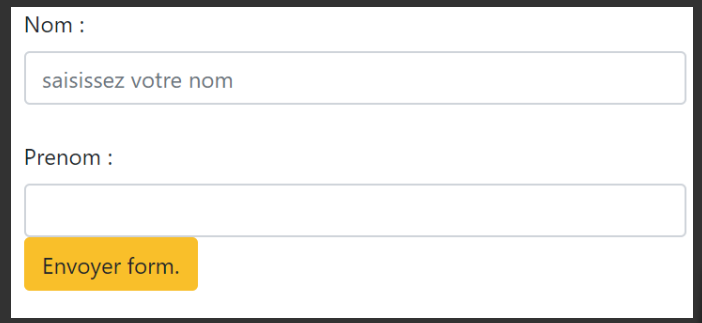
```php
// extrait
echo "<h1>Parcours sans récupération de la clé </h1>";
foreach ($tableau3 as $uneCase)
{
    echo "<strong>une case... </strong>";
    var_dump($uneCase);
}

echo "<h1>Parcours avec récupération de la clé </h1>";
foreach ($tableau3 as $uneCle => $uneValeur)
{
    echo "<strong>Case $uneCle</strong>";
    var_dump($uneValeur);
}
```

3 TABLEAUX ASSOCIATIFS PARTICULIERS : $_GET, $_POST ET $_SESSION

RÉCEPTION DE DONNÉES DE FORMULAIRES

- Deux tableaux associatifs particuliers, gérés par le système : `$_GET` et `$_POST`



Création du formulaire : HTML

extrait du code

```
<form action="traiterForm.php" method="get">
    <input id="inputNom" name="nom" >
    <input id="inputPrenom" name="prenom" >
    <button type="submit">Envoyer form.</button>
</form>
```

Traitement du formulaire : PHP

```
<?php
    // fichier : traiterForm.php
    echo "champ 'nom' reçu : ";
    echo $_GET["nom"];
    echo "champ 'nom' reçu : ";
    echo $_GET["prenom"];
```

Code complet : 07_formulaire.zip

RÉCEPTION DE DONNÉES DE FORMULAIRES

- Envoi en GET :
  - Visible dans l'URL
  - Limitation due à la longueur maximale de l'URL : 2000 caractères
- Envoi en POST :
  - Pas visible dans l'URL
  - … mais visible dans la requête HTTP (outils de dev du navigateur)
  - Pas de limitation de longueur

- Pas nécessairement de sémantique associée à GET (« obtenir » : lecture) et POST (« envoyer » : écriture).

SESSIONS

Principe, utilité

- Durée de vie d'une variable : un fichier (un script)
- Comment :
  - transmettre une donnée d'un fichier(script) à l'autre ?
  - disposer de variables communes à tous les fichiers ?
- Une solution : les sessions
- Session : un espace de stockage associé à une « session de navigation »

SESSIONS

- Espace de stockage « réel » : un emplacement dans le système de fichiers du serveur
- Accès par programmation à cet espace : un tableau associatif appelé `$_SESSION`
- Fonctionnement, principe :
  1. on demande le démarrage du mécanisme de session en tout début de script
  2. on accède librement ensuite au tableau, en lecture et écriture

Le point 1. est important : les informations de session figurent dans les entêtes HTTP, il ne faut pas que les entêtes soient générées avant que la session ait été démarrée.

SESSIONS

- Exemple minimaliste : 2 pages, une écriture et une lecture dans la session.

```php
<?php
    // fichier page1.php
    session_start();

    $_SESSION["login"] = "Kirikou";

    echo "page 1";
```

08_sessions_page1.php

```php
<?php
    // fichier page2.php
    session_start();

    echo "login : ".$_SESSION["login"];

    var_dump($_SESSION);

    echo "id de session : ".session_id();
```

08_sessions_page2.php

SESSIONS : PRINCIPE POUR L'IDENTIFICATION (LOGIN)

- Etapes de l'authentification :
  - L'utilisateur saisit un login/pass dans un formulaire
  - Les données du formulaire sont envoyées au serveur
  - Le serveur cherche dans la base de données le couple login/pass, ainsi que le rôle associé par exemple
  - Si le couple n'est pas trouvé : erreur d'identification
  - Si le couple est trouvé : stockage en session du login et du rôle
- Utilisation sur les pages à accès restreint :
  - Démarrage de la session
  - Récupération du rôle dans $_SESSION
  - Refus de l'accès (redirection, die…) si le rôle ne donne pas droit à la page

VARIABLES SUPER GLOBALES

- Variables super globales : disponibles pour tous les scripts
- Entre autres :
  - $_GET
  - $_POST
  - $_SESSION
  - $_SERVER

FONCTION UTILISATEUR

- Déclaration avec le mot clé function
- Assez similaire à une fonction Java ou C

- Le nom des fonctions n'est pas sensible à la casse… mais il est de bonne pratique d'invoquer une fonction en respectant la casse de sa définition.
- On peut définir une fonction dans une fonction.

FONCTIONS : ARGUMENTS

- On peut indiquer des valeurs par défaut pour les arguments. Si des arguments ont des valeurs par défaut ils doivent être placés à la fin de la liste des arguments.
- On peut indiquer le type que doit avoir un argument (type hinting)
- On peut indiquer qu'une fonction admet un nombre quelconque d'arguments, en utilisant le « mot-clé » ...

FONCTIONS : EXEMPLE

06_fonctions_A.php

```php
function maFonction($texte, $occurences=NULL)
    {
        if ($occurences!=NULL)
        {
            for ($i=0; $i<$occurences; $i++)
            {
                echo $texte."<br>";
            }
        }
        else
        {
            echo "...<br>";
        }
    }
```

PORTÉE DES VARIABLES : POINT PARTICULIER

- Une variable « globale » existe dans l'ensemble du fichier où elle est définie.
- Attention : pour utiliser une variable globale dans une fonction, il faut spécifier que l'on veut y faire référence.

Extrait de : 06_fonctions_B.php

```php
$nombre1 = 20;

function maFonction2()
{
    global $nombre1; // pour accéder à la var globale
    echo "20 x 10 = ";
    echo $nombre1 * 10;
    echo "<br><br>";
}
```