Créer des formulaires en XHTML pour son site web

Date de dernière mise à jour : 27/06/2007 à 19:36

Les formulaires

Les formulaires sont la base de l'interactivité d'un site web. Le XHTML va vous permettre au mieux d'envoyer des formulaires par mail, il ne vous permettra pas d'afficher des messages en fonction de cases cochées ou non, de vous faire des calculs ou autres opérations. Vous aurez besoin d'un langage plus évolué tel que PHP, pour effectuer ceci. Nous allons quand même voir comment créer des zones de texte, des cases à cocher, etc ... Tout ça pour créer de beaux formulaires accessibles à tous. Commençons par les bases !

Ajout d'un formulaire et méthode d'envoi

Un formulaire se définit par une balise de type **block**. Si vous vous souvenez de ce que j'ai écrit aux chapitres précédents, cela veut dire que vous ne pouvez pas la placer n'importe comment. En l'occurence, vous ne pouvez pas la mettre dans un paragraphe.

La balise utilisée pour délimiter un formulaire est la balise **form>**
. Tout ce qui se trouve dans cette balise fera partie du formulaire. Les formulaires étant généralement destinés à être interactifs, il faut qu'ils soient traités. Pour ce faire, il faut que les données puissent être envoyées, soit par mail, soit à une page web qui sera chargée de les traiter. Cette page ne pourra être en XHTML seule, car le XHTML ne permet pas de récupérer des informations sur les formulaires. L'attribut qui permet de spécifier l'action qui sera faite en cas de clic sur le bouton "Envoyer" (par défaut) est l'attribut **action** qui prend pour valeur soit une adresse mail précédée de l'attribut mailto:, comme nous l'avons vu sur le chapitre concernant les liens, ou alors vous mettez l'adresse d'une page web qui sera chargée de traiter le formulaire. Voyons ces deux cas :

form action
="mailto:mail@fai.fr">

Dans ce cas le formulaire sera envoyé par mail à l'adresse mail@fai.fr. Voyons maintenant le cas où une page web (on la supposera en PHP) traite les données du formulaire :

form action
="page.php">

Ici, j'ai placé un lien relatif vers la page **page.php**, si vous ne comprennez pas cette notion, il faut que vous revoyez le chapitre sur les liens relatifs et absolus. Vous auriez pu placer une adresse web absolue.

Tout ça c'est bien beau, mais il va falloir indiquer à notre formulaire quelle méthode adopter pour l'envoi des données à la destination. Il existe deux méthodes :

-La méthode GET, qui fait passer toutes les données par l'url (adresse) de la page cible. Vous vous retrouverez donc avec une adresse de ce style : page.php?nom=dupont&prenom=pierre. Cette méthode a plusieurs inconvénients : outre le fait qu'un visiteur peut très bien mettre n'importe quoi dans la barre d'adresses, cette méthode est également limitée en nombre de caractères. La limite varie en fonction du navigateur, elle peut aller de 255 caractères à 65536 en général. Bref il ne vaut mieux pas compter sur cette méthode pour les envois de textes longs.
-La méthode POST, bien plus adaptée, va tout envoyer à la page web dans les en-têtes HTTP que renvoie votre navigateur. Vous n'aurez pas à vous en préoccuper, je vous dis juste que ça se passe comme ça :). Cette méthode est légèrement plus sécurisée que la méthode GET contre les pirates, mais n'espérez pas vous dispenser de vérifications côté serveur, qui doivent **toujours**- être faites. N'ayez jamais confiance en une donnée provenant du visiteur.

L'attribut permettant de spécifier la méthode d'envoi est method. Il peut prendre deux valeurs :

-get
-post

Voici le code de notre formulaire de tout à l'heure qui nous choisirons d'envoyer en méthode post :

form action
="page.php" method
="post">

Voyons maintenant comment ajouter des zones interactives à notre formulaire. On va commencer par les zones de texte basiques.

Les zones de texte simples

J'entends par zone de texte simple une case à une ligne dans laquelle vous pouvez saisir du texte. La plupart des zones que nous allons voir s'ajoutent grâce à la balise de type **input />** même si il y a beaucoup d'exceptions. Les attributs que vont prendre cette balise vont déterminer le type de notre champ. On appelle un champ une "zone" interactive du formulaire.

Commençons par une zone de texte :

L'attribut qui permet de dire que l'on insère une zone de texte simple est **type** et il prend la valeur **text**. Vous devez ensuite donner un nom à votre zone de texte, car ne perdez pas de vue que vous devez ensuite traiter les données. Pour récupérer les données, il faut bien pouvoir identifier vos différents champs, et donc leur donner des noms différents. L'attribut permettant de donner un nom à un champ est name et il prend la valeur du nom que vous souhaitez donner à votre attribut. Si je veux nommer mon champ texte **prenom** on aura ceci : **name ="prenom"**.

Les balises input doivent être placées dans des éléments de type block, en plus d'être dans le formulaire. On utilisera ici des paragraphes. Voici ce que ça donne au niveau du code :

form action
="page.php" method
="post">
p>input type
="text" name
="prenom" />
p>
form>

Quelques attributs existent en plus :

-L'attribut **size**- : il va vous permettre de spécifier la largeur du champ en nombre de caractères.
-L'attribut **maxlength**- : il va vous permettre de spécifier le nombre de caractères maximum que peut rentrer le visiteur (via le navigateur, encore une fois cela ne vous dispense pas de faire un traitement en PHP derrière).
-L'attribut **value**- : il va vous permettre de spécifier une valeur par défaut à votre champ, cette valeur sera affichée dans le champ texte et le visiteur pourra la modifier bien entendu.
-L'attribut **readonly**- : il va vous permettre de placer le champ texte en lecture seule, le visiteur ne pourra donc pas modifier la valeur s'y trouvant (tout du moins pas via le navigateur, donc attention, ne vous servez pas de cela pour une prétendue sécurité).

Voilà maintenant un exemple combinant toutes les propriétés que je viens de vous énoncer. Il s'agit d'un champ texte portant le nom "prenom", qui sera large de 20 caractères. Il pourra contenir 25 caractères maximum, contiendra la valeur "Anthony" par défaut et sera en

lecture seule :

```
form action
="page.php"          method
="post">
            p>input          type
="text" name
="prenom"          value
="Anthony" size
="20"          maxlength
="25" readonly
="readonly"          />
p>
form>
```

Vous obtenez ceci :

Les labels

Vue comme ça, la zone de texte ne présente aucun attrait. Il faudrait que le visiteur sache ce qu'il va remplir. Il faut donc lui apporter des informations supplémentaires. On utilise pour cela la balise **label>**
. Si vous souhaitez ajouter comme information "Votre prénom :", vous allez faire ceci en XHTML :

```
form action
="page.php"          method
="post">
            p>label
>
```
Votre prénom :

```
input          type
="text" name
="prenom"

/>
label
>
p>
form>
```

La balise **label>**
doit entourer l'information et le champ de formulaire. Cela permet aux navigateurs autres que graphiques de savoir que telle information est liée à tel champ de formulaire.

Zone de mot de passe

La zone de mot de passe est une zone de texte dans laquelle les caractères sont remplacés par des étoiles ou des points. Vous pouvez donc l'utiliser comme zone dans laquelle un visiteur doit entrer un mot de passe pour s'identifier par exemple. A la place du type text, vous

allez cette fois utiliser le type password. Voilà ce que ça donne :

```
form action
="page.php" method
="post">
p>input type
="password" name
="mot_de_passe"

/>
p>
form>
```

Les attributs que nous avons vus tout à l'heure fonctionnent aussi pour cette zone.

Zone de texte multiligne

La zone de texte multiligne va être utile dès que vous voudrez taper un long message, comme pour poster un message sur le forum par exemple. Elle n'utilise pas la balise **input />**
mais a sa propre balise, la balise **textarea>**
. Elle dispose de deux autres attributs :

-cols : qui indique le nombre de caractères pouvant être placés sur une ligne dans afficher l'ascenseur horizontal.
-rows : qui indique le nombre de lignes pouvant être placées dans la textarea sans afficher l'ascenseur vertical.

Voici ce que ça donne en XHTML pour créer une textarea portant le nom de "message", contenant la valeur par défaut "Tapez ici votre message" et qui permet de taper 3 lignes et 50 caractères par ligne, sans afficher les ascenseurs :

```
form action
="page.php" method
="post">
p>label
>
Votre message :

textarea name
="message" rows
="3" cols
="50"

/>Tapez ici votre message

textarea
>
label
>
```

p>
form>

Si vous ne souhaitez pas placer de valeur par défaut dans la zone de texte multiligne, les deux balises ouvrant et fermant la textarea seront accolées. Ceci est parfaitement normal :

form action
="page.php" method
="post">
p>
label
>
Votre message :

textarea name
="message" rows
="3" cols
="50"

/>

textarea
>
label
>

p>
form>

Vous obtenez ceci à l'écran :

Les cases à cocher
Les cases à cocher, aussi appelées checkbox, sont souvent utilisées dans les sondages par exemple. On utilise cette fois la balise input qui prendra comme type la valeur "checkbox". Il faut comme toujours donner un nom à notre checkbox, un complément d'informations. Un attribut apparaît : **checked**. Vous allez pouvoir l'utiliser si vous souhaitez que votre case soit cochée ou non par défaut.

Voici l'exemple d'une case à cocher ayant pour nom "abonnement" et cochée par défaut :

form action
="page.php" method
="post">
p>
label

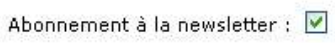
>
Abonnement à la newsletter :

input type
="checkbox" name
="abonnement" checked
="checked"

/>
label
>

p>
form>

Voici ce que ça donne à l'écran :



Les boutons radios
Les boutons radios vont être utiles lorsque vous aurez un choix d'options et que le visiteur ne pourra en choisir qu'une seule. Là aussi, le code est extrêmement simple puisqu'il se base encore une fois sur la balise input, et encore une fois, seul le type diffère. Il s'agit cette fois du type "radio". Voici ce que ça donne :

form action
="page.php" method
="post">
p>
S'abonner ?

p>
p>
label
>
Oui :

input type
="radio" name
="confirmation" checked
="checked" value
="oui"
/>
label

>

p> p
>label
>Non :

input

type
="radio"
name
="confirmation"
value
="non"

/>
label
>
p
>

form>

Il est important de noter que les boutons radios doivent avoir le même nom pour pouvoir "communiquer" ensemble, c'est à dire que si le visiteur sélectionne "Oui", l'option "Non" sera désélectionnée. Si vous souhaitez gérer plusieurs zones d'options (deux questions par exemple), il suffit de donner des noms différents à vos boutons radios qui n'appartiennent pas à la même question. Ici, l'option "Oui" sera sélectionnée par défaut grâce à l'attribut checked
. Voilà ce que vous obtenez à l'écran :

Les listes déroulantes

Les listes déroulantes sont utiles quand vous avez beaucoup de choix disponibles et une faible place en hauteur pour y mettre des listes d'articles avec une case à cocher à côté par exemple. Il s'agit cette fois d'une nouvelle balise que nous allons utiliser. Cette balise porte le nom de . Elle dispose toujours d'un attribut name. Pour indiquer les options que l'utilisateur peut choisir, on utilise des balises qui portent comme attributs "value" et si il s'agit de l'attribut par défaut, "selected". Nous allons faire un exemple regroupant tout cela :

form action
="page.php" method
="post">

label>Votre couleur préférée :

select name
="couleur">
option value
="rouge"
>
Rouge

option>
option value
="vert"
>Vert

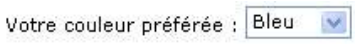
option
> option value
="bleu" selected
="selected"
>Bleu

option
>

select
>
label
>

form>

Ici, on demande quelle est la couleur préférée du visiteur. Par défaut, la couleur sélectionnée est le bleu. Voici ce que ça donne à l'écran :



Et si vous souhaitez que l'utilisateur puisse sélectionner plusieurs couleurs ? Il va falloir utiliser l'attribut **multiple** . Voici un exemple :

form action
="page.php" method
="post">
label>Votre (ou vos) couleur(s) préférée(s) :

select name

="couleur" multiple
="multiple">
option value
="rouge"
>
Rouge

option>
option value
="vert"
>Vert

option
> option value
="bleu" selected
="selected"
>Bleu

option
>

select
>
label
>

form>

Le visiteur pourra à l'aide de la touche CTRL sélectionner plusieurs valeurs. Voyons maintenant comment créer des groupes de valeurs. On utilise pour cela la balise **optgroup>**
qui porte un attribut **label**
permettant d'indiquer des informations supplémentaires. Voici comment ça marche :

form action
="page.php" method
="post">
label>Votre (ou vos) couleur(s) préférée(s) :

select name
="couleur" multiple
="multiple">
optgroup label
="Tons bleus"
>

option value
="bleu_clair"
>Bleu clair

option
> **option**
value
="bleu_fonce"
>Bleu foncé

option
>

optgroup
>
option **value**
="rouge" **selected**
="selected"
>
Rouge

option
>

select
>
label
>

form>

La soumission des formulaires
Une fois votre formulaire préparé, il faut que le visiteur puisse l'envoyer. Pour ce faire, on utilise un bouton qui se code avec une balise input de type "submit". L'attribue "value" comporte quant à lui le texte qui sera affiché pour envoyer le formulaire.

Voici ce que ça donne avec un formulaire comportant une zone de texte simple :

form action
="page.php" **method**
="post">
p>label
>

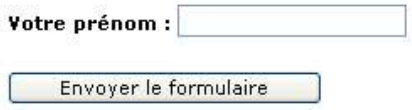
Votre prénom :

```
input type
="text" name
="prenom"

/>
label
>
p>
p
>input
type
="submit"
value
="Envoyer le formulaire"
/>
p
>

form>
```

Voici ce que vous obtenez à l'écran :



Chapitre suivant

-Annexes

Transférez votre site web chez un hébergeur

Optimiser ses images pour le Web

Validez le code XHTML de vos pages web

CSS et design web