

# PHP

TRANSMISSION DE DONNÉES ENTRE PAGES WEB

# Préambule – le web dans tous ses états...

## Pourquoi as-t-on besoin de transmettre des données à une page web ?

Le protocole HTTP à la base de la communication entre le navigateur web et le serveur (Apache / nginx, etc...) est un « protocole sans état ».

## Quèsaco un « protocole sans état » (stateless protocol) ?

Un protocole sans état est un protocole de communication qui n'enregistre pas son état entre deux requêtes successives. Chaque requêtes sont indépendantes les unes des autres.

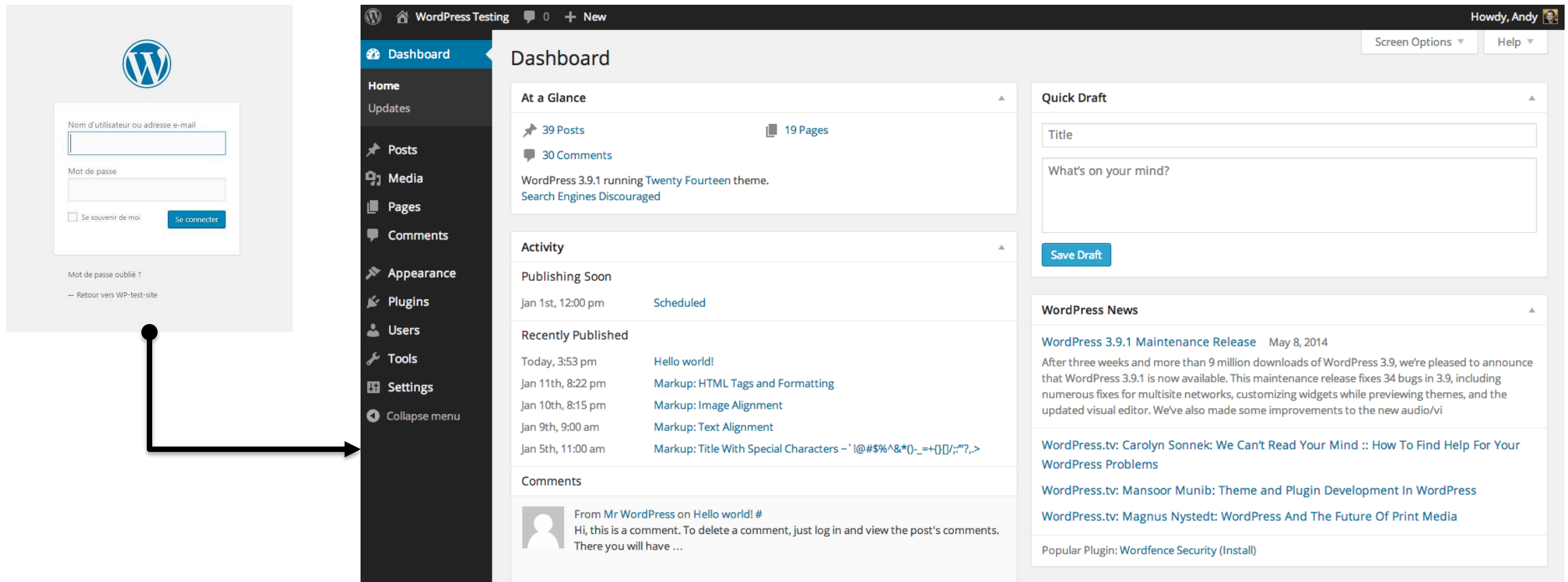
Dans le cas d'un « protocole avec état » (stateful protocol), le serveur conserve l'ensemble des états entre chaque requête. Par exemple, le protocole FTP est un « protocole avec état ».



# Préambule – le web dans tous ses états...

**Il est donc nécessaire de transmettre des données entre les pages pour pouvoir faire évoluer l'état de notre application durant une session de navigation.**

Ici par exemple, même si le protocole est stateless, l'application elle est stateful !



Transmission de données

# UTILISATION DE L'URI

# Un mot sur...

## **URI, URL et URN**

<https://www.w3.org/TR/uri-clarification/>

<https://danielmiessler.com/study/url-uri/>

En un mot

Une URL identifie une ressource en terme de localisation

Une URN identifie une ressource en terme de nom

une URI identifie une ressource, en terme de localisation, de nom, ou les deux.

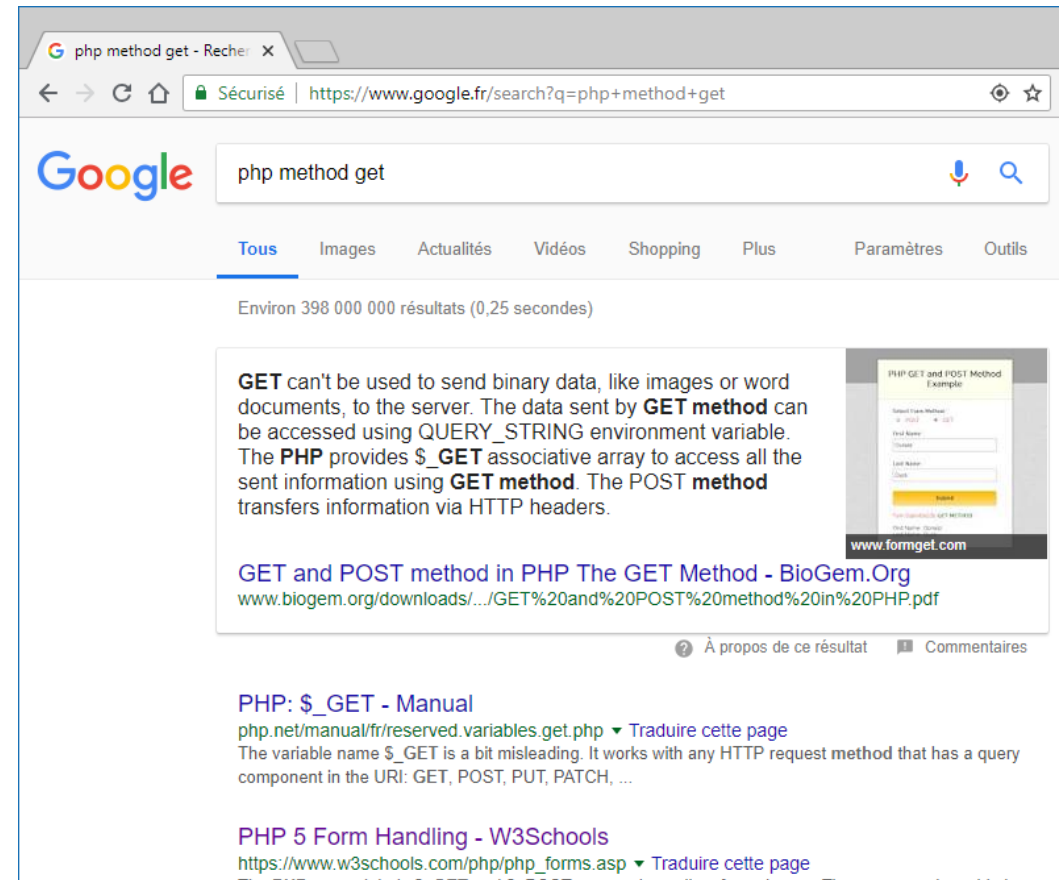
Une URL est donc une URI, l'inverse n'est pas forcément vrai : Une voiture roule, ce qui roule n'est pas forcément une voiture...

# Envoie de données via l'URI

Lorsque l'on fait une recherche sur google, la page est rechargée avec le résultat de la requête. Le serveur a donc bien reçu les paramètres de la recherche « php method get ». Mais, comment ?

**C'est la capacité à ajouter des paramètres dans l'URI du protocole HTTP qui a été utilisé ici.**

Les paramètres sont récupérés côté serveur à partir de la requête et sont utilisés lors de la génération de la réponse.



# Envoie de données via l'URI

**Pour construire une URI avec des paramètres, il suffit d'ajouter les paires clé/valeur à la suite de votre URI, après un « ? », séparées par des « & ».**

**Par exemple, si l'accès à votre page PHP se fait via l'URI `http://monsite.com/mapage.php` alors pour passer comme paramètres un nom et un âge l'URI sera :**

`http://monsite.com/mapage.php?nom=Marcel&age=16`

**Pour récupérer les paramètres côté serveur (donc dans la page `mapage.php`), il suffit d'utiliser la super globale `$_GET`. `$_GET` est un tableau associatif dont la portée est globale.**

```
<?php
echo $_GET['nom']. " a " . $_GET['age']. " an(s).";
?>
```

# Envoie de données via l'URI

**Lorsque l'on traite les paramètres reçus via l'URI, il faut vérifier s'ils existent, soit avec la fonction `isset`, soit avec l'opérateur Null coalescent (`??`, PHP 7.x).**

```
<?php
```

```
// Récupère la valeur de $_GET['utilisateur'] retourne 'aucun'
```

```
// s'il n'existe pas.
```

```
$identifiant = $_GET['utilisateur'] ?? 'aucun';
```

```
// Ceci est équivalent à :
```

```
$identifiant = isset($_GET['utilisateur']) ? $_GET['utilisateur'] : 'aucun';
```

```
// L'opérateur permet de faire du chaînage : Ceci va retourner la première
```

```
// valeur définie respectivement dans $_GET['utilisateur'], $_POST['utilisateur']
```

```
// et 'aucun'.
```

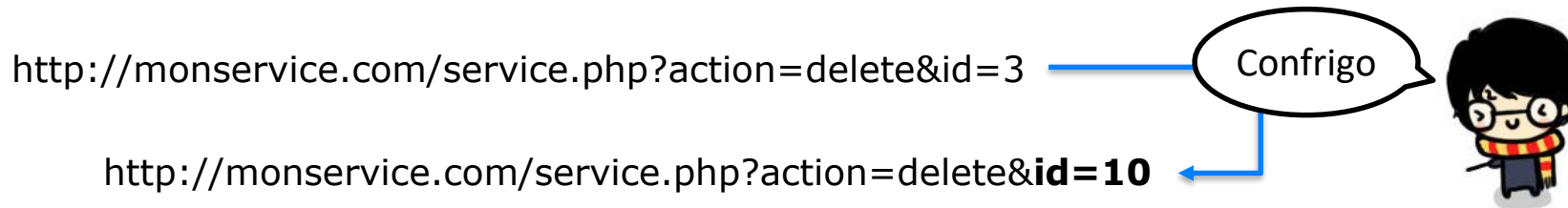
```
$identifiant = $_GET['utilisateur'] ?? $_POST['utilisateur'] ?? 'aucun';
```

```
?>
```



# Un mot sur la sécurité

**Les paramètres utilisés dans l'URI sont visibles de l'utilisateur. Il est donc très facile de les modifier.**



**Ils est obligatoire de vérifier et la nature et la valeur des paramètres que vous recevez !**

**Il peut dans certains cas être utile de signer vos paramètres.**

**Par exemple en utilisant une fonction de hachage :**

« action=delete&id=3 » —SHA1—> f5b981492c6380a46369452f33d779fef0b827d2

http://monservice.com/service.php?action=delete&id=3&token=f5b981492c6380a46369452f33d779fef0b827d2

En vérifiant côté serveur la valeur du paramètre token (en fonction de la valeur id), il devient simple de rejeter l'URI suivante :

http://monservice.com/service.php?action=delete&id=10&token=f5b981492c6380a46369452f33d779fef0b827d2

Transmission de données

# UTILISATION DE L'ENTÊTE HTTP

# Envoie de données via l'entête HTTP

**Le protocole HTTP prévoit l'envoi de données via les méthodes POST et PUT.**

**Exemple (source <http://www.jmarshall.com/easy/http/>) :**

POST /path/script.cgi HTTP/1.0

From: frog@jmarshall.com

User-Agent: HTTPTool/1.0

Content-Type: application/x-www-form-urlencoded

Content-Length: 32

home=Cosby&favorite+flavor=flies

**Bon à savoir :**

**Quelles différences entre POST et PUT ???**

<https://blog.xebia.fr/2014/03/17/post-vs-put-la-confusion/>

<https://www.journaldunet.fr/web-tech/developpement/1202893-rest-put-vs-post/>

<https://tools.ietf.org/html/rfc2616> (ben oui !)

# Envoie de données via l'entête HTTP

**Lors de l'envoi du formulaire de connexion, le login et le mot de passe ne sont pas visibles dans l'URI... OUF !**

**Ces données sont envoyées dans l'entête HTTP, visible ci-dessous... Oh My Gosh !**

The image shows a web browser window displaying the login page of Aix-Marseille University. The page title is "Aix-Marseille Université - Service d'authentification". It features a login form with fields for "Identifiant:" (username) and "Mot de passe:" (password). The username field contains "desbenoit" and the password field is masked with dots. A blue "SE CONNECTER" button is visible. Below the form, there is a security warning and a link for "Problème de connexion".

An arrow points from the "SE CONNECTER" button to the "Headers" tab in the browser's developer tools. The "Headers" tab shows the request details for the URL "https://ident.univ-amu.fr/cas/login?service=https%3A%2F%2Fent.univ-amu.fr%2FuPortal%2FLogin". The "General" section shows the Request URL, Request Method (POST), Status Code (200 OK), Remote Address (139.124.244.65:443), and Referrer Policy (no-referrer-when-downgrade). The "Query String Parameters" section shows "service: https://ent.univ-amu.fr/uPortal/Login". The "Form Data" section, highlighted with a red box, shows the following data:

```
username: desbenoit
password: mon mot de passe
lt: LT-3792701-ACjdPRDsSIUjTKuI10FMkWMQcdcGCI-ident.univ-amu.fr
execution: e1s2
_eventId: submit
submit: SE CONNECTER
```

# Envoie de données via l'entête HTTP

## Exemple de script permettant l'envoi de données via l'entête HTTP

```
<?php
$server = '127.0.0.1';
$host = "www.example.com";
$port = 80;
$target = "/test/cible.php";
$formdata = array ( "login" => "Marie", "pwd" => "mon mot de passe");

//build the post string
foreach($formdata AS $key => $val) {
    $poststring .= urlencode($key) . "=" . urlencode($val) . "&";
}

// strip off trailing ampersand
$poststring = substr($poststring, 0, -1);
$fp = fsockopen($server, $port, $errno, $errstr, $timeout = 30);

if (!$fp) {
    echo "$errstr ($errno)\n";
}
```

```
else {
    //send the server request
    fputs($fp, "POST $target HTTP/1.1\r\n");
    fputs($fp, "Host: $host\r\n");
    fputs($fp, "Content-type: application/x-www-form-urlencoded\r\n");
    fputs($fp, "Content-length: ".strlen($poststring)."\r\n");
    fputs($fp, "Connection: close\r\n\r\n");
    fputs($fp, $poststring . "\r\n\r\n");

    //loop through the response from the server
    $response = "";
    while(!feof($fp)) {
        $response .= fgets($fp, 4096);
    }

    //remove response header
    $response = substr($response, strpos($response, "\r\n\r\n") + 4);
    echo $response;

    //close fp - we are done with it
    fclose($fp);
}
?>
```

# Envoie de données via l'entête HTTP

**Pour récupérer les paramètres côté serveur (donc dans la page cible.php), il suffit d'utiliser la super globale `$_POST`. Comme pour `$_GET`, `$_POST` est une super globale. Il faut également vérifier que la variable existe, soit avec `isset`, soit avec l'opérateur Null coalescent.**

*[dans cible.php]*

```
<?php
    $login=$_POST['login'] ?? "";
    $pwd=$_POST['pwd'] ?? "";
?>
<!doctype html>
<html class="no-js" lang="">
    <head>
        <meta charset="utf-8">
        <title>Cible</title>
    </head>
    <body>
        Login: <?=$login?> <br/>
        Mot de passe: <?=$pwd?>
    </body>
</html>
```

Transmission de données

# UTILISATION DES FORMULAIRES

# Déclaration d'un formulaire

## Élément form

Pour déclarer un nouveau formulaire, on utilise l'élément form

```
<!doctype html>  
<html lang="fr">  
  <head>  
    <meta charset="utf-8" />  
    <title>Formulaire HTML5</title>  
  </head>  
  
  <body>
```

```
    <form action="..." method="...">  
  
    </form>
```

```
  </body>  
</html>
```



# Méthode d'envoi et traitement

```
<form action="traitement.php" method="post">  
  
</form>
```

## Attribut action

Chemin vers le fichier qui va traiter les données envoyées par le formulaire.

## Attribut method

Mode de transmission des données.

Deux valeurs sont possibles :

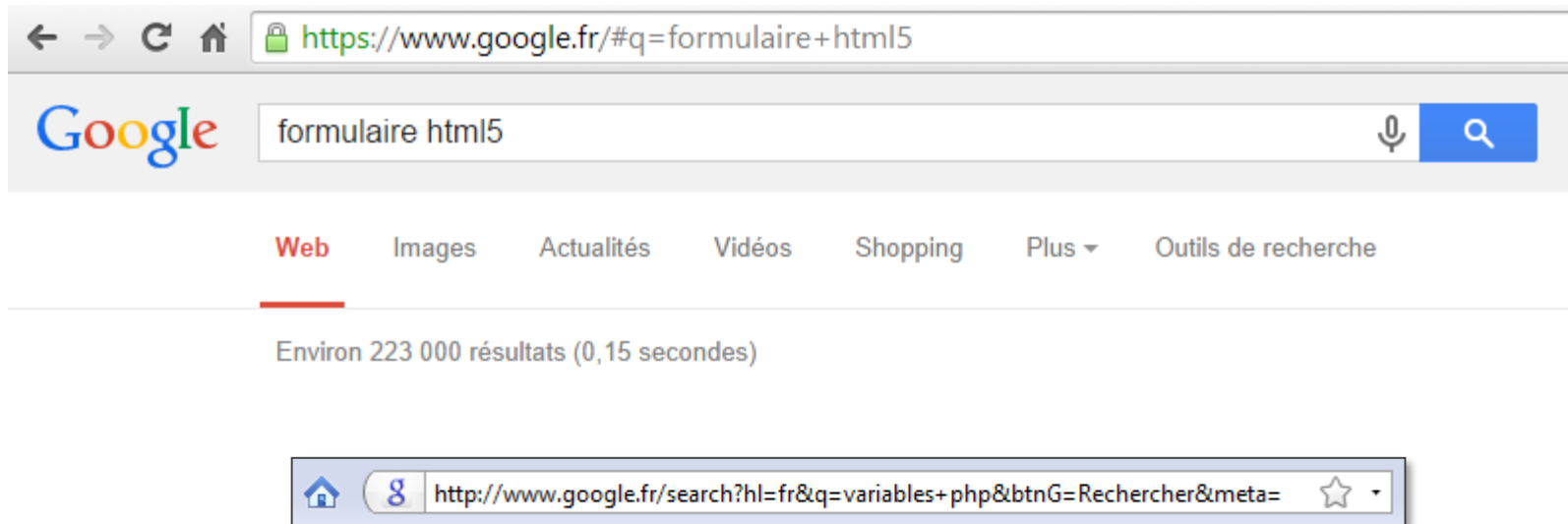
- ⦿ GET ou
- ⦿ POST

# Méthode d'envoi et traitement

## Passage des données en GET

Envoie les variables à la suite de l'adresse.

Les paramètres sont donc visibles pour l'utilisateur.



# Méthode d'envoi et traitement

## Passage des données en POST

Envoie les variables dans l'entête de la requête HTTP.

Les paramètres ne sont donc pas visibles pour l'utilisateur.

The screenshot shows a web browser with the URL `https://fr-fr.facebook.com`. The page displays the Facebook login interface with fields for 'Adresse électronique ou téléphone' (containing 'marcel') and 'Mot de passe' (containing '\*\*\*\*\*'), and a 'Connexion' button. Below the login fields is the 'Inscription' (Sign Up) section. The browser's developer tools are open, showing the 'Headers' tab. The request details are as follows:

- Remote Address: 31.13.93.241:443
- Request URL: `https://www.facebook.com/login.php?login_attempt=1`
- Request Method: POST
- Status Code: 200 OK
- Request Headers (15)
- Query String Parameters (view parsed):
  - `login_attempt=1`
- Form Data (view parsed):
  - `lsd=AVo1JCcW&email=marcel&pass=pagnol&default_persistent=0&timezon`
  - `e=-120&lgnrnd=024356_JN-X&lgnjs=1410687838&locale=fr_FR`
- Response Headers (17)

A red arrow points from the 'Connexion' button on the page to the 'Request Headers' section in the developer tools, illustrating the flow of data from the user interface to the backend.

# Déclencher l'envoi d'un formulaire

## Bouton submit

Elément bouton, attribut type = submit

L'appuie sur le bouton déclenche l'appel du script défini par l'attribut action de l'élément form.

```
<form action="script_default.php" method="post">  
  
  <input type="submit" value="Envoyer">  
</form>
```

## L'attribut formaction

Si cet attribut est spécifié pour un bouton de type submit, c'est le script qui y est défini qui sera appelé en lieu et place de celui défini pour l'élément form.

```
<form action="script_default.php" method="post">  
  
  <input type="submit" value="Envoyer" formaction="script_send.php">  
</form>
```

# Définir des données dans un formulaire

## Différent éléments permettent de définir des valeurs dans un formulaire

- ◉ input
- ◉ textarea
- ◉ select
- ◉ ...



Les objets de formulaires doivent-êtré placés entre les balises `<form>` et `</form>`. Cependant, l'utilisation de l'attribut `form` d'un champ de formulaire permet de spécifier son rattachement lorsque celui-ci se trouve en dehors de la définition d'un formulaire.

# Les champs texte

## Sur une ligne : balise input - type="text"

```
<form action="traitement_formulaire.php" method="post">  
  <label for="form-nom">Entrez votre nom :</label><br />  
  <input name="nom" id="form-nom" type="text" value="?" size="25" /><br />  
  <input type="submit" value="Envoyer" />  
</form>
```

Entrez votre nom :

## Mot de passe : balise input - type="password"

```
<form action="traitement_formulaire.php" method="post">  
  <label for="form-mdp">Entrez votre mot de passe :</label><br />  
  <input name="mdp" id="form-mdp" type="password" value="" size="25" /><br />  
  <input type="submit" value="Envoyer" />  
</form>
```

Entrez votre mot de passe :

# Les champs texte

## Nouveaux champs introduit par HTML5

Champ de type tel

Champ de type url

Champ de type email

Champ de type search

Champ de types date, time, datetime, datetime-local

Champ de type month et week

Champ de type number

Champ de type range

Champ de type color

**HTML 5 Form Elements**

Email:

URL:

Telephone:

Password:

Number:

Range:

Date:

Month:

Week:

Time:

Datetime:

Datetime-local:

Search:

Color:

lun.	mar.	mer.	jeu.	ven.	sam.	dim.
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

# Zone de texte

## Créer une zone de texte sur plusieurs lignes

- balise textarea

```
<form action="traitement_formulaire.php" method="post">
<label for="form-msg">Entrez votre commentaire
:</label><br />
  <textarea id="form-msg" name="comment" cols="35"
rows="5">
Ceci est un commentaire
sur plusieurs
lignes.
  </textarea><br />
  <input type="submit" value="Envoyer" /><br />
</form>
```

Entrez votre commentaire :

Ceci est un commentaire  
sur plusieurs  
lignes.

Envoyer



# Boutons de formulaire

## Créer des boutons

- balise input - type="submit", type="reset", type="button"

```
<form action="form_send.php" method="post">
  <p>
    Entrez votre e-mail :<br />
    <input type="text" name="mail" size="35" /><br />
    <input type="submit" name="envoyer" value="Valider" />
    <input type="reset" name="RAZ" value="Effacer" />
    <input type="button" name="Annu" value="Annuler" />
  </p>
</form>
```

Entrez votre e-mail :

Les boutons de type *submit* et *reset* doivent obligatoirement être placés entre les balises `<form>` et `</form>`, ou au moins y faire référence, sinon ils n'auront aucun effet.


# Image cliquable

## Créer un bouton de validation avec une image

- balise input - type="image"

```
<form action="form_send.php" method="post">
  <p>
    Entrez votre e-mail :<br />
    <input type="text" name="mail" size="35" />
    <input type="image" name="img_send" src="images/send.gif" alt="logo"/>
  </p>
</form>
```

Entrez votre e-mail :



(img\_send\_x, img\_send\_y)

Lorsque l'on clique sur l'image, les données du formulaire sont transmises ainsi que les coordonnées du point cliqué accessible par **name\_x** et **name\_y**.

# Cases à cocher et boutons radio

## Case à cocher

- balise input - type="checkbox"

```
<form action="./form.php" method="post">
  <p>
    Recevoir les news par e-mail
    <input type="checkbox" name="q1" value="oui" checked="checked" />
    <br /><input type="submit" value="Envoyer" /><br />
  </p>
</form>
```

Recevoir les news par e-mail ☒

Envoyer

## Boutons radio

- balise input - type="radio"

```
<form action="./form.php" method="post">
  <p>
    Vous êtes :
    <br /><input type="radio" name="genre" value="h" checked="checked" /> Un homme
    <br /><input type="radio" name="genre" value="m" /> Une femme
    <br /><input type="radio" name="genre" value="" /> Autre...
    <br /><input type="submit" value="Soumettre" /><br />
  </p>
</form>
```

Vous êtes :

☒ Un homme

☐ Une femme

☐ Autre...

Soumettre

# Liste déroulante / Liste de sélection

## Définir une liste déroulante

- balise select

```
<form action="./form.php" method="post">
  <p>Vous gagnés : <br />
  <select name="salaire">
    <option value="1">Moins de 1000€</option>
    <option value="2">Entre 1000€ et 2000€</option>
    <option value="3">Entre 2000€ et 3000€</option>
    <option value="4">Plus de 4000€</option>
  </select>
</p>
</form>
```

Vous gagnés :

Moins de 1000€

Moins de 1000€

Entre 1000€ et 2000€

Entre 2000€ et 3000€

Plus de 4000€

- Attribut size - liste de sélection :

```
<form action="./form.php" method="post">
  <p>Vous gagnés : <br />
  <select name="salaire" size="5">
    <option value="1">Moins de 1000€</option>
    <option value="2">Entre 1000€ et 2000€</option>
    <option value="3">Entre 2000€ et 3000€</option>
    <option value="4">Plus de 4000€</option>
  </select>
</p>
</form>
```

Vous gagnés :

Moins de 1000€

Entre 1000€ et 2000€

Entre 2000€ et 3000€

Plus de 4000€

# Liste multiple

## Définir une liste de sélection multiple

- balise select - attributs size et multiple

```
<form action="./form.php" method="post">
  <p>Selectionnez les langages que vous connaissez :<br />
  <select name="id_langage[]" multiple size="6">
    <option value="1">PHP</option>
    <option value="2">JAVA</option>
    <option value="3">Javascript</option>
    <option value="4">C</option>
    <option value="5">C++</option>
    <option value="6">C#</option>
  </select>
</p>
</form>
```

Selectionnez les langages que vous connaissez :

PHP	
JAVA	
Javascript	
C	
C++	
C#	



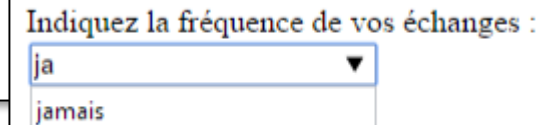
Attention au nom de l'identifiant, c'est un choix multiple, il faut donc utiliser un tableau pour pouvoir récupérer les données avec php. C'est pour cela que l'on utilise des crochets [ ]

# Suggestion de choix

## Élément datalist

L'élément **datalist** associé à un champ de type **text** permet de définir un champ de saisie avec une autocomplétion.

```
<label for="form-frq">Indiquez la fréquence de vos échanges :</label>  
<input list="frq" type="text" id="form-frq">  
<datalist id="frq">  
  <option value="jamais">  
  <option value="rarement">  
  <option value="1 à 2 fois par semaine">  
  <option value="3 à 4 fois par semaine">  
  <option value="plus de 5 fois par semaine">  
</datalist>
```



A utiliser avec une liste modérée d'éléments.

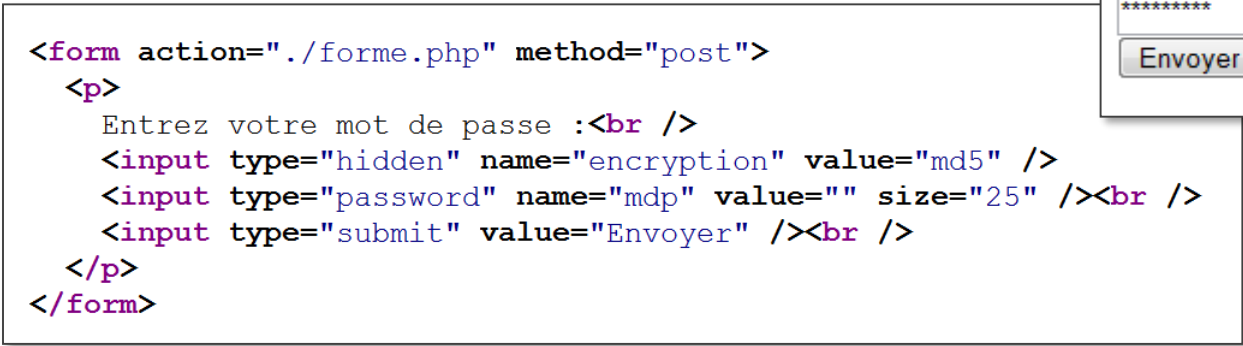
Il ne s'agit pas de dumper l'intégralité d'une table d'une base de donnée : Utilisation d'une requête AJAX.

# Champs cachés

## Champs cachés – type="hidden"

- Les champs cachés servent à envoyer une valeur sans que l'utilisateur ne le voie et sans qu'il puisse la modifier.

```
<form action="./forme.php" method="post">
  <p>
    Entrez votre mot de passe :<br />
    <input type="hidden" name="encryption" value="md5" />
    <input type="password" name="mdp" value="" size="25" /><br />
    <input type="submit" value="Envoyer" /><br />
  </p>
</form>
```



Entrez votre mot de passe :

\*\*\*\*\*

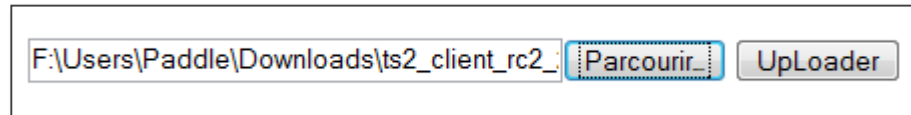
Envoyer



Attention, la valeur d'un champ caché est accessible et modifiable dans le source de la page HTML côté client !  
Il faut donc faire attention à ce que l'on y met.

# Envoi de fichiers

## Formulaire d'envoi de fichier



A screenshot of a web form for file upload. It features a text input field containing the file path "F:\Users\Paddle\Downloads\ts2\_client\_rc2\_". To the right of the input field is a button labeled "Parcourir..." (Browse...). Further to the right is a button labeled "UpLoader".

## balise input - type="file"

```
<form action="upload.php" method="post" enctype="multipart/form-data">
  <p>
    <input type="file" name="fichier" size="40" />
    <input type="submit" value="UpLoader" />
  </p>
</form>
```



Il est nécessaire de spécifier dans la balise *form* qu'il ne s'agit pas de données textuelles. C'est le rôle de l'instruction ***enctype="multipart/form-data"***



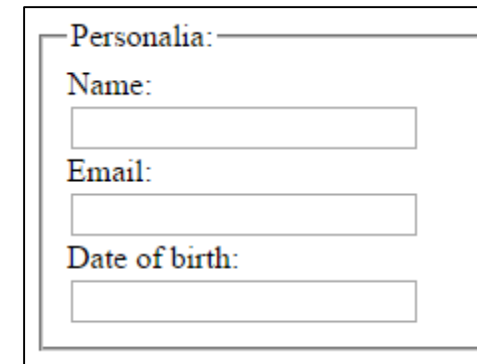
# Regrouper des champs - <fieldset>

## Element fieldset

Pour des raisons d'ergonomie, il est possible de regrouper les champs d'un formulaire grâce à l'élément fieldset.

**L 'élément legend permet de définir le nom du groupe.**

```
<form>
  <fieldset>
    <legend>Personalia:</legend>
    Name: <input type="text"><br>
    Email: <input type="text"><br>
    Date of birth: <input type="text">
  </fieldset>
</form>
```



# HTML5 - API de validation

Avec HTML5, la validation des formulaires par le navigateur apparait (sans utiliser explicitement javascript).

## L'attribut require

```
<input type="email" id="email_addr" name="email_addr" required />
```

Email

Envoyer

! Veuillez renseigner ce champ.

## L'attribut pattern

```
<input type="text" id="part" name="part" required pattern="[A-Z]{3}[0-9]{4}"  
title="Part numbers consist of 3 uppercase letters followed by 4 digits." />
```

Part numbers

Envoyer

! Veuillez respecter le format requis.  
Part numbers consist of 3 uppercase letters followed by 4 digits.

# HTML5 - API de validation

## L'événement oninput

```
<input type="email" id="email_addr" name="email_addr" required />

<input type="text" id="part" name="part" required pattern="[A-Z]{3}[0-9]{4}"
       title="Part numbers consist of 3 uppercase letters followed by 4 digits."/>

<label>Email:</label>
<input type="email" id="email_addr" name="email_addr">

<label>Repeat Email Address:</label>
<input type="email" id="email_addr_repeat" name="email_addr_repeat" oninput="check(this)">

<script>
  function check(input) {
    if (input.value !== document.getElementById('email_addr').value) {
      input.setCustomValidity('The two email addresses must match.');
```

# Traitement des données avec PHP

**Les données envoyées par un formulaire se retrouvent dans des superglobales.**

**Les superglobales sont des tableaux associatifs dont la portée est globale.**

**Pour récupérer les informations d'un formulaire, on peut utiliser l'une des superglobales suivantes :**

- ⦿ `$_GET[]` : données envoyées via l'URL
- ⦿ `$_POST[]` : données envoyées via l'entête HTTP
- ⦿ `$_FILES[]` : informations sur les fichiers envoyés par le visiteur.
- ⦿ `$_REQUEST[]` : fusion de `$_GET[]`, `$_POST[]` et `$_COOKIE[]`.  
En cas de conflits, `$_COOKIE[]` est prioritaire, puis `$_POST[]`,...

# Exemple de récupération d'une donnée

## Fichier form\_mail.html - création du formulaire

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

  <form action="form_mail.php" method="post">
    <p>
      Entrez votre e-mail :<br />
      <input type="text" name="mail" size="35" />
      <input type="submit" name="send_mail" value="Envoyer" />
    </p>
  </form>

</body>
</html>
```

Entrez votre e-mail :

Envoyer

# Exemple de récupération d'une donnée

## Fichier form\_mail.php - traitement des données

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
  if (isset($_POST['send_mail'])) {
    echo "<p>Votre mail est " . $_POST['mail'] . "</p>";
  } else {
    echo '<p>Utilisez la page <a href="./form_mail.html">form_mail.html</a>
    pour renseigner votre mail</p>';
  }
?>

</body>
</html>
```

# `$_REQUEST[]`, `$_POST[]` ou `$_GET[]` ?

## **`$_POST[]`**

- ⦿ si les données du formulaire sont transmises par la méthode post

## **`$_GET[]`**

- ⦿ si les données du formulaire sont transmises par la méthode get

## **`$_REQUEST[]`**

- ⦿ utilisable quelque soit la méthode de passage utilisée, à la condition que l'on soit certain qu'il n'y ait pas de collisions possibles ie deux variables de même nom.

Avantage : On peut modifier la façon de passer les données sans avoir à modifier le code de la page PHP qui les traite. Attention toutefois à ne pas abuser de son utilisation. Il est préférable de savoir d'où vient la données (entête HTTP, URI ou cookie).

# Traitement du texte

## Attention aux données provenant d'un champ texte

- ⦿ Il peut être nécessaire de les traiter (ex. pour produire une page W3C valide)
- ⦿ il est indispensable de les contrôler (ex. sécurité – contrôle d'injection de codes)

## Quelques fonctions utiles pour les traiter

- ⦿ Les fonctions addslashes() et stripslashes() permettent, respectivement, d'ajouter et de supprimer les backslashes ("\") devant les caractères le nécessitant (guillemets simples : ', guillemets doubles : ", backslash : \, valeur NULL)
- ⦿ [Obsolète depuis php 5.3] La directive PHP magic\_quotes\_gpc est activée par défaut, et elle appelle addslashes() sur toutes les données GET, POST et COOKIE.
- ⦿ La fonction get\_magic\_quotes\_gpc() retourne TRUE si la directive magic\_quotes est activée, FALSE sinon.
- ⦿ La fonction nl2br() transforme les changements de lignes (\n) en "<br/>"
- ⦿ La fonction htmlentities() transforme les caractères interprétables afin qu'ils soient affichés et non analysés.



## Exemple : form\_textarea.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

  <form action="form_textarea.php" method="post">
    <p>
      Entrez votre texte :<br />
      <textarea name="comment" cols="35" rows="6">
        Si j'écris comme ça en sautant des
        lignes et en utilisant des apostrophes
        la phrase b<a est vrai mais pas b>a,
        je peux avoir des problèmes.
      </textarea>
      <br /><input type="submit" name="send_comment" value="Envoyer" />
      <input type="reset" name="reset_comment" value="RaZ" />
    </p>
  </form>

</body>
</html>
```

Entrez votre texte :

Si j'écris comme ça en sautant des  
lignes et en utilisant des  
apostrophes  
la phrase b<a est vrai mais pas b>a,  
je peux avoir des problèmes.

Envoyer

RaZ

## Exemple : form\_textarea.php simpliste !

```
<?php
    if (!isset($_REQUEST['send_comment'])) header('location: form_textarea.html');
    else $texte=$_REQUEST['comment'];
?>

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
    <title>Page vide</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
    if (isset($texte)) echo "<p>".$texte."</p>";
?>

</body>
</html>
```

# Exemple : form\_textarea.php simpliste !

Le code précédent génère le code suivant :

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
5
6  <head>
7      <title>Page vide</title>
8      <meta http-equiv="content-type"
9          content="text/html; charset=utf-8" />
10     <meta http-equiv="Content-Style-Type" content="text/css" />
11 </head>
12
13 <body>
14
15 <p>Si j\'écris comme ça en sautant des
16 lignes et en utilisant des apostrophes
17 la phrase b<a est vrai mais pas b>a,
18 je peux avoir des problèmes.
19     </p>
20 </body>
21 </html>
22
```

Ce qui produit cet affichage dans le navigateur :

Si j'écis comme ça en sautant des lignes et en utilisant des apostrophes la phrase ba, je peux avoir des problèmes.

## Exemple : form\_textarea.php corrigé !

```
<?php
    if (!isset($_REQUEST['send_comment'])) header('location: form_textarea.html');
    else
    {
        $texte=$_REQUEST['comment'];
        $texte=htmlentities($texte, ENT_COMPAT, "UTF-8");
    }
?>

<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
    <title>Page vide</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
    if (isset($texte)) echo nl2br("<p>".$texte."</p>");
?>

</body>
</html>
```

# Exemple : form\_textarea.php corrigé !

Le code précédent génère le code suivant :

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
5
6  <head>
7      <title>Page vide</title>
8      <meta http-equiv="content-type"
9          content="text/html; charset=utf-8" />
10     <meta http-equiv="Content-Style-Type" content="text/css" />
11 </head>
12
13 <body>
14
15     <p>Si j'&eacute;cris comme &ccedil;a en sautant des <br />
16 lignes et en utilisant des apostrophes <br />
17 la phrase b<lt;a est vrai mais pas b<gt;a, <br />
18 je peux avoir des probl&egrave;mes.<br />
19     </p>
20 </body>
21 </html>
22
```

Si j'écris comme ça en sautant des lignes et en utilisant des apostrophes la phrase b<a est vrai mais pas b>a, je peux avoir des problèmes.

Ce qui produit cet affichage dans le navigateur

# Le cas des cases à cocher

## Cases à cocher – fichier HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

  <form action="form_case.php" method="get">
    <p>Recevoir les news par e-mail
      <input type="checkbox" name="case_offre_on" value="ok" checked="checked" />
      <br /><input type="submit" value="Envoyer" /><br />
    </p>
  </form>

</body>
</html>
```

# Le cas des cases à cocher

## Cases à cocher – le fichier php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">

<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
  if (isset($_REQUEST['case_offre_on'])) {
    echo "La case est cochée et sa valeur est : ".$_REQUEST['case_offre_on'];
  } else {
    echo "La case n'est pas cochée !";
  }
?>

</body>
</html>
```

# L'envoi de fichiers - \$\_FILES[]

Etant donné le formulaire d'envoi de fichier suivant :

```
<form action="upload.php" method="post" enctype="multipart/form-data">
  <p>
    <input type="file" name="fichier" size="40" />
    <input type="submit" value="UpLoader" />
  </p>
</form>
```

## \$\_FILES['fichier']

- ⊙ Tableau associatif qui permet d'accéder aux données concernant l'état du téléchargement.

\$_FILES['fichier']['la clé']	Description
\$_FILES['fichier'][name]	nom du fichier stocké sur la machine du client.
\$_FILES['fichier'][tmp_name]	nom du fichier stocké sur le serveur lorsqu'il a été reçu.
\$_FILES['fichier'][type]	type mime du fichier envoyé.
\$_FILES['fichier'][size]	taille du fichier (en octets).
\$_FILES['fichier'][error]	code d'erreur s'il y a eu un problème lors du transfert.



# L'envoi de fichiers - \$\_FILES[]

Etant donné le formulaire d'envoi de fichier suivant :

```
<form action="upload.php" method="post" enctype="multipart/form-data">
  <p>
    <input type="file" name="fichier" size="40" />
    <input type="submit" value="UpLoader" />
  </p>
</form>
```

**\$\_FILES['fichier']**

- ⊙ Tableau associatif qui permet d'accéder aux données concernant l'état du téléchargement.

\$_FILES['fichier']['la clé']	Description
\$_FILES['fichier'][name]	nom du fichier stocké sur la machine du client.
\$_FILES['fichier'][tmp_name]	nom du fichier stocké sur le serveur lorsqu'il a été reçu.
\$_FILES['fichier'][type]	type mime du fichier envoyé.
\$_FILES['fichier'][size]	taille du fichier (en octets).
\$_FILES['fichier'][error]	code d'erreur s'il y a eu un problème lors du transfert.

# L'envoi de fichiers - \$\_FILES[]

## Les messages d'erreurs

UPLOAD_ERR_OK	Valeur : 0. Aucune erreur, le téléchargement est correct.
UPLOAD_ERR_INI_SIZE	Valeur : 1. La taille du fichier téléchargé excède la valeur de <u>upload_max_filesize</u> , configurée dans le <i>php.ini</i> .
UPLOAD_ERR_FORM_SIZE	Valeur : 2. La taille du fichier téléchargé excède la valeur de <i>MAX_FILE_SIZE</i> , qui a été spécifiée dans le formulaire HTML. <code>&lt;input type="hidden" name="MAX_FILE_SIZE" value="X" /&gt;</code>
UPLOAD_ERR_PARTIAL	Valeur : 3. Le fichier n'a été que partiellement téléchargé.
UPLOAD_ERR_NO_FILE	Valeur : 4. Aucun fichier n'a été téléchargé.
UPLOAD_ERR_NO_TMP_DIR	Valeur : 6. Un dossier temporaire est manquant. Introduit en PHP 5.0.3.
UPLOAD_ERR_CANT_WRITE	Valeur : 7. Échec de l'écriture du fichier sur le disque. Introduit en PHP 5.1.0.
UPLOAD_ERR_EXTENSION	Valeur : 8. Une extension PHP a arrêté l'envoi de fichier. PHP ne propose aucun moyen de déterminer quelle extension est en cause. L'examen du <u>phpinfo()</u> peut aider

# Génération d'un formulaire

**PHP peut envoyer au navigateur des balises HTML et donc, à fortiori des balises de formulaires.**

**Il est donc bien évidemment possible de générer le formulaire dans le code PHP.**

# Génération d'un formulaire

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
  //On teste si l'on vient du formulaire ou s'il faut le créer
  if (isset($_REQUEST['send_mail']) && (!empty($_REQUEST['mail'])))
  { //Si on vient du formulaire
    //On traite les données
    echo "<p>Votre mail est ".$_REQUEST['mail']. "</p>";
  }
  else { //Sinon, génération de la page avec le formulaire
    echo '<form action="form_mail2.php" method="post">
      <p>Entrez votre e-mail :<br />
      <input type="text" name="mail" size="35" />
      <input type="submit" name="send_mail" value="Envoyer" /></p>
      </form>';
  }
?>

</body>
</html>
```

Ici on utilise qu'un seul fichier PHP pour afficher et traiter le formulaire.

## Avantage :

Cela permet par exemple de traiter les erreurs comme la saisie d'un mail non valide.

# Génération d'un formulaire

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Page vide</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
    //On teste si l'on vient du formulaire ou s'il faut le créer
    if (isset($_REQUEST['send_note']) && (!empty($_REQUEST['note'])))
    { //Si on vient du formulaire
        //On traite les données
        echo "<p>Vous avez donné la note " . $_REQUEST['note'] . "</p>";
    }
    else { //Sinon, génération de la page avec le formulaire
        echo '<form action="form_case2.php" method="get">
            <p>Choisissez une note :';

        for ($i=1; $i<10; $i++) echo '<input type="radio" name="note" value="'. $i . ' ">'. $i . ' ';
        echo '<input type="radio" name="note" value="'. $i . ' " checked="checked" />'. $i;

        echo '<br /><input type="submit" name="send_note" value="Envoyer" /></p></form>';
    }
?>

</body>
</html>
```

# Génération d'un formulaire

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Page vide</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

<?php
  //On teste si l'on vient du formulaire ou s'il faut le créer
  if (isset($_REQUEST['send_note']) && (!empty($_REQUEST['note'])))
  { //Si on vient du formulaire
    //On traite les données
    echo "<p>Vous avez donné la note ".$_REQUEST['note']."</p>";
  }
  else { //Sinon, génération de la page avec le formulaire
    echo '<form action="form_case2.php"
      <p>Choisissez une note :';

    for ($i=1; $i<10; $i++) echo '<input type="radio" name="note" value="'.$i.'" /> '.$i.' ';

    echo '<br /><input type="submit" name="send_note" value="Envoyer" /></p></form>';
  }
?>

</body>
</html>
```

Vous avez donné la note 3

Choisissez une note : ☐ 1 ☐ 2 ☒ 3 ☐ 4 ☐ 5 ☐ 6 ☐ 7 ☐ 8 ☐ 9 ☐ 10

Envoyer

# Génération d'un formulaire

```
function displayListesOfUsers() {
    $liste_items="";
    $sql = "select * from users";
    if ($link=getLinkBDD()) $result=mysql_query($sql,$link);

    while($user=mysql_fetch_assoc($result)) {
        $user_login=($user['webmaster']?"<strong>$user[login]</strong>":"$user[login]");
        $ctrl=md5($user['id_user']);
        $item=<<<ITEM
        $user_login - <em>$user[mail]</em><span class="floatright">
            <form class="floatleft" action="" method="post">
                <input type="hidden" name="id_u" value="$user[id_user]" />
                <input type="hidden" name="ctrl" value="$ctrl" />
                <button type="submit" class="edit small" value="user:edit2update" name="action" >Editer</button>
                <button type="submit" class="delete small" value="user:delete" name="action" >Supprimer</button>
            </form>
        </span>
    ITEM;

        $liste_items.='<li style="height:24px;">'. $item .'</li>';
    }

    $liste_users=<<<USERS
    <ul style="list-style: none;">
        $liste_items
        <li>
            <br/><em>Ajouter un utilisateur</em><span class="floatright"><form class="floatleft" action="" method="post">
                <button type="submit" class="add small" value="user:edit2add" name="action" >Ajouter</button>
            </form></span>
        </li>
    </ul>
    USERS;
    echo $liste_users;
}
```

# Génération d'un formulaire

**webmaster** - *brett.desbenoit@univ-provence.fr*

desbenoit - *brett.desbenoit@univ-amu.fr*






*Ajouter un utilisateur*



```
<ul style="list-style: none;">
  <li style="height:24px;">    <strong>webmaster</strong> - <em>brett.desbenoit@univ-provence.fr</em><span class="floatright">
    <form class="floatleft" action="" method="post">
      <input type="hidden" name="id_u" value="1" />
      <input type="hidden" name="ctrl" value="c4ca4238a0b923820dcc509a6f75849b" />
      <button type="submit" class="edit small" value="user:edit2update" name="action" >Editer</button>
      <button type="submit" class="delete small" value="user:delete" name="action" >Supprimer</button>
    </form>
  </span>    </li><li style="height:24px;">    desbenoit - <em>brett.desbenoit@univ-amu.fr</em><span class="floatright">
    <form class="floatleft" action="" method="post">
      <input type="hidden" name="id_u" value="2" />
      <input type="hidden" name="ctrl" value="c81e728d9d4c2f636f067f89cc14862c" />
      <button type="submit" class="edit small" value="user:edit2update" name="action" >Editer</button>
      <button type="submit" class="delete small" value="user:delete" name="action" >Supprimer</button>
    </form>
  </span>    </li>
  <li>
    <br/><em>Ajouter un utilisateur</em><span class="floatright"><form class="floatleft" action="" method="post">
      <button type="submit" class="add small" value="user:edit2add" name="action" >Ajouter</button>
    </form></span>
  </li>
</ul>
```



# Génération d'un formulaire

webmaster - brett.desbenoit@univ-provence.fr	 
desbenoit - brett.desbenoit@univ-amu.fr	 
Ajouter un utilisateur	

```
<ul style="list-style: none;">
  <li style="height:24px;">    <strong>webmaster</strong> - <em>brett.desbenoit@univ-provence.fr</em><span class="floatright">
    <form class="floatleft" action="" method="post">
      <input type="hidden" name="id_u" value="1" />
      <input type="hidden" name="ctrl" value="c4ca4238a0b923820dcc509a6f75849b" />
      <button type="submit" class="edit small" value="user:edit2update" name="action" >Editer</button>
      <button type="submit" class="delete small" value="user:delete" name="action" >Supprimer</button>
    </form>
  </span>    </li><li style="height:24px;">    desbenoit - <em>brett.desbenoit@univ-amu.fr</em><span class="floatright">
    <form class="floatleft" action="" method="post">
      <input type="hidden" name="id_u" value="2" />
      <input type="hidden" name="ctrl" value="c81e728d9d4c2f636f067f89cc14862c" />
      <button type="submit" class="edit small" value="user:edit2update" name="action" >Editer</button>
      <button type="submit" class="delete small" value="user:delete" name="action" >Supprimer</button>
    </form>
  </span>    </li>
  <li>
    <br/><em>Ajouter un utilisateur</em><span class="floatright"><form class="floatleft" action="" method="post">
      <button type="submit" class="add small" value="user:edit2add" name="action" >Ajouter</button>
    </form></span>
  </li>
</ul>
```

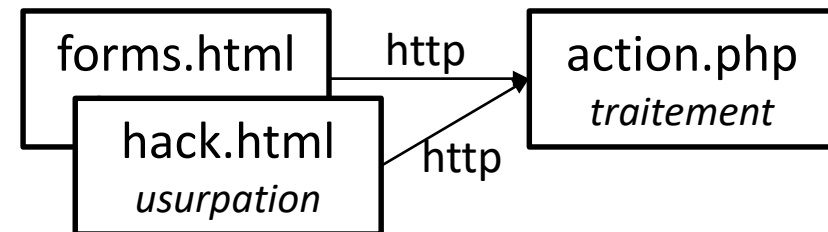
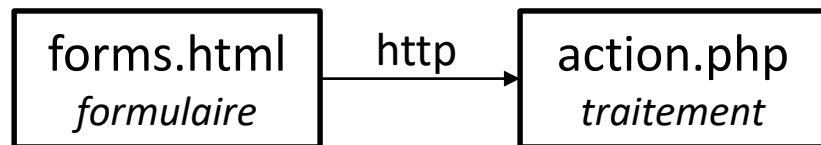
# Génération d'un formulaire

```
$action=(isset($_POST['action'])?$_POST['action']:"no-action");  
switch($action) {  
    case 'user:update':  
        ...  
  
        break;  
    case 'user:edit2update':  
        ...  
  
        break;  
    case 'user:edit2add':  
        ...  
  
        break;  
    case 'user:add':  
        ...  
  
        break;  
    case 'user:delete':  
        ...  
  
        break;  
    case 'no-action':  
        ...  
  
        break;  
}
```

# Un mot sur la sécurité

**Lorsque l'on traite les données d'un formulaire, il faut garder en têtes les éléments ci-dessous.**

A priori on ne sait pas d'où viennent les données que l'on traite. Même si la cible reste identique (attribut action), l'origine du formulaire peut être différente.



En outre, le formulaire a pu être modifié. Par exemple, un champ hidden peut facilement être modifié.

Les formulaires sont également sujet aux attaques XSS (cross-site scripting). Cette attaque consiste à injecter du code javascript dans un champ de formulaire.

Il est important de ne jamais faire confiance aux données de l'utilisateur (même avec une validation HTML5). Les données doivent être validées (côté client et côté serveur) et signées (par exemple via l'utilisation d'une fonction de hachage).

Transmission de données

# UTILISATION DES COOKIES

# Les Cookies

**Un cookie est un fichier texte dont le contenu est créée par le serveur.**

**C'est le serveur via la directive Set-cookie (dans l'en-tête de sa réponse) qui indique au client qu'il souhaite y stocker un cookie.**

**Si le client l'accepte alors le navigateur web va ensuite joindre le cookie à l'en-tête de toutes ses requêtes (vers le même domaine).**



# Fonctions PHP

## Manipulation des cookies en PHP

### Création/mise à jour d'un COOKIES

```
<?php
    setcookie("name", "value", time()+$int);
    /*name est le nom du cookies et value sa valeur, name[] pour définir plusieurs valeurs
    $int durée après laquelle le cookie expire (timestamp unix)*/
?>
```

### Récupération d'un Cookies : on utilise la super global \$\_COOKIE

```
<?php
    echo $_COOKIE["your cookie name"];
?>
```

### Suppression

```
<?php
    unset($_COOKIE["yourcookie"]);
    /*Ou*/
    setcookie("yourcookie", "yourvalue", time()-1);
    /*Il a expiré, il va être détruit*/
?>
```

# Limites des COOKIES

**Les Cookies sont stockés sur la machine du client et sont faciles à modifier on peut ajouter un hash pour les signer mais cela reste insuffisant**

- ◉ Via un éditeur de texte
- ◉ Via Javascript
- ◉ Etc.

**Ils transitent sans arrêt entre le client et le serveur.**

**Le client peut ne pas les accepter.**

Transmission de données

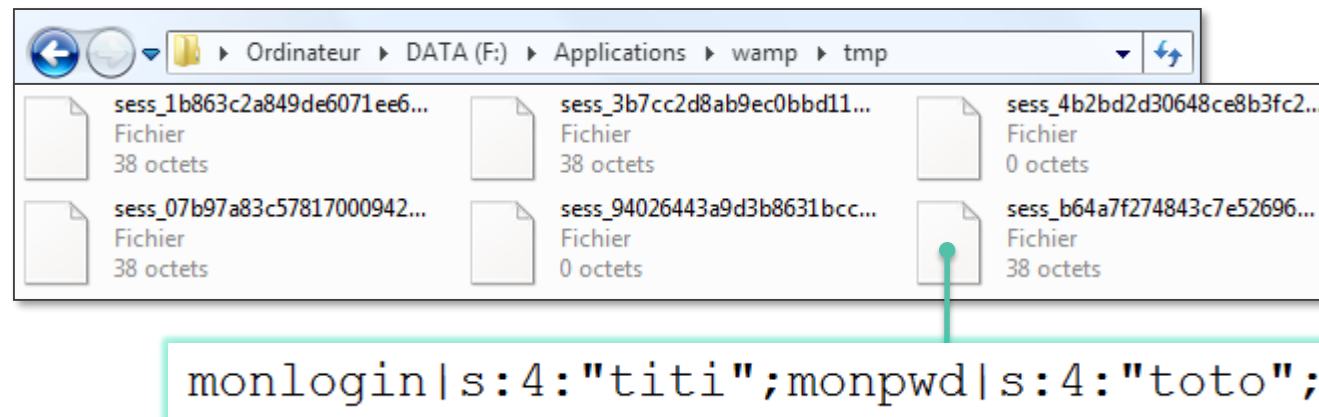
# UTILISATION DES SESSIONS



# Les sessions

## Principe

- Une session est personnelle à chaque visiteur et permet de stocker temporairement des variables.
- Les données de session sont stockées dans un fichier sur le serveur : on ne peut pas y avoir accès du côté client par le biais d'un script JavaScript.



Finalement, une session c'est un peu comme un cookie mais coté serveur.

# A quoi ça sert une session ?

**Pister un utilisateur pour établir des statistiques de navigation**

**Transmettre des données de page en page comme un panier**

**Stocker les données d'identification tels que le login et le mot de passe crypté**

# Fonctionnement d'une session

**Lorsque l'on démarre une session, le serveur génère un identifiant unique le PHPSESSID.**

**PHP se charge de transmettre cet identifiant d'une page à l'autre. Il est possible de spécifier (dans php.ini) comment l'identifiant sera transmit :**

- ⦿ `session.use_cookies` : l'identifiant est stocké dans un cookie
- ⦿ `session.use_only_cookies` : dans ce cas, php ignore les identifiants transmis via l'url et n'utilisera que ceux stockés dans les cookies
- ⦿ `session.use_trans_sid` : Permet d'indiquer à PHP d'insérer l'identifiant dans tout les liens relatifs
- ⦿ `session.rewrite_tags` : Indique où et dans quelles balises HTML insérer l'identifiant de session

# Démarrer une session

## Automatiquement

- Si l'option `session.auto_start` est à 1.

## Explicitement

- en utilisant l'instruction `session_start()` qui permet de créer une nouvelle session, ou restaurer une session existante.



Si les données sont transmises par le biais de cookies, `session_start()` doit être appelée avant d'afficher quoi que ce soit !!!

Attention aux pages qui commencent par une ligne vide.

# Durée de vie d'une session

## La durée de vie d'une session dépend de trois paramètres :

- ⦿ La fréquence à laquelle va être vérifié sa validité

```
; Define the probability that the 'garbage collection' process is started  
; on every session initialization.  
; The probability is calculated by using gc_probability/gc_divisor,  
; e.g. 1/100 means there is a 1% chance that the GC process starts  
; on each request.  
session.gc_probability = 1  
session.gc_divisor     = 100
```

- ⦿ Sa durée maximum d'existence

```
; After this number of seconds, stored data will be seen as 'garbage' and  
; cleaned up by the garbage collection process.  
session.gc_maxlifetime = 1440
```

**Au delà de ce temps, les données de la session seront considérées comme périmées.**

# Manipulation de données dans une session

**Les données d'une session sont stockée dans la super globale `$_SESSION[...]`**

**Pour ajouter, tester l'existence ou supprimer une donnée dans une session, on procède donc de la même manière qu'avec un tableau associatif**

```
<?php
    session_start();                                //Création/Récupération de la session
    $_SESSION['nom'] = $_REQUEST['nom'];             //Création de la variable "nom" dans la session

    if (!isset($_SESSION['count'])) {               //Test d'existence de la variable "count"
        $_SESSION['count'] = 0;                     //Création de la variable "count"
    } else $_SESSION['count']++;                     //Incrémentation de la variable "count"
?>
```

```
<?php
    session_start();                                //Récupération de la session et destruction de la
    unset($_SESSION['nom']);                         //variable "nom" : on ne peut donc plus y accéder
?>
```

# Détruire une session

## Vider le contenu de la session

- la session est toujours accessible mais elle est vide.

```
<?php
    $_SESSION = array(); //$_SESSION est désormais un tableau vide.
?>
```

## Détruire la session

- la session n'est plus accessible.

```
<?php //session_destroy() retourne true si la session est détruite, false sinon.
    if (session_destroy()) echo 'Session détruite !';
    else echo 'Erreur : impossible de détruire la session !';
?>
```

## Régénérer l'identifiant de session

- L'identifiant est modifié mais les données ne sont pas effacées.

```
<?php
    session_regenerate_id(); //Regenère un id de session sans la détruire
?>
```

# Ouvrir plusieurs sessions

**On ne peut pas ouvrir plusieurs sessions simultanément mais on peut les ouvrir les unes après les autres.**

- Sélectionner la session

```
<?php
    session_name('utilisateur');
?>
```

- Fermer une session et sauvegarde ses données

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

## Exemple

```
<?php
    session_name('utilisateur');
    session_start(); // Création de la première session
    [...] // Utilisation de la première session
    session_write_close(); // Fermeture de la première session, ses données sont sauvegardées.
    session_name('admin'); // Indication du nom de la seconde session
    session_start(); // Ouverture de la seconde session
    [...] // Utilisation de la seconde session.
?>
```



# Sessions et sécurité

## Plusieurs directives de configuration permettent de jouer sur la sécurité des sessions

### **session.referer\_check**

Contient une sous-chaîne que vous souhaitez retrouver dans tous les en-têtes HTTP Referer. Si cet en-tête a été envoyé par le client et que la sous-chaîne n'a pas été trouvée, l'identifiant de session sera considéré comme invalide. Par défaut, cette option est une chaîne vide.

### **session.hash\_function**

Permet de spécifier la fonction de hachage à utiliser pour générer les identifiants de session. '0' signifie MD5 (128 bits) et '1' signifie SHA-1 (160 bits).

Depuis PHP 5.3.0, il est également possible de spécifier n'importe quel algorithme fourni par l'extension hash (s'il est disponible), comme sha512 ou whirlpool.

Une liste complète d'algorithmes peut être obtenue avec la fonction `hash_algos()`.

### **session.entropy\_file**

Est un chemin jusqu'à une source externe (un fichier), qui sera utilisée comme source additionnelle d'entropie pour la création de l'identifiant de session. Des exemples valides sont `/dev/random` et `/dev/urandom`, qui sont disponibles sur tous les systèmes Unix. Cette fonctionnalité est supportée sous Windows depuis PHP 5.3.3. Le fait de définir `session.entropy_length` à une valeur différente de zéro fera que PHP utilisera l'API aléatoire de Windows comme source d'entropie.

### **session.save\_handler**

Définit le nom du gestionnaire de session qui est utilisé pour stocker et relire les données. Par défaut, c'est le système intégré par fichiers : `files`. Noter que les extensions individuelles doivent enregistrer leurs propres gestionnaires de session. Voir aussi `session_set_save_handler()`.

# Sessions et sécurité

## Exemple d'utilisation de session\_set\_save\_handler() : stockage de la session dans une BDD

```
1 //Ouvre la session
2 function ouvrir_session($chemin_de_stockage, $nom_de_session) {
3     $_ENV['nom_de_session'] = $nom_de_session;
4     return true;
5 }
6
7 //Ferme la session
8 function fermer_session() {
9     return true; // Rien à faire
10 }
11
12 //Renvoie les données de sessions stockées en BDD
13 function lire_session($identifiant_de_session) {
14     //connection à la BDD
15     //SELECT <donnees> FROM SESSIONS_TABLE WHERE identifiant = $identifiant_de_session AND nom = $_ENV['nom_de_session']
16     //retourne le résultat de la requête
17 }
18
19 //Met à jour les données de session en BDD
20 function ecrire_session($identifiant_de_session, $donnees_de_session) {
21     //Connection à la BDD
22     //UPDATE SESSIONS_TABLE SET donnees = $donnees_de_session WHERE identifiant = $identifiant_de_session AND nom = $_ENV['nom_de_session']
23     //retourne l'etat de l'action
24 }
25
26 //Ferme la session
```

Transmission de données

# **EXEMPLE D'APPLICATION, CRÉATION D'UN ESPACE MEMBRE**

# Préambule

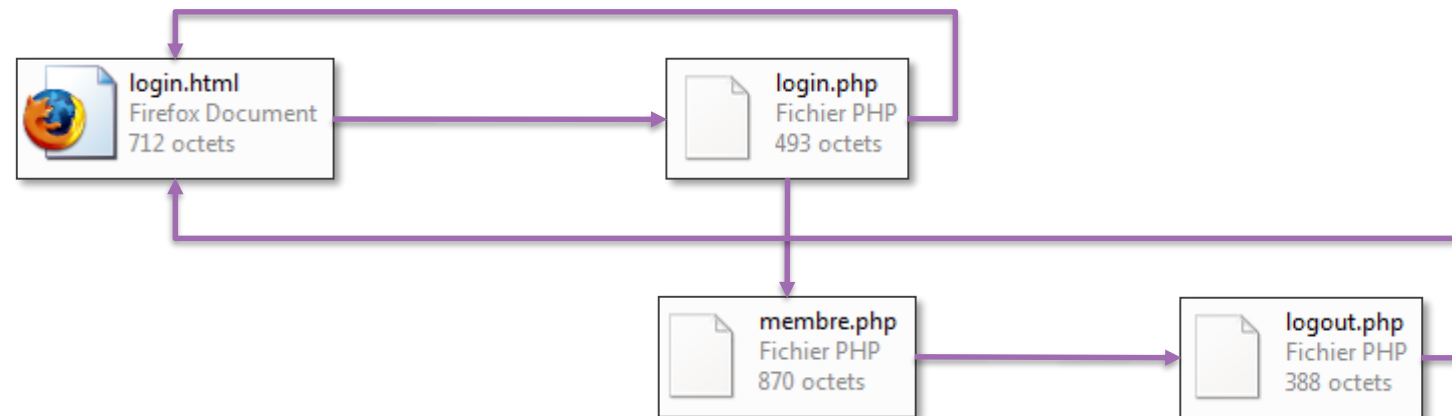
**Les exemples qui suivent sont là pour expliquer comment fonctionne les sessions et donne des pistes d'implantation fonction de l'état actuel du cours et de vos connaissances.**

**Nous verrons par la suite comment faire BEAUCOUP mieux !**

# Création d'un espace membre - Solution 1

**Dans cet exemple, on utilise une session pour stocker les informations relatives à l'identification d'un utilisateur et les transmettre de page en page.**

**L'architecture est la suivante :**



# Création d'un espace membre - Solution 1

## Login.html

- Création du formulaire pour se connecter.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Page login</title>
  <meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
  <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>
  <form action="login.php" method="post">
    <p>Login : <input type="text" name="login" size="35" />
    <br />Mot de passe : <input type="password" name="pwd" size="35" />
    <br /><input type="submit" name="from_log" value="Connexion" /></p>
  </form>

</body>
</html>
```

# Création d'un espace membre - Solution 1

## Login.php

- ⦿ Traitement des données issues du formulaire de connexion.
- ⦿ Si l'utilisateur est valide, on le redirige vers la page membre, sinon, on le redirige vers la page de login.

```
<?php
$login_valide = "titi";
$pwd_valide = "toto";

if (($_REQUEST['login']==$login_valide) && ($_REQUEST['pwd']==$pwd_valide ))
{
    //On démarre la session
    session_start();
    $_SESSION['monlogin'] = $_REQUEST['login'];
    $_SESSION['monpwd'] = $_REQUEST['pwd'];

    //on redirige le visiteur vers une page membre
    //echo '<a href="membre.php">espace membre</a>';
    header('location: membre.php');
} else {
    header('location: login.html');
}
?>
```

# Création d'un espace membre - Solution 1

```
<?php
//On utilise session_start pour démarrer la session
//ou récupérer une session existante
session_start();

//S'il n'y a rien dans la session,
//c'est que l'on ne vient pas de login
if (!isset($_SESSION['monlogin'])) header("location: ./login.html");
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Page des membres</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

    <?php
        // On récupère les variables de session
        echo "<p>Bonjour ".$_SESSION['monlogin']. " !<br/>";

        // On affiche un lien pour fermer notre session
        echo '<a href="./logout.php">Déconnexion</a></p>';
    ?>

</body>
</html>
```

## *Membre.php*

On essaye de récupérer une session existante

Si elle n'existe pas, on redirige vers la page de login

Sinon, on affiche les données et un lien pour se déconnecter i.e. pour supprimer la session.



# Création d'un espace membre - Solution 1

## Logout.php

- ⦿ On récupère la session ;
- ⦿ Et la vide des données qu'elle contient.
- ⦿ Enfin, on détruit la session et redirige vers la page de connexion.

```
<?php
//On démarre la session
session_start();

//On détruit toute les variables de notre session
//Rq : cette fonction n'est pas officiellement dépréciée mais
//on lui préférera $_SESSION[]=array();
session_unset();

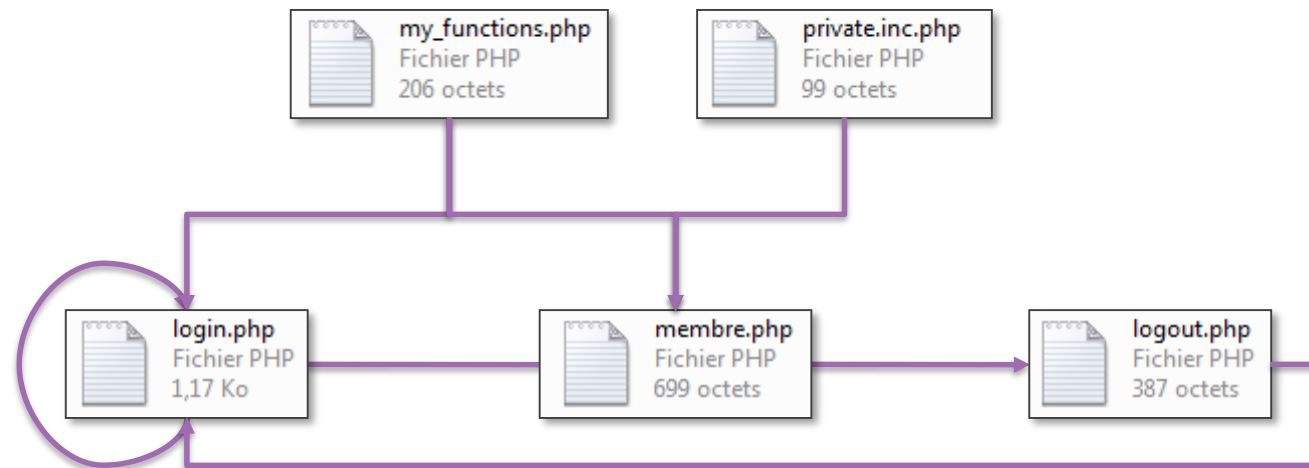
//On détruit notre session
session_destroy();

//On redirige le visiteur vers la page d'accueil
header('location: login.html');
?>
```

## Création d'un espace membre - Solution 2

**Dans cet exemple, on utilise une session pour stocker les informations relatives à l'identification d'un utilisateur et un fichier pour gérer l'authentification**

**L'architecture est la suivante :**



# Création d'un espace membre - Solution 2

## Fichiers de fonctions et d'authentification

### my\_functions.php

- ⦿ Ce fichier contient les fonctions utiles pour l'authentification.

```
<?php
function isValidUser($form_login, $form_pwd)
{
    $login_valide = "titi";
    $pwd_valide = "toto";

    return (($form_login=== $login_valide) && ($form_pwd=== $pwd_valide));
}
?>
```

### private.inc.php

- ⦿ Si ce fichier est inclut dans une page, alors on est certain que seul un utilisateur provenant de login et authentifié peut continuer.

```
<?php
session_start();
if (!isset($_SESSION['id_user'])) header("location: login.php");
?>
```

# Création d'un espace membre - Solution 2

```
<?php
require_once "my_functions.php";

$login="";

if (isset($_REQUEST['from_log']))
{
    $login=$_REQUEST['login'];
    if (isValidUser($_REQUEST['login'],$_REQUEST['pwd']))
    {
        session_start();
        $_SESSION['id_user'] = $_REQUEST['login'];

        //on redirige le visiteur vers une page membre
        header('location: membre.php');
    } else $error_log = "Erreur d'authentification !";
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Page login</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>
    <form action="" method="post">
        <p><?php if (isset($error_log)) echo '<span style="color:red;">'.$error_log.'</span>';?>
        <br />Login : <input type="text" name="login" value="<?php echo $login; ?>" size="35" />
        <br />Mot de passe : <input type="password" name="pwd" size="35" />
        <br /><input type="submit" name="from_log" value="Connexion" /></p>
    </form>

</body>
</html>
```

## Login.php

Traitement des données du formulaire si l'on en vient

Si l'utilisateur est valide on démarre une session et on y stock son id puis on redirige vers la page membre.

Sinon on réaffiche le formulaire avec un message d'erreur.

Sinon affichage du formulaire de login.

# Création d'un espace membre - Solution 2

```
<?php
    require_once "private.inc.php";
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Page des membres</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

    <?php
        // On récupère les variables de session
        echo "<p>Votre login est : ".$_SESSION['id_user']. "<br/>";

        // On affiche un lien pour fermer notre session
        echo '<a href="./logout.php">Déconnexion</a></p>';
    ?>

</body>
</html>
```

## Membre.php

On utilise le fichier private.inc.php qui vérifie que l'utilisateur est bien un utilisateur authentifié.

Si c'est bien le cas on affiche les données et un lien pour se déconnecter i.e. pour supprimer la session.

# Création d'un espace membre - Solution 2

## Logout.php

- ⦿ On récupère la session;
- ⦿ Et la vide des données qu'elle contient.
- ⦿ Enfin, on détruit la session et redirige vers la page de connexion.

```
<?php
//On démarre la session
session_start();

//On détruit toute les variables de notre session
//Rq : cette fonction n'est pas officiellement dépréciée mais
//on lui préférera $_SESSION[]=array();
session_unset();

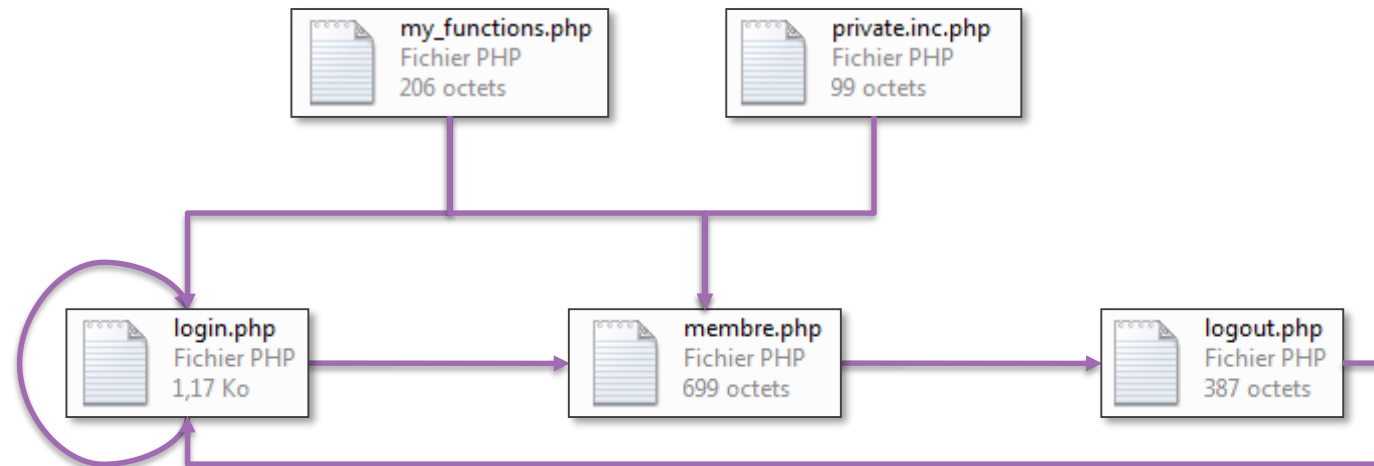
//On détruit notre session
session_destroy();

//On redirige le visiteur vers la page d'accueil
header('location: login.html');
?>
```

# Création d'un espace membre - Solution 3

**Dans cet exemple, authentification et session sont dissociés.**

**L'architecture reste cependant la même que précédemment :**



# Création d'un espace membre - Solution 3

```
<?php
session_start();
if (isset($_SESSION[nb_tentative])) $_SESSION[nb_tentative]++;
else $_SESSION[nb_tentative]=1;

require_once "my_functions.php";
$login="";

if (isset($_REQUEST['from_log']))
{
    $login=$_REQUEST['login'];
    if (isValidUser($_REQUEST['login'],$_REQUEST['pwd']))
    {
        session_regenerate_id();
        unset($_SESSION[nb_tentative]);
        $_SESSION['id_user'] = $_REQUEST['login'];

        //on redirige le visiteur vers une page membre
        header('location: membre.php');
    } else $error_log = "Erreur d'authentification !";
}
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<title>Page login</title>
<meta http-equiv="content-type"
    content="text/html; charset=utf-8" />
<meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>
<?php
$form='<form action="" method="post">';
$form.= '<span style="color:red;"> Tentative: '.$_SESSION['nb_tentative'].' /3</span>';
$form.= '<br />Login : <input type="text" name="login" value="'.$login.'" size="35" />';
$form.= '<br />Mot de passe : <input type="password" name="pwd" size="35" />';
$form.= '<br /><input type="submit" name="from_log" value="Connexion" /></p>';
$form.= '</form>';

if ($_SESSION[nb_tentative]>3) echo "<p>TROP DE TENTAVIES VEUILLEZ RE-ESSAYER PLUS TARD </p>";
else echo $form;
?>
</body>
</html>
```

## Login.php

Contrairement aux solutions 1&2 dans cette version la session n'est plus directement liée à un utilisateur authentifié.

On l'utilise également pour stocker par exemple le nombre de tentatives de connexion et avoir une réponse appropriée.



# Création d'un espace membre - Solution 3

```
<?php
    require_once "private.inc.php";
?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <title>Page des membres</title>
    <meta http-equiv="content-type"
        content="text/html; charset=utf-8" />
    <meta http-equiv="Content-Style-Type" content="text/css" />
</head>

<body>

    <?php
        // On récupère les variables de session
        echo "<p>Votre login est : ".$_SESSION['id_user']. "<br/>";

        // On affiche un lien pour fermer notre session
        echo '<a href="./logout.php">Déconnexion</a></p>';
    ?>

</body>
</html>
```

## Membre.php

On utilise le fichier private.inc.php qui vérifie que l'utilisateur est bien un utilisateur logué.

Si c'est bien le cas on affiche les données et un lien pour ce déconnecter i.e. pour supprimer la session.

# Création d'un espace membre - Solution 3

## Logout.php

- ⦿ On récupère la session;
- ⦿ Et la vide des données qu'elle contient.
- ⦿ Enfin, on détruit la session et redirige vers la page de connexion.

```
<?php
//On démarre la session
session_start();

//On détruit toute les variables de notre session
//Rq : cette fonction n'est pas officiellement dépréciée mais
//on lui préférera $_SESSION[]=array();
session_unset();

//On détruit notre session
session_destroy();

//On redirige le visiteur vers la page d'accueil
header('location: login.html');
?>
```

## Quelques notes sur ces exemples !

**Il ne s'agit que d'un exemple visant à montrer comment fonctionne les sessions. Je vous déconseille fortement de créer une page de membre en stockant le login et le mot de passe dans la session et en clair !!!**



### Oui mais alors ? Comment qu'on fait ?

- ⦿ Tout d'abord, les données des utilisateurs devraient être stockées dans une base de données.
- ⦿ Le mot de passe, devrait être crypté par un algorithme de hachage tels que MD5 ou SHA1 (A préférer) et ce, côté serveur (php) et côté client (javascript). Vous pouvez également utiliser la technique du grain de sel (salt) pour vous protéger des dictionnaires de mots de passe.
- ⦿ Il est également possible de créer une table session dans la base de données pour associer à l'utilisateur son identifiant de session.
- ⦿ ...