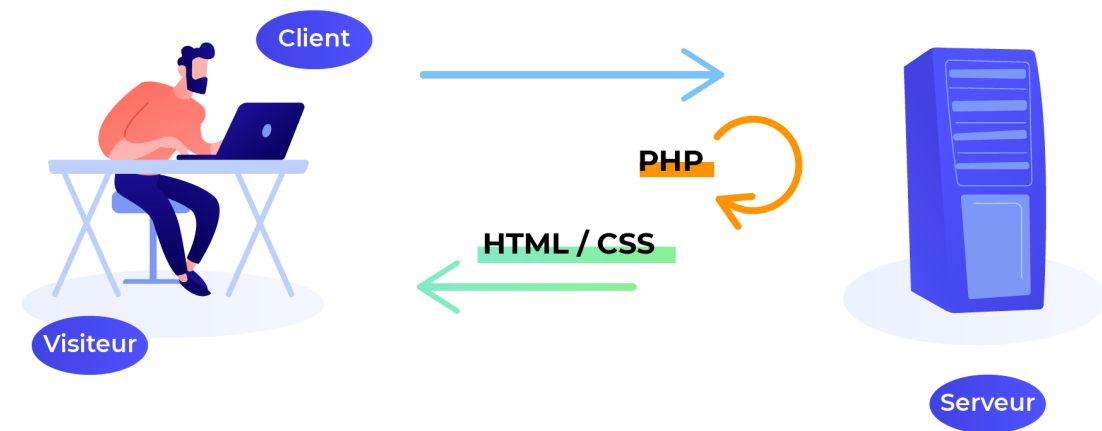


Cours Programmation web en PHP

Semestre 2



Plan

Partie 2: Les fichiers et les formulaires

- Les fonctions de gestion de fichiers
- Parcours et copie d'un fichier
- Les formulaires
- Structure de la requête et réponse http
- Récupération des données

Gestion des fichiers en php

- PHP vous permet d'accéder au système de gestion de fichier au niveau du serveur pour réaliser des opérations sur les fichiers.
- Les opérations sur les fichiers concernent **la création, l'ouverture, la suppression, la copie, la lecture et l'écriture** de fichiers.
- Ces opérations sont régies par des droits et varient selon le mode d'accès au fichier.
- Il existe plusieurs fonctions prédéfinies vous permettant de manipuler facilement et efficacement les fichiers en PHP.

Procédure d'utilisation des fichiers

1-Ouverture du
fichier

2-Lire le contenu
du fichier

3-Effectuer les
traitements
nécessaires

4- Fermer le fichier

Les fichiers - quelques fonctions

- **`$fp= fopen($file,$mode)`** : ouverture du fichier **`$file`**
- **`fclose($fp)`** : ferme le fichier **`$fp`**.
- **`fgets($fp, $length)`** : lit une ligne de **`$length`** caractères
- **`fputs($fp, $str)`** : écrit la chaîne **`$str`** dans le fichier
- **`fgetc($fp)`** : lit un caractère.
- **`fseek ($fp, p)`** : modifie la position du pointeur vers la position **`p`**
- **`feof($fp)`** : teste la fin du fichier.

Les fichiers - quelques fonctions

- **file_exists(\$file)** : indique si le fichier **\$file** existe.
- **filesize(\$file)** : retourne la taille du fichier **\$file**.
- **filetype(\$file)** : retourne le type du fichier **\$file**.
- **unlink(\$file)** : détruit le fichier **\$file**.
- **is_dir()** : permet de savoir si le fichier dont le nom est passé en paramètre correspond à un répertoire
- **is_executable()** : permet de savoir si le fichier dont le nom est passé en paramètre est exécutable.

Fichiers : *fopen*

r : ouvre en lecture avec le pointeur de fichier au début

r+ : ouvre en lecture et écriture avec le pointeur au début

w : ouvre en écriture seule; place le pointeur au début du fichier (Si le fichier n'existe pas, elle tente de le créer).

w+ : ouvre en lecture et écriture; place le pointeur de fichier au début du fichier (Si le fichier n'existe pas, elle tente de le créer).

Fichiers : *fopen*

a : ouvre en écriture seule; le pointeur de fichier à la fin du fichier file. (Si le fichier n'existe pas, elle tente de le créer).

a+ : ouvre en lecture et écriture; place le pointeur de fichier à la fin du fichier. (Si le fichier n'existe pas, on tente de le créer).

Parcours d'un fichier

```
<?php
```

```
$s="";
```

```
$fp = fopen ("fichier.txt", "r+");
```

```
fseek ($fp, 0);
```

```
while (!feof($fp))
```

```
{
```

```
$s=$s.fgetc($fp);
```

```
}
```

```
fclose ($fp);
```

```
echo $s;
```

```
?>
```

Ouvre un fichier ou une URL

Mode d'ouverture

Modifie la position du pointeur de fichier

Lit un caractère dans un fichier

Teste la fin du fichier

Ferme un fichier

Parcours d'un fichier

```
<!DOCTYPE html>
<html>
<body>

<?php
$file = fopen("test.txt","r");
//Afficher les lignes d'un fichier tant que nous n'avons pas atteint la fin du fichier
while(! feof($file)) {
    $line = fgets($file);
    //Récupère la ligne courante à partir de l'emplacement du pointeur sur le fichier
    echo $line. "<br>";
}
fclose($file);
?>

</body>
</html>
```

Fichiers : Existence d'un fichier

```
<?php
$fp="fichier.txt";
if (file_exists($fp)) {
echo " Le fichier ".$fp." existe "; echo"<br>";
echo "sa taille est ".filesize($fp);echo"<br>";
echo "son type est ".filetype($fp);
}
else
echo "Le fichier $fp n'existe pas sur la machine";
?>
```

Fichiers : Ajout chaine

<?php

```
$fp = fopen ("fichier.txt", "r+");
```

```
$chaine = fgets ($fp, 11);
```

```
$chaine= $chaine."bonjour";
```

```
fseek ($fp, 0); // se positionner à la position 0
```

```
fputs ($fp,$chaine);
```

← Ecrire la chaine dans le fichier

```
fclose ($fp);
```

```
echo $chaine;
```

?>

Fonctions des fichiers

- **copy(\$source, \$dest)** : copie le fichier **\$source** vers **\$dest**.
- **readfile(\$file)** : affiche le fichier **\$file**.
- **rename(\$old, \$new)** : renomme le fichier **\$old** en **\$new**.

Copie de Fichiers

```
<?php
```

```
$fp="fichier.txt";
```

```
$fp1="fichier1.txt";
```

```
unlink($fp1);
```

```
copy($fp,$fp1);
```

```
$s=readfile("fichier1.txt");
```

```
echo $s;
```

```
rename($fp1, "fichier02.txt");
```

```
?>
```

Détruire le fichier

Copier \$fp dans fp1 et le créer s'il n'existe pas

lit le fichier et le met dans \$s

Renommer le fichier fichier 1 en fichier 2

Formulaires

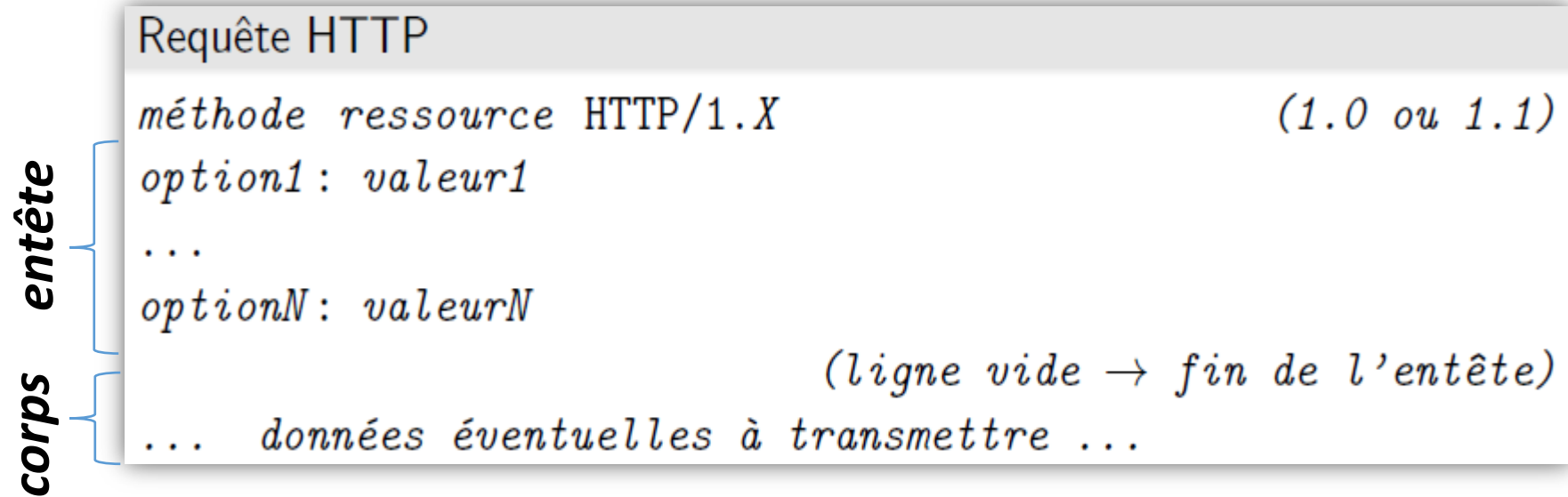
- Les applications dynamiques sont basées sur des échanges entre le client et le serveur.
- Ces échanges sont appelés des requêtes http.
- Une requête du client peut être effectuée par les manières suivantes:
 - Ecrire directement l'adresse de la page demandée dans la zone URL du navigateur.
 - Cliquer sur un lien hypertexte de la page courante.
 - Envoyer un formulaire de la page courante (Bouton submit).
 - Provoquée par le rafraichissement de la page courante.

Formulaires

La requête contient des informations envoyées par le client. Ces informations peuvent être:

- Des données envoyées par une méthode spécifique (GET/POST);
- Un ensemble d'informations sur sa machine (adresse IP, langue, type de navigateur....)
- Le client peut envoyer également au serveur des données enregistrées préalablement, par le serveur, dans le poste du client sous forme de COOKIES

Structure de la requête client



Structure de la requête client

Transmission des données avec la méthode http GET ou POST

- La méthode GET place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.

<http://www.esi.ac.ma/inscription.php?champ=valeur&champ2=valeur>

- **inconvénients :**

- Rendre visibles les données dans la barre d'adresse du navigateur.
 - La longueur totale est limitée à 2048 caractères, ce qui rend impossible la transmission d'un volume de données important.
 - L'application est sujette à des injections SQL.
- La méthode POST regroupe les informations dans l'entête d'une requête HTTP
 - Assure une confidentialité efficace des données

Structure de la requête client

Exemple d'une requête POST : HTTP

```
POST /login HTTP/1.1
HOST : www.domaine.net
ACCEPT : application/json
Content-type : application/x-www-form-urlencoded
Cookie : JSESSIONID : C4512152018CFDDOP345677
```

// Saut de ligne

```
username=admin & password=1234 & action=login
```

Entête de la requête

Corps de la requête

Structure de la réponse serveur

Réponse HTTP

```
HTTP/1.X statut description (1.0 ou 1.1)
option1: valeur1
...
optionN: valeurN
(ligne vide → fin de l'entête)
... données à transmettre ... (contenu utile)
```

Statut /Description:

- 1XX : information
- 2XX : succès (200 OK, 201 Created . . .)
- 3XX : redirection (301 Moved Permanently . . .)
- 4XX : erreur du client (403 Forbidden, 404 Not Found . . .)
- 5XX : erreur du serveur (501 Not Implemented . . .)

Structure de la réponse serveur

Protocole : HTTP

Exemple d'une réponse : HTTP

HTTP/1.1 200 OK

Date : Wed, 05Feb02 15:02:01 GMT

Server : Apache/1.3.24

Last-Modified : Wed, 02Oct01 24:02:01 GMT

Content-type : application/json

Content-length : **77**

SET-Cookie : JSESSIONID : C4512152018CFDDOP345677

// Saut de ligne

```
[  
  { "id" : 1 , "TaskName" : "T1"}, { "id" : 2 , "TaskName" : "T2"},  
  { "id" : 3 , "TaskName" : "T3"}  
]
```

Entête de la réponse

Corps de la Réponse

Exemple de la requête GET avec réponse

Le client veut consulter `http://www.ietf.org/rfc.html`

Il se connecte à `www.ietf.org:80` et envoie la requête :

`GET /rfc.html HTTP/1.1`

`Host: www.ietf.org:80`

`Connection: close` *(fermer après la réponse)*

(ligne vide → fin de l'entête)

Le serveur accepte la connexion, analyse la requête et répond :

`HTTP/1.0 200 OK`

`Server: Apache/2.2.4 (Linux/SUSE) ...`

`...`

(autres options non indiquées ici)

`Content-Length: 13671`

`Content-Type: text/html`

`Connection: close`

(ligne vide → fin de l'entête)

`<HTML> ... </HTML>`

(contenu de la ressource)

(le serveur ferme la connexion)

Exemple de la requête POST

Méthode souvent utilisée pour les formulaires

```
POST /cgi-bin/login.cgi HTTP/1.1
```

```
Host: localhost:80
```

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

```
Content-Length: 28
```

```
Connection: close
```

(ligne vide → fin de l'entête)

```
login=myname&password=mypass
```

(données POST)

- L'avantage de cette méthode par rapport à GET, c'est qu'on peut envoyer autant de caractères que l'on souhaite.
- En plus les données sont encapsulées dans le corps de la requête http. Ce qui fait que les données envoyées ne sont pas visibles au niveau de l'URL du navigateur.

Formulaires (Rappel HTML)

- Les formulaires introduits dans le HTML depuis ses plus anciennes versions sont l'élément essentiel qui permet l'interactivité entre un site et ses visiteurs. Ils constituent pour cette raison la base de la création de sites web dynamiques.
- Tout échange entre le visiteur et le serveur passe par un formulaire, dans lequel l'utilisateur peut :
 - saisir les textes ou les mots de passe;
 - opérer des choix grâce à des boutons radio, des cases à cocher ou des listes de sélection;
 - envoyer ses propres fichiers depuis le poste client;
 - ...

Formulaires (Rappel HTML)

```
<form method="GET | POST" action="page.php">
<input type="text" name="Champ_saisie" value="Texte"/>
<select name="Liste_Choix" size="3">
  <option value="Option_1">Option_1</option>
  <option value="Option_2">Option_2</option>
  <option value="Option_3">Option_3</option>
</select>
<textarea name="Zone_Texte" cols="30" rows="5"> Texte par défaut
</textarea>
<input type="checkbox" name="Case_Cocher[]" value="Case_1"> Case 1<br>
<input type="checkbox" name="Case_Cocher[]" value="Case_2"> Case 2<br>
<input type="checkbox" name="Case_Cocher[]" value="Case_3"> Case 3<br>
<input type="radio" name="Case_Radio" value="Case radio 1"> radio 1<br>
<input type="radio" name="Case_Radio" value="Case radio 2"> radio 2<br>
<input type="radio" name="Case_Radio" value="Case radio 3"> radio 3<br>
<input type="reset" name="Annulation" value="Annuler">
<input type="submit" name="Valider" value="Valider">
</form>
```

Attributs d'un formulaire




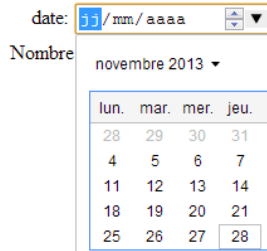
- **Method**: la méthode d'envoi des données vers le serveur (POST ou GET)
- **Action**: le fichier qui va traiter les informations saisies. L'attribut peut aussi avoir la valeur "**mailto:**", qui provoque l'envoi des données vers une adresse e-mail, qu'il faut préciser à la suite du mot **mailto** en écrivant, par exemple :
action="mailto:nom@esi.ac.ma"
- **Name** = "chaîne_de_caractères" : nom du formulaire (optionnel)

Structure d'un formulaire: `<form>` `</form>`


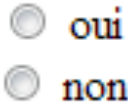
- La balise `<input/>` permet de créer les composants classiques des formulaires et dont les aspects et les rôles sont très différents.
- La différenciation de ces composants s'effectue simplement en définissant la valeur de leurs attributs, et notamment de l'attribut ***type*** (*obligatoire*).
- On trouve des éléments de ***type*** "text", "password", "checkbox", "radio", "submit", "reset", "file" et "hidden".
- L'attribut ***name*** est obligatoire, car c'est lui qui permet d'identifier les champs côté serveur et ainsi de récupérer leur contenu.
- L'attribut ***value*** (facultatif) : c'est ce que contient la zone de texte au départ. Par défaut, la zone de texte est vide mais il peut être pratique de pré-remplir le champ. Exemple : `<input type="text" name="Ecole" value="ESI" />`.

Structure d'un formulaire: `<form>` `</form>`

- HTML permet de créer des champs de type:

Type bouton	
<code><input type= "button" value = "Clic " name= "btnClic " /></code>	
Type checkbox : cases à cocher	
<code><input type ="checkbox" value="choix1" name= "choix[]"/></code>	
<code><input type ="checkbox" value="choix1" name= "choix[]"/></code> <code><label >Texte</label></code> Si vous voulez que la case soit cochée par défaut: <code><input type="checkbox" name="choix1" checked="checked" /></code>	 Texte
Type Date	
<code><input type="date" name= "date " /></code>	
<code><input type="date" value="2013-11-28" name= "date " /></code>	

Structure d'un formulaire: `<form> </form>`

Type File: Permet le transfert de fichiers du poste client vers le serveur. L'attribut accept : définit le ou les types de fichier acceptés en transfert	
<code><input type= "file" accept="image/gif,image/jpeg" name= "file " /></code>	
Type image	
<code><input type ="image" src="exemple.png" /></code>	
Type text: champ de saisie de texte d'une seule ligne. Value: valeur par défaut	
<code><input type="text" value="exemple" /></code>	
Type password: champ de saisie de mot de passe	
<code><input type="password" value="password" name= "pass " /></code>	
Type radio. L'attribut checked : la valeur cochée par défaut	
<code><input name="radGroupe" type="radio" value ="oui"/> <label> oui </label> <input name="radGroupe" type="radio" value ="non"/> <label > non</label></code>	

Structure d'un formulaire: `<form>` `</form>`

Type submit: Crée un bouton sur lequel l'utilisateur doit cliquer pour déclencher l'envoi des données de tout le formulaire vers le serveur.

```
<input type= "submit" value ="envoyer" name= "envoyer " />
```

Type reset: Crée un bouton de réinitialisation du formulaire

```
<input type ="reset" value = "Annuler " name= "annuler " />
```

Type hidden: Crée un champ caché n'ayant, comme son nom l'indique, aucun rendu visuel dans le formulaire mais permettant de transmettre des informations invisibles pour l'utilisateur.

```
<input type="hidden" name="exemple" value="Mateo21" />
```

textarea: champ de saisie de texte permettant la saisie sur plusieurs lignes.

```
<textarea name="zone_de_texte" cols="2" rows="2"> Tapez votre texte ici  
</textarea>
```

Structure d'un formulaire: `<form> </form>`

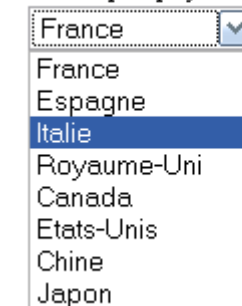
Liste déroulante:

```
<select name="choix">
  <option value="choix1">Choix 1</option>
  <option value="choix2">Choix 2</option>
  <option value="choix3">Choix 3</option>
  <option value="choix4">Choix 4</option>
</select>
```

Vous pouvez définir le choix par défaut de la liste. avec l'attribut **`selected="selected"`** à une balise `<option>`, alors ce sera le choix par défaut. On pourrait par exemple écrire :

```
<option value="choix3" selected="selected">Choix 3</option>
```

Dans quel pays habitez-vous ?



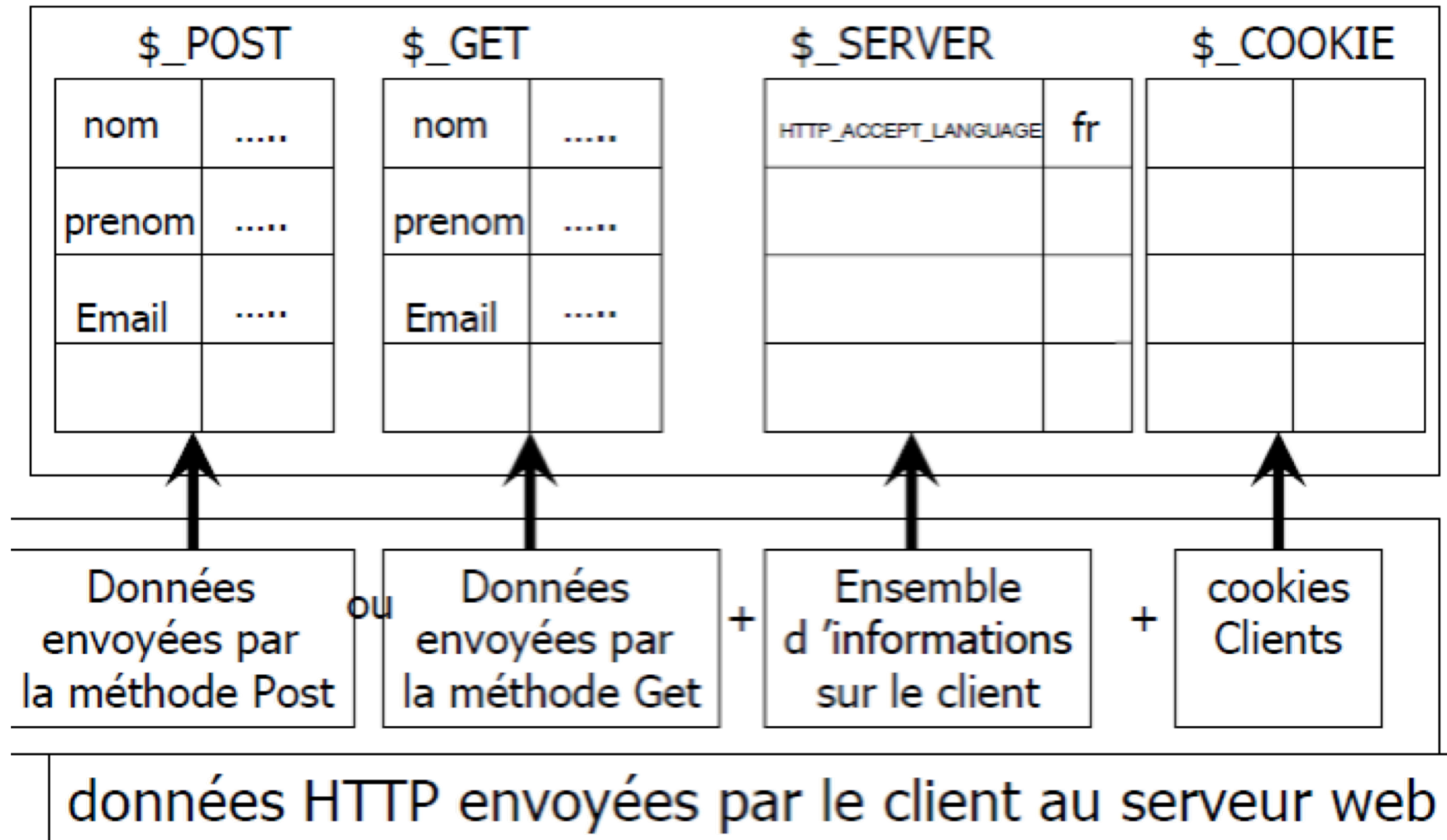
Exemple de formulaire

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Mon premier formulaire</title>
  </head>
  <body>
    <!-- attributs obligatoires de Form method et action-->
    <form method="POST" action="formulairePOST.php">
      <!-- Contenu du formulaire-->
      Nom:<input type="text" name="nom"></br>
      Prénom:<input type="text" name="prenom"></br>
      Email:<input type="text" name="email"></br>
      <input type="submit" name="ok" value="OK">
    </form>
  </body>
</html>
```


Récupération des données – Valeurs uniques

- Quand l'utilisateur clique sur le bouton « **Submit** », Une requête HTTP est envoyée au serveur à destination du script désigné par l'attribut **action** de l'élément **<form>**.
- La requête contient toutes les associations entre les noms des champs et leur valeur. Ces associations se trouvent dans l'en-tête HTTP si la méthode POST est utilisée et dans l'URL s'il s'agit de la méthode GET.
- Depuis la version 4.1 de PHP, les valeurs des formulaires sont contenues sur le serveur dans des tableaux associatifs dits super globaux appelés **\$_POST** et **\$_GET**, selon la méthode choisie. Les clés de ces tableaux sont les noms associés aux champs par l'attribut **name**.

Récupération des données



Récupération des données avec \$_GET

- Le tableau **\$_GET** permet de stocker les données envoyées par la méthode **GET**.
- Si on suppose, par exemple, qu'un client a envoyé une requête GET de type
<http://localhost/formulaire.php?nom=azerty>
- Nous pouvons écrire l'instruction PHP suivante pour récupérer la valeur du paramètre nom de l'URL.

```
<?php echo $_GET['nom']; ?>
```

Récupération des données avec \$_POST

- Le tableau **\$_POST** permet de stocker les données envoyées par la méthode **POST**.
- Si on suppose, par exemple, qu'un client a envoyé un formulaire qui contient les champs nom, prénom et email, en utilisant la méthode POST, nous pouvons récupérer, côté serveur avec un script PHP en écrivant :

```
< ?php
    echo $_POST['nom'];
    echo $_POST['prenom'];
    echo $_POST['email'];
?>
```

Récupération des données – Valeurs multiples

- Certains champs de formulaire peuvent permettre aux visiteurs de saisir plusieurs valeurs sous un même nom de composant.
- Cela peut concerner un groupe de **cases à cocher** ayant le même attribut name, par exemple, dont il est possible de cocher une ou plusieurs cases simultanément.
- Dans tous les cas, ce n'est pas une valeur scalaire mais un tableau qui est récupéré côté serveur. Il faut pour cela faire suivre le nom du composant de crochets, comme pour créer une variable de type array().

Récupération des données – Valeurs multiples

- Dans l'exemple suivant :

Bleu: `<input type="checkbox" name="choix[]" value="bleu" />`

Blanc: `<input type="checkbox" name="choix[]" value="Blanc" />`

- l'utilisateur peut cocher les deux cases simultanément. Le programmeur récupère ces valeurs dans les variables suivantes :

`$_POST["choix"][0]` qui contient la valeur »Bleu"

`$_POST["choix"][1]` qui contient la valeur « Blanc"

Récupération des données – le contrôle de saisie

- Avant de pouvoir récupérer les données dans le script PHP, on peut vérifier si ces données ont été saisies par l'utilisateur.

```
<?php
```

```
//vérifier l'existence des variables
```

```
if ( isset($_POST['prenom']) && isset($_POST['nom'])) {  
echo 'Bonjour ' . $_POST['prenom'] , $_POST['nom'] . ' !';  
}
```

```
else
```

```
{
```

```
echo 'Il faut renseigner un nom et un prénom !';
```

```
}
```

```
?>
```

Exercice

- Réaliser une page html contenant le formulaire suivant.
- Réaliser un script PHP permettant de récupérer le login et le mot de passe et les comparer avec ceux préalablement stockés dans un tableau associatif.
- Afficher un message d'erreur si le login ou mot de passe est incorrect sinon, afficher la page d'accueil de l'application

The image displays two versions of a login form, illustrating a successful login and an error state.

Top Mockup (Normal State):

- Title:** Login
- Instructions:** Pour accéder à votre espace personnel, merci de saisir votre login/email et password
- Form Fields:**
 - Email:
 - Password:
- Submit Button:** Envoyer

Bottom Mockup (Error State):

- Title:** Login
- Instructions:** Pour accéder à votre espace personnel, merci de saisir votre login/email et password
- Form Fields:**
 - Email:
 - Password:
- Submit Button:** Envoyer
- Error Message:** Login ou mot de passe incorrect .