Chapitre 9

# Gestion des formulaires

# I. Introduction

Les formulaires constituent le principal moyen pour vos visiteurs d'entrer des informations sur votre site. Ils constituent donc l'élément essentiel qui permet l'interactivité entre un site et ses visiteurs.

Tout échange entre visiteur et serveur passe par un formulaire, dans lequel l'utilisateur peut saisir **textes** ou **mots de passe**, opérer des **choix** grâce à des boutons **radio**, des **cases à cocher** ou des **listes** de sélection, voire envoyer ses propres **fichiers** depuis le poste client.

Le langage HTML ne peut pas analyser les informations que le visiteur a saisi dans un formulaire. Alors, le traitement des résultats doit s'effectuer dans un autre langage, par exemple le **PHP**.

# II. Création d'un formulaire HTML

## 1. Structure d'un formulaire HTML

```
<!DOCTYPE html>
<html>
	<head>
		<meta charset="UTF-8">
		<title>Titre de la page</title>
	</head>
	<body>
	<form method="POST" action="<?=$_SERVER['PHP_SELF']?>">
		<fieldset><legend>Titre du formulaire</legend>
			<!--Corps du formulaire-->
		</fieldset>
	</form>
	</body>
</html>
```

## 2. Attributs de l'élément <form>

`method`="**POST|GET**" **:** indique la méthode de transmission HTTP des données saisies dans le formulaire.

- «**POST**» est la valeur qui correspond à un envoi de données stockées dans le corps de la requête (les données sont cachées).
- «**GET**» correspond à un envoi des données ***visibles dans l'URL*** (séparées de l'adresse du script par un point d'interrogation) : ***à éviter pour des raisons évidentes de sécurité***.

La méthode **GET** est la ***méthode par défaut*** et présente deux inconvénients:

1) L'ajout des données du formulaire à l'adresse URL du fichier qui les traite, ce qui les rend visibles par le visiteur.
2) Il existe une limite à la longueur des URL (2 à 4ko)

Ces problèmes ne se retrouvent pas avec la valeur POST, que vous utiliserez dans la plupart des cas.

## 2. Attributs de l'élément <form>

`action`="**nom_de_fichier.php**" permet d'indiquer l'**URL** qui va recevoir les informations entrées dans le formulaire, lorsque l'on cliquera sur le bouton de validation. Plus précisément, l'URL est l'adresse d'un programme (un script) qui va récupérer les données et les traiter.

Si le fichier qui traite les données est celui qui contient le formulaire vous pouvez utiliser les syntaxes suivants:

1) `action="<?=$_SERVER['PHP_SELF']?>"`
2) `action="#"`
3) `action=""`

L'attribut `action` 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@gmail.com"`

## 3. Attributs optionnels de l'élément <form>

name="**chaine_de_caractères**" Attribue un nom au formulaire. Cet attribut est surtout utilisé pour accéder aux éléments du formulaire via un script JavaScript.

enctype="**type_d'encodage**" **:** C'est le type d'encodage des données transmises au serveur.

1) "**application/x-www-form-urlencoded**" **:** c'est la valeur par défaut. C'est celle qu'il faut utiliser dans la plupart des cas.
2) "**multipart/form-data**" **:** C'est la valeur à utiliser pour envoyer des fichiers.

***Remarque*:** Si l'attribut action a la valeur "**mailto**:", l'attribut **enctype** a pour valeur "text/plain" ou "text/html", selon que le contenu envoyé à une adresse e-mail au format texte ou XHTML.

## 4. Ecrire un formulaire

La balise **<input>** définit des champs d'information. L'attribut **name** associe un nom au champ, l'attribut **value** désigne éventuellement une valeur initiale, et l'attribut **type** décrit le type de champ de formulaire. Les valeurs possibles incluent:

- ✓ **`text`** : champ de texte qui permet à l'utilisateur de saisir un nombre ou une phrase.
- ✓ **`checkbox`** : cases à cocher.
- ✓ **`radio`** **:** boutons radio.
- ✓ **`password`** :pour un champ de mot de passe.
- ✓ **`hidden`** : champ caché pour transmettre des informations vers le serveur, sans que l'utilisateur ne le sache.

## 4. Ecrire un formulaire

- ✓ **textarea** pour une zone de texte multi-lignes : `<textarea> </textarea>`
- ✓ **select** :pour une liste de sélection (ou liste déroulante).
- ✓ **submit** : pour un bouton de soumission.
- ✓ **image** : même principe que submit mais c'est une image au lieu d'un bouton.
- ✓ **file** : Affiche un bouton permettant de sélectionner un fichier de l'ordinateur.
- ✓ **email,url,number,date,etc**. : pour des types de données spécifiques.
- ✓ **reset** :pour remettre le formulaire dans son état initial.

**Remarque :**

Les **checkbox** et **radios** n'envoient rien s'ils ne sont pas cochés.

# *Exemple*

Créer une page php (**formulaire.php**) contenant un formulaire avec les champs suivants:

- Vos coordonnées qui contient:
  - Nom (zone de texte), Prénom (zone de texte),Mail (zone de texte).
  - Sexe (boutons radio): Homme ou Femme.
  - Ville (liste déroulante).

- Vos loisirs
  - Loisirs (cases à cocher): Lecture, Sport, Voyage.
  - Zone de texte multilignes pour décrire vos loisirs.

- Envoyer nous votre photo
  - Champ d'envoi d'un fichier du client vers le serveur.
  - Champ caché contenant la taille maximale des fichiers transférables.
  - Boutons de réinitialisation.
  - Bouton d'envoi.

# L’interface du formulaire

**Vos coordonnées**

**Nom :**

**Prénom :**

**Mail :**

**Sexe :** ○ Homme ○ Femme

**Ville** Paris

**Vos loisirs**

**Loisirs :** ☐ Voyage ☐ Lecture ☐ Sport

**Commentaire :** Décrivez vos loisires

**Envoyer nous votre photo**

Choisir un fichier Aucun fichier choisi

Envoyer Effacer

# III. Récupération des données du formulaire

## 1. valeurs uniques

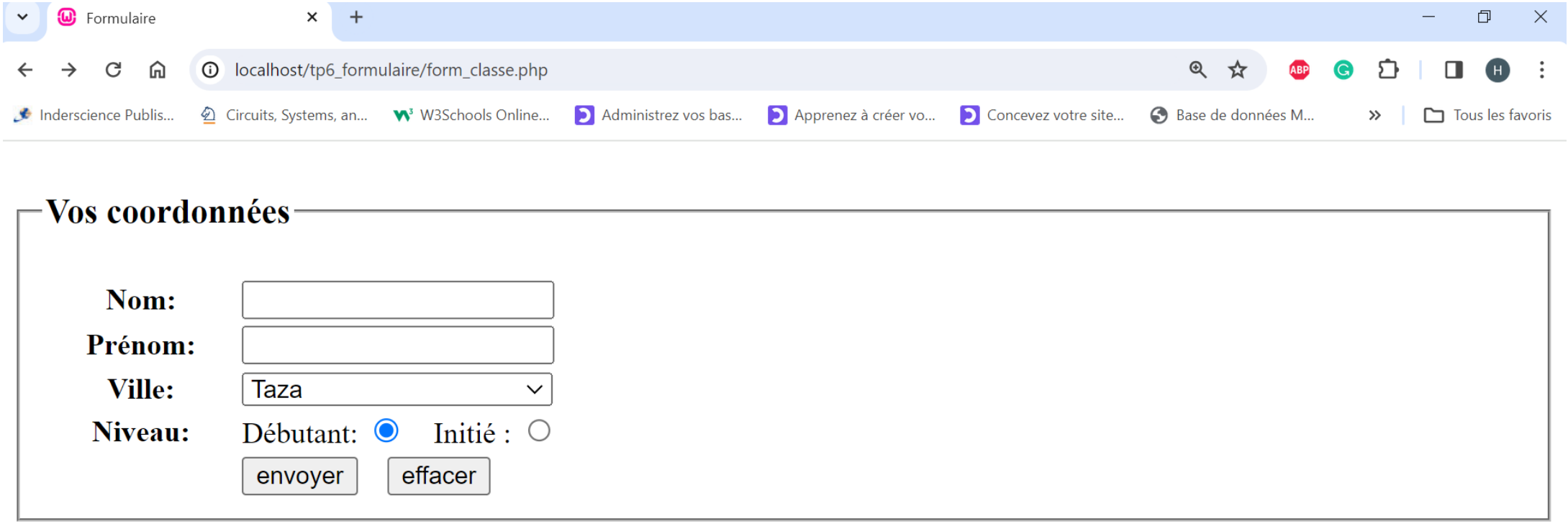
- Les valeurs uniques proviennent des champs de formulaire dans lesquels l'utilisateur ne peut entrer qu'une valeur, un texte par exemple, ou ne peut faire qu'un seul choix (*bouton radio, liste de sélection à choix unique*).

- Selon la méthode choisie, ces valeurs sont contenues sur le serveur dans des tableaux associatifs **$_POST** et **$_GET**. Les clés de ces tableaux sont les noms associés aux champs par l'attribut **name**.

# Cas de la méthode POST

## Exemple 1: Récupération des valeurs dans un formulaire élémentaire

1) Ecrire un code HTML qui permet d'avoir le formulaire de la figure suivante :



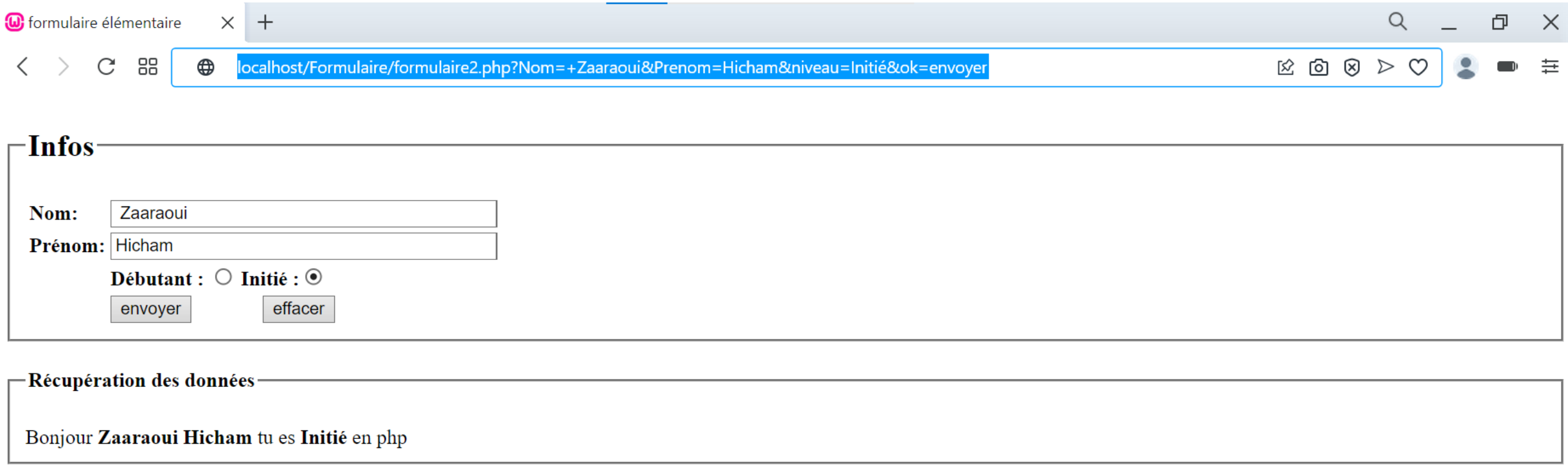

2) Ecrire un code PHP qui permet de récupérer le résultat dans le même fichier, comme suit :

**Récupération des données**

Bonjour **Zaaraoui Hicham** tu es **Initié** en php

# Cas de la méthode GET

**Exemple 2:** On remplace la méthode POST par la méthode GET.




## Remarque:

Lors du clic sur le bouton d'envoi l'adresse de la page cible désignée par l'attribut **action** est suivie par le caractère ? puis par le nom de chaque champ et la valeur qui y est associée.

## 2. Les valeurs multiples

- Cela concerne un groupe de ***cases à cocher*** ayant le même attribut `name` ou ***une liste de sélection*** ayant toujours un nom unique mais dans laquelle l'attribut `multiple` est défini.

**Exemple:**

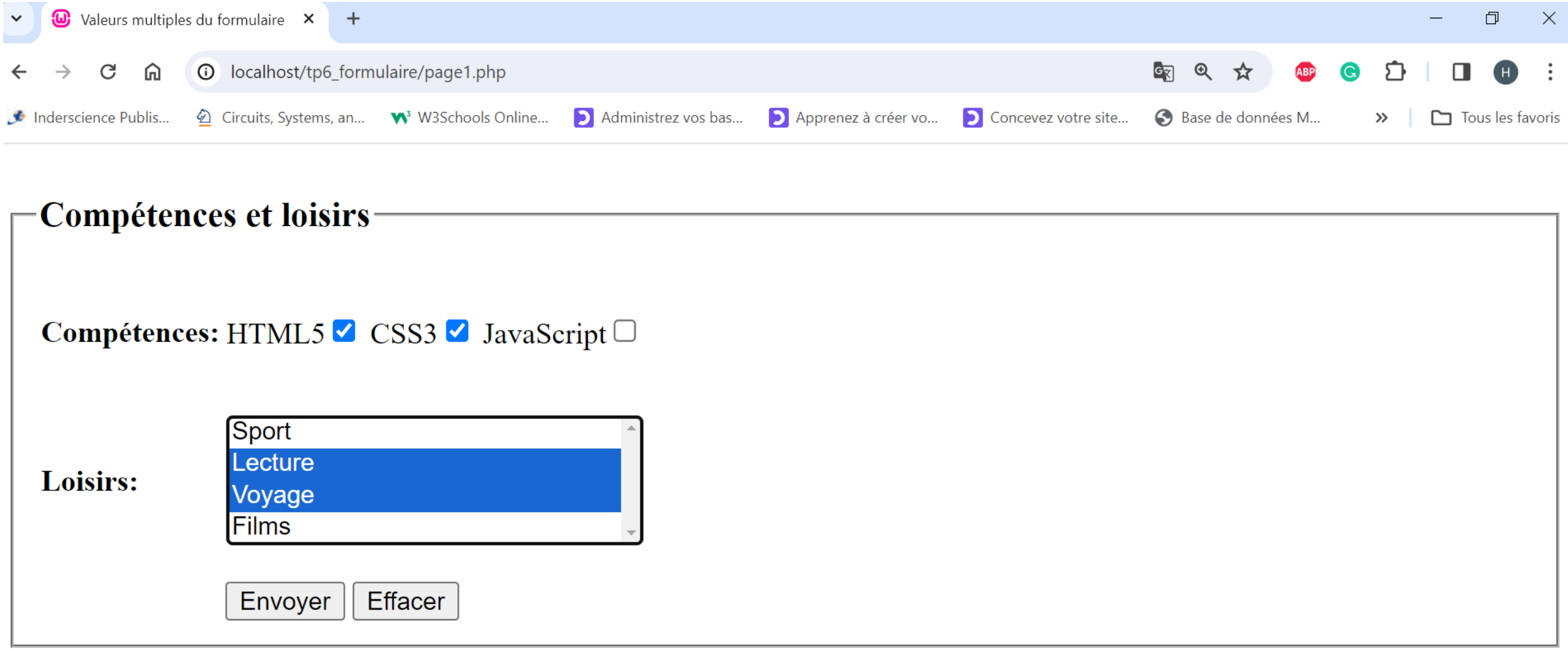
**Bleu :** `<input type="checkbox" name="choix[]" value="bleu" />`

**Rouge:** `<input type="checkbox" name="choix[]" value="rouge" />`

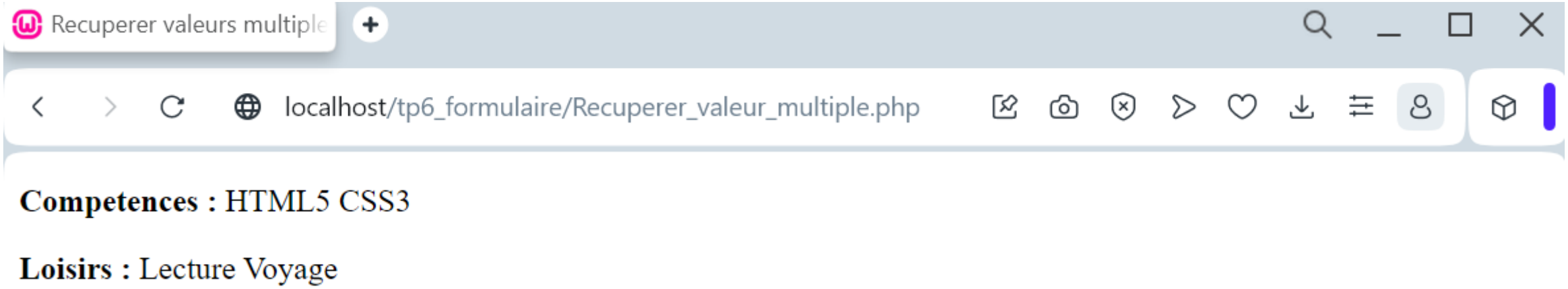
- Pour récupérer ces valeurs on utilise:
  - ✓ `$_POST["choix"][0]` qui contient la valeur **bleu**.
  - ✓ `$_POST["choix"][1]` qui contient la valeur **rouge**.

# Exemple 3 : Récupération des valeurs multiple du formulaire

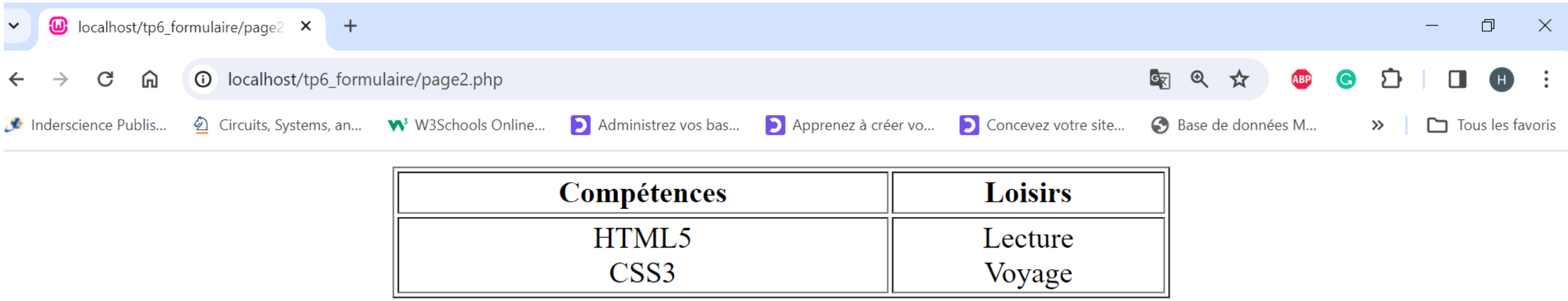
1) Ecrire une page **page1.php** qui permet d'afficher le formulaire suivant:



2) Ecrire le script de la page **page2.php** qui permet d'afficher l'interface suivante :




3) Ecrire un autre code dans la page **page3.php** qui permet d'afficher l'interface suivante :




| Compétences | Loisirs |
|---|---|
| HTML5<br>CSS3 | Lecture<br>Voyage |

## 1) page1.php Solution

```
<body>
<fieldset> <legend><h3>Compétences et loisirs</h3></legend>
<table>
<form method='post' action="page2.php">
<tr> <th>Compétences:</th>
<td> HTML5<input type="checkbox" name="Competences[]" value="HTML5"/>
CSS3<input type="checkbox" name="Competences[]" value="CSS3"/>
JavaScript<input type="checkbox" name="Competences[]" value="JavaScript"/>
</td> </tr>
<tr> <th>Loisirs:</th> <td> <select name="Loisirs[]" multiple>
<option value="Sport">Sport</option>
<option value="Lecture">Lecture</option>
<option value="Voyage">Voyage</option>
<option value="Films">Films</option>
</select> </td> </tr>
<tr> <th colspan="2" class="al"> <input type="submit" value="Envoyer"
name="ok"/> <input type="reset" value="Effacer" name="no"/> </th> </tr>
</form> </table>
</fieldset> </body>
```

## 2) page2.php — Solution

```
<meta charset="UTF-8"/>
<?php
if(isset($_POST['ok'])){
    $ch="Compétences : ";
        foreach($_POST['Competences'] as $val){
            $ch.="<b>".$val." </b>";
        }
$ch.="<br>";
$ch.="Loisirs : ";
        foreach($_POST['Loisirs'] as $v){
            $ch.="<b>".$v." </b>";
        }
echo $ch;
}
?>
```

## 3) page3.php — Solution

```php
<?php
if(isset($_POST['ok'])){
     $ch="<table width='50%' align='center' border='1'>";
$ch.="<tr><th>Compétences</th><th>Loisirs</th></tr>";
$ch.="<tr align='center'><td>";
     foreach($_POST['Competences'] as $val){
        $ch.=$val."<br>";
     }
$ch.="</td><td>";
      foreach($_POST['Loisirs'] as $v){
        $ch.=$v."<br>";
      }
$ch.="</td></tr></table>";
echo $ch; }
?>
```

# IV. Maintien de l'état du formulaire

- Après le traitement, le formulaire se retrouve alors dans son état initial, et toutes les saisies effectuées sont effacées.

- En cas d'erreur de saisie sur un seul champ, l'utilisateur est obligé de recommencer l'ensemble de la saisie.

*Pour éviter cela,*

- Pour ***le texte,*** il suffit de définir l'attribut **value** avec la variable $_POST[name] ou $_GET[name]:

**Exemple:**

```
value="<?php if(isset($_POST["nom"])) echo $_POST["nom"]?>"
```

- Pour ***les boutons radio***, il faut définir l'attribut du bouton choisi à la valeur "**checked**" en fonction de la valeur de la variable `$_POST[name](ou $_GET[name]).`

**Exemple:**

```
<?php if(isset($_POST["niveau"])&& $_POST["niveau"]=="debutant")
echo "checked"; ?>
```

- Pour ***les listes déroulante***, il faut définir l'attribut du bouton choisi à la valeur "**selected**" en fonction de la valeur de la variable `$_POST[name](ou $_GET[name]).`

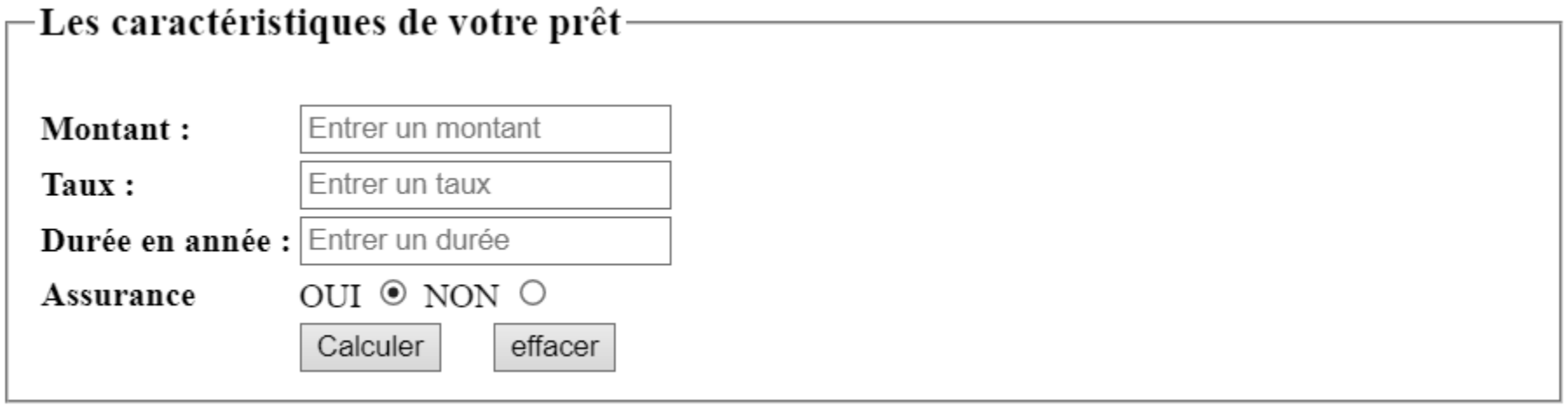
**Exemple:**

```
<?php if(isset($_POST["ville"])&&$_POST["ville"]=="Paris")
echo "selected"; ?>
```

# Exercice d'application:

Soit un site d'annonces immobilières proposant un plan de financement aux visiteurs.

Créer un formulaire permettant de calculer le prêt bancaire ?




- Le **Montant** correspond au capital emprunté.
- Le **Taux** désigne le taux mensuel sous forme décimale. Par exemple, si l'utilisateur saisit 6 pour le taux annuel, la variable $taux vaut 6/100/12, soit 0,005, ou 0,5 % par mois.

- La **Durée en année** qui sera convertie en mois pour calculer la mensualité.

- L'**Assurance** a la valeur 1 puisque le bouton radio « OUI » est coché par défaut. La variable $assur renvoie au montant de l'assurance mensuelle, soit 0,035 % du capital emprunté. Cette variable prend la valeur 0 si le bouton radio « NON » est coché.

- la **Mensualité** est calculée selon la formule financière suivante:

$$mens = \frac{(capital \times taux)}{1 - (1 + taux)^{-duree}}$$

# Pour un prêt de 300000 DH à 2 % sur 10 ans, la mensualité est de : 2760.4DH hors assurence

## Tableau d'amortissement

| Mois | Capitale restant | Mensualité hors assurance | Amortissement | Intêret | Assurance | Mensualité avec assurance |
|---|---|---|---|---|---|---|
| 1 | 300000 | 2760.4 | 2260.4 | 500 | 105 | 2865.4 |
| 2 | 297739.6 | 2760.4 | 2264.17 | 496.23 | 105 | 2865.4 |
| 3 | 295475.43 | 2760.4 | 2267.94 | 492.46 | 105 | 2865.4 |
| 4 | 293207.48 | 2760.4 | 2271.72 | 488.68 | 105 | 2865.4 |
| 5 | 290935.76 | 2760.4 | 2275.51 | 484.89 | 105 | 2865.4 |
| 6 | 288660.25 | 2760.4 | 2279.3 | 481.1 | 105 | 2865.4 |
| 7 | 286380.94 | 2760.4 | 2283.1 | 477.3 | 105 | 2865.4 |
| 8 | 284097.84 | 2760.4 | 2286.91 | 473.5 | 105 | 2865.4 |
| 9 | 281810.93 | 2760.4 | 2290.72 | 469.68 | 105 | 2865.4 |
| 10 | 279520.21 | 2760.4 | 2294.54 | 465.87 | 105 | 2865.4 |
| 11 | 277225.68 | 2760.4 | 2298.36 | 462.04 | 105 | 2865.4 |

# TP : Formulaire

## Exercice 1

Créez un formulaire comprenant un groupe de champs ayant pour titre "**Adresse client**". Le groupe doit permettre la saisie du ***nom***, du ***prénom***, du **sexe**, de la ***ville*** et des **loisirs**. Les données sont ensuite traitées par un fichier PHP séparé récupérant les données et les affichant dans un tableau.

## Exercice 2

Améliorez le script précédent en vérifiant l'existence des données et en affichant une ***boîte d'alerte JavaScript*** si l'une des données est manquante.

## Exercice 3

Créez un formulaire de saisie d'***adresse e-mail*** contenant un champ caché destiné à récupérer le ***type du navigateur*** de l'utilisateur. Le code PHP affiche l'adresse et le nom du navigateur dans la même page après vérification de l'existence des données.

## Exercice 4

Créez un formulaire demandant la saisie d'un prix HT et d'un taux de TVA. Le script affiche le montant de la TVA et le prix TTC. Le formulaire ***maintient les données saisies***.

# V. Transfert de fichiers vers le serveur

```
<input type="file" />
```

- Il ne s'agit plus de transmettre une ou plusieurs valeurs scalaires au serveur mais **l'intégralité d'un fichier**, lequel peut avoir une **taille** importante et un **type** quelconque. Ce fichier doit évidemment être présent sur l'ordinateur du visiteur.

***Remarque*:**

Problème de sécurité pour votre site, car les fichiers vont être écrits et éventuellement exécutés sur le serveur.

***Exemple*:**

Un utilisateur mal intentionné transfère un fichier nommé index.php ou index.html sur le serveur.

## Solution :

- Vérifier l'extension du fichier préalablement à la définition des extensions autorisées lors de la création du formulaire avec l'attribut : `accept` de l'élément `<input/>`

- L'élément `<form>` doit avoir l'attribut `method` à la valeur **`POST`** et l'attribut `enctype` à la valeur **`multipart/form-data`**.

### Précautions supplémentaires:

- Limiter la taille des fichiers à télécharger en ajoutant un champ caché nommé MAX_FILE_SIZE, dont l'attribut value contienne la taille maximale admise en octet (récupérée dans `$_POST["MAX_FILE_SIZE"]`).

```
<input type="hidden" name="MAX_FILE_SIZE" value="1000000"/>
```

***Remarque :***

- L'utilisation de ce champ n'est pas obligatoire puisque le fichier **php.ini** du serveur contient la directive "**upload_max_filesize**", dont la valeur est un entier indiquant la taille maximale en octet admise par défaut par le serveur. En local avec **Wampserver**, cette valeur vaut **2 Mo**.

- Le fichier est bien transféré sur le serveur, mais il se trouve dans un répertoire tampon défini par la directive "**upload_tmp_dir**" du fichier **php.ini.** Si vous n'en faites rien, il est perdu lors de la déconnexion du client.

- Le nom du fichier sur le poste client est différent de celui qu'il est enregistré sur le serveur, puisque celui-ci est créé arbitrairement par le serveur.

❑ Pour récupérer les informations nécessaires au traitement du fichier transféré. Il faut utiliser les tableaux associatifs multidimensionnels `$_FILES`.

```
<input type="file" name="fich" accept=".doc" />
```

- `$_FILES["fich"]["name"]` contient le nom qu'avait le fichier sur le poste client.
- `$_FILES["fich"]["type"]` contient le type MIME initial du fichier.
- `$_FILES["fich"]["size"]` donne la taille réelle en octet du fichier transféré.
- `$_FILES["fich"]["tmp_name"]` donne le nom temporaire que le serveur attribue automatiquement au fichier en l'enregistrant dans le répertoire tampon.

- `$_FILES["fich"]["error"]` donne le code d'erreur éventuel associé au fichier téléchargé et permet d'afficher un message d'erreur en clair en créant un tableau indicé de 0 à 4 contenant les messages appropriés.

  **`UPLOAD_ERR_OK:0`** indique que le transfert est bien réalisé.

  **`UPLOAD_ERR_INI_SIZE:1`** indique que la taille du fichier est supérieure à celle définie dans la php.ini

  **`UPLOAD_ERR_FORM_SIZE:2`** indique que la taille est supérieure à celle définie dans le champ caché MAX_FILE_SIZE.

  **`UPLOAD_ERR_PARTIAL:3`** indique que le fichier n'a été que partiellement téléchargé.

  **`UPLOAD_ERR_NO_FILE:4`** indique qu'aucun fichier n'a été téléchargé.

- En dernier lieu, vous devez procéder à l'enregistrement et au renommage éventuel du fichier sur le serveur.

```
boolean move_uploaded_file(string "fichtmp", string "fichfinal")
```

## Exercice 5

Créez un formulaire n'effectuant que le transfert de fichiers **pdf** et d'une taille limitée à 1 Mo. Le script affiche le **nom** du fichier du poste client ainsi que **la taille** du fichier transféré et la **confirmation de réception**.

Il est possible de proposer à l'utilisateur de transférer plusieurs fichiers simultanément. Dans ce cas, l'éléments <input type="file"…/> soient tous écrits de la manière suivante :

```
<input type="file" name="fich[]"  accept="image/gif" />
```

## Exemple :

Ecrire un script qui permet d'avoir les informations liées aux deux fichiers téléchargées à partir du poste client?