



# PHP

Variables super globales



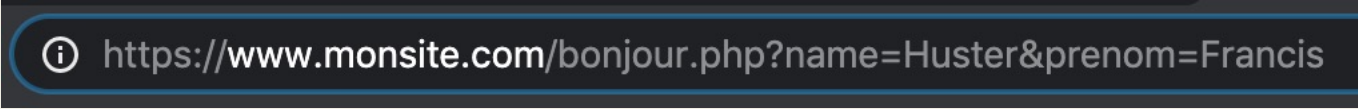


# \$\_GET

Passer des données en URL

# Passer des données à la main

Vous avez déjà probablement déjà remarqué que certaines URL étaient parfois légèrement à rallonge...



<https://www.monsite.com/bonjour.php?name=Huster&prenom=Francis>

Le début de l'URL est relativement familière, jusqu'au « .php » puis après se trouve un « ? » et des données, visiblement un nom et un prénom

Avec cette structure on peut passer théoriquement autant de données que l'on souhaite sauf que quelques navigateurs ne gèrent pas les URLs de plus de 256 caractères.

# Récupérer ces données en PHP

On peut maintenant imaginer faire un lien d'une page à une autre avec des données passées en URL.

```
<a href="https://www.monsite.com/bonjour.php?name=Huster&prenom=Francis">un lien</a>
```

Notez que « & » s'est transformé en « &amp; » pour respecter les standards W3C.

Il va falloir demander à PHP de les récupérer de l'autre côté pour en faire quelque chose

# Récupérer un \$\_GET

Pour récupérer des données passées en URL, on utilise la variable PHP `$_GET`.

Cette variable est appelée **Variable Super Globale** elle est en effet récupérable absolument partout, dans une fonction ou une classe.

Cette variable se présente sous la forme d'un **array associatif**, donc pour récupérer le nom et le prénom dans notre exemple précédent il faudra donc taper

```
echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'];
```

# Isset()

- Il ne faut absolument jamais faire confiance à l'utilisateur. Surtout avec des données passées en URL qui sont facilement modifiable.
- L'utilisateur pourrait déjà simplement supprimer des infos de l'URL ce qui afficherait un gros message d'erreur très moche.
- La fonction **isset()** teste la présence d'un paramètre et renvoie un booléen **true** si il est renseigné, **false** autrement.

```
if ( isset($_GET['prenom']) && isset($_GET['nom']) ) {  
    echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'];  
}  
else {  
    echo 'Je ne sais pas qui tu es...';  
}
```

# Tester les valeurs reçues

```
// Je m'assure que la valeur est bien un int et si
// ce n'est pas le cas je la transtype vers un int
$_GET['repeter'] = (int) $_GET['repeter'];

if ( $_GET['repeter'] < 100 ) {
    // Je fais des choses
}
```

- Imaginons que nous passions un paramètre « répéter » qui déclenche une boucle « for » pour répéter plusieurs fois le « echo 'bonjour' »
- Que se passerait il si l'utilisateur rentrait une valeur énorme ou rentrait une string ou un booléen...
- Grâce à « (int) » on va forcer la valeur reçue à être un entier. La valeur ne sera pas modifiée si c'est déjà un entier, si c'est une string ou un booléen, elle sera transtypée vers un entier.
- On vérifie ensuite que la valeur est dans une plage de valeur raisonnable pour notre serveur.

# Petit exercice

Faites un petit script que vous appellerez directement avec l'URL et qui affichera autant de fois que vous lui demanderez

« Bonjour NOM PRENOM, tu as AGE ans »



# Correction

```
// Je commence par vérifier que ces valeurs existent
if ( isset($_GET['nom']) && isset($_GET['prenom']) && isset($_GET['age']) && isset($_GET['repete']) ) :
    // Je transtype les valeurs pour que ce soient des int
    $_GET['repete'] = (int) $_GET['repete'];
    $_GET['age'] = (int) $_GET['age'];

    // Si les valeurs sont raisonnables
    if ( $_GET['repete'] <= 100 ) :
        // Alors je lance ma loop
        for ( $i = 1 ; $i <= $_GET['repete'] ; $i++ ) :
            echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' tu as ' . $_GET['age'] . ' ans !<br />';
        endfor;
    else : // Si les valeurs sont trop grandes
        echo "Vous avez rentré un nombre trop grand !";
    endif;

else : // Si les valeurs ne sont pas renseignées
    echo 'Toutes les valeurs n\'ont pas été renseignées...';
endif;
```

# FAIRE UN FORMULAIRE

Enfin un peu d'interaction !



# Refaisons un peu de HTML

- Avant de vouloir passer des données par un formulaire il faut commencer par le créer, et c'est en HTML que ça se passe.
- L'élément `<form>` permet de créer un formulaire, il va avoir besoin de deux attributs :
  - **Method** : qui sera soit **get** (passer les infos en URL) ou **post** qui permet de passer les informations directement dans le corps de la requête, de façon sécurisée. On aura tendance à privilégier **post** tant que possible.
  - **Action** : c'est la page qui sera appelée et vers laquelle les infos seront passées une fois le formulaire validé.

```
<form method="post" action="fichier-cible.php">  
|   <!-- Le formulaire ira ici -->  
</form>
```

# <input>

- L'élément <input> est l'élément le plus utilisé dans un formulaire
- Il prend généralement 3 arguments :
  - *Name* : ca sera le nom de la variable récupéré par `$_GET` ou `$_POST`
  - *Id* : c'est l'identifiant unique qui nous permettra de pointer vers l'élément, notamment avec <label>
  - *Value* : dans le cas d'un bouton radio ou d'une checkbox, ca sera la valeur passée par le formulaire, dans le cas du submit ça sera le texte affiché sur le bouton d'envoi, dans le cas d'un text, ce sera la valeur pré-remplie

```
<input type="text" name="pseudo" id="form_pseudo">
<input type="radio" name="voiture" id="form_voiture" value="voiture">
<input type="checkbox">
<input type="password">
<input type="submit" value="Envoyer le formulaire">
<!-- etc... -->
```

# <label>

- L'élément <label> sert à afficher le nom du champs. Il est aussi pratique pour les options d'accessibilité qui vont le lire à haute voix.
- Il prend généralement un argument « for » qui permet de le lier à un élément en particulier. Il faut donc renseigner l'id de l'élément en question

```
<label for="form_pseudo"></label>  
<input type="text" name="pseudo" id="form_pseudo">
```

# Un exemple :

- Ne pensez pas les checkbox comme un élément à choix multiples mais plutôt comme plusieurs choix indépendants dont la réponse ne peut être que vrai ou faux
- Il sera plus simple de récupérer les choix de l'autre côté du formulaire ainsi
- Exemple :
  - As-tu une voiture ? : Oui / Non
  - As-tu un vélo ? : Oui / Non
  - ...

```
<form method="get" action="fichier-cible.php">
  <!-- Petit champs de texte -->
  <label for="form_pseudo">Pseudo</label>
  <input type="text" name="pseudo" id="form_pseudo" value="coucou"><br />

  <!-- Un bouton radio : choix unique -->
  <input type="radio" name="sexe" id="form_sexe_H" value="H">
  <label for="form_sexe_H">Homme</label><br />
  <input type="radio" name="sexe" id="form_sexe_F" value="F">
  <label for="form_sexe_F">Femme</label><br />
  <input type="radio" name="sexe" id="form_sexe_A" value="A">
  <label for="form_sexe_A">Autre</label><br />

  <!-- Checkbox : choix multiple -->
  <!-- Notez le "name" différent à chaque fois -->
  <input type="checkbox" name="voiture" id="form_voiture" value="voiture">
  <label for="form_voiture">Voiture</label><br />
  <input type="checkbox" name="moto" id="form_moto" value="moto">
  <label for="form_moto">Moto</label><br />
  <input type="checkbox" name="velo" id="form_velo" value="velo">
  <label for="form_velo">Velo</label><br />

  <!-- Email -->
  <label for="form_email">Email</label>
  <input type="email" name="email" id="form_email"><br />

  <!-- Date avec un calendrier -->
  <label for="form_date">Date de naissance</label>
  <input type="date" name="date_naissance" id="form_date"><br />

  <!-- Password : affiche des bullets -->
  <label for="form_password">Password</label>
  <input type="password" name="password" id="form_password"><br />

  <!-- Bouton submit -->
  <input type="submit" value="Envoyer !">
</form>
```

```
<form method="get" action="test.php">
  <!-- Checkbox : choix multiple -->
  <!-- J'ajoute des crochets au "name" pour forcer le comportement array -->
  <input type="checkbox" name="vehicule[]" id="form_voiture" value="voiture">
  <label for="form_voiture">Voiture</label><br />
  <input type="checkbox" name="vehicule[]" id="form_moto" value="moto">
  <label for="form_moto">Moto</label><br />
  <input type="checkbox" name="vehicule[]" id="form_velo" value="velo">
  <label for="form_velo">Velo</label><br />

  <!-- Bouton submit -->
  <input type="submit" value="Envoyer !" />
</form>

<?php

var_dump($_GET['vehicule']);
```

## Une astuce pour les checkbox

- Si vous voulez quand même récupérer les différents choix d'une checkbox dans un même array, il va falloir « tricher » un petit peu :



```

<form action="test.php" method="get">
  <!-- Non modifiable mais passé par le formulaire -->
  <label for="pseudo">Pseudo</label>
  <input type="text" name="pseudo" id="pseudo" value="Francis" readonly>
  <br />
  <!-- Requis et sélectionné au chargement de la page -->
  <label for="prenom">Prénom</label>
  <input type="text" name="prenom" id="prenom" placeholder="John" required autofocus>
  <br />
  <!-- Affiché mais non modifiable et non passé par le formulaire -->
  <label for="nom">Nom de famille</label>
  <input type="text" name="nom" id="nom" placeholder="Doe" disabled>
  <br />
  <!-- Champs caché mais quand même envoyé -->
  <input type="text" name="hidden" id="hidden" value="Je suis caché" hidden>
  <input type="submit" value="Send...">
</form>

```

## Les attributs des champs

Vous pouvez aussi tout à fait rajouter des attributs à la fin des champs pour les rendre obligatoires ou déjà sélectionnés de base...

- **Readonly** : Permet de rendre le champs non modifiable par l'utilisateur. Il faut par contre avoir une valeur « value="" » renseignée sinon le champs restera vide
- **Disabled** : Désactive le champs, il restera visible mais ne sera ni modifiable par l'utilisateur ni passé par le formulaire
- **Placeholder=""** : Permet d'avoir une valeur exemple en transparence lorsque le champs n'est pas rempli
- **Required** : Rend le champs obligatoire
- **Autofocus** : Mettre un champs sélectionné de base au load de la page



# Vérifier les données de l'autre côté

- On ne le répètera jamais assez, en faites pas confiance à votre utilisateur. Pire, n'importe qui peut envoyer un formulaire « maison » sur votre page cible qui ne saura pas faire la différence entre votre formulaire ou celui malveillant.
- Vérifiez que les données que vous recevez sont du bon type et que elles ne dépassent pas certaines limites.
- Une fonction **TRÈS** utile est la fonction **htmlspecialchars()** qui va neutraliser les balises **HTML** pour qu'elles ne soient pas actives

```
// Le script sera exécuté
echo "<script>window.onload = () => { alert('Je suis un script') };</script>";

// Le script sera compris comme une string de caractères
echo htmlspecialchars("<script>window.onload = () => { alert('Je suis un script') };</script>");
```

# Exercice

Nous allons faire un formulaire un peu à rallonge demandant toutes ces informations personnelles :

- Nom, Prénom, genre (avec un choix radio), email, site web, styles de musiques préférés (sous forme de checkbox)

Et une page de récupération affichant ces données :

- Affichez un message particulier si la valeur n'a pas été renseignée
- Protégez chaque champs pour éviter les injections de code
- Faites en sorte que les champs email et site web soient cliquables sur la page de réception

```

<form method="get" action="retour.php">
  <!-- Nom -->
  <label for="nom">Nom :</label><br />
  <input type="text" id="nom" name="nom" required><br />
  <!-- Prénom -->
  <label for="prenom">Prénom :</label><br />
  <input type="text" id="prenom" name="prenom" required><br />
  <!-- Genre -->
  <label>Genre :</label><br />
  <input type="radio" id="Autre" name="genre" value="Autre" checked>
  <label for="Autre">Autre</label>
  <input type="radio" id="Femme" name="genre" value="Femme">
  <label for="Femme">Femme</label>
  <input type="radio" id="Homme" name="genre" value="Homme">
  <label for="Homme">Homme</label><br />
  <!-- email -->
  <label for="email">email :</label><br />
  <input type="email" id="email" name="email"><br />
  <!-- Site Web -->
  <label for="website">Site Web :</label><br />
  <input type="text" id="website" name="website"><br />
  <!-- Style Musique -->
  <label>Styles préférés :</label><br />
  <label for="electro">Electro</label>
  <input type="checkbox" id="electro" name="musique[]" value="electro">
  <label for="rock">Rock</label>
  <input type="checkbox" id="rock" name="musique[]" value="rock">
  <label for="rap">Rap</label>
  <input type="checkbox" id="rap" name="musique[]" value="rap"><br />
  <label for="classique">Classique</label>
  <input type="checkbox" id="classique" name="musique[]" value="classique">
  <label for="jazz">Jazz</label>
  <input type="checkbox" id="jazz" name="musique[]" value="jazz">
  <label for="blues">Blues</label>
  <input type="checkbox" id="blues" name="musique[]" value="blues"><br />
  <!-- Submit -->
  <input type="submit" value="Envoyer !" />
</form>

```

# Correction : formulaire

- Utiliser l'attribut « checked » sur un bouton radio permet de s'assurer que le champs sera rempli
- Notez l'utilisation des crochets pour la checkbox

# Correction : traitement

04\_formulaire/retour.php

```
<?php

// Si la variable existe et qu'elle n'est pas vide
if ( isset($_GET['nom']) && $_GET['nom'] != NULL ) {
    echo "<p>Votre nom : " . htmlspecialchars($_GET['nom']) . "</p>";
    // htmlspecialchars() pour éviter l'injection de balises html
}
else {
    echo "<p>Vous n'avez pas renseigné votre nom</p>";
}

if ( isset($_GET['prenom']) && $_GET['prenom'] != NULL ) {
    echo "<p>Votre prénom : " . htmlspecialchars($_GET['prenom']) . "</p>";
}
else {
    echo "<p>Vous n'avez pas renseigné votre prénom</p>";
}

if ( isset($_GET['genre']) && $_GET['genre'] != NULL ) {
    echo "<p>Votre genre : " . htmlspecialchars($_GET['genre']) . "</p>";
}
else {
    echo "<p>Vous n'avez pas renseigné votre genre</p>";
}

if ( isset($_GET['email']) && $_GET['email'] != NULL ) {
    echo "<p>Votre email : <a href=\"mailto:" . htmlspecialchars($_GET['email']) . "\">\" . htmlspecialchars($_GET['email']) . "</a></p>";
}
else {
    echo "<p>Vous n'avez pas renseigné votre email</p>";
}

if ( isset($_GET['website']) && $_GET['website'] != NULL ) {
    $search = array('https://', 'http://', 'www.');
```

// Comme vu dernière fois, je vais normaliser l'url

```
$url = str_replace($search, '', htmlspecialchars($_GET['website']));
    echo "<p>Votre site web : <a href=\"https://www.\" . $url . \"\" target=\"_blank\">\" . $url . "</a></p>"; // et l'utiliser dans le href
}
else {
    echo "<p>Vous n'avez pas renseigné votre site web</p>";
}

if ( isset($_GET['musique']) && $_GET['musique'] != NULL ) {
    $style = $_GET['musique'];
    echo "Vos styles de musique préférés : <ul>";
    foreach ( $style as $value ) {
        echo "<li>\" . $value . "</li>";
    }
    echo "</ul>";
}
else {
    echo "<p>Vous n'avez pas renseigné vos styles préférés</p>";
}
```

# Exercice

Pour mettre tout ça en application, nous allons faire une page qui est protégée par un password.

Elle doit pouvoir s'afficher seulement avec les combinaisons suivantes :

- Login: Admin / Psw: Password
- Login: Francis / Psw: Huster

Essayez de tout faire sur un seul fichier

# Correction

- 05\_password.php
- Bien que très robuste de conception, ce mécanisme de protection est très efficace puisque l'utilisateur ne verra jamais le fichier php mais seulement le html généré depuis celui-ci
- Il est possible de stocker des informations sensibles dans un fichier php sans trop de craintes

```
<body>
  <?php
    // Si les variables sont renseignées et on les bonnes valeurs
    if ( ( isset($_POST['login']) && isset($_POST['password']) ) &&
        ( ($_POST['login'] == 'Admin' && $_POST['password'] == 'Password' ) ||
          ( $_POST['login'] == 'Francis' && $_POST['password'] == 'Huster' ) ) ) :
      ?>

      <!-- Alors j'affiche la page -->
      <h1>Bienvenue ici</h1>
      <p>Si vous lisez ceci c'est que vous êtes connecté</p>

      <?php

    else : // Sinon
      ?>

      <!-- J'affiche le formulaire de connexion -->
      <h1>Page protégée</h1>

      <form method="post" action="05_password.php">
        <label for="login">Login :</label><br />
        <input type="text" id="login" name="login"><br />
        <label for="password">Mot de passe :</label><br />
        <input type="password" id="password" name="password"><br />
        <input type="submit" value="Entrer">
      </form>

      <?php
    endif;
  ?>
</body>
```



```
// Je vais hasher mon password (passé en premier paramètre)
// avec l'algorithme de hash (passé en second paramètre)
// Attention le second paramètre est une constante et ne prend
// pas de guillemets
$hash = password_hash('Maurice', PASSWORD_DEFAULT);
// Un truc du genre $2y$10$aPLU.zsTXym6fswXZp/CbuKqRekaitHs9VlUUKWgm10Cu81sB.FZu

// Vérifier le hash pour tester un password
// Je passe le password à tester en premier paramètre
// et je le compare à mon password hashé (stocké en base de données)
if ( password_verify('Maurice', $hash) ) {
    echo "C'est le bon password";
}
else {
    echo "Ce n'est pas le bon password";
}
```

## Protéger un password

- Il existe cependant une méthode très efficace pour protéger un password pour éviter que une liste de passwords utilisateurs ne fuite en cas de piratage
- Le hachage va transformer le password de votre user en une chaîne hexadécimale complètement illisible, ce qui va sécuriser ce que vous stockez en base de données



# SESSIONS & COOKIES

Miom Miom Miom !



# Sessions

Les sessions sont un moyen de transporter des variables sur plusieurs pages. Comme pour `$_GET` et `$_POST`, `$_SESSION` est une variable super globale

- Un visiteur arrive, une session est créée avec un **PHPSESSID** unique automatiquement, cet ID de session est transporté de page en page (généralement avec un cookie)
- Une fois la session générée, on peut y stocker une infinité de variables
- Lorsque le visiteur se déconnecte, la session est fermée et détruite. Par défaut, les sessions se déconnectent automatiquement après un certain temps d'inactivité : c'est un **timeout**

# Sessions

- Pour initialiser une session, il faut placer la commande **session\_start()** en tout début de code, avant le HTML.
- Il faut aussi penser à l'instancier sur toutes les pages sinon vous ne pourrez pas accéder à la variable **\$\_SESSION**
- Exemple : 06\_sessions

```
1  <?php
2      session_start(); // J'initialise ma session
3  ?>
4  <!DOCTYPE html>
5  <html lang="fr">
6  <head>
7      <title>Une page</title>
8  </head>
9  <body>
10     <?php
11         $_SESSION['prenom'] = "Maurice";
12         echo $_SESSION['prenom'];
13     ?>
14 </body>
15 </html>
```

# Session\_destroy()

- La fonction session\_destroy() permet d'arrêter une session, même si cette fonction est appelée automatiquement après un certain temps d'inactivité (timeout), elle est utile pour créer un bouton de déconnexion.

```
session_destroy();
```

- Les session ne sont pas utiles seulement pour stocker un nom et un prénom comme dans cet exemple mais elles peuvent aussi servir à savoir si un utilisateur est authentifié, pour accéder à plusieurs pages protégées sur un site ou alors aussi pour créer un panier dans un site de e-commerce

# Redirection en PHP

- La fonction `header()` en PHP permet de spécifier l'en-tête HTTP.
- Elle peut servir à spécifier le statut HTTP à envoyer (une erreur 404 par exemple)
- Elle peut servir à spécifier le type de données envoyées par le document (un JSON par exemple)
- Ou elle peut aussi servir de redirection
- N'oubliez jamais que `header()` doit être appelée avant que le moindre contenu ne soit envoyé, soit par des lignes HTML habituelles dans le fichier, soit par des affichages PHP.

```
<?php
header("Location: http://www.example.com/"); /* Redirection du navigateur */

/* Assurez-vous que la suite du code ne soit pas exécutée une fois la redirection effectuée. */
exit;
?>
```

# Exercice

- Nous avons précédemment vu comment faire une page protégée par un mot de passe
- En utilisant les sessions, faites un ensemble de pages protégées par un mot de passe, comme pour un espace membre.
- Nous allons utiliser header() comme une fonction de redirection pour avoir une page dédiée au login
- Correction : 07\_sessions\_pwd

# C'est quoi un cookie ?

- Un cookie est un **petit fichier texte** que l'on enregistre **sur la machine du visiteur**. Il permet de retenir des informations sur le visiteur comme son pseudo ou sa date de naissance.
- Chaque cookie stocke **une information à la fois** donc si vous voulez stocker un pseudo et une date de naissance il faudra créer deux cookies
- Les cookies ont aussi **une date d'expiration**, ils sont automatiquement supprimés par le navigateur après cette date

# Ecrire un cookie

Un cookie stocke un nom à une valeur. Pour écrire un cookie on utilise la fonction **setcookie()**.

Cette fonction prend en général 3 paramètres dans l'ordre suivant :

- Le nom du cookie (par exemple : pseudo)
- La valeur du cookie (par exemple : Francis\_Huster)
- La date d'expiration du cookie sous forme de timestamp

# Timestamp

Le timestamp (ou Unix Time) est le nombre de secondes écoulées depuis le 1<sup>er</sup> Janvier 1970 à minuit UTC

Pour récupérer sa valeur actuelle on utilise la fonction **time()**

Si on veut donc un cookie d'une validité de un an, nous taperons donc : « `time() + 365 * 24 * 3600` »

C'est-à-dire : le timestamp actuel plus 365 jours de 24 heures, chacune de 3600 secondes.

Notre cookies aura donc cette tête là :

```
setcookie('pseudo', 'Francis_Huster', time() + 365*24*3600);
```



# Lire le cookie

Pour lire ensuite un cookie, on utilise la variable `$_COOKIE`.

Comme les autres, c'est une super globale et elle se présente sous la forme d'un array associatif

# Petit exercice

Pour mettre tout ça en application, créez deux pages, une qui crée un cookie avec un pseudo et un lien menant vers une autre page affichant ce cookie

Correction : 08\_cookies

# Pour résumer :

- Les variables **GET**, **POST**, **SESSION** et **COOKIE** sont 4 **Super Globales** qui permettent de passer des variables de page en page. Les 4 sont des **array associatifs**.
- **\$\_GET** permet de passer des informations **via l'URL**. La structure de l'URL sera `monsite.com?variable=valeur&autrevariable=valeur`
- **\$\_POST** permet de passer des variables de page en page de façon **sécurisée et invisible** pour l'utilisateur
- **\$\_SESSION** permet créer un **PHPSESSID** qui est transporté de page en page et avec lui, toutes les informations de la session
- **\$\_COOKIE** crée un petit fichier **stocké sur l'ordinateur** de l'utilisateur. Chaque cookie ne contient qu'une information et possède une date d'expiration notée au format **Timestamp**