Les superglobales Get, Post et Files

Table des matières

I. Contexte 3

II. La Superglobale $_GET 3

A. La Superglobale $_GET 3

B. Exercice : Quiz 5

III. La Superglobale $_POST 6

A. La Superglobale $_POST 6

B. Exercice : Quiz 7

IV. La Superglobale $_FILES 8

A. La Superglobale $_FILES 8

B. Exercice : Quiz 9

V. Essentiel 10

VI. Auto-évaluation 11

A. Exercice 11

B. Test 11

Solutions des exercices 13

I. Contexte

Durée : 1 h

Prérequis : bases de PHP - HTML

Environnement de travail : Visual studio Code - WAMPServer - PHP 8.1.16

Contexte

Bienvenue dans ce cours sur les superglobales en PHP ! Mais, que sont-elles ? Et bien ce sont des variables prédéfinies qui stockent des informations globales accessibles depuis **n'importe où** dans le code PHP. Elles sont au nombre de neuf : **$GLOBALS, $_SERVER, $_GET, $_POST, $_FILES, $_COOKIE, $_SESSION, $_REQUEST, et $_ENV.** Elles se distinguent les unes des autres et se complètent entre elles.

Chacune de ces variables stocke des informations spécifiques comme les données du formulaire soumis, les données de session ou les informations du serveur. Ce sont des outils essentiels pour les développeurs web, car ces variables permettent de récupérer les données envoyées par les utilisateurs et de les traiter côté serveur. D'autant plus qu'elles ne nécessitent pas de déclaration ou d'initialisation supplémentaire. Enfin, les superglobales doivent être utilisées avec prudence, car elles peuvent potentiellement causer des failles de sécurité.

Que vous soyez débutant en programmation web ou que vous avez déjà une expérience, ce cours sur les superglobales **$_GET**, **$_POST, $_FILES** vous permettra de comprendre comment les utiliser pour créer des sites web dynamiques et interactifs.

II. La Superglobale $_GET

A. La Superglobale $_GET

Récupérer des données

La superglobale $_GET en PHP stocke les paramètres de requête HTTP via la méthode GET. Une requête HTTP est un message envoyé par un client, comme un navigateur, à un serveur web pour demander une ressource qui peut être une page web, une image ou un fichier. C'est la base de l'interaction client-serveur.

La méthode GET est couramment utilisée pour récupérer les données transmises via l'URL et comme toutes les superglobales. C'est une variable qui contient des informations sous la forme d'un **tableau associatif**. Mais pourquoi des tableaux ? Parce qu'ils permettent une grande flexibilité dans la manière dont les informations sont traitées. De plus, c'est une structure de données qui permet de stocker des informations avec une relation paire « *clé-valeur* ». De ce fait, il est possible de faire appel à des fonctions de ce tableau pour manipuler ou réorganiser ces données. Les données pouvant varier en termes de nombre, de type et de valeurs, il est donc difficile de les stocker dans une variable unique.

Dans le cas de la méthode GET, chaque clé représente le nom de l'élément de formulaire ou le nom du paramètre d'URL, et la valeur associée est la donnée correspondante à celle envoyée par le client. À noter que la méthode GET est très répandue sur le web et qu'elle est prise en charge par la plupart des serveurs et clients HTTP.

Exemple

Voici un exemple de code qui permet de récupérer les valeurs d'un paramètre en faisant appel à la requête $_GET dans une URL. Le code présenté montre comment créer un tableau associatif en PHP, où chaque paramètre est composé d'une clé et d'une valeur. Le tableau contient trois éléments avec les clés "*clé1*", "*clé2*" et "*clé3*", et respectivement les valeurs "*valeur1*", "*valeur2*" et "*valeur 3*".

```php
<?php

///index.html?id=123 //  PAGE WEB index.html

$_GET['id']; // requête pour récupérer l'ID de l'URL


 // Tableau associatif

$tableau = [
    "clé1" => "valeur1",
    "clé2" => "valeur2",
    "clé3" => "valeur3"
 ];
?>
```

Les paramètres de requête seront spécifiés en utilisant le symbole « *?* ». La requête GET suivante récupèrera la page web index.html avec le paramètre de requête « *id = 123* ».

Cette requête est utilisée pour effectuer des recherches, trier des données, filtrer des résultats ou pour des opérations similaires. Cependant, elle a certaines limitations et restrictions, comme la longueur maximale de l'URL et le fait que les données envoyées soient visibles dans l'URL. La syntaxe pour accéder aux données de la superglobale **$_GET** est simple, comme le montre le code suivant.

Exemple

```php
<?php
if(isset($_GET['plat']) && !empty($_GET['plat'])) {
  $plat = $_GET['plat'];
} else {
  $plat = 'Plat non défini';
}
?>

<a href="?plat=pizza"> Plat 1 </a><br>
<a href="?plat=salade"> Plat 2 </a><br>

<?php
echo '<br>';
echo $plat;
echo '</br>';
?>

```

Voici un exemple de code qui présente deux liens qui contiennent des informations différentes. Le premier a pour paramètre ou information "*plat*" et comme valeur "*pizza*". De même que le deuxième lien contient le même paramètre, mais avec une valeur différente. Ce code PHP récupère la valeur du paramètre $_GET['plat'] selon le clic de l'utilisateur et l'affiche sur la page.

Il est possible d'inclure également des informations supplémentaires dans l'URL, comme des paramètres de suivi ou des identifiants de session pour améliorer les fonctionnalités de l'application.

Exemple

```php
<?php
  // Vérifie si le paramètre "id" est présent dans l'URL
  if(isset($_GET['id'])) {
    // Vérifie si l'id afficher dans l'URL correspond à la valeur 1
    if($_GET['id'] === '1') {
      // Affiche la div 1
      echo '<div><p>Contenu de la div 1</p></div>';
    } else if($_GET['id'] === '2')  {
      // Affiche la div 2
      echo '<div ><p>Contenu de la div 2</p></div>';

        }
}
?>
```

Ce code PHP vérifie si le paramètre "*id*" est présent dans l'URL via la superglobale $_GET. Si c'est le cas, il vérifie si la valeur de l'id correspond à la valeur 1 ou 2. Si l'id correspond à 1, il affiche une div avec le contenu "*Contenu de la div 1*". Sinon, si l'id correspond à 2, il affiche une autre div avec le contenu "*Contenu de la div 2*". En utilisant la méthode GET, il est donc possible de récupérer des informations depuis l'URL pour adapter l'affichage de la page, en fonction des données fournies. La fonction **isset ()** permet de vérifier si une variable a été définie et si elle n'est pas nulle. C'est une façon de vérifier si une variable est initialisée avant de l'utiliser dans une instruction.

Méthode GET et URL

Il est vrai que la visibilité de certaines des informations dans l'URL peut être sujette à controverse, mais elle permet aussi une transparence notable, tant pour l'utilisateur que pour les développeurs. La navigation sur un site web devient claire, c'est pourquoi il est nécessaire que ces informations circulent bien entre les requetés. C'est là tout le travail du développeur que de paramétrer des scripts via méthode GET qui répondent à un besoin de clarté, mais aussi, et surtout à des questions de sécurité.

B. Exercice : Quiz

[solution n°1 p.15]

Question 1

Les données envoyées via l'URL peuvent être récupérées en utilisant :

- ○ En utilisant la fonction GET() ;
- ○ Avec la requête $GET['data']
- ○ En utilisant $_GET[‘data’]

Question 2

Quelle superglobale contient les informations des requêtes GET ?

- ○ $_REQUEST
- ○ $_GET
- ○ $_POST

Question 3

Il est préférable de transmettre des données dites sensibles via :

- SMS
- $_POST
- $_GET

Question 4

À quoi sert la fonction isset () ?

- À vérifier si une variable est false
- À afficher les données
- À vérifier si ma variable est définie et non nulle

Question 5

Qu'est-ce qu'une superglobale ?

- Une variable définie super cool
- Une variable prédéfinie comme tableau associatif et qui contient des données
- Une variable qui doit être déclarée avant d'être utilisé

III. La Superglobale $_POST

A. La Superglobale $_POST

Comprendre et utiliser la méthode $_POST

$_GET est une superglobale utile et couramment utilisée en PHP, mais le fait de transmettre les informations par l'URL la rende peu sûre. C'est pourquoi il est préférable d'utiliser la méthode **$_POST** pour des données de formulaire.

Lors de la création d'un site web, il est fréquent d'avoir besoin de collecter des données à partir de l'utilisateur. Pour ce faire, nous utilisons des **formulaires HTML**. Ces formulaires peuvent contenir différents types d'éléments tels que des zones de texte, des cases à cocher, des boutons radio, etc.

Ce formulaire doit inclure un champ d'action qui indique l'URL du serveur qui recevra les données, ainsi qu'un champ méthode qui spécifie la méthode utilisée. Vous devez également inclure des champs de formulaire pour les données que vous souhaitez envoyer au serveur.

Une fois que l'utilisateur remplit le formulaire et le soumet, les données sont **envoyées au serveur web** pour traitement. Pour récupérer ces données côté serveur, nous utilisons la superglobale $_POST. Contrairement à GET, les données ne sont pas visibles dans l'URL. Cela rend la méthode plus sécurisée pour l'envoi de données sensibles telles que **les informations de connexion ou mot de passe.**

Cependant, il est essentiel de vérifier que ces données soient définies avant de les utiliser dans votre code. Pour cela, nous utiliserons la fonction **isset ()** et/ou avec **empty ()** en complément.

$_POST est une superglobale fondamentale en PHP, il est important de comprendre comment l'utiliser correctement et cette vidéo vous y aidera.

```
    <?php
        // SCRIPT PHP
          ?>
<form action="form.php" method="POST">
      <label for="pseudo">Pseudo :</label>
      <input type="text" id="pseudo" name="pseudo">
      <label for="mot de passe">Mot de passe :</label>
```

```
      <input type="password" id="password" name="password"><br><br>
      <input type="submit" value="Envoyer">
    </form>
```

Exemple

Voici l'exemple d'un formulaire de connexion basique qui permet à l'utilisateur de saisir son pseudo et son mot de passe pour se connecter à un site web, défini avec la méthode POST indiquant que les données seront envoyées de manière sécurisée au fichier form.php lorsqu'il sera soumis.

```php
<?php
// Vérifie si les champs ont été soumis
if (isset($_POST['pseudo']) && isset($_POST['password']))
{
  $pseudo = $_POST['pseudo'];
  $password = $_POST['password'];
       // exemple de validation
  if (empty($pseudo) || empty($password))
  {
      echo "Veuillez entrer un pseudo et un mot de passe";
  }
  else
  {
    echo "Bienvenue, $pseudo !";
  }
}
?>
```

Exemple

Ce code PHP vérifie si les champs "*pseudo*" et "*password*" ont été soumis via la superglobale $_POST. S'ils ont été soumis, il récupère leur valeur dans les variables $pseudo et $password respectivement. Ensuite, il effectue une validation simple pour s'assurer que les deux champs ne sont pas vides. Si l'un des champs est vide, il affiche un message d'erreur "*Veuillez entrer un pseudo et un mot de passe*". Si les deux champs sont remplis, il affiche un message de bienvenue personnalisé en utilisant la variable $pseudo.

En utilisant la méthode POST, il est donc possible de récupérer des informations soumises via un formulaire et de les traiter dans le code PHP pour effectuer des actions spécifiques, en fonction des données fournies.

Si vous voulez faire une validation plus complète sur les champs, **vous pouvez ajouter des conditions supplémentaires**. Par exemple, vous pouvez vérifier si le pseudo respecte certaines règles, si le mot de passe est suffisamment fort ou si l'authentification est réussie. Vous pouvez également rediriger l'utilisateur vers une page de bienvenue. Voici une vidéo qui vous permettra de voir comment renforcer la sécurité au niveau d'un formulaire.

B. Exercice : Quiz

[solution n°2 p.16]

Question 1

Quelle est la différence entre la méthode POST et la méthode GET ?

- ○ Aucune, les deux fonctionnent de la même manière
- ○ La méthode GET est plus rapide que la méthode POST
- ○ La méthode POST est plus sécurisée que la méthode GET

Question 2

Comment peut-on récupérer les données par méthode POST ?

- O En utilisant la fonction POST ()
- O Avec la superglobale GET
- O Avec la superglobale POST

Question 3

Dans cette URL, combien de paramètres sont transmis ? “ *http://example.com/search.php?query=apple&type=fruit*[1] “

- O 1
- O 2
- O Aucun

Question 4

Vous créez une fonctionnalité de connexion sur votre site web, quelle méthode est la plus appropriée ?

- O GET
- O POST
- O FILTER

Question 5

Dans un formulaire, quelle est l'utilité de l’attribut action ?

- O Indique l’emplacement où les données seront reçues
- O Définit le type de données envoyées via le formulaire
- O Actionne la soumission du formulaire

IV. La Superglobale $_FILES

A. La Superglobale $_FILES

L’envoi de fichier avec la superglobale $_FILES

La superglobale **$_FILES** est utilisée dans un formulaire pour envoyer ou télécharger des fichiers. L’envoi de fichier est une pratique courante sur le web, par exemple pour modifier son aspect de profil ou pour permettre le partage de documents, c’est la requête $_FILES entre en jeu. L’un de ces avantages clés est qu’elle permet de stocker des informations détaillées d’un fichier comme le nom, le type, la taille et le chemin d’accès. Simple à utiliser et fonctionnant comme toutes les superglobales avec un tableau associatif, si vous avez compris comment fonctionnent les méthodes GET et POST alors vous ne craignez rien, mais ne vous inquiétez pas, tout sera expliqué en détail en vidéo.

Il est important de noter qu’elle comporte des risques de sécurité. Il faudra donc s’assurer de mettre en place des mesures, telles que la vérification des fichiers téléchargés ou des règles au niveau du type ou de la taille des fichiers autorisés.

1 http://example.com/search.php?query=apple&type=fruit

```html
<form action="upload.php" method="post" enctype="multipart/form-data">
  <label for="image">Choisir une image :</label>
  <input type="file" name="photo" id="image">
  <button type="submit" name="submit">Envoyer</button>
</form>
```

Exemple

Voici un exemple de formulaire qui permet le téléchargement d'une image.

Lorsque l'on crée un formulaire HTML, nous avons vu que nous pouvions spécifier la méthode en utilisant l'attribut **method**. Pour le cas de $_**FILES,** nous allons utiliser la méthode POST et un champ input de type files. À la soumission du formulaire, les données seront stockées dans la superglobale $_FILES.

Pour accéder à ces données, nous utiliserons le **name** comme clé. Par exemple, si le nom de notre fichier est 'photo', nous accèderons à son contenu en utilisant **$_FILES['photo'].**

Ici, l'attribut action de notre formulaire est défini sur "*upload.php*". Cela signifie que lorsque l'utilisateur enverra le formulaire, les données seront envoyées à notre fichier PHP "*upload.php*".

Quant à l'attribut method, il est défini sur "*post*", ce qui signifie que les données du formulaire seront envoyées en utilisant la méthode POST.

Enfin, l'attribut enctype est défini sur "*multipart/form-data*". Cela indique au navigateur que nous envoyons des fichiers, et non simplement du texte, et que le navigateur doit utiliser un encodage spécifique pour envoyer ces fichiers.

```php
<?php
if (isset($_POST['submit']) && (isset($_FILES['photo'] ))) {

// récupérer des informations sur notre image
$image_name = $_FILES['photo']['name']; // nom de notre fichier
$image_tmp_name = $_FILES['photo']['tmp_name']; // dossier temporaire
$image_error = $_FILES['photo']['error']; // valeur d'erreur de notre image
if($image_error === 0 ) {
// Enregistrer l'image dans notre dossier uploads
$destination = "uploads/".$image_name ; // uploads/1.png
move_uploaded_file($image_tmp_name, $destination);
echo " L'image a bien été enregistrée";}
else {
echo " Il y a eu une erreur lors du téléchargement de l'image";
}}?>
```

Ce code PHP permet de gérer l'envoi de fichiers depuis un formulaire HTML. Le formulaire permet à l'utilisateur de sélectionner un fichier image sur son ordinateur, puis de l'envoyer au serveur en cliquant sur le bouton "*Envoyer*". Le code commence par vérifier si le formulaire a bien été soumis en utilisant la méthode POST et en vérifiant si le bouton "*submit*" a été cliqué.

Ensuite, il récupère différentes informations sur le fichier envoyé : le nom du fichier, son emplacement temporaire sur le serveur, et une valeur d'erreur éventuelle. Si aucune erreur n'a été détectée lors de l'envoi du fichier, le code enregistre l'image dans un dossier spécifié (ici, "*uploads*") en utilisant la fonction PHP "*move_uploaded_file*". Si une erreur a été détectée, un message d'erreur est affiché à l'utilisateur.

Il est important de noter que le formulaire HTML doit inclure l'attribut "*enctype*" avec la valeur "*multipart/form-data*" pour permettre l'envoi de fichiers.

B. Exercice : Quiz

[solution n°3 p.17]

Question 1

Comment peut-on limiter la taille d'un fichier téléchargé via un formulaire ?

- ○ En utilisant la fonction filesize()
- ○ En utilisant le paramètre size de la superglobale $_FILES
- ○ En utilisant un filtre

Question 2

Comment vérifier si notre téléchargement de fichier est un succès ?

- ☐ Vérifier si l'image est dans le dossier de destination
- ☐ Vérifier si le paramètre error est égal à 0
- ☐ Avec un echo $_FILES

Question 3

Quelle fonction permet de déplacer le fichier jusqu'à son dossier final ?

- ○ File_uploads()
- ○ Move_uploaded_file()
- ○ Move_please()

Question 4

Vous créez une fonctionnalité de recherche sur votre site web, quelle méthode est la plus appropriée ?

- ○ GET
- ○ POST
- ○ FILTER

Question 5

Supposons que vous ayez un formulaire qui permet aux utilisateurs de télécharger un fichier image. Comment définir le formulaire pour permettre le téléchargement de l'image ?

- ○ <form method= "*GET*">
- ○ <form method= "*POST*">
- ○ <form method= "*POST*" enctype="*multipart/form-data*">

V. Essentiel

Les méthodes GET et POST sont souvent présentées comme des alternatives l'une à l'autre, mais en réalité elles sont complémentaires et peuvent être utilisées de manière efficace, selon le besoin d'une application. GET est idéale pour les requêtes de lecture de données telles que la recherche ou la pagination via l'URL, et facilite la navigation d'un utilisateur ; tandis que POST se voit souvent utilisée pour les requêtes de modification de données, en les cryptant dans le corps de la requête HTTP. En utilisant ces deux méthodes de manière complémentaire, nous pouvons créer des scripts PHP efficaces et sécurisés. Ce sera alors à vous de choisir la méthode appropriée en fonction des exigences attendues.

Nous avons vu que les superglobales sont des variables très importantes dans la programmation web avec PHP. Petit rappel sur les notions vues dans ce cours :

- La superglobale $_POST permet de récupérer les données soumises par les utilisateurs via un formulaire. Ces données sont dans une requête HTTP de type POST et peuvent contenir des informations confidentielles telles que des noms d'utilisateur et des mots de passe.
- Quant à la superglobale $_GET, elle permet de récupérer les données transmises par les utilisateurs via l'URL de la page. Ces données dans une requête HTTP de type GET et sont moins confidentielles, ce qui est pratique pour faire des recherches ou du suivi via des identifiants.
- Enfin, la superglobale $_FILES permet de récupérer les fichiers soumis par les utilisateurs via un formulaire. Ces fichiers sont dans une requête HTTP de type POST.

Pour conclure, les superglobales GET, POST, et FILES sont des outils très pratiques pour récupérer et traiter les données soumises par les utilisateurs dans les formulaires. Cependant, il est important de rester vigilant quant à la qualité et à la sécurité, afin de garantir un fonctionnement optimal de nos applications. Faire preuve de prudence et de rigueur lors de la manipulation de ces informations.

En tant que jeunes développeurs, vous devez assimiler le fonctionnement de ces superglobales et savoir les utiliser de manière pertinente. Il est donc important de se former sur les techniques de validation et de sécurisation de ces données. Je vous invite donc à continuer à pratiquer et à explorer ces notions avec curiosité et enthousiasme. En comprenant bien ces outils, vous créerez des applications web réussies et performantes.

VI. Auto-évaluation

A. Exercice

Vous êtes un développeur en charge de la création d'un formulaire de connexion pour un site web. Votre client vous a fourni un nom d'utilisateur et un mot de passe prédéfini. Il demande à ce que vous vérifiiez la connexion et à ce que, si elle est réussie, l'utilisateur soit redirigé sur une autre page avec un message de bienvenue. Aussi, dans le cas où les identifiants sont incorrects, que vous en informiez l'utilisateur.

Question 1

[solution n°4 p.18]

Dans un premier temps, vous devrez définir des variables et fournir un formulaire.

Question 2

[solution n°5 p.18]

Dans un second temps, il vous demande de vérifier la bonne connexion de l'utilisateur suivi d'une redirection et d'un message d'erreur en cas d'échec, comment procédez-vous ?

B. Test

Exercice 1 : Quiz

[solution n°6 p.18]

Question 1

Vous avez créé un formulaire de réservation de billets pour un évènement. Vous souhaitez vous assurer que chaque utilisateur ne peut réserver qu'un seul billet. Quelle sera la meilleure méthode ?

- ○ Utilisez les cookies pour stocker le nombre de billets réservés par chaque utilisateur
- ○ Demander aux utilisateurs de créer un compte et de se connecter avant de pouvoir réserver un billet
- ○ Utilisez une base de données pour stocker les informations de réservation de chaque utilisateur

Question 2

Vous avez créé un site de e-commerce électronique et vous voulez permettre aux utilisateurs de se connecter ? Comment pouvez-vous sécuriser l'authentification des utilisateurs ?

- Stocker les mots de passe en clair dans une base de données
- Utiliser un algorithme de hachage pour stocker les mots de passe dans la base de données
- Envoyer les mots de passe par e-mail aux utilisateurs lorsqu'ils s'inscrivent

Question 3

Vous avez créé une application de gestion de projet et vous souhaitez permettre aux utilisateurs d'ajouter des fichiers ? Comment gérer le téléchargement des fichiers pour assurer la sécurité ?

- Accepter automatique le téléchargement de tout type de fichier
- Vérifier le type de fichier avant de le télécharger sur le serveur
- À la confiance

Question 4

Quel message conviendrait dans l'echo à la place de //code ?

```php
?php
if (isset($_FILES['image'])) {
  $file_name = $_FILES['image']['name'];
  $file_size = $_FILES['image']['size'];
  $file_tmp = $_FILES['image']['tmp_name'];
  $file_type = $_FILES['image']['type'];
  $file_ext = strtolower(end(explode('.',$_FILES['image']['name'])));

  $extensions = ["jpeg","jpg","png"];

  if (in_array($file_ext,$extensions) === false) {
    echo " //code";
  }

  elseif ($file_size > 2097152) {
    echo 'Le fichier doit être inférieur à 2 MB';
  }

  move_uploaded_file($file_tmp, "uploads/" . $file_name);
}
?>
```

- Format autorisé
- Format refusé, choisissez un fichier jpg ou png
- Les formats jpg ou png ne sont pas autorisés

Question 5

Quelle est l'erreur dans ce code et comment la corriger ?

```
  <form method="get" action=" <?php $_SERVER['PHP_SELF']?>">
  <label for="id">ID:</label>
  <input type="text" id="id" name="id">
  <button type="submit">Rechercher</button>
              </form>
```

- L'ID dans le label n'est pas le même que dans l'attribut name
- La balise PHP n'est pas bien fermée
- La ligne action doit être corrigée

Solutions des exercices

Exercice p. 5 **Solution n°1**

Question 1

Les données envoyées via l'URL peuvent être récupérées en utilisant :

- ○ En utilisant la fonction GET() ;
- ○ Avec la requête $GET['data']
- ◉ En utilisant $_GET[‘data’]

Q Les données transmises dans l'URL peuvent être récupérées en utilisant la superglobale $_GET.

Question 2

Quelle superglobale contient les informations des requêtes GET ?

- ○ $_REQUEST
- ◉ $_GET
- ○ $_POST

Q $_GET contient spécifiquement les données des requêtes $_GET tandis que $_REQUEST contient les données de toutes les requêtes.

Question 3

Il est préférable de transmettre des données dites sensibles via :

- ○ SMS
- ◉ $_POST
- ○ $_GET

Q La méthode $_POST ne transmet pas les données via l'URL contrairement à la méthode $_GET.

Question 4

À quoi sert la fonction isset () ?

- ○ À vérifier si une variable est false
- ○ À afficher les données
- ◉ À vérifier si ma variable est définie et non nulle

Q La méthode isset () retourne une valeur booléenne true si la variable est définie et non nulle.

Question 5

Qu'est-ce qu'une superglobale ?

- ○ Une variable définie super cool
- ◉ Une variable prédéfinie comme tableau associatif et qui contient des données
- ○ Une variable qui doit être déclarée avant d'être utilisé

Q Les superglobales sont des variables prédéfinies et qui ne doivent pas être déclarées. Elles sont accessibles partout dans un script PHP et sont des tableaux associatifs contenant des informations.

Exercice p. 7 **Solution n°2**

Question 1

Quelle est la différence entre la méthode POST et la méthode GET ?

- ○ Aucune, les deux fonctionnent de la même manière
- ○ La méthode GET est plus rapide que la méthode POST
- ◉ La méthode POST est plus sécurisée que la méthode GET

Q La méthode POST est considérée comme plus sûre et transmet de plus larges données contrairement à GET qui peut être limité par la longueur de l'URL.

Question 2

Comment peut-on récupérer les données par méthode POST ?

- ○ En utilisant la fonction POST ()
- ○ Avec la superglobale GET
- ◉ Avec la superglobale POST

Q Pour récupérer des données par méthode POST, il faut faire appel à la superglobale $_POST.

Question 3

Dans cette URL, combien de paramètres sont transmis ? " *http://example.com/search.php?query=apple&type=fruit*[1] "

- ○ 1
- ◉ 2
- ○ Aucun

Q Dans cette URL, 2 paramètres ont été transmis par méthode GET : "*query*" et "*type*".

Question 4

Vous créez une fonctionnalité de connexion sur votre site web, quelle méthode est la plus appropriée ?

- ○ GET
- ◉ POST
- ○ FILTER

1 http://example.com/search.php?query=apple&type=fruit

Il est préférable d'utiliser la méthode POST pour envoyer et récupérer des données dites sensibles.

Question 5

Dans un formulaire, quelle est l'utilité de l'attribut action ?

- (•) Indique l'emplacement où les données seront reçues
- ( ) Définit le type de données envoyées via le formulaire
- ( ) Actionne la soumission du formulaire

L'attribut action indique que les données seront envoyées au fichier où l'URL est spécifiée après soumission du formulaire.

Exercice p. 9 **Solution n°3**

Question 1

Comment peut-on limiter la taille d'un fichier téléchargé via un formulaire ?

- ( ) En utilisant la fonction filesize()
- (•) En utilisant le paramètre size de la superglobale $_FILES
- ( ) En utilisant un filtre

On peut utiliser la propriété size de notre superglobale $_FILES qui permet d'obtenir la taille en octets et vérifier si la taille est inférieure ou supérieure à l'aide d'une condition.

Question 2

Comment vérifier si notre téléchargement de fichier est un succès ?

- [x] Vérifier si l'image est dans le dossier de destination
- [x] Vérifier si le paramètre error est égal à 0
- [ ] Avec un echo $_FILES

Si l'image est dans le dossier de destination, c'est que le téléchargement s'est bien déroule, et si le paramètre error renvoie 0, alors c'est aussi une preuve du bon téléchargement de l'image.

Question 3

Quelle fonction permet de déplacer le fichier jusqu'à son dossier final ?

- ( ) File_uploads()
- (•) Move_uploaded_file()
- ( ) Move_please()

La fonction move_uploaded_file est utilisée pour déplacer un fichier téléchargé depuis son emplacement temporaire vers son emplacement final sur le serveur.

Question 4

Vous créez une fonctionnalité de recherche sur votre site web, quelle méthode est la plus appropriée ?

- [x] GET
- [ ] POST
- [ ] FILTER

La méthode GET est généralement utilisée pour les requêtes de recherche, car elle permet d'inclure les termes de recherche dans l'URL.

Question 5

Supposons que vous ayez un formulaire qui permet aux utilisateurs de télécharger un fichier image. Comment définir le formulaire pour permettre le téléchargement de l'image ?

- [ ] <form method= "*GET*">
- [ ] <form method= "*POST*">
- [x] <form method= "*POST*" enctype="*multipart/form-data*">

La méthode POST ne suffit pas, il est nécessaire de définir l'enctype pour permettre le téléchargement d'images.

p. 11 Solution n°4

```php
<?php
// variables prédéfinies
$user = "utilisateur";
$pass = "mdp123";
  // Affichage du formulaire
echo "<form method='POST'>
<label>Nom d'utilisateur :</label>
<input type='text' name='username'><br>
<label>Mot de passe :</label>
<input type='password' name='password'><br>
<input type='submit' value='Se connecter'>
</form>";
```

p. 11 Solution n°5

```php
if(isset($_POST['username']) && isset($_POST['password'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    // Vérification des identifiants
    if($username === $user && $password === $pass) {
  header('location:page.php');
            echo "Bienvenue, $user !";
    } else {
        echo "Nom d'utilisateur ou mot de passe incorrect.";
    }
} ;

```

Exercice p. 11 Solution n°6

Question 1

Vous avez créé un formulaire de réservation de billets pour un évènement. Vous souhaitez vous assurer que chaque utilisateur ne peut réserver qu'un seul billet. Quelle sera la meilleure méthode ?

- ○ Utilisez les cookies pour stocker le nombre de billets réservés par chaque utilisateur
- ◉ Demander aux utilisateurs de créer un compte et de se connecter avant de pouvoir réserver un billet
- ○ Utilisez une base de données pour stocker les informations de réservation de chaque utilisateur

En demandant à chaque utilisateur de créer un compte, vous pourrez stocker les informations de connexion et de réservation dans une base de données, afin de voir si un utilisateur a déjà réservé ou non un billet. Cela permettra de limiter l'achat de billet.

Question 2

Vous avez créé un site de e-commerce électronique et vous voulez permettre aux utilisateurs de se connecter ? Comment pouvez-vous sécuriser l'authentification des utilisateurs ?

- ○ Stocker les mots de passe en clair dans une base de données
- ◉ Utiliser un algorithme de hachage pour stocker les mots de passe dans la base de données
- ○ Envoyer les mots de passe par e-mail aux utilisateurs lorsqu'ils s'inscrivent

Pour sécuriser l'authentification des utilisateurs sur un site de e-commerce, il est recommandé d'utiliser un algorithme de hachage pour stocker les mots de passe dans la base de données. Stocker les mots de passe en clair est risqué, et l'envoi des mots de passe par e-mail est également déconseillé.

Question 3

Vous avez créé une application de gestion de projet et vous souhaitez permettre aux utilisateurs d'ajouter des fichiers ? Comment gérer le téléchargement des fichiers pour assurer la sécurité ?

- ○ Accepter automatique le téléchargement de tout type de fichier
- ◉ Vérifier le type de fichier avant de le télécharger sur le serveur
- ○ À la confiance

En ajoutant des règles de sécurité en vérifiant le type, la taille, la présence de virus ou d'autres paramètres, vous vous assurez que les utilisateurs ne téléchargent que des fichiers autorisés.

Question 4

Quel message conviendrait dans l'echo à la place de //code ?

```php
?php
if (isset($_FILES['image'])) {
  $file_name = $_FILES['image']['name'];
  $file_size = $_FILES['image']['size'];
  $file_tmp = $_FILES['image']['tmp_name'];
  $file_type = $_FILES['image']['type'];
  $file_ext = strtolower(end(explode('.',$_FILES['image']['name'])));

  $extensions = ["jpeg","jpg","png"];

  if (in_array($file_ext,$extensions) === false) {
    echo " //code";
  }

```

```
15   elseif ($file_size > 2097152) {
16     echo ‘Le fichier doit être inférieur à 2 MB';
17   }
18
19   move_uploaded_file($file_tmp, "uploads/" . $file_name);
20 }
21 ?>
```

- ○ Format autorisé
- ◉ Format refusé, choisissez un fichier jpg ou png
- ○ Les formats jpg ou png ne sont pas autorisés

Q La ligne " *$extensions = array(jpeg…)* " décrit un tableau qui contient les extensions de fichier autorisées.

Question 5

Quelle est l’erreur dans ce code et comment la corriger ?

```
1   <form method="get" action=" <?php $_SERVER['PHP_SELF']?>">
2   <label for="id">ID:</label>
3   <input type="text" id="id" name="id">
4   <button type="submit">Rechercher</button>
5               </form>
```

- ○ L’ID dans le label n’est pas le même que dans l’attribut name
- ○ La balise PHP n’est pas bien fermée
- ◉ La ligne action doit être corrigée

Q La valeur de l’attribut n’est pas correctement définie, la ligne action doit être corrigée en < ?php echo $_Server …