PHP

PHP Web

Dans ce module

- phpinfo() et $_SERVER
- $_GET et $_POST
- $_POST et les redirections
- Upload et $_FILES
- Cookie et $_COOKIE
- Manipulation de cookies
- $_SESSION
- Pratique

phpinfo() et $_SERVER

- phpinfo() affiche toutes information de configurations de php et de son serveur d'exécution
- La variable super globale $_SERVER est un tableau contenant toutes les information que PHP à pu récupérer de la requête client.

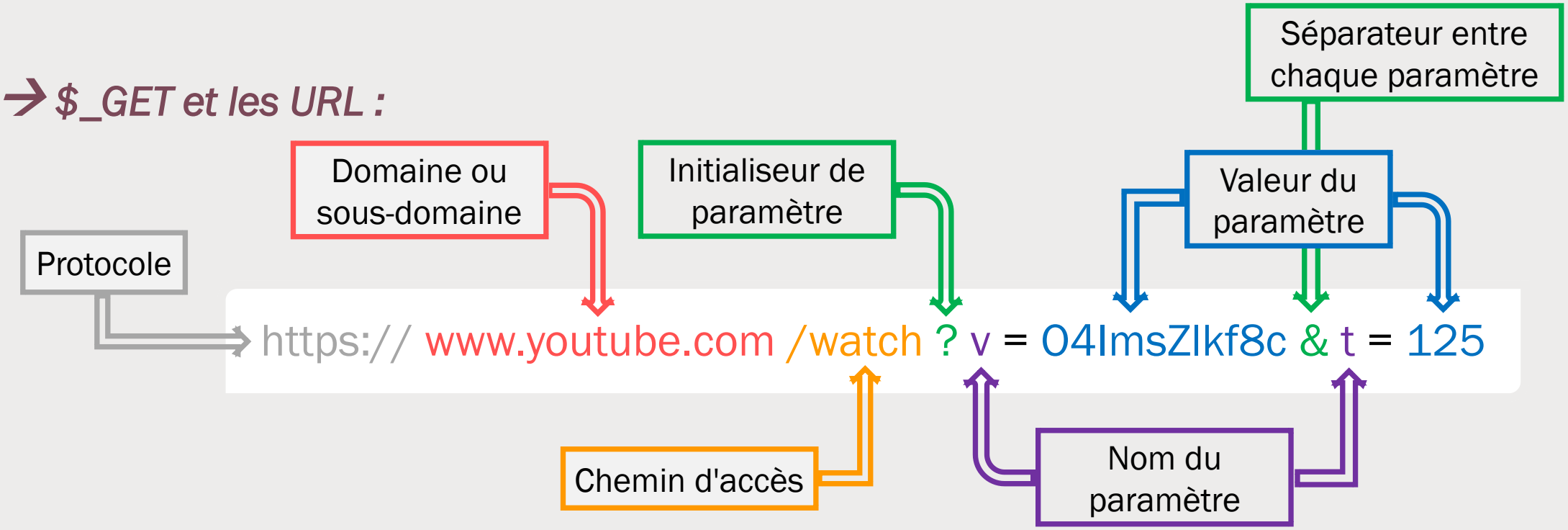
```php
// Display $_SERVER content
var_dump($_SERVER);

// Appel phpinfo()
phpinfo();
```

$_GET et $_POST

- $_GET et $_POST sont des variables super globales. Deux tableaux contenant tout les données key/value transmise dans la requête client.
- $_GET contient les données key/value de l'url de la requête HTTP
- $_POST contient les données key/value du body de la requête HTTP

→ *$_GET et les URL :*



$_GET - Exemple

http://localhost/index.php?name=jean&age=25

```php
var_dump($_GET);

$name = $_GET['name'];
$age = $_GET['age'];

echo "L'utilisateur s'appel ".$name." et à ".$age." ans.";
```

array *(size=2)*

'name' => string 'jean' *(length=4)*

'age' => string '25' *(length=2)*

L'utilisateur s'appelle jean et à 25 ans.

$_POST - Exemple

→ http://localhost/**index.php**

1

```
<form action="contact.php" method="POST">
    <input type="text" name="name">
    <input type="text" name="age">
    <input type="submit" value="Envoyer">
</form>
```

→ http://localhost/**contact.php**

2

```
<?php
    var_dump($_POST);
    $name = $_POST['name'];
    $age = $_POST['age'];
    echo "L'utilisateur s'appel ".$name." et à ".$age." ans.";
?>
```

array *(size=2)*

'name' => string 'jean' *(length=4)*

'age' => string '25' *(length=2)*

L'utilisateur s'appelle jean et à 25 ans.

$_POST et les redirections - Exemple

→ http://localhost/**index.php**

1

```php
<form action="contact.php" method="POST">
    <input type="text" name="name">
    <input type="text" name="age">
    <input type="submit" value="Envoyer">
</form>
```

3

→ http://localhost/**contact.php**

2

```php
<?php
    // Utiliser les données de $_POST
    //    - Envoyer un email
    //    - Ajouter les données en BDD
    //    - Mettre à jour des données en BDD
    //    - Stocker les données dans un fichier (ex: JSON)
    //    - ...

    // Puis une redirection PHP de la requête
    header("Location: http://localhost/index.php");
    exit();
```

Upload et $_FILES

- La super globale $_FILES est un tableau contenant des données utilises les fichiers uploadés. Ces données sont elle-même sous la forme d'un tableau. $_FILES est donc un tableau à 2 dimensions.
- Dans un formulaire HTML, il faut indiquer au navigateur et au serveur que l'on va réaliser l'envoi d'un fichier par l'ajout de **`enctype="multipart/form-data"`** dans la balise form.

→ http://localhost/**index.php**

```
<form action="upload.php" method="POST" enctype="multipart/form-data">
    <input type="file" name="cv">
    ...
```

→ http://localhost/**upload.php**

```
<?php
var_dump($_FILES['cv']);
?>
```

```
array (size=5)
    'name' => 'CV MEYER Marc.pdf'
    'type' => string 'application/pdf' (length=10)
    'tmp_name' => string '/tmp/php3E7A.tmp' (length=25)
    'error' => int 0
    'size' => int 1024352
```

→ *Pour info : HTTP MIME Types*

Upload et $_FILES - Exemple

→ http://localhost/**index.php**

1

```
<form action="upload.php" method="POST" enctype="multipart/form-data" >
    <input type="email" name="email">
    <input type="file" name="cv">
    <input type="submit" value="Envoyer">
</form>
```

→ http://localhost/**upload.php**

2

```
<?php
if (array_key_exists('cv', $_FILES) and $_FILES['cv']['error'] === 0) {
    $tmpPath = $_FILES['cv']['tmp_name'];
    $name = $_FILES['cv']['name'];
    $result = move_uploaded_file($tmpPath, 'storage/'.$name);
    echo ($result ? 'Upload OK' : 'Upload fail !');
}

$email = $_POST['email'];
```

Cookie et $_COOKIE

- Les cookies ne sont pas propres à PHP mais font partie du protocole HTTP. Ce sont des données key/value stockées par le client (navigateur).
- Ces derniers sont retransmis à chaque échange entre le client (navigateur) et le serveur web.
- En PHP (ou tout autre langage serveur) ou en JS Front, on les utilise pour enregistrer des données persistantes sur le navigateur client.
- Les cookies expirent à une date et heure choisie par le serveur ou à défaut par le navigateur (le navigateur aura toujours le dernier mot sur leur conservation).
- Par défaut, il n'y a aucun cookie.
- Pour des raisons de sécurité, ces derniers ne doivent pas contenir de données sensibles.

```php
var_dump($_COOKIE);

$lang = array_key_exists('lang', $_COOKIE)
        ? $_COOKIE['lang']
        : null;
```

Manipulation de cookies

- Pour créer, modifier ou supprimer un cookie en PHP on utilisera la fonction setcookie() :

→ Créer un cookie en PHP avec expiration : maintenant + 1an

```
setcookie('NAME', VAL);
```

→ Créer un cookie en PHP avec expiration définie : maintenant + NUMBER (seconds)

```
setcookie('NAME', VAL, time() + NUMBER);
```

```
setcookie('NAME', VAL, time() + (3600 * 24 * 30))
```

→ Supprime un cookie en PHP

```
setcookie('NAME', null, 1);
```

Session

- Il y a plusieurs concept de sessions. Ici nous parlons de la session gérée via des cookies. Ce mécanisme varie beaucoup d'un langage backend à l'autre.
- La session contrairement au cookie permet de stocker des informations key/value de l'utilisateur directement sur le serveur avec le tableau super global $_SESSION.
- On utilisera la session pour finir le concept d'authentification (login) d'un utilisateur.
- Donc tout comme les cookies, une session sera liée à un utilisateur précis. En réalité, le langage backend (PHP) va créer un cookie identification qui servira à lier la session entre le navigateur et le serveur.
- Pour utiliser une session PHP, il faut la démarrer avec la fonction session_start() :
  1. *On appel la fonction **session_start()***
  2. *Si le cookie de session n'existe, **PHP génère un cookie appelé PHPSESSID** contenant un string aléatoire et unique qui servira d'identifiant. **S'il existe, PHP utilise la session existante**.*
  3. *PHP charge alors toutes les données de session liées à l'identifiant qu'il connait.*
  4. *Après ça, l'utilisation des fonctions de sessions est disponible. **Sans session_start(), l'utilisation d'une fonction de sessions générera une erreur !***

```
// Session non dispnoble
session_start();
// Session dispnoble
```

Session - Exemple

```php
// Utilisation de la session n'est pas dispnoble
// $_SESSION n'existe pas
session_start();
// Utilisation de la session est dispnoble
// $_SESSION existe
var_dump($_SESSION);
```

```php
// Exemple de manipulations des données
session_start();
$_SESSION['email'] = 'user@gmail.com'; // Add or Update
$_SESSION['name'] = 'Marc'; // Add or Update
unset($_SESSION['test']) // Remove
```

```php
// Destruction des données de session
session_start();
session_destroy();
```

→ *Pour info : PHP Session Fonctions*

Session - Exemple

→ http://localhost/**login.php**

```php
// Authentication example
session_start(); // Start
$email = 'admin@my-site.com'; // user log en brut
$pwd = 'jesuisadmin99'; // user log en brut

// Check login
if ($_POST['email'] === $email and $_POST['password'] === $pwd ) {
    $_SESSION['auth'] = true; // Logged
    header('Location: http://localhost/home.php'); // Go home.php
} else {
    $_SESSION['auth'] = false; // Not logged
    header('Location: http://localhost/login-form.php'); // Back
}
```

→ http://localhost/**home.php**

```php
// Page for authenticated user
session_start(); // Start
if ( !array_key_exist('auth', $_SESSION) or !$_SESSION['auth']) {
    header('Location: http://localhost/login-form.php'); // Not logged !
    exit();
}

// CODE IF LOGGED
```

Pratique

- ✓ *Quand on parle de « **page** », c'est pour un fichier PHP qui affiche de l'**HTML** mais qui peut aussi contenir du php pour un affichage « dynamique » de l'HTML, pour afficher des paramètres GET venant de l'URL ou alors pour des restrictions d'accès à la page.*
- ✓ *Quand on parle de « **script** » ou « **page de traitement** », c'est pour un fichier PHP qui **traite des données de formulaires ou des interactions extérieurs (mail, paiement, ...)** et qui après traitement des données redirige l'utilisateur sur une « **page** » pour l'affichage.*

- ❖ Créer un mini-site avec une authentification. Vous devrez utiliser les exemples et les connaissances du cours pour réaliser ce mini-site. Le but est d'avoir une page home.php qui n'est accessible que par un utilisateur que l'on considère connecté. L'authentification est un concept de droit d'accès, il y a donc plusieurs solutions à l'application de ce concept avec les éléments de ce cours.
- ❖ Il est demandé :
  - ➢ 1 page d'accueil simpliste
  - ➢ 1 page de login avec le formulaire (email + password)
  - ➢ 1 script de traitement du formulaire de login avec les redirections nécessaires
  - ➢ 1 page home pour les utilisateurs connectés. Sur cette page home, il serait intéressant de réfléchir à un bouton (lien) de déconnexion pour déconnecter l'utilisateur et de le rediriger sur la page d'accueil. Il est probablement nécessaire de créer un nouveau script.
- ❖ Il peut ensuite être intéressant d'améliorer les comportements du site. Par exemple, un utilisateur connecté ne devrait plus pouvoir accéder à la page de connexion (login) ...

PHP

PHP Web

Fin du module