Chap 14_1. Formulaires : Création

Livre p 381 – Chap 28 Requêtes http et formulaires

1. Création d'un formulaire HTML

a. Introduction

On a vu, dans le chapitre 5, comment créer des pages web en HTML, puis dans le chapitre 6, comment les mettre en page grâce au CSS. Mais ces pages sont statiques : il y a peu d'interactions avec le visiteur.
Grâce aux formulaires, on peut créer des pages dynamiques qui vont interagir avec le visiteur.
Pour cela deux étapes sont nécessaires :

- La demande de saisie de données, avec de nouvelles balises HTML…
- Le traitement des données récoltées en utilisant un langage de programmation tel que le PHP, le JavaScript ou même le Python !

```
<!DOCTYPE html>
<html>
    <head>
        …
    </head>
    <body>
        …
        <form>
        Contenu du formulaire
        </form>
        …
    </body>
</html>
```

b. Structure du document

Le formulaire est une zone particulière du corps d'une page web, délimitée par les balises <**form**> et <**/form**>.

Attention : il faut y ajouter deux attributs :

- La méthode : Comment envoyer les données ?
  - GET : Les données sont visibles dans l'adresse URL de la page (C'est la méthode utilisée par défaut, ce n'est donc pas la peine de la préciser dans ce cas)
  - POST : Les données ne sont pas visibles (mais attention, elles ne sont pas cryptées, elles peuvent être interceptées)
- L'action : Comment traiter les données ?
  On doit indiquer l'adresse de la page qui va traiter les données
  voir la deuxième partie : « **14_2. Formulaires : Traitement** »

Exemple : <form **method**="post" **action**="test.php"> … </form>.

c. Zones de saisie (ou champ)

Entre les deux balises définissant le formulaire, on va pouvoir écrire différentes zones de saisie :

- sur une ligne : avec les balises <**input** type="…" name="…" />
- sur plusieurs lignes : avec les balises <**textarea** name="…" > </**textarea**>

Attention : Il faudra mettre tout cela en forme avec du code CSS (taille de la zone, couleur, police…)
Remarque :

- **name**="…" : permet de donner un nom aux différentes données qui vont être envoyées
- **type**="…" : permet d'avoir un format prédéfini pour les données saisies et de sélectionner un clavier approprié lorsqu'on utilise un smartphone ou une tablette.

d. Label et id

Il est fortement conseillé d'ajouter un descriptif (label) à chaque zone de saisies, il faut alors lier le descriptif à la zone en utilisant un identifiant unique (id).
Exemple : <**label** for="zone"> Écrire ici : </**label**> <input type="text" name="texte" **id**="zone"/>
Attention : Ne pas confondre le nom de la variable contenant les données envoyées (name="…") et l'identifiant de la zone de saisie utilisée pour la lier à son étiquette (id="…")

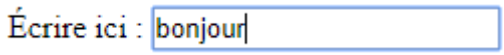
e. Envoie des données

À la fin du formulaire, il est indispensable de créer un bouton pour envoyer les données !
Exemple : <input type="**submit**"/> ou <button type="**submit**"> Envoyer </button >

2. Types de zones de saisie

a. Texte standard

Un cadre pour une envoyer une ligne de texte classique : <input type="**text**" name="reponse" />



b. Texte caché

Un cadre pour envoyer une ligne de texte dans laquelle tous les caractères sont remplacés par un symbole * ou • (cela dépend du navigateur utilisé) : <input type="**password**" name="reponse" />
Remarque : Certains navigateur proposent une icône à droite pour voir ou cacher le texte saisi.

c. Nombre

Un cadre pour envoyer un nombre entier : <input type="**number**" name="reponse" />
On peut même préciser une valeur minimale ou maximale autorisée…
Remarque : Certains navigateur proposent une icône pour augmenter ou diminuer le nombre saisi, mais on peut aussi utiliser un curseur, mais il n'y a pas d'indications sur le curseur !
Exemple : <input type="**range**" name=" reponse " value="10" max="20" min="0" step="2">

d. Numéro de téléphone, adresse mail, URL

Un cadre pour envoyer un numéro de téléphone : <input type="**tel**" name="reponse" />
ou pour envoyer une adresse mail : <input type="**email**" name="reponse" />
ou pour envoyer une adresse URL : <input type="**url**" name="reponse" />
Remarque : En plus d'obtenir un clavier adapté si on est sur smartphone ou tablette, lors de l'envoie des données, le navigateur vérifie format du texte entré ("…@..." pour un email ou "http://..." pour une URL)

e. Valeurs prédéfinies

On peut demander à l'utilisateur de choisir parmi :

- des cases à cocher : (plusieurs cases possibles en même temps)
  <input type="**checkbox**" name="choix_1" value="1" id="c_1" /> <label for="c_1"> Choix 1 </label>
  <input type="**checkbox**" name="choix_2" value="2" id="c_2" /> <label for="c_2"> Choix 2 </label>
- des options à sélectionner : (une seule option possible)
  <input type="**radio**" name="choix" value="1" id="c_1" /> <label for="c_1"> Choix 1 </label>
  <input type="**radio**" name="choix" value="2" id="c_2" /> <label for="c_2"> Choix 2 </label>
- une liste déroulante : (une seule ligne à choisir dans une liste)
  <label for="liste"> Liste ci-dessous
  <**select** name="choix" id="liste">
  <**option** value="1"> Choix 1 </**option**>
  <**option** value="2"> Choix 2 </**option**>
  </**select**>

f. Autres types :

On peut aussi demander une date ("**date**"), un mois ("**month**"), une semaine ("**week**"), une heure ("**time**") et même une couleur ("**color**"). Attention : Certains navigateurs ne sont pas compatibles !

g. Attributs :

On peut ajouter un ou plusieurs attributs à la balise <input … /> pour personnaliser son fonctionnement :

- **autofocus** : Active directement une zone de saisie (un seul autofocus par formulaire).
- **required** : L'envoi des données du formulaire n'est possible que si cette zone est complétée.
- **maxlength** : Permet de limiter le nombre de caractères à saisir.
- **value** : Permet de pré-remplir la zone par une valeur.
- **holder** : Permet de donner une indication sur le contenu de la zone à remplir.
- **size** : Permet de spécifier la taille de la zone, mais on peut aussi le faire en CSS…
  Remarque : Utiliser **rows** et **cols** pour les zones de plusieurs lignes avec la balise <textarea>

Chap 14_2. Formulaires : Traitement

Livre p 395 – Chap 29 Le web, côté serveur

3. Traitement des données en PHP

a. Introduction

Une fois que le formulaire a été rempli et que l'utilisateur a cliqué sur le bouton « Soumettre », comment traiter les données récoltées ?
On peut traiter les données directement, côté client, avec le langage JavaScript ou de façon plus sûre, côté serveur, avec le langage PHP. Nous utiliserons le langage PHP dans ce chapitre !
PHP signifie : « PHP : Hypertext Preprocessor » (PHP signifiant cette fois : « Personal Home Page »)
On peut aussi traiter les données en Python, le module Django facilite alors la création de formulaires.
Remarque : On peut également associer le tout à une base de données MySQL…

b. Structure du document

Lors de la création du formulaire, nous avons écrit en dans la page HTML la ligne suivante :

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

Il faut donc créer la page nommée : "**test.php**" : (On peut changer le nom de la page dans le formulaire)
C'est en fait une page HTML avec sa structure classique, dans laquelle on va ajouter des commandes en langage PHP dans des balises <?**php** … ?>.
Attention : Dans le langage PHP, on utilise des points virgules à la fin de chaque instruction et des accolades pour les instructions plus complexes. On peut ajouter des commentaires ainsi : / … /

c. Variables

Le nom d'une variable en PHP doit toujours commencer par **$**.
Le symbole utilisé pour l'affectation est **=** comme en Python.
Exemples : $n = 1 ; $prenom = "Nicolas" ;

Les données envoyées par le formulaire sont stockées dans un tableau nommé **$_GET** ou **$_POST**, selon la méthode utilisée.
Généralement, il récupère leurs valeurs au début du document.

Remarque : On peut aussi définir des constantes, leur nom ne commence pas par $ et on doit passer par l'instruction **define**(nom, valeur).
Exemple : define("pi", 3.14) ;

```
<?php
    $a = $_POST["age"];
    $nom = $_POST["nom"];
?>
<!DOCTYPE html>
<html>
    <head>
        …
    </head>
    <body>
        …
        Traitement et affichage
        …
    </body>
</html>
```

d. Opérateurs

- Opérateurs mathématiques : On retrouve comme en Python, l'addition (**+**), la soustraction (**–**), la multiplication (*****), la division (/), le reste de la division (**%**) et les puissances (******).
- Opérateurs de comparaison : On retrouve aussi comme en Python, le test d'égalité (**==**) ou de différence (**!=**) et les inégalités classiques (<, >, <=, >=)
  Remarque : La comparaison se fait après conversion, il existe d'autres opérateurs pour une comparaison sans conversion **(===** et **!==**)
  Exemple : ("1" == 1) renvoie TRUE alors que ("1" === 1) renvoie FALSE
- Opérateurs booléens : On retrouve la conjonction (**and** ou **&&),** la disjonction (**or** ou ||), la négation (**!**) et le ou exclusif (**xor**)
  Remarque : Vrai et Faux peuvent s'écrire en minuscule ou en majuscule (**true**, **True**, **TRUE**)
- Opérateur pour les chaînes de caractères : la concaténation est un point (**.**). (c'était + en Python)
- Affectation : Le symbole d'affectation (=) peut-être combiné avec une opération (**+=**, ***=**, **.=**, …).

e. Affichage

Pour afficher un texte, un nombre ou le contenu d'une variable, on utilise l'instruction **echo**.
S'il y a plusieurs arguments à afficher, les séparer par une virgule (qui n'ajoutera pas un espace).
Attention : On affiche le texte dans une page HTML, donc le saut de ligne est "
" et non pas "\n"
Remarque : On peut mixer le code HTML et le code PHP pour afficher les données.
Exemple : Les deux cadres ci-dessous afficheront la même chose.

```
Bonjour < ?php echo $nom ; ?>
```

```
< ?php echo "Bonjour ", $nom ; ?>
```

f. Tests et boucles

- Test « Si … alors … sinon … » : **if** (test) {instructions_si_vrai} **else** {instructions _si_faux}
  Exemple : Le cadre ci-dessous permet de tester si une donnée est vide avant de l'affecter

```
<?php if (empty($_POST["a"])) {$a = 0 ;} else {$a = $_POST["a"] ;} ?>
```

- Boucle conditionnelle : **while** (test) {instructions}
- Boucle inconditionnelle : **for** (début ; fin ; pas) {instructions}
  Exemple : Les deux cadres ci-dessous afficheront la même chose.

```
<?php
    $i=0 ;
    while ($i < 10)
        {echo $i, "<br>" ; $i++ ; }
?>
```

```
<?php
    for ($i = 0 ; $i < 10 ; $i++)
        {echo $i, "<br>" ; }
?>
```

Attention : Les tabulations sont là pour faciliter la lecture, les blocs sont délimités par les accolades.

g. Gestion d'un fichier texte

- Ouverture : $fichier = **fopen**(nom_fichier, mode).
  mode peut prendre les valeurs : "r" pour lecture seule, "r+" pour lecture + écriture
  "w" pour écriture seule, "w+" pour écriture + lecture
  "a" pour ajout en écriture, le curseur se place à la fin du fichier
- Fermeture : **fclose**($fichier).
- Lecture d'une ligne : $variable = **fgets**($fichier)
- Lecture du fichier entier : $variable = **fread**($fichier)
- Ecriture : **fputs**($fichier, chaine) ou **fwrite**($fichier, chaine)
- Retour au début : **fseek**($fichier, 0) (sauf si le fichier est ouvert en mode "a")

h. Envoi d'emails

On peut envoyer directement un mail avec la commande : **mail**($to, $subject, $message).
$to : L'adresse mail du destinataire
$subject : Le sujet du message
$message : Le texte du message (avec des "\n" pour les sauts de ligne)

4. Serveur local

a. Introduction

Le traitement du formulaire en PHP ne peut se faire que si le serveur sur lequel est stocké la page en PHP l'autorise… Le site alwaysdata.com utilisé dans le TP du chapitre 5 sur le langage HTML l'autorise.
Attention : Windows ne l'autorise pas, il faut installer un serveur local (localhost) qui va permettre de tester ses pages dynamiques en PHP avant de les envoyer sur Internet !

b. Logiciels

- UWamp : https://www.uwamp.com/ se décompresse directement, pas d'installation à faire !
- Laragon : https://laragon.org/ installation et utilisation très simple !