PROGRAMMATION PHP-8 – MYSQL-8
ECHANGE D'INFORMATIONS PHP
$_GET
L2S – 7-9 AVRIL 2025
3 JOURNEES

http://php.net/manual/fr/langref.php
http://www.w3schools.com/php/
https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql

Prof : Bertrand Liaudet
Supports de cours en ligne : http://bliaudet.free.fr/rubrique.php3?id_rubrique=276

SOMMAIRE

Edition : septembre 2021 – avril 2025

ECHANGES D'INFORMATIONS EN PHP

0. Présentation du document

- Les exemples sont présentés dans un chapitre en vert avec le mot clé : exemple-
- Les dossiers d'exemples sont fournis dans l'article qui contient ce fichier de cours.
  - ➔ http://bliaudet.free.fr/IMG/zip/PHP-02-Echanges_d_informations.zip
  - ➔ Chargez ce fichier et mettez-le dans le dossier « php » du répertoire web « www » du serveur WEB. Vous pouvez aussi structurer les choses avec des dossiers J1, J2, etc. correspondant aux journées de travail.

- Les exercices à faire sont présentés dans un chapitre en jaune avec les mots-clés : TP- et exercice-
  - ➔ Les sources pour les exercices, quand il y en a, sont fournis dans le dossier des exemples.

Remarque

- Certains fichiers d'exemple commencent par ces trois lignes :

```
echo '<h1>CODE PHP</h1>';
highlight_file('fichier.php');
echo '<h1>RESULTATS</h1>';
```

- Ce code affiche deux balises h1 avec CODE PHP puis RESULTATS
- La fonction « highlight_file » permet d'afficher le contenu du fichier proposé. Quand on teste le code, on commence par affiche le code. Ca permet de voir le code en même temps que les résultats.
- Pour généraliser le code, on écrit : highlight_file(basename(__FILE__));
- basename(__FILE__) permet de récupérer le nom du fichier en cours de traitement.

Document « slidé »

- Ce document Word est en partie « slidé » : chaque page tient, en général, sur un écran, comme un slide.

1. Introduction : passer des variables dans l'URL

Recherche dans google de : « php variable $_GET »

Résultats

- L'URL du résultat est la suivante (ou au moins y ressemble) :
- https://www.google.fr/search?q=php+variable+%24_GET

Analyse de l'URL

- Dans l'URL, on a :
  - La partie adresse : https://www.google.fr/
  - La partie fichier : search
  - Le ? avant les paramètres
  - Les paramètres : q=php+variable+%24_GET
    - ➔ Il peut y avoir plusieurs paramètres séparés par des « & ».

- Chaque paramètre est constitué d'un couple clé=valeur.
- Avec notre recherche, on trouve la clé « q » :
  - q=php+variable+%24_GET
    - ➔ La clé « q » contient les informations « php » + « variable » + « %24_get »
    - ➔ Le %24 vient remplacer le $.

Modification de l'URL

- On peut modifier directement l'URL.
- On met par exemple POST à la place de GET.
- On obtient la page de recherche de « php variable $_POST »
- https://www.google.fr/search?q=php+variable+%24_POST

TP-0-1 : petits exercices d'URL

Google

- Chercher « java » directement dans l'URL de google.

Page du cours

- Regardez l'URL de la page du cours : ici.
  - Quelle est l'adresse ?
  - Quel est le fichier ?
  - Quels sont les paramètres de l'URL ?
  - Est-ce que l'adresse fonctionne sans paramètre ?
  - Changez la valeur du paramètre.
    - ➔ Mettez 233. Que constatez-vous ?
    - ➔ Mettez 500. Que constatez-vous ?
  - Changez la valeur du fichier : mettez rubrique.php puis mettez un paramètre id_rublique=77

2. Méthode Get - URL - Le tableau $_GET

Problématique

- Comment faire passer des informations d'une page web à une autre ?
  - ➔ Passer d'une page à une autre peut se faire en HTML avec des <a href="URL">.
  - ➔ On vient de voir qu'on peut passer des informations sur l'URL.

    - ➔ On va donc pouvoir passer des informations d'une page à une autre de cette manière.

exemple-1 : <a href="URL"> et tableau $_GET

Objectif

L'objectif est, à partir d'une première page (la page d'appel, index) de faire appel à une deuxième page (la page appelée) en passant des informations qui circuleront à travers l'URL.

Principes : un passage de paramètres

- On utilise un link text
- Dans l'URL, on va mettre les paramètres.
- Dans la page appelée, les paramètres se retrouve dans un tableau associatif de paramètres : $_GET

Page appelante : a href

- Fichier : index_appel_bonjour.php
- C'est une page HTML ou PHP (il faudra l'exécuter via le serveur web) avec un <a href> :

```
<DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>$_GET-Appelant</title>
    </head>

    <body>
        <h1>Test de $_GET</h1>
        <p>Rendez-vous sur
            <a href="bonjour.php?prenom=Bertrand&amp;nom=Liaudet">
                un bonjour personnalisé avec $_GET
            </a>
            <!-- syntaxe : ?nom de variable=valeur&nom de variable=etc. -->
        </p>
    </body>
</html>
```

href="bonjour.php?prenom=Bertrand&nom=Liaudet">

- La partie adresse : c'est par défaut l'adresse dont on part
- La partie fichier : bonjour.php
- Le ? avant les paramètres
- Les paramètres : prenom=Bertrand et nom=Liaudet
- Il peut y avoir plusieurs paramètres séparés par des « & ». **On écrit « &amp ; »** car le « & » tout seul crée une confusion en HTML.

➔ Les informations envoyées vont remplir le tableau associatif $_GET

Page appelée : $_GET

- Fichier : bonjour.php

```
< <DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" />
        <title>$_GET-Appelé</title>
    </head>

    <body>
        <h1>Page appelée par un  &lt;a href="URL"&gt; </h1>
        <h2>Des paramètres sont passées l'URL dans le tableau $_GET : regardez l'URL
 !</h2>
        <p> Bonjour
            <?php
                echo $_GET['prenom']. ' ' .$_GET['nom'];
            ?>
        </p>
    </body>
</html>
```

Explications

- Une seule ligne de PHP :
  ➔ echo $_GET['prenom']. ' ' .$_GET['nom'];
- Dans la page appelée, on a un tableau associatif $_GET qui contient les clés et les valeurs fournies dans la page appelante :

```
 array (
     [prenom] => Bertrand
     [nom] => Liaudet
 )
```

  ➔ On peut donc exploiter ce tableau associatif comme on veut dans la page appelée.
- HTML
  ➔ &alt ;a permet d'avoir un <
  ➔ &gt ; permet d'avoir un >
  ➔ code spéciaux HTML : ici.

Précisions

- URL : on voit les paramètres
  - ➔ …bonjour.php?prenom=Bertrand&nom=Liaudet
  - ➔ (dans l'adresse les espaces sont traduits par %20)

- Sécurité – Confidentialité
  - ➔ L'information qui circule est visible sur le lien URL. Il faut donc faire attention à la limiter à une information non confidentielle.

- Limite
  - ➔ En général, les navigateurs n'acceptent pas des url de plus de 256 caractères.
  - ➔ Il faut donc éviter de passer trop d'informations par ce moyen.

Débogage

- Avant le code HTML, on peut ajouter cette partie en PHP :

```php
<?php
    echo '<h1>CODE PHP : </h1>';
    highlight_file(basename(__FILE__));

    echo '<h1>RESULTATS : </h1>';
    echo '<h3>TABLEAU $_GET : </h3>';
    echo '<pre>' ; print_r($_GET); echo '</pre>';
?>
```

  - ➔ highlight permet d'afficher le code
  - ➔ print-r($_GET) permet d'afficher le contenu de $_GET

3. Toujours vérifier les informations fournies par l'utilisateur – 1

Présentation

- On a vu qu'on pouvait voir dans l'URL les informations fournies par l'utilisateur.
- On peut aussi modifier ces informations ou les supprimer directement dans l'URL.
- Ca peut être gênant pour le fonctionnement du site.
  ➔ Il faut donc faire des vérifications.

exemple-2 : href-GET-sans-verifications

Code sans vérifications

```
<h1>Test de $_GET</h1>
<?php
    echo '<p> Nous allons compter jusqu\'à ' .$nombre. '</p>';
    for($i=1; $i<=$_GET['nombre']; $i++){
        echo('<p>' .$i. '</p>');
    }
?>
```

- On boucle sur la taille du paramètre et on fait des affichages.
- Si on donne de mauvaises valeurs au paramètre en changeant l'URL, on aura de mauvais affichages :
- Par exemple :
  - 100000 à la place de 10 : ça ralentit le serveur
  - Toto à la place de 10 : ça affiche n'importe quoi
  - Pas de paramètres : ça génère une erreur

exemple-3 : href-GET-avec-vérifications

Code avec vérifications

- On vérifie tous les cas à problème possibles :

```
<h1>Test de $_GET</h1>
<?php
    echo '<h2>on attend un nombre du programme appelant</h2>'   ;
    if( !isset($_GET['nombre']) ){
        echo '<p> Pas de nombre reçu !</p>';
    }
    else{
        // cast: si $_GET n'est pas un entier, on obtient 0
        $nombre=(int)$_GET['nombre'];
        if( $_GET['nombre'] >10000 OR $_GET['nombre'] <=0){
            echo '<p> Le nombre reçu n\'est pas valide !</p>';
        }
        else{
            echo '<p> Nous allons compter jusqu\'à ' .$nombre. '</p>';
            for($i=1; $i<=$_GET['nombre']; $i++){
                echo('<p>' .$i. '</p>');
            }
        }
    }
?>
```

Vérifier que les informations attendues existent : isset($var)

- isset() est vrai s'il y a quelque chose dans la variable, faux sinon. On pourrait aussi utiliser la fonction empy()

Transtypage (cast) : forcer le type d'une variable : (int)

- $_GET['nombre'] est forcément une chaine de caractères.
- On pourrait tester si elle contient des chiffres avec la fonction ctype_digit().

- On peut aussi « caster » directement $_GET['nombre'] avec un (int). Si $_GET['nombre'] n'est pas un entier, le cast renvoie 0.

Fonction de gestion des variables

- Il existe de nombreuses fonctions qui permettent de faire ces vérifications.
- Ce sont particulièrement les fonctions de gestion des variables :
- http://php.net/manual/fr/ref.var.php

Principe: Never trust user input ! Faille XSS

Principe : envoyer du code : faille XSS

- Never trust user input : ce qui vient de l'URL pourrait être saisie manuellement par l'utilisateur !
- On peut envoyer n'importe quoi et particulièrement du code HTML ou JavaScript qui peut nuire à l'utilisation normale du site, en affichant par exemple des contenus inadaptés.
- L'envoi de code à la place d'un texte, c'est ce qu'on appelle la **faille XSS**.

Protection : htmlspecialchars()

- PHP fournit une fonction qui permettent de traiter le code interprétable comme une simple chaîne de caractères.
  - ➔ **htmlspecialchars()** : http://php.net/manual/fr/function.htmlspecialchars.php
  - ➔ $nombre = htmlspecialchars($_GET['nombre'])

Méthode : commencer par récupérer une valeur vérifiée

- On divise le code en 3 parties : Modèle, Contrôleur et Vue
  - ➔ Le modèle s'occupe de récupérer les données et de les vérifier.
  - ➔ Le contrôleur fait les traitements de données.
  - ➔ La vue affiche les résultats en fonctions des données.
- Dans notre code, quand on a un $_GET, par exemple $_GET['nombre'], on va commencer par récupérer une valeur vérifiée et propre :

```
<h1>Test de $_GET</h1>
<?php
    // Modèle : on récupère et on traite les données
    echo '<h2>on attend un nombre du programme appelant</h2>';
    if( !isset($_GET['nombre']) ){
        echo '<p> Pas de nombre reçu !</p>';
        // on choisira ce qu'on fait dans ce cas là
        // c'est un problème d'interface utilisateur
        // il y a beaucoup de solutions possibles !
    }
    else{
        $nombre = htmlspecialchars($_GET['nombre']) ;
        $nombre=(int)$_GET['nombre'];
        if( $_GET['nombre'] >10000 OR $_GET['nombre'] <=0){
            echo '<p> Le nombre reçu n\'est pas valide !</p>';
            // on choisira ce qu'on fait dans ce cas là
            // c'est un problème d'interface utilisateur
            // il y a beaucoup de solutions possibles !
        }
    }

    // Contrôleur : on ne fait rien dans ce cas

    // Vue : à ce stade on a un $nombre « propre »
    echo '<p> Nous allons compter jusqu\'à ' .$nombre. '</p>';
    for($i=1; $i<=$_GET['nombre']; $i++){
        echo('<p>' .$i. '</p>');
    }
?>
```

TP 2 - $_GET

TP-0-1 : petits exercices d'URL

Google

- Chercher « java » directement dans l'URL de google.

Page du cours

- Regardez l'URL de la page du cours : ici.
  - Quelle est l'adresse ?
  - Quel est le fichier ?
  - Quels sont les paramètres de l'URL ?
  - Est-ce que l'adresse fonctionne sans paramètre ?
  - Changez la valeur du paramètre.
    - ➔ Mettez 233. Que constatez-vous ?
    - ➔ Mettez 500. Que constatez-vous ?
  - Changez la valeur du fichier : mettez rubrique.php puis mettez un paramètre id_rublique=77

TP-0-2 : $_GET, MVC et XSS

Objectif

- Coder le dernier exemple du cours avec un fichier appelant et un fichier appelé.
- Le fichier appelant va « includer sa vue »
- Le fichier appelé va « includer sa vue »
- En cas d'erreur, le fichier appeler revient sur le fichier appelant en passant en paramètre un message d'erreur. Il y a 2 messages possibles : « pas de valeur fournie » et « valeur fournie non conforme ».
- Ce message sera affiché dans la page appelé en bas de page et en rouge (cet unique CSS est mis directement dans la balise : style=«color :red » dans la balise). Ce n'est pas l'usage, mais c'est plus simple pour notre exercice.