Contexte

Nous avons vu la semaine dernière que la portée d’une variable est le script dans lequel elle est définie. La durée de vie d’une variable est donc le temps de l’exécution du script. Lorsque le script se termine, les variables sont supprimées.

La question qui se pose est : comment interagir avec l’utilisateur dans une session Web, récupérer ses données saisies, conserver le contexte et ses identifiants etc ?

PHP, conçu dès le départ dans un environnement Web, propose un grand nombre d’outils pour répondre à cette problématique, que nous allons détailler aujourd’hui.

Liens et formulaires

En HTML, il existe principalement deux méthodes pour interagir avec un utilisateur :

* les liens (balise <a>) ;
* les formulaires (balise <form>).

Des scripts PHP peuvent aussi bien gérer :

* Les clics utilisateurs, au moyen de lien et de la méthode GET du protocole HTTP
* Les données saisies dans un formulaire par un utilisateur, envoyées et traitées grâce à la méthode POST du protocole http

## Les variables Super Globales

Les variables super globales sont une fonctionnalité cruciale de PHP.

Manuel PHP :

« Les Superglobales — Les Superglobales sont des variables internes qui sont toujours disponibles, quel que soit le contexte

Description

Plusieurs variables prédéfinies en PHP sont "superglobales", ce qui signifie qu'elles sont disponibles quel que soit le contexte du script. Il est inutile de faire global $variable; avant d'y accéder dans les fonctions ou les méthodes.

Les variables superglobales sont :

$\_SERVER, $\_GET, $\_POST, $\_FILES, $\_COOKIE, $\_SESSION, $\_REQUEST, $\_ENV, $GLOBALS

## Utilisation des variables globales du tableau $\_SERVER

**$\_SERVER** est un tableau créé automatiquement contenant des informations comme les chemins du script, les en-têtes, etc.

Voici quelques-unes des informations les plus utiles :

* **$\_SERVER[’HTTP\_USER\_AGENT’]** : permet de connaître la version du navigateur et de ses bibliothèques utilisées pour afficher la page courante.
* **$\_SERVER[’REQUEST\_METHOD’]** : permet de connaître la méthode de requête utilisée.

Ces méthodes sont GET, POST, PUT et HEAD. Nous allons les aborder plus en détail

* **$\_SERVER[’REMOTE\_ADDR’]** : permet de connaître l’adresse IP de l’utilisateur qui a demandé la page.
* **$\_SERVER[’PHP\_SELF’]** : permet de connaître le nom du script actuellement utilisé.
* **$\_SERVER[’QUERY\_STRING’]** : permet de connaître la chaîne de requête utilisée, c’est-à-dire les variables et leurs valeurs passées dans l’URL.
* **$\_SERVER[’HTTP\_ACCEPT\_LANGUAGE’]** : permet de connaître la langue utilisée par votre navigateur.
* **$\_SERVER[’HTTP\_HOST’]** : permet de connaître le nom du serveur ou son adresse IP.
* **$\_SERVER[’PHP\_AUTH\_USER’]** et **$\_SERVER[’PHP\_AUTH\_PW’]** : permet de connaître respectivement le nom de l’utilisateur et son mot de passe saisis dans le fenêtre générée par la fonction header().

L’ensemble des informations disponibles dans le tableau $\_SERVER est présenté ici : <http://www.php.net/manual/en/reserved.variables.server.php>

## Interaction via un lien : $\_GET

Les adresses contenant des liens peuvent également contenir des paramètres GET : la syntaxe à utiliser consiste à ajouter à l’URL, à la suite du caractère ? une liste de variables sous forme variable=valeur, séparées par des &.

Exemple n°1 :script : professeur.php

Les données sont réceptionnées dans le script cible mentionné par le lien au moyen de la variable tableau superglobale $\_GET

Pour passer des valeurs aux paramètres via la méthode GET, il est recommandé d’utiliser la fonction PHP **urlencode**().

Extrait du manuel PHP :

« urlencode — Encode une chaîne en URL

Description ¶

string urlencode ( string $str )

Cette fonction est utile lors de l'encodage d'une chaîne de caractères à utiliser dans la partie d'une URL, comme façon simple de passer des variables vers la page suivante.

Liste de paramètres ¶

str

La chaîne de caractères à encoder.

Valeurs de retour ¶

Retourne une chaîne dont les caractères non alphanumériques (hormis -\_.) sont remplacés par des séquences commençant par un caractère pourcentage (%), suivi de deux chiffres hexadécimaux. Les espaces sont remplacés par des signes plus (+). Ce codage est celui qui est utilisé pour poster des informations dans les formulaires HTML. Le type MIME est application/x-www-form-urlencoded. Ce codage est différent de celui spécifié dans la » RFC 3986 (voir rawurlencode()) : pour des raisons historiques, les espaces sont remplacés par des signes plus (+). »

Pour construire le lien HTML qui va utiliser la méthode GET, il est recommandé d’utiliser la fonction PHP **htmlspecialchars**

Cf la documentation <http://php.net/manual/fr/function.htmlspecialchars.php>

***Il faut s’y reporter de manière régulière quand vous voulez construire des liens URL.***

## Exploitation de formulaires Web: la méthode $\_POST

## Utilisation

On utilise généralement la méthode POST lorsque :

* On a besoin d’envoyer un fichier en attachement
* Les données à envoyer sont volumineuses
* Les données à envoyer sont confidentielles, typiquement un mot de passe qui ne doit pas figurer dans une URL avec la méthode GET
* Lorsque l’actions déclenchée par le formulaire doit être renouvelée, par exemple modération d’un blog / article

## Spécification du script cible

Le script qui va récupérer les données du formulaire est spécifié dans le champ **action** du formulaire.

Exemple : <**form action="analyseFormulaireComplet.php" method="post"**>

Le code d’utilisation des données du formulaire figurera dans le script analyseFormulaireComplet.php

## Champs récupérés dans la variable superglobale $\_POST

Tous les champs valide du formulaire, répondant au standard HTML5, entre la déclaration du formulaire (<FORM>) et la fin (</FORM>), sont passés à la variable PHP superglobale $\_POST

Ces éléments sont accessibles à partir du tableau associatif $\_POST. Par exemple si un champ de type text nom a été défini, comme ceci :

<**input type="text" name="nomEleve" value="" size="20" maxlength="20"** />

alors, la valeur du champ saisi par l’utilisateur sera accessible, dans le script cible, de la manière suivante : $\_POST[**'nomEleve'**]

## Tests de l’existence des champs de formulaire pour construire des variables

Existence

On peut tester l’existence d’un champ de formulaire via la fonction isset .

Cela est utile par exemple pour les cases à cocher (type checkbox), car si aucune case n’est cochée, rien n’est envoyé et donc l’entrée correspondante dans le tableau $\_POST n’est pas définie.

Contrôle

Les fonctions ctype permettent de valider les types de données simples. Cf le manuel PHP <http://php.net/manual/en/ref.ctype.php> pour la liste des fonctions ctype permettant de valider différents types de données

Filtres

Les fonctions de filtre PHP permettent des validations plus complexes, typiquement pour des saisies d’email.

Cf le manuel PHP : <http://php.net/manual/en/ref.filter.php>

## Redirection de page

On souhaite parfois effectuer une redirection sans appuyer sur un bouton. Pour cela, vous pouvez rediriger sur une page PHP en utilisant la fonction header().

Pour rediriger sur une page PHP ou HTML, il faut utiliser l’attribut location de la fonction header().

Par exemple :

<?php

header("Location:http://www.google.fr"); // Redirige sur

                                         // http://www.google.fr

?>

ou :

<?php

header("Location:redirection.php"); // Redirige sur la page

                                    // redirection.php

?>

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. Une erreur très classique est de lire un fichier avec include ou require, et de laisser des espaces ou des lignes vides, qui produiront un affichage avant que la fonction header() ne soit appelée. Le même problème existe avec les fichiers PHP/HTML standards. »

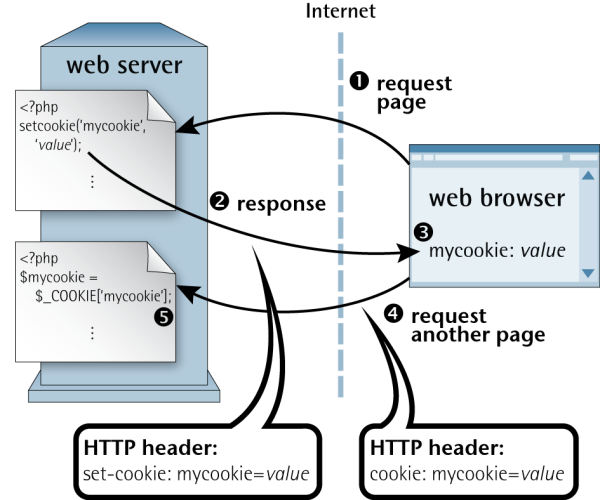
Cookies et sessions

Introduction aux Cookies

Les cookies permettent de stocker des informations sur un utilisateur, ***sur le poste de celui-ci*** , par exemple pour :

* Se souvenir du nom ou d’une information concernant l’utilisateur et éviter de la ressaisir
* Conserver les informations saisies dans un formulaire et permettre de pré remplir lors d’une consultation ultérieure
* Identifier chaque visiteur de manière unique

Principe de fonctionnement des cookies avec PHP



Ecriture d’un cookie

On écrit un cookie avec la fonction PHP setcookie(nom,valeur).

Cf : <http://php.net/manual/fr/function.setcookie.php>

Cette function doit être envoyée avant tout contenu Web donc doit figurer avant toute instruction HTML.

Exemple : setCookieEtudiant.php

Lecture d’un cookie

Les cookies sont accessibles via la variable superglobale $\_COOKIE.

Suppression d’un cookie

Pour effacer un cookie, il faut re définir un cookie du même nom mais avec une valeur dans le passé.

Exemple :

*setcookie*(**'matieresInfo'**,**''**,1);

## Sessions

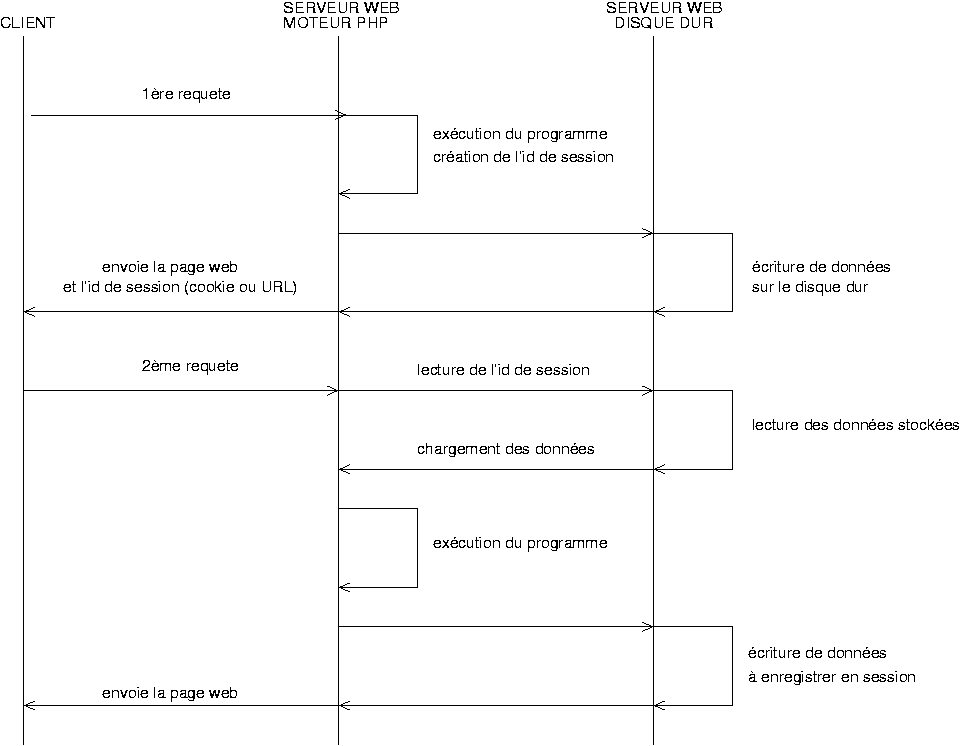
Introduction

La session permet de stocker des informations sur le serveur, au lieu du poste visiteur (cookie), ce qui est plus sûr, et facile d’utilisation au moyen de la variable super globale $\_SESSION.

Comme les données sont stockées sur le serveur, la sécurité est améliorée.

En revanche la durée est limitée (timeout), contrairement aux cookies.

Principe de fonctionnement sous forme de diagramme de séquence (source Université de Caen Normandie)



Démarrage et fermeture d’une session

* session\_start()  : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui. Vous devez appeler cette fonction au tout début de chacune des pages où vous avez besoin des variables de session.
* session\_destroy()  : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes

Utilisation

L’utilisation est très simple pour le développeur, puisqu’il suffit de lire et écrire dans le tableau association de la variable superglobale $\_SESSION pour propager des informations d’une page PHP à l’autre.