Le **Web** avec **PHP**

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

PHP

Un formulaire

- un outil graphique
  - que nous créons avec le langage de description **HTML**
  - que nous gérons avec un langage de programmation comme **PHP**, **Java**...
- il permet à l'utilisateur de saisir des données
- et de les envoyer vers une autre page, vers une base de données...

PHP

Déclaration d'un formulaire :

```
<form method="POST ou GET" action="destination">
   ...
</form>
```

PHP

Déclaration d'un formulaire :

```
<form method="POST ou GET" action="destination">
   ...
</form>
```


Les attributs d'un formulaire

- `action` : indique le nom du fichier (**PHP**) qui reçoit les données après soumission
- `method` : concerne l'envoi de données et peut prendre deux valeurs.
  - `GET` : transmet les données dans la barre d'adresse
  - `POST` : transmet les données dans le corps de la requête **HTTP**

PHP

Considérons la page `formulaire.php` suivante

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Premier formulaire PHP</title>
</head>

<body>
    <form method="post" action="personne.php">
        <H2>Formulaire</H2>
        <div>
            <label for="nom">Nom *</label>
            <input type="text" id="nom" name="nom">
        </div>
        <div>
            <label for="prenom">Prénom *</label>
            <input type="text" id="prenom" name="prenom">
        </div>
        <button>Enoyer</button>
    </form>
</body>

</html>
```

PHP

Dans `personne.php`, on utilise la variable super-globale `$_POST` pour récupérer les valeurs transmises par le formulaire

```
<!DOCTYPE html>
<html lang="fr">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale
      =1.0">
    <title>Page personne</title>
</head>

<body>
    <p>
        <?php
        echo "Bonjour ", $_POST['prenom'], " ", $_POST['nom'];
        ?>
    </p>
</body>

</html>
```

PHP

Les variables super-globales

- `$_POST` : tableau associatif contenant les valeurs transmises via le protocole **HTTP** et la méthode **POST**
- `$_GET` : tableau associatif contenant les valeurs transmises via la barre d'adresse (= le protocole **HTTP** et la méthode **GET**)
- `$_COOKIE` : tableau associatif contenant les cookies **HTTP**
- `$_REQUEST` = `$_COOKIE` + `$_GET` + `$_POST`
- `$_SESSION` : tableau associatif contenant les valeurs stockées dans une session (coté serveur)
- `$_SERVER` : tableau associatif contenant des informations sur le serveur (protocole, numéro de port...)
- `$_FILES` : tableau associatif contenant des informations sur les fichiers soumis (via un formulaire avec le protocole **HTTP** et la méthode **POST**)
- `$_ENV` : tableau associatif contenant des informations sur les variables d'environnement

PHP

Pour explorer les différentes variables/valeurs de `$_SERVER`

ajoutez l'instruction `var_dump($_SERVER);` dans votre fichier **PHP**.

PHP

Pour explorer les différentes variables/valeurs de `$_SERVER`

ajoutez l'instruction `var_dump($_SERVER);` dans votre fichier **PHP**.

Quelques variables définies dans `$_SERVER`

- `$_SERVER["HTTP_REFERER"]` : contient l'adresse de la page (si elle existe) qui a conduit le client à la page courante
- `$_SERVER["REQUEST_METHOD"]` : contient le nom de la méthode (`GET`, `POST`...) de requête utilisée pour accéder à la page.
- ...

PHP

Pour explorer les différentes valeurs de `$_ENV`

- Si vous utilisez **WAMP**
  - Allez dans `php.ini`
  - Cherchez `variables_order = "GPCS"`
  - Remplacez `"GPCS"` (pour GET, POST, COOKIE and SERVER) par `"EGPCS"` (pour GET, POST, COOKIE, ENV and SERVER)
  - Sauvegardez puis redémarrez tous les services
- ajoutez l'instruction `var_dump($_ENV);` dans votre fichier **PHP**.

PHP

Dans le cas où certaines variables d'environnement sont masquées ou non chargées automatiquement, on peut utiliser la fonction `getenv()` pour les récupérer explicitement

```php
var_dump(getenv());
```

PHP

Considérons le lien avec les paramètres `nom` et `prenom`

```
<a href="personne.php?nom=wick&prenom=john">
  découvrez le personnage
</a>
```

PHP

Considérons le lien avec les paramètres `nom` et `prenom`

```
<a href="personne.php?nom=wick&prenom=john">
  découvrez le personnage
</a>
```


Question

Comment récupérer les valeurs de ces paramètres dans `personne.php` ?

PHP

Considérons le lien avec les paramètres `nom` et `prenom`

```
<a href="personne.php?nom=wick&prenom=john">
  découvrez le personnage
</a>
```


Question

Comment récupérer les valeurs de ces paramètres dans `personne.php` ?

Réponse

Utilisez la variable super-globale `$_GET`

PHP

Dans `personne.php`, on utilise la variable super-globale `$_GET` pour récupérer les paramètres de requête

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Page personne</title>
  </head>
  <body>
    <p>
      <?php
        echo "Bonjour ", $_GET['prenom'], " ", $_GET['nom'];
      ?>
    </p>
  </body>
</html>
```

PHP

Objectif

- Savoir stocker et récupérer des données quel que soit l'emplacement.
- Sécuriser l'accès à notre application **Web**.

PHP

Les cookies

- Un cookie est un fichier que l'on enregistre sur l'ordinateur du visiteur pour une longue durée.
- Il a une durée de vie réglable (par défaut détruit lors de la fermeture du navigateur).
- Il permet généralement de sauvegarder des données sur le visiteur qui nous permet de l'identifier lors de sa prochaine visite.
- Les informations sont stockés sous la forme : clé = valeur.
- Les types acceptés sont uniquement des chaînes de caractères.
- Disponibles, en **PHP**, via la variable super-globale `$_COOKIE`.

PHP

Pour stocker un cookie, on utilise la fonction `setcookie` (instruction à placer au tout début de la page, avant même la balise `<!DOCTYPE html>`)

```
setcookie("nom", "wick");
```

PHP

Pour stocker un cookie, on utilise la fonction `setcookie` (instruction à placer au tout début de la page, avant même la balise `<!DOCTYPE html>`)

```
setcookie("nom", "wick");
```

Pour stocker un cookie pour une durée d'un jour

```
setcookie("nom", "wick", time() + (86400 * 30));
```

PHP

Pour récupérer un cookie

```
echo $_COOKIE["nom"];
/* affiche wick */
```

PHP

Pour récupérer un cookie

```
echo $_COOKIE["nom"];
/* affiche wick */
```

Les valeurs récupérées d'un cookie sont toujours de type chaîne de caractères

```
setcookie("nombre", 10);
echo $_COOKIE["nombre"];
/* affiche 10 */

echo gettype($_COOKIE["nombre"]);
/* affiche string */
```

PHP

La fonction `setcookie` peut accepter plusieurs paramètres

1. `name`
2. `value`
3. `espires` : indique le timestamp d'expiration
4. `path` : indique le chemin relatif pour accéder à ce cookie
5. `domain` : indique le·s domaine·s (`www.mon-site.fr`, `w2.www.mon-site.fr`,... `mon-site.fr` si on veut tout) pour le·squel·s le cookie est disponible
6. `secure` : si `true`, transmet un cookie uniquement si la connexion est sécurisée (**HTTPS**)
7. `httponly` : si `true`, rend un cookie accessible seulement par le protocole **HTTP** (inaccessible depuis **JavaScript**)

PHP

Remarque

Il n'existe pas de méthode pour détruire un cookie, il faut attendre sa date d'expiration ou lui affecter une date d'expiration dans le passé.

PHP

Remarque

Il n'existe pas de méthode pour détruire un cookie, il faut attendre sa date d'expiration ou lui affecter une date d'expiration dans le passé.

Exemple

```
setcookie("nom", "", time() - 3600);
```

PHP

Les sessions

- Un moyen de conserver des données relatives à un visiteur sur toutes les pages de notre site. .
- Les données seront enregistrées sur le serveur.
- Dans une session, on peut enregistrer une infinité de variables de tout type (scalaire, objet, tableau d'objets...).
- Ces variables seront détruites lorsque le visiteur se déconnecte ou dépasse une certaine durée...
- Elles sont disponibles, en **PHP**, via la variable super-globale `$_SESSION`.

PHP

Avant de stocker des données dans la session, il faut la démarrer (instruction à placer au tout début de la page, avant même la balise `<!DOCTYPE html>`)

```
session_start();
```

PHP

Avant de stocker des données dans la session, il faut la démarrer (instruction à placer au tout début de la page, avant même la balise `<!DOCTYPE html>`)

```
session_start();
```

Pour ajouter une variable dans la session

```
$_SESSION['prenom'] = "john";
```

PHP

Avant de stocker des données dans la session, il faut la démarrer (instruction à placer au tout début de la page, avant même la balise `<!DOCTYPE html>`)

```
session_start();
```

Pour ajouter une variable dans la session

```
$_SESSION['prenom'] = "john";
```

Pour récupérer et afficher une donnée de session

```
echo $_SESSION["nom"];
```

PHP

Avant de stocker des données dans la session, il faut la démarrer (instruction à placer au tout début de la page, avant même la balise `<!DOCTYPE html>`)

```
session_start();
```

Pour ajouter une variable dans la session

```
$_SESSION['prenom'] = "john";
```

Pour récupérer et afficher une donnée de session

```
echo $_SESSION["nom"];
```

Pour détruire une session

```
// pour supprimer toutes les variables session
session_unset();

// détruit la session
session_destroy();
```

PHP

Remarques

- Par défaut, une session **PHP** reste active tant que le navigateur de l'utilisateur est ouvert. Une fois que l'utilisateur ferme son navigateur, la session se termine.
- `session_destroy()` est automatiquement appelée si le visiteur reste inactif pendant plusieurs minutes.
- La durée d'une session en **PHP** est définie dans `php.ini` :
  `session.gc_maxlifetime = 1440`

PHP

Remarques

- Par défaut, une session **PHP** reste active tant que le navigateur de l'utilisateur est ouvert. Une fois que l'utilisateur ferme son navigateur, la session se termine.
- `session_destroy()` est automatiquement appelée si le visiteur reste inactif pendant plusieurs minutes.
- La durée d'une session en **PHP** est définie dans `php.ini` : `session.gc_maxlifetime = 1440`

Pour modifier la durée d'une session

```
ini_set('session.gc_maxlifetime', 3600);
// 3600 = 1 jour
```

PHP

Considérons le menu suivant

```
<nav id="menu">
    <ul>
        <li><a href="page1.php">page1</a></li>
        <li><a href="page2.php">page2</a></li>
        <li><a href="page3.php">page3</a></li>
    </ul>
</nav>
```

PHP

Considérons le menu suivant

```
<nav id="menu">
    <ul>
        <li><a href="page1.php">page1</a></li>
        <li><a href="page2.php">page2</a></li>
        <li><a href="page3.php">page3</a></li>
    </ul>
</nav>
```

- Objectif : afficher ce menu dans toutes les pages de l'application
- Deux solutions possibles
  1. dupliquer ce code dans toutes les pages de l'application
  2. **déplacer ce code dans un fichier `menu.php` et l'inclure dans toutes les pages de l'application**

PHP

Pour inclure le menu précédent défini dans `menu.php`, il suffit d'ajouter dans tous les fichiers le code suivant

```
<?php include("menu.php"); ?>
```

PHP

Pour inclure le menu précédent défini dans `menu.php`, il suffit d'ajouter dans tous les fichiers le code suivant

```
<?php include("menu.php"); ?>
```

Remarques

- `include` s'utilise avec et sans parenthèses.
- On peut la faire avec la même chose, l'entête et le pied de page.

PHP

Les instructions d'inclusion

- `include` produit un **E_WARNING** si le fichier est introuvable et continue l'exécution du script
- `require` produit une erreur fatale et arrête l'exécution du script
- `include_once` même chose que include, mais n'inclut qu'une seule fois un fichier donné dans un même document si `include` a déjà été appelé auparavant avec le même nom de fichier
- `require_once` même chose que include, mais n'inclut qu'une seule fois un fichier donné dans un même document si `require` a déjà été appelé auparavant avec le même nom de fichier

PHP

Menu inclus deux fois

```
<?php include("menu.php"); ?>
<?php include("menu.php"); ?>
```

PHP

Menu inclus deux fois

```
<?php include("menu.php"); ?>
<?php include("menu.php"); ?>
```

Menu inclus deux fois aussi

```
<?php include("menu.php"); ?>
<?php require("menu.php"); ?>
```

PHP

Menu inclus deux fois

```
<?php include("menu.php"); ?>
<?php include("menu.php"); ?>
```

Menu inclus deux fois aussi

```
<?php include("menu.php"); ?>
<?php require("menu.php"); ?>
```

Menu inclus une seule fois

```
<?php include("menu.php"); ?>
<?php include_once("menu.php"); ?>
```

PHP

Menu inclus deux fois

```
<?php include("menu.php"); ?>
<?php include("menu.php"); ?>
```

Menu inclus deux fois aussi

```
<?php include("menu.php"); ?>
<?php require("menu.php"); ?>
```

Menu inclus une seule fois

```
<?php include("menu.php"); ?>
<?php include_once("menu.php"); ?>
```

Menu inclus une seule fois

```
<?php require("menu.php"); ?>
<?php include_once("menu.php"); ?>
```

PHP

Pour rediriger vers une page `connexion.php`

```
header('location: connexion.php');
```

PHP

Pour rediriger vers une page `connexion.php`

```
header('location: connexion.php');
```


Remarques

- La fonction ne fonctionne que si elle est placée au tout début du fichier (avant même la balise `<!DOCTYPE html>`).
- Pas d'espace entre `location` et " :"
- Mettez un `exit()` ou un `die` après `header` pour arrêter l'exécution du script.

PHP

Lien vers la liste des caractères spéciaux les plus utilisés dans les URLs

https://www.w3schools.com/tags/ref_urlencode.asp?_sm_au_=iVVDMg0TSmrMV6Dm