Activités autour de PHP

Site officiel de référence : www.php.net

Travail 1 :

Un bon technicien réseau aime tout savoir sur le serveur web/php qu'il est chargé d'administrer. Voici une commande qui permet de vérifier le bon fonctionnement de php tout en fournissant toutes les informations utiles sur la version installée :

```
<?php
      phpinfo();
?>
```

Testez le résultat de ce script.

Travail 2 : Les variables prédéfinies

Les variables prédéfinies sont en majuscules.
Comme toutes les variables php, elles commencent par le caractère $
Il existe un certains nombre de variables contenant des informations utilisables par le programmeur php :

Elle sont regroupées dans un tableau (array) nommé $_SERVER[ ]
Ce lien http://php.net/manual/fr/reserved.variables.server.php en donne la liste.

Testez le script :

```
<?php
echo "</br>Vous avez demand&eacute; la page ".$_SERVER['REQUEST_URI'];
echo "</br>je vous donne : ".$_SERVER['PHP_SELF'];
echo "</br>Mon adresse serveur : ".$_SERVER['SERVER_ADDR'];
echo "</br>Protocole : ".$_SERVER['SERVER_PROTOCOL'];
?>
```

A FAIRE : Ajoutez l'affichage de l'adresse IP du client (celle de votre ordinateur).
ATTENTION : Si votre navigateur utilise un proxy, l'information sera faussée : ce sera l'adresse du proxy ! Dans ce cas il faut prévoir aussi l'affichage de la variable $_SERVER['HTTP_X_FORWARDED_FOR'] si elle existe.

TSVP →

Travail 3 : Rechargement de page (Si vous n'avez pas fait le TP Formulaire)

Nous avons déjà vu comment utiliser les variables de formulaire.
Nous allons créer un formulaire qui n'aura pas l'attribut « *action* ». Dans ce cas, l'action du bouton « *submit* » sera de recharger la même page. Mais au rechargement, les variables du formulaire existeront et seront utilisable par le php :

Testez le script :

```
<!DOCTYPE html>

<html>
<body>
	<form method='GET'>
		Votre age :
		<input type="text" name="zone1"/>
		<input type="submit"/>
	</form>

<?php
	if ( isset($_GET['zone1']) )
	{
		$annee = date('Y') - $_GET['zone1'];
		echo " Ann&eacute;e de naissance : ".$annee;
	}
?>
</body>
</html>
```

Lors du premier chargement de ce script, la partie PHP est exécutée MAIS comme on n'a pas encore validé le formulaire, le test d'existence (*isset*) renvoi FAUX.

Saisissez un nombre, validez, et observez l'adresse URL dans le navigateur : vous devez voir la variable « zone1 » et son contenu : La page vient de se recharger et cette fois les variables du formulaire existent.

Testez le même formulaire avec la **méthode POST** et observez à nouveau l'adresse URL affichée par le navigateur. On en déduit l'utilité de la méthode POST pour un premier niveau de confidentialité. La variable a été transmise de façon discrète dans l'en-tête http, qui n'est pas directement visible par l'utilisateur (Il lui faudrait WireShark).

Le formulaire HTML5 propose de contrôler la saisie au moment du SUBMIT. Dans notre exemple, on demande la saisie d'un nombre. La saisie est obligatoire. On veut même éviter les nombres extravagants.

Modifiez la balise de saisie en ajoutant ou modifiant les **attributs** :

- Ajouter l'attribut « required » pour obliger une saisie.
- Remplacer le type « text » par le type « number » pour forcer la saisie d'un nombre
- Ajouter les attributs « min » et « max » pour définir les limites de la saisie
- Ajouter l'attribut « step » =1 pour empêcher la saisie de nombres à virgule

Travail 4 : Petit projet

Créer un nouveau formulaire où on saisie son année de naissance et son mois de naissance. Le code PHP calculera l'âge.
Exemple : Un code PHP « moderne » utilisera la classe DateTime. Voici le code PHP à adapter et à intégrer au formulaire :

```
$annee = 2000; // A obtenir du formulaire par $_GET ou $_POST
$mois = 05; // A obtenir du formulaire par $_GET ou $_POST
$maDate = new DateTime("$annee-$mois-10"); // On crée un objet DateTime 10/05/2000. Le jour est pris au hasard
$now = new DateTime('NOW'); // On crée un objet date du jour (NOW)
$diff = $now->diff($maDate); // On calcule la différence. Le résultat est un objet DateTime
echo "Votre age:" . $diff->y; // On ne prend que la propriété « y » (year = année) de l'objet.
```

« Have a break, have a KitKat » : J'aimerais bien un « debug » de mes erreurs PHP

Les erreurs de PHP n'apparaissent pas sur le Navigateur. Elles restent sur le serveur PHP. Et comme on travaille « à distance » avec Filezilla …

Solution sympa :

Créer un fichier PHP que vous appelez comme vous voulez ; par exemple : debug.php et qui contient :

```
<?php
      error_reporting(E_ALL);
      ini_set("display_errors", 1);
      require ("votreFichierAtester.php");
?>
```

Et ensuite, au lieu d'exécuter *votreFichierAtester.php* , vous exécutez debug.php.

… Fin de la pause …

Travail 5 : Redirection de page : Fonction HEADER( )

On peut demander au php de se dérouter sur une autre page. La commande *header* permet cela. Comme il s'agit d'une commande qui modifie le header HTTP, elle doit être utilisée SEULEMENT SI AUCUNE donnée d'affichage n'a été envoyée. Cela doit donc se passer AVANT le <body>.

Saisissez le script **info.php**

```
<!DOCTYPE html>
<html>
<body>
      <form method='GET' action="info2.php">
            Votre nom :
            <input type="text" name="zone1"/>
            <input type="submit"/>
      </form>
</body>
</html>
```

Saisissez le script **info2.php**

```
<!DOCTYPE html>

<?php
		if ( !isset($_GET['zone1']) )
		{
			header("Location:info.php"); die();
		}

		if ( $_GET['zone1'] != "dupont" )
		{
			header("Location:info.php"); die();
		}
?>

<html>
	<head>
		<title> Page 2 </title>
	</head>

<body>
	<h1 style="text-align:center">Bienvenue dans cet espace priv&eacute;</h1>

</body>
</html>
```

Testez : Le message de bienvenue ne s'affiche que si vous avez saisi « dupont ». C'est un premier pas vers les espaces sécurisés.

Modifiez *info.php* :
Actuellement, aucun message n'indique que l'on n'a pas saisi le bon nom.
On va donc ajouter un message :

Pour cela, dans *info2.php*, ajoutez une variable GET nommée *status* dans l'URL du *header* :

```
header("Location:info.php?status=echec");
```

Puis dans *info.php* créez le code php qui affichera un message si la variable *status* existe (*isset ()*).

Travail 6 : Accès aux fichiers

Voici le code php d'un compteur de visites simple : Le nombre de visites est stocké dans un fichier texte dans le dossier du site.

```
<?php
     if(file_exists('visites.txt'))
     {          //Ouverture en lecture
                $fic = fopen('visites.txt', 'r+');
                if (!$fic)
                     echo "Erreur ouverture compteur";
                $visites = fgets($fic); // lecture du fichier
     }
     else
     {          // Creation du fichier si besoin
                $fic = fopen('visites.txt', 'a+');
                if (!$fic)
                     echo "Erreur creation compteur";
                $visites = 0;
     }
     $visites++; // Une visite de + !!
     fseek($fic, 0); // Début du fichier
     fputs($fic, $visites); // Ecriture
     fclose($fic); // Fermeture
?>
```

- Insérez ce code au bon endroit

ATTENTION : A cause des restrictions de droits d'accès, le fichier « visites.txt » doit être créé avec un éditeur de texte et contenir le chiffre 0. Ensuite, modifiez ses droits (propriétés) pour que les « autres » puissent écrire dedans.

- Ajoutez l'affichage du nombre de visites sous la bienvenue HTML, avec un « echo » de la variable $visites.