INITIATION AU LANGAGE PHP

Actualisé à la version PHP 7.4 qui sera maintenue jusqu'en Novembre 2022

Depuis le 20 Novembre 2020 : PHP 8

https://www.php.net/docs.php

- PHP (**PHP HyperText Preprocessor** (Acronyme récursif)) est un langage récent spécifiquement orienté **programmation Web (1994 -> …)**

- Il s'agit d'un langage de script **interprété**

- Les scripts **PHP** sont exécutés du **coté du serveur** Web

- Le but de PHP est essentiellement d'effectuer des traitements côté serveur Web et de générer/personnaliser le code HTML à retourner au client

- Le script n'est jamais retourné tel quel au client ... c'est le résultat de l'exécution qui est renvoyé

- ASP, JSP et PHP ont une finalité identique :
  générer du code HTML à la volée pour obtenir un contenu de page Web "**dynamique**"

Le Langage PHP

- Simplicité d'écriture des scripts (langage simple mais riche, et maintenant **orienté objet** depuis **PHP 5**)

- Possibilité de **structurer** le code **PHP** (sous-programmes, modules externes, classes, écriture modulaire …)
- Simplicité d'interfaçage avec des bases de données (de nombreux **SGBD** sont supportés, mais le plus utilisé avec ce langage est **MySQL**).

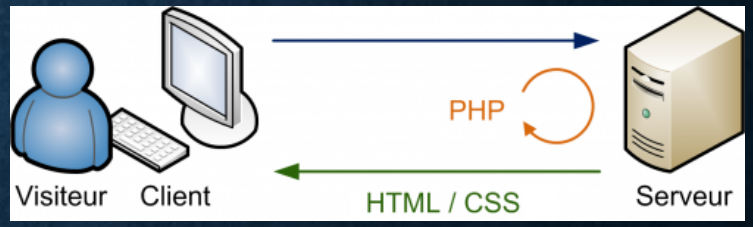
- Intégration du **moteur PHP** au sein de nombreux logiciels serveur Web (**Apache**, Microsoft IIS, ...)
- Gratuité et disponibilité du code source
- Très nombreuses extensions, bibliothèques et ‘Framework' (eux aussi Open Source) autour du langage PHP

Le Langage PHP

- Un script PHP est un **simple fichier texte** contenant des instructions, parfois entremêlées dans un code HTML/CSS/JavaScript, à l'aide de balises spéciales

- Ce fichier doit :
  - être stocké sur le serveur Web (dans le répertoire **www** avec Wamp)

  - avoir l'extension ".php", ".php3" , ".php4" … pour pouvoir être **interprété** par un **module spécifique** du serveur Web



- Lorsque le client demande une page réalisée en PHP :
  - le serveur « reconnaît » qu'il s'agit d'un fichier PHP grâce à son **extension**
  - il scanne le contenu du fichier
  - tant qu'il s'agit de code HTML, il transmet ce code dans le flot HTML résultant
  - lorsqu'il rencontre une instruction PHP, il la transmet à **l'interpréteur** PHP
  - l'interpréteur exécute l'instruction et alimente le flot HTML résultant
  - à la fin du script, le serveur transmet le code HTML/CSS/JavaScript final au client

La Forme du Langage PHP

- Utilisation de balises spécifiques : **<?php …… ?>**

- Marque de début : **<?php**
- Marque de fin **?>**

- Entre les deux marques, un **nombre quelconque de lignes** de code PHP

- Il peut y avoir **plusieurs séquences successives** <?php … ?> au sein d'une même page de script de code PHP

  - Pour du traitement pur (tests, boucles, accès BDD…)
  - Pour insérer des données variables dans le code HTML

La Forme du Langage PHP …

- Séparation des instructions **par un point virgule** à chaque fin d'instruction
  la marque de fin (?>) implique la fin d'instruction PHP

```
<?php
        echo 'Coucou et recoucou.';
?>
```

ou

```
<?php echo 'Coucou et recoucou.' ; ?>
```

- Les **commentaires :**

```
<?php
    echo 'Coucou';     // Ceci est un commentaire sur une ligne

                       /* Ceci est un commentaire sur plusieurs lignes,
                       voilà la deuxième ligne …
                       */
    echo 'Ceci est encore un test';
?>
```

- PHP différencie **MAJUSCULES** et **minuscules**
- Séparateur de bloc : **{…}**

Les Variables en PHP…

- Pas de type ni de déclaration initiale obligatoire (mais PHP 5 introduit des prémices de typage dans les classes d'objets, PHP 7 le permet sans l'imposer)

- PHP supporte les types de données suivant :
  - nombres entiers (1-2- 6 - 1000) et à virgule flottante (0.002 - 10.5)
  - chaînes de caractères
  - booléen
  - date (date et heure)
  - tableaux (array)
  - objet
- Le type d'une variable est décidé au moment de l'exécution par le PHP, en fonction du contexte dans lequel la variable est utilisée
- La manipulation des variables se fait par nom symbolique
- Le nom d'une variable commence toujours par $

```
$chaine = "Coucou";          // variable chaîne
$compteur = 1;               // variable nombre entier
```

Les Chaînes de caractères en PHP…

- Sont délimitées par des simple-quotes ou guillemets

  ‘Coucou numéro : ’ . $nombre ➔ PHP reproduit tel quel la chaîne *Coucou numéro :* et il y a **concaténation** avec la valeur de la variable $nombre (concaténation explicite). Le . en PHP est l’opérateur de **concaténation**.

  "Coucou numéro : $nombre" ➔ PHP **évalue la chaîne** puis retourne le résultat (ici, la concaténation est donc implicite)

- Le caractère backslash (\) est utilisé pour "protéger" un caractère spécial (comme en Java, JavaScript, C#) :

\n	nouvelle ligne
\t	tabulation horizontale
\\	backslash
\$	caractère $
\"	double-quotes (guillemet)
\’	simple-quotes (apostrophe)

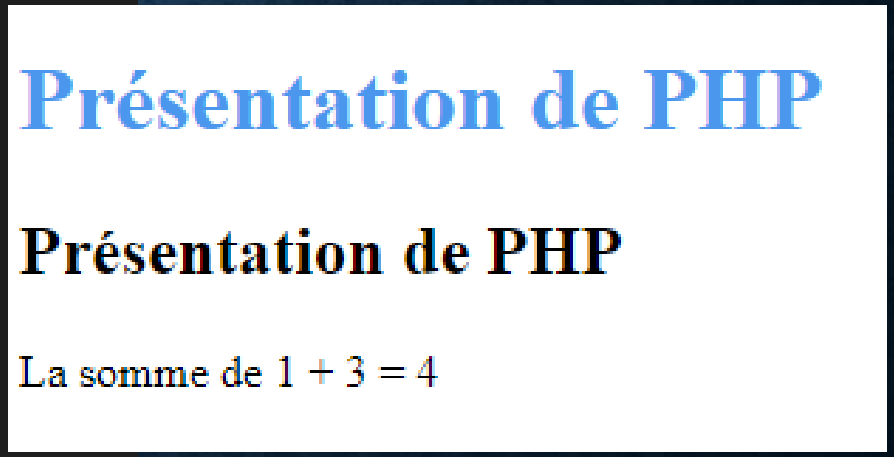
- Il existe un grand nombre de *fonctions du langage* s'appliquant aux chaînes de caractères
  strlen(), trim(), substr()…

Mise en œuvre d'un script PHP

index.php

Back > Test_PHP > Presentation_1 > index.php > html

```php
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Présentation PHP 1</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>

<h1><?php $titre = "Présentation de PHP"; echo $titre; ?></h1>
<h2><?= $titre ?></h2>

<?php

$a = 1;
$b = 3;

$somme = $a + $b;

echo "La somme de $a + $b = " . $somme;  // Calcul de la Somme de $a + $b

?>

</body>
</html>
```



Les Fonctions en PHP

Une **fonction** est un **bloc de code** qui peut être exécuté à chaque fois que nous en avons besoin.

• Création de fonctions en PHP :

– Toutes les fonctions commencent par le mot ’**function ()**’
– Le nom des fonctions doit être explicite
– Les noms de fonctions commencent par une lettre minuscule ou underscore
– () pour définir les **paramètres**
– {} pour le corps de la fonction
– **return** est utilisé pour retourner une valeur au programme appelant.

Les Fonctions en PHP …

```php
<?php

    function afficherUnTruc($prenom, $age, $ponctuation){

        if($age<18){
            echo $prenom. ", vous êtes mineur".$ponctuation."<br>";

        }else{
            echo $prenom. ", vous êtes majeur ".$ponctuation."<br>";
        }
    }

    afficherUnTruc("Sacha",52, ".");
    afficherUnTruc("Donald",16, " !!!");
?>
```

Sacha, vous êtes majeur .
Donald, vous êtes mineur !!!

```php
<?php

    function additionner($a, $b){

        $total = $a+$b;
        return $total;
    }

    echo "La somme de 8 + 10 = ".additionner(8,10);

?>
```

La somme de 8 + 10 = 18

Fonctions particulières PHP pour le Web

- PHP introduit des fonctions particulières afin :

  - de faciliter les échanges **client Web** / **Serveur Web** selon le protocole **HTTP**
  - de protéger l'application contre des saisies malveillantes dans les formulaires (Sécurité)

Fonction PHP	Rôle
htmlspecialchars()	Convertit les caractères spéciaux comme les lettres accentuées ou les signes < et > en 'entités' HTML (é >…)
htmlentities()	Même principe mais étendu à plus de cas de figure (tester au cas par cas pour choisir entre ces deux fonctions)
html_entity_decode()	Convertit à l'inverse les entités HTML en caractères ordinaires
urlencode()	Convertit certains caractères de manière à respecter la syntaxe des URL (ex : espace devient %20)

Les Tableaux de variables en PHP

- **Pas de type ni de déclaration initiale obligatoire**
- Accès par **indice initialisé à zéro**
- Création d'un tableau
  en affectant explicitement chacune des valeurs : **$tabx[0] = ‘abc’;**
  en énumérant son contenu : **$tabx= array('abc', 'def', 'Coucou');**
- Tableaux à une dimension :

```
$tabx[0] = ‘abc’;
$tabx[1] = ‘def’;
$tabx[ ] = ‘Coucou’;  // ‘Coucou’ est stocké dans $tabx[2]
$tabx[ ] = ‘Recoucou’;   // $tabx[3] == ‘Recoucou’
```

- Tableaux à plusieurs dimensions :
  Pour chaque dimension du tableau, ajouter une paire de crochets

```
$taby[1][0]     = $f;        // tableau à deux dimensions
```

- Taille d'un tableau (= nombre d'éléments) : fonction count(…)

Tableaux indicés et associatifs

- **Les indices commencent toujours à [0]**

- Les indices peuvent être des **chaînes de caractères** correspondant à des noms symboliques (clés)
  on parle de "**tableaux associatifs**« , ils sont plus explicites en utilisation :

```
$tpersonne[ "age" ] = 32;
$tpersonne[ "nom" ] = "Dupont";
```

NB : de nombreuses données système sont disponibles sous forme de tableaux associatifs (date, variables d'environnement serveur, accès aux BDD…)

La syntaxe (barbare) ***clé => valeur*** définit/obtient une clé et la valeur associée. Une clé peut être une chaîne ou un nombre :

```
$arr = array("foo" => "bar", 12 => true);
```

Tableau que l'on peut parcourir avec une boucle **foreach**

```
foreach($arr as $item=>$valeur) {…}
```

Tableaux indicés et associatifs …

```php
$coordonnees = array(
                        "prenom"=>"Pierre",
                        "nom"=>"KIROULE",
                        "adresse"=>"8 rue du Paradis",
                        "ville"=>"Brive"
                    );

foreach($coordonnees as $clef => $valeur){

    echo "[".$clef."]"."  vaut  " .$valeur."<br>";
}
```

Tableau associatif en PHP

[prenom] vaut Pierre
[nom] vaut KIROULE
[adresse] vaut 8 rue du Paradis
[ville] vaut Brive

Les variables super-globales en PHP

- **PHP met à disposition des variables dite ‘super-globales’ accessibles et utilisables à tout moment dans tout script PHP :**

  - nom en **MAJUSCULES** précédé du signe _
  - sous forme de **tableaux associatifs**
  - informations sur le serveur, le client, les données saisies en formulaire…

  $_SERVER, $_SESSION, $_GET, $_POST…

```php
<?php

echo "<pre>";
print_r($_SERVER);
echo"</pre>";

?>
```

Saisir ce code dans un script PHP et exécutez-le depuis votre navigateur :

Pour la mise au point des scripts en PHP

- Pour aider à la mise au point de vos scripts :

var_dump($xxx) : retourne le contenu et le type de la variable
$xxx peut être aussi bien une variable simple qu'un tableau ou un objet

```php
<?php

var_dump($_COOKIE);

?>
```

```
C: \ wamp64 \ www \ TP_DWWM \ Back \ Test_PHP \ Presentation_1 \ info.php: 4:
array  (size = 5)
  'cookielawinfo-checkbox-needed' =>  string  'yes'  (length = 3)
  'cookielawinfo-checkbox- non nécessaire ' =>  string  'oui'  (longueur = 3)
  'CookieLawInfoConsent' =>  string  'eyJuZWNlc3NhcnkiOnRydWUsIm5vbi1uZWNlc3NhcnkiOnRydWV9'  (longueur = 52)
  'viewed_cookie_policy' =>  string  'oui'  (longueur = 3)
  'PHPSESSID' =>  string  ' m18nml40qsosjseg4bk6h7dnfg '  (longueur = 26)
```

Bufferisation du flux généré

- Par défaut :
  echo alimente directement le flux de sortie; le flux est envoyé au fur et à mesure au client

- Pour **préparer entièrement** la (portion de) page et s'assurer que tout s'est bien déroulé avant d'envoyer un résultat au client : alimenter un **buffer** de sortie PHP (*o*utput *b*uffer) puis envoyer au final son contenu au client (ou tout abandonner…)

Exemple :

```
ob_start(); // initialise la bufferisation

echo 'coucou';
echo ' vous allez bien ?';
echo '<br>';
$content = ob_get_clean();
ob_end_flush(); // on vide le tampon et on retourne le contenu au client

echo $content;
```

PHP et HTTP la fonction header()

- Les pages Web sont véhiculées selon le "***protocole HTTP*** " entre le logiciel "**client Web**" (= browser) et le logiciel "**serveur Web**" (**Apache**, IIS…)

  Le client adresse une "***requête HTTP*** " contenant principalement l'URL de la page demandée (et paramètres éventuels après le signe ?) + des infos de service (adresse IP du demandeur, durée de vie de la page, autorisation de mise en cache par les routeurs…) ➔ message de type **entête HTTP**

  Le serveur Web retourne une "***réponse HTTP***" : page HTML dans un "**corps de message HTTP**", précédée d'un "**entête HTTP**" contenant des infos de service (dont le code erreur 200, 403, 404, 500…)

- ***header()*** permet de jouer sur le ***paramétrage de l'ENTETE du message "réponse HTTP"*** retourné au browser qui a exprimé la "**requête HTTP**"

PHP et HTTP la fonction header() …

header("location:…");

Effectue une ***redirection*** vers une autre page ;
NB: normalement, aucun flux HTML (même pas un saut de ligne !) ne doit avoir été émis au moment où l'interpréteur PHP exécute header("location:…")

header("refresh:…");

Personnalise la ***durée de vie de la page sur le navigateur*** de manière à effectuer une ***redirection ou un rafraîchissement automatique*** de la page.
Même action et même syntaxe que <meta http-equiv="refresh" content="…" /> en HTML

Autres exemples :

```
header("Content-Type: application/pdf");
header("Cache-Control: no-cache, must-revalidate");
header("Expires: Sat, 26 Jul 2021 05:00:00 GMT");
```

Structures conditionnelles if

- L'instruction **if ... else** :

if (condition**) {**
liste d'instructions constituant un bloc, à exécuter si la **condition** est remplie
} else {
autre série d'instructions, à exécuter si la **condition** n'est pas remplie
}

- L'instruction **if… elseif … else**

Test de plusieurs conditions de façon ***exclusive***, une seule des conditions sera réalisée ...

Evite d'avoir à imbriquer des instructions *if (et les {} qui vont avec)*
Plusieurs **elseif** peuvent s'imbriquer les uns dans les autres ... après un **if** initial

Structure conditionnelle Switch

- équivaut à une série d'instruction if (exclusives ou non) :

```
if ($i == 0) {echo ‘$i égal 0'; } ;
if ($i == 1) {echo ‘$i égal 1'; } ;
if ($i == 2) {echo ‘$i égal 2'; } ;
```

- qui peut s'écrire :

```php
$i=3;

switch ($i) {
    case 0: echo '$i égal 0'; break;
    case 1: echo '$i égal 1'; break;
    case 2: echo '$i égal 2'; break;

    default : echo '$i est différent de 0, 1 et 2';
}
```

- utiliser "**break**" dans chaque cas pour éviter des tests inutiles quand les cas sont exclusifs les uns des autres

- mot-clé : **default :** "case" spécial correspondant à **tous les cas non cités** précédemment

Les Boucles en PHP

- La boucle **while** : moyen le plus simple d'implémenter une boucle en PHP

```php
$i = 1;
while ($i <= 10) {
    echo $i++;  // La valeur affichée est $i avant l'incrémentation
}
```

- La boucle **for** : un autre moyen d'implémenter une boucle en PHP

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

Les Boucles en PHP …

- La boucle **foreach** : précédemment rencontrée au sujet des tableaux associatifs

```php
$tab_animaux = array("Lion","Grizzly","Ablette","Chevreuil", "Panthère");

foreach($tab_animaux as $animal){

    echo $animal."<br>";
}
```

Lion
Grizzly
Ablette
Chevreuil
Panthère

Les Gestion des Erreurs en PHP

PHP intègre le même système de gestion des erreurs que C#, Java ou JavaScript:

```
try { // bloc protégé

    if(…)  // détection d'une erreur
        {
          $error = '…';
          throw new Exception($error); // levée d'exception
        }else{…traitement normal…}
    }
catch (Exception $e) { // gestion de l'erreur
    echo 'Exception attrappée : '.  $e->getMessage(). "<br>";
}
```

Les Gestion des Erreurs en PHP …

```php
function division($num,$denom){

    try{
        if($denom === 0){
            throw new Exception("Division par 0 impossible !!!");
        }else{
            return $num/$denom;
        }
    }catch(Exception $e){

        echo "Capture de l'erreur : ".$e->getMessage()."<br>";
    }

}

echo division(4,0);
echo division(6,3);
```

Capture de l'erreur : Division par 0 impossible !!!
2

La Portée des variables en PHP

En PHP, une variable ***globale*** doit être déclarée « **global** » à l'intérieur de chaque fonction afin de pouvoir être utilisée dans cette fonction (sinon, il s'agit d'un autre variable, de portée ***locale***)

```php
$a = 1;
$b = 2;
function Sum () {
    global $a, $b;
    $b = $a + $b;
    echo $b;
} ;
Sum ();
echo $b;  //ce script va afficher "3" puis "3"
```

```php
$a = 1;
$b = 2;
function Sum () {
    $a = 10; $b = 20;
    $b = $a + $b;
    echo $b;
} ;
Sum ();
echo $b; // ce script va afficher "30" et "2"
```

```php
$a = 1;
$b = 2;
function Sum ($x,$y) {
    $y = $x + $y;
    echo $y;
} ;
Sum ($a,$b);
echo $b;   // ce script va afficher "3" et "2"
```

Modularisation du code en PHP

En Web, chaque page est indépendante de la précédente (et de la suivante) ➔ doit donc être traitée comme un tout par le serveur Web

historiquement : écriture "***spaghetti***" mêlant code HTML/CSS/JavaScript et PHP/SQL dans le même script.

Conseillé : **modularisation du code source dans différents scripts**
qui assurent **chacun une fonction spécifique** (contrôles d'accès utilisateur, accès à la base de données, calculs divers, constitution de la page HTML…)
qui doivent être **fusionnés avant interprétation** de la page
fonction PHP **include (…)** : fusionne en lieu et place sans provoquer d'erreur si le module de script est absent
fonction PHP **require(…)** : fusionne en lieu et place et provoque une erreur si le module de script est absent sur serveur ou déjà fusionné
fonctions PHP include_once() et require_once(…) : fusionne en lieu et place si ce module n'a pas déjà été fusionnée dans cette page et provoque une erreur si le module de script est absent sur serveur

Modularisation du code en PHP …

index.php | menu.php

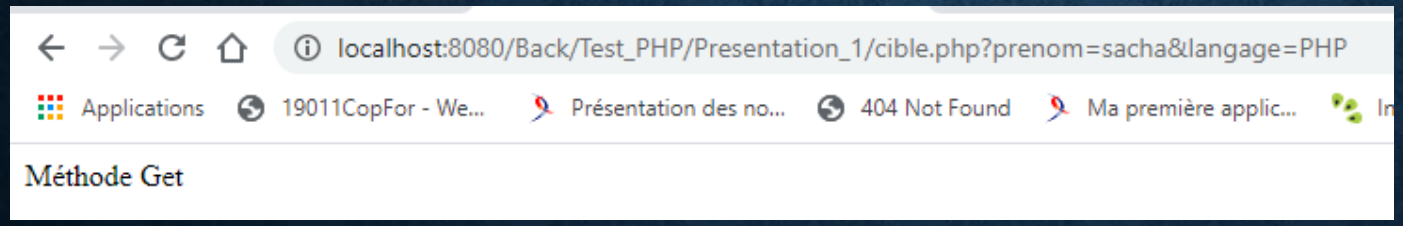
Back > Test_PHP > Presentation_1 > index.php > html > body

```php
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Présentation PHP 1</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<?php
include("menu.php");
?>

<div class="moncontenu">
    <h1>Le Langage PHP</h1>
    <p>
        Modularistaion du code, qui permet de diviser le code en
        divers unités spécifiques qui peuvent être appelées au sein
        d'un même script PHP.
    </p>
</div>
</body>
</html>
```

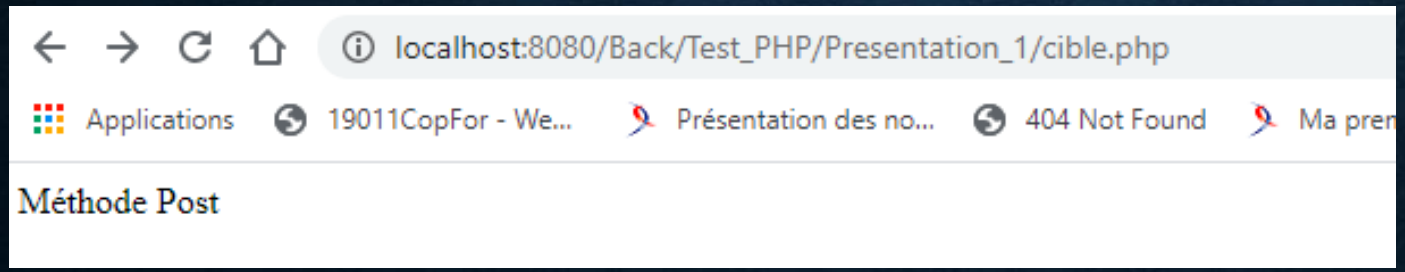
```php
<ul>
    <li>Accueil</li>
    <li>PHP</li>
    <li>JavaScript</li>
    <li>SQL</li>
</ul>
```

PHP : propagation des données de page en page

- HTTP supporte deux méthodes d'envoi de demandes de pages (requêtes HTTP) **GET** et **POST**
- Chacune exploite la structure du message HTTP de manière différente : les données sont passées en **entête** ou en **corps** du message HTTP
- Dans les deux cas l'URL de la page demandée se situe dans l'entête HTTP, c'est ce qu'on voit dans la barre d'adresse de l'explorateur
- Les données à transmettre sont :



Dans **l'entête** de message HTTP. (Jusqu'à 2000 caractères dans la barre d'adresse).
Cas du **GET**



Dans **le corps** de message HTTP.
Cas du **POST**

Passage des données depuis un Formulaire HTML

- Utilisation des fonctionnalités HTTP en **POST** tout comme en **GET** (attribut **method** des formulaires HTML)
- Le contenu d'un formulaire HTML est passé lors de la demande de page (soumission du formulaire HTML par bouton type "**submit**" ou méthode "**.submit()**" JavaScript)
- La valeur de chaque composant du formulaire HTML de saisie est disponible dans le code PHP invoqué :

  Dans 2 tableaux associatifs ("*super-globaux*") qui référencent toutes les données reçues :

  $_GET et/ou **$_POST**

  Exemple : echo "Bonjour " . $_GET["nom"] . " ! " ;

  Le tableau associatif *super-global* **$_REQUEST** regroupe quant à lui l'ensemble des données passées au serveur, en **get**, en **post** et même par **cookies**

Passage des données depuis un Formulaire HTML …

- Seuls les contrôles graphiques HTML ***nommés*** (attribut HTML '**name**') sont adressés au serveur en général, inutile de nommer les boutons de commande

- Si plusieurs contrôles graphiques HTML portent le *même nom pour des données différentes*, il est prudent de *les nommer en HTML avec des crochets* (ex : name="matricule[]" value="…" afin que PHP les récupère bien dans des structures PHP ***array***

Tester éventuellement le nombre de données reçues avec la fonction PHP **count(…)** ➔ count($_GET["matricule"])

Visiter toutes les valeurs reçues :

. avec une boucle **foreach** : foreach ($matricule as $unMatricule) {…}

Passage des données depuis un Formulaire HTML …

index.php | traitement.php

Back > Test_PHP > Presentation_1 > index.php > html > body > form

```
1   <!DOCTYPE html>
2   <html lang="fr">
3   <head>···
8   </head>
9   <body>
10  <h1>Soumission d'un Formulaire en POST</h1>
11  <br><br>
12  <form action="traitement.php" method="post">
13      <p>
14          <label for="nom">Nom :</label>
15          <input type="text" name="nom" id="nom">
16      </p>
17      <br>
18      <p>
19          <label for="prenom">Prénom :</label>
20          <input type="text" name="prenom" id="prenom">
21      </p>
22      <br>
23      <p>
24          <label for="email">Email :</label>
25          <input type="text" name="email" id="email">
26      </p>
27      <br>
28      <p>
29          <button type="reset">Annuler</button>   
30          <button type="submit">Envoyer</button>
31      </p>
32  </form>
33  </body>
34  </html>
```

Passage des données depuis un Formulaire HTML ...

index.php | traitement.php

Back > Test_PHP > Presentation_1 > traitement.php > ...

```php
<?php
// print_r est très utile pour afficher un Tableau à des fins de debuggage !!!
echo "<pre>";
print_r($_POST);
echo "</pre>";

// Boucle foreach très utile pour parcourir tous les éléments d'un tableau associatif !!!
foreach($_POST as $clef => $value ){

    echo $clef ." : ". $value."<br>";
}

echo "<br><br>";

// Pattern de Regex pour vérifier que le prénom est constitué de 3 à 30 caractères accentués ou pas.
$pattern_prenom = '/^[a-zA-ZáàâäãåçéèêëíìîïñóòôöõúùûüýÿæœÁÀÂÄÃÅÇÉÈÊËÍÌÎÏÑÓÒÔÖÕÚÙÛÜÝŸÆŒ._\s-]{3,30}$/';

// Il faut tester s'il y a une valeur dans la variable $_POST["prenom"]
if(empty($_POST["prenom"])){

    echo "Vous devez saisir un prenom";
}else{

    if(!preg_match($pattern_prenom, $_POST["prenom"])){

        echo "Veuillez entrer un prénom valide !!!";
    }else{

        echo 'Salut <strong>'.$_POST["prenom"]. '</strong>, Bienvenue dans le système !!!';
    }
}
?>
```

```
Array
(
    [nom] => RESTOUEIX
    [prenom] => Sacha
    [email] => sacha8milo@gmail.com
)
```

nom : RESTOUEIX
prenom : Sacha
email : sacha8milo@gmail.com

Salut **Sacha**, Bienvenue dans le système !!!

Les Sessions en PHP

- La gestion des **sessions** est un moyen de sauver des informations entre deux accès HTTP aux pages d'un site
- Rappel : HTTP est un protocole sans conservation d'état ("sitôt servi, sitôt oublié")
- Mémoriser et propager des informations de page en page n'est pas évident dans ces conditions …
- En standard, seuls les **cookies** et le **passage de paramètres** dans **l'URL** le permettent, les **sessions** sont là pour résoudre le problème …
- Les **sessions** permettent d'enregistrer des variables, de les préserver pour les réutiliser au fil des pages visitées par un même utilisateur manipulant un même navigateur
- Une session ‘tombe’ d’elle-même après un certain temps d’inactivité
- Attention que ce mécanisme est gourmand en ressources serveur ➔ il n’est pas activé par défaut en PHP

Les Sessions en PHP …

- *Toujours activer sur chaque page* le mécanisme de gestion des sessions :
  session_start(); // ***permet d'utiliser*** le mécanisme des sessions

- Création d'une variable de session :
  - par ajout dans le tableau associatif super-global **$_SESSION**

  $_SESSION['nom'] = htmlentities($_POST['nom']) ;

- Tester leur *existence* grâce à la fonction **isset()**
  if (isset($_SESSION["login"])) {…} else {
  header("Location: http://www.monsite.fr/login.php") }

- Fermer la session (pour détruire toutes les variables de session existantes) :
  session_destroy();

Astuces pour propager les données de page en page

- Les variables de **session** sont aussi un bon moyen de mémoriser côté serveur des données de l'utilisateur mais les mécanismes induits consomment des ressources serveur Web

- Le développeur peut aussi propager des données de page en page :

  - en les insérant dynamiquement en PHP, ***dans la balise form,*** de manière invisible pour l'utilisateur, à l'aide de tag HTML **<input type="hidden" …/>** :
  <input type="hidden" name="prenom" value="<?php echo $_POST['prenom']; ?>" />
  en les accolant à l'URL de la page suivante, à l'aide de PHP/JavaScript par exemple :

```
<script >
    <?php $prenom = "Sacha"; ?>
    let param = "?prenom=<?php echo $prenom; ?>";

    window.location.href = "pagesuite.php" + param ;
</script>
```

```
<body>

<?php
    $prenom = "Sacha";
    $param = "pagesuite.php?prenom=".$prenom;
?>

<a href=<?=$param;?>>Page Suivante</a>

</body>
```

CRÉDITS

OEUVRE COLLECTIVE DE L'AFPA

Sous le pilotage de la DIIP
et du centre sectoriel Tertiaire

EQUIPE DE CONCEPTION

M. Restoueix Sacha (Formateur)

Date de mise à jour : 28/12/2020
Date de dépôt légal : 2020
