Apprendre le langage Html

Les formulaires

Table des matières

Les formulaires

Chapitre 1 : L'interactivité (cgi / mailto / Javascript)

1.1 L'interactivité, mais comment ?

Avec les formulaires, Html vous ouvre les portes de l'interactivité et vous permet de recevoir des informations provenant directement de votre lecteur et éventuellement de lui répondre directement.

Mais encore fallait-il trouver le moyen de renvoyer cette information! En effet, lorsque vous surfez, vous demandez à votre provider (serveur connecté au Web) de vous envoyer la page se trouvant à l'adresse (URL) indiquée. Et vous consultez la page, sur votre ordinateur, en mode **read only** soit en lecture seule.

Pour sortir de la page ou de votre ordinateur, les moyens disponibles sont :

- permettre à certains utilisateurs triés sur le volet, généralement des professionnels, d'écrire sur le serveur et d'en exploiter les ressources. Tout ceci dans les meilleures conditions de sécurité pour le provider. C'est la procédure des CGI.

- utiliser une autre ressource d'Internet disponible pour l'utilisateur, c'est le courrier électronique ou l'e-mail. C'est la procédure mailto.

- une dernière procédure (à laquelle je tiens) qui permet de transférer les informations en interne, à l'intérieur d'une page ou d'un site Web, C'est le Javascript.

1.2 CGI

La procédure de CGI [Common Gateway Interface] est quelque chose de complexe. Nous y consacrons un chapitre, plus loin dans ce site.

Citons quand même ici qu'une CGI est écrite dans un langage de programmation (C, C++, Perl...). Ce programme est alors chargé sur le disque dur du serveur dans un répertoire bien déterminé et qui, généralement, s'appelle
cgi-bin. L'accès à ce répertoire est limité par l'administrateur du serveur, pour d'évidentes raisons de sécurité, aux seuls utilisateurs habilités.
Une fois chargée sur le serveur, on peut y accéder de façon classique par l'URL "http:// www.nom du serveur/ cgi-bin/ ma_cgi.pl". Le programme de la CGI s'effectue en utilisant les ressources de l'ordinateur du serveur, éventuellement pour vous préparer une réponse immédiate (avec si le programme est mal conçu, un risque de plantage complet du serveur lui-même).

1.3 Mailto

Netscape avec Navigator 3 a imaginé le premier ce moyen d'exploiter les formulaires et qui présente l'énorme avantage de pouvoir être utilisé par tout un chacun. Depuis, cette possibilité a été reprise par Microsoft Explorer 4 et bien entendu Netscape 4 (Communicator).

Ainsi, la procédure d'envoi de formulaires par le protocole mailto, ne fonctionne pas sous Microsoft Explorer 3.0. Permettez-moi d'insister pour vous éviter des interrogations inutiles :

Mailto ne fonctionne pas sous Microsoft Explorer 3.0 !

Les formulaires présentent l'avantage par rapport au simple courrier électronique de pouvoir prédéfinir ou de structurer l'information qui vous intéresse. Cette prédéfinition des données est très utilisée dans les applications commerciales du Web

1.4 Javascript

Avec du Javascript, on peut utiliser les formulaires pour transférer des informations à l'intérieur d'une page ou même à l'intérieur d'un site (par l'usage des frames). En outre, Javascript, propose des outils particulièrement adaptés pour la vérification des données introduites par l'utilisateur dans les formulaires avant l'envoi et le traitement de celles-ci.

Vous pouvez en savoir plus sur le Javascript ou sur les formulaires avec Javascript en consultant "Apprendre le Javascript" du même auteur (www.ccim.be/ccim328/js/index.htm).

Chapitre 2 : Définition d'un formulaire

2.1 Définition

Avant de pouvoir utiliser les différentes sortes de formulaires (ligne de texte, liste déroulante, cases à cocher...), il faut déclarer au browser qu'il devra gérer des formulaires et ce qu'il devra en faire.

```
<FORM method="post" action="URL d'expédition" enctype="text/plain">
... les formulaires proprement dit ...
</FORM>
```

2.2 Commentaires

- L'attribut "method" vous offre le choix entre get et post. La différence entre ces deux méthodes repose sur la façon dont les données seront transmises au serveur et exploitées par celui-ci. Avec le temps, la méthode post s'est imposée car elle apparaît plus efficace et permet le traitement d'une quantité plus importante de données.

- L'attribut "action" spécifie l'adresse d'expédition des données.

Dans le cas d'un traitement des données par une CGI, on spécifie le répertoire CGI du serveur et le nom de la CGI.
<FORM method="post" action="http://www.serveur/cgi-bin/ma_cgi.pl">

Dans le cas d'un envoi vers en adresse électronique (e-mail), on utilise le protocole mailto: suivi de l'adresse électronique de destinataire (généralement votre adresse e-mail).
<FORM method="post" action="mailto:Vanlancker.Luc@ccim.be">
(sans espace entre mailto: et l'adresse e-mail !)

- L'attribut "enctype" (optionnel) spécifie l'encodage utilisé pour le contenu du formulaire. Ce paramètre ne peut être utilisé qu'accompagné par la méthode post. Ainsi enctype="text/plain" encode le contenu du formulaire en format texte lisible par le destinataire. Cette option est particulièrement adaptée à l'action du type mailto.

- Il n'est pas inutile de prévoir l'attribut name="nom" si la page comporte plusieurs formulaires.

- Attention !!! Je suis certain que, emporté par votre impatience à encoder les formulaires, vous allez oublier la balise de fin </FORM>. Dans ce cas, à la visualisation dans le navigateur, rien ne sera affiché.

- Dans le cas de l'utilisation en interne des formulaires par du Javascript, les attributs method, action et enctype sont inutiles car on ne fait pas appel au serveur.

Chapitre 3 : Ligne de texte

3.1 Définition

INPUT type="text" indique un champ de saisie d'une seule ligne. C'est assurément le formulaire le plus simple à mettre en oeuvre :

```
<FORM>
<INPUT type="text" name="nom" size="50">
</FORM>
```

3.2 Commentaires

L'attribut name="nom du formulaire" est quasiment obligatoire car on n'utilise que rarement un seul formulaire. Le nom va identifier la chaîne de caractères du champ de saisie. De façon schématique, nom = (ce qui est introduit dans la ligne de texte).

- L'attribut size (optionnel) définit la longueur du champ de saisie. Notons que l'on peut introduire un nombre de caractères plus élevé que celui de la longueur.

- Il existe l'attribut maxlength="x" (optionnel) qui limite le nombre réel de caractères que l'on peut introduire dans le champ de saisie. La balise <INPUT> n'a pas de balise de fin.

Chapitre 4 : Zone de saisie

4.1 Définition

La balise <TEXTAREA></TEXTAREA> introduit une zone de texte multilignes et non plus une simple ligne de texte. La syntaxe est :

```
<FORM>
<TEXTAREA name="nom" rows=4 cols=40>Valeur par défaut</TEXTAREA>
</FORM>
```

4.2 Commentaires

- L'attribut name permet de donner un nom au formulaire.
- L'attribut rows=x détermine le nombre de lignes de la zone de texte.
- L'attribut cols=y détermine le nombre de caractères visibles sur chaque ligne ou si vous préférez le nombre de colonnes.
- L'attribut wrap (optionnel) détermine la façon dont les sauts de ligne seront traités lors d'un changement de ligne.

  * Avec wrap=virtual, les changements de lignes sont effectués automatiquement dans la zone de texte mais le tout sera transmis en une seule ligne.
  * Avec wrap=physical, les changements de lignes sont effectués automatiquement dans la zone de texte et ceux-ci sont également transmis.
  * Avec wrap=off, il n'y a aucun changement de ligne.

- Attention !!! La balise <TEXTAREA> a une balise de fin, soit </TEXTAREA>.

Chapitre 5 : Liste déroulante

5.1 Définition

La balise <SELECT></SELECT> indique au browser l'usage d'une liste déroulante. Les éléments de la liste sont introduits par la balise <OPTION> ... (</OPTION> facultatif).

```
<FORM>
<SELECT name="nom" size="1">
<OPTION>lundi
<OPTION>mardi
<OPTION>mercredi
<OPTION>jeudi
<OPTION>vendredi
</SELECT>
</FORM>
```

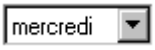
Si vous cliquez sur la petite flèche vers le bas, vous obtiendrez la liste déroulante où on retrouve les éléments de la liste (<OPTION>).

5.2 Commentaires

- L'attribut name="nom" définit le nom du formulaire.
- L'attribut size="x" détermine le nombre d'éléments de liste affiché dans la boite d'entrée. En fait, size="1" est optionnel car "1" est la valeur par défaut. Le même exemple avec size="3" donne :

- On peut présélectionner l'élément affiché dans la boite d'entrée (par défaut, le premier élément de la liste sera retenu). On utilise pour ce faire l'attribut selected de la balise <OPTION>. Pour faire afficher d'entrée mercredi au lieu de lundi notre exemple devient :
-

```
<FORM>
<SELECT name="nom" size="1">
<OPTION>lundi
<OPTION>mardi
<OPTION selected>mercredi
<OPTION>jeudi
<OPTION>vendredi
</SELECT>
</FORM>
```



- La balise <SELECT> a un balise de fin </SELECT> tandis que le balise de fin </OPTION> est facultative.

Chapitre 6 : Bouton d'option

6.1 Définition

Il serait plus logique de parler de boutons d'option car ils n'ont de sens que s'ils sont plusieurs. Ainsi on parle d'un groupe de boutons d'options.

Les boutons d'option, aussi appelés boutons radio, ont comme particularité qu'une seule option à la fois peut être activée (le "ou" exclusif).

La syntaxe de base est :

```
<FORM>
<INPUT type="radio" name="nom du groupe" value="valeur du bouton">
</FORM>
```

Ainsi, si on propose un choix entre, ou le tarif de jour ou le tarif de nuit ou le tarif de week-end, l'exemple devient

```
<FORM>
<INPUT type= "radio" name="tarif" value="jour"> tarif de jour
<INPUT type= "radio" name="tarif" value="nuit"> tarif de nuit
<INPUT type= "radio" name="tarif" value="week-end"> tarif de week-end
</FORM>
```

○ tarif de jour ○ tarif de nuit ○ tarif de week-end

Commentaires

- Vous avez compris que l'attribut name="nom" doit avoir le même nom pour tout le groupe de boutons d'option.
    - L'attribut "checked" (optionnel) permet de présélectionner un bouton radio. Ainsi

<INPUT type= "radio" name="tarif" value="jour" checked> tarif de jour

présenterait le bouton radio "tarif de jour" en position présélectionnée.

◉ tarif de jour ○ tarif de nuit ○ tarif de week-end

- Le contenu de l'attribut "value" du bouton retenu sera renvoyé par mailto ou utilisé par le Javascript.
- La balise <INPUT> n'a pas de balise de fin.
- Pour la petite histoire le terme radio ferait référence aux anciens postes de radio sur lesquels le fait d'appuyer sur un bouton désactivait le bouton précédemment enfoncé.

Chapitre 7 : Case à cocher

7.1 Définition

La philosophie des cases à cocher [checkbox] est assez similaire aux boites d'option. Ici, cependant, plusieurs choix simultanés peuvent être réalisés.

La syntaxe de base est :

```
<FORM>
<INPUT type="checkbox" name="nom" value="valeur attachée au bouton">
</FORM>
```

Comme exemple :

```
<FORM>
<INPUT type="checkbox" name="choix1" value="1"> glace vanille
<INPUT type="checkbox" name="choix2" value="2"> chantilly
<INPUT type="checkbox" name="choix3" value="3"> chocolat chaud
<INPUT type="checkbox" name="choix4" value="4"> biscuit
</FORM>
```

☐ glace vanille ☐ chantilly ☐ chocolat chaud ☐ biscuit

7.2 Commentaires

- Les règles pour l'attribut name="nom" sont moins précises que pour les boutons d'option. Vous pouvez employer des noms identiques ou des noms différents pour chaque case à cocher. Cependant des noms différents sont nécessaires pour l'utilisation d'un script.

- L'attribut checked (optionnel) permet de présélectionner une case à cocher. Ainsi

<INPUT type="checkbox" name="choix1" value="1" checked > glace vanille

présenterait la case à cocher "glace vanille" en position présélectionnée.

☑ glace vanille ☐ chantilly ☐ chocolat chaud ☐ biscuit

- Le contenu de l'attribut "value" du ou des boutons retenus seront renvoyés par mailto ou utilisé par le Javascript.

- La balise <INPUT> n'a pas de balise de fin.

Chapitre 8 : Bouton de commande

8.1 Définition

Avec l'introduction des langages de scripts (Javascript et VBscript) l'usage du bouton de commande présente un intérêt certain. Simplement à titre d'illustration ou pour vous donner envie d'en savoir plus sur le Javascript, je vous propose d'en découvrir la syntaxe :

```
<FORM>
<INPUT type="button" name="nom" value="texte du bouton" onclick="fonction Javascript">
</FORM>
```

Voyons un petit exemple.

```
<FORM>
<INPUT type="button" name="nom" value= "Bouton de test" onclick="alert('Test réussi !')">
</FORM>
```

En cliquant sur le bouton "Bouton de test", on va déclencher une fonction Javascript élémentaire [mon cher Watson] qui consiste à afficher une petite boite (dite d'alerte) avec le texte "Test réussi!".

8.2 Commentaires

- L'attribut "value" permet d'adapter le texte du bouton à vos souhaits.
- Avec le bouton de commande, il n'est pas nécessaire d'avertir le browser qu'on utilisera du Javascript par une balise du genre <SCRIPT language="javascript">. En effet, le navigateur (compatible Javascript) reconnaît par défaut les fonctions en Javascript. Par contre, si vous utilisez du VBscript, il faudra utiliser l'encodage <INPUT type="button" name="test" value="Pour un test" language="VBscript" OnClick="MsgBox 'Test réussi!'"> (Pour rappel, Javascript par défaut). Mais je m'emporte et je m'éloigne ainsi des objectifs de ce chapitre...

Chapitre 9 : Submit et Reset

9.1 Submit

Le bouton Submit a la tâche spécifique de transmettre toutes les informations contenues dans le formulaire à l'URL désignée dans les attributs ACTION et METHOD du tag <FORM>.

la syntaxe Html est :

```
<FORM>
<INPUT TYPE="submit" NAME="nom" VALUE="texte du bouton">
</FORM>
```

Soit par exemple :

```
<FORM>
<INPUT TYPE="submit" NAME="nom" VALUE=" Envoyer ">
</FORM>
```

Commentaires

- Les modifications seront peu nombreuses car le bouton Submit a une fonction Html très spécifique. Seul le texte du bouton pourra être modifié (par défaut Submit).
- La balise <INPUT> n'a pas de balise de fin.

9.2 Reset

Le bouton Reset permet d'annuler les modifications apportées aux contrôles d'un formulaire et de restaurer les valeurs par défaut.

la syntaxe Html est :

```
<FORM>
<INPUT TYPE="reset" NAME="nom" VALUE="texte du bouton">
</FORM>
```

Soit par exemple :

```
<FORM>
<INPUT TYPE="reset" NAME="nom" VALUE=" Annuler ">
</FORM>
```

Commentaires

- Les modifications seront peu nombreuses car le bouton Reset a une fonction Html très spécifique. Seul le texte du bouton pourra être modifié (par défaut Reset).
- La balise <INPUT> n'a pas de balise de fin.

Chapitre 10 : Un livre d'or

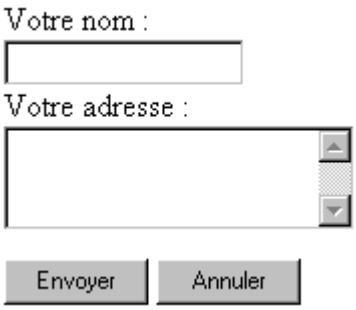
Une des premières applications des formulaires est celle d'un livre d'or [guestbook]. Vous trouverez ci-après un exemple complet de l'utilisation des formulaires par mailto et qui peut être adapté par chacun pour en faire "son" livre d'or.

N'oubliez pas que mailto ne fonctionne pas sous Microsoft Explorer 3.0 mais seulement sous Explorer 4.0 et Netscape 3.0 et plus. Pour être précis, mailto sous Explorer 3.0 n'entraîne pas de message d'erreur mais ouvre simplement une fenêtre de courrier électronique sans tenir compte du formulaire.

Pour ma part, j'ai retiré la page de livre d'or de mon site car à mon avis, pour un site d'amateur ou d'hobbyiste, une prédéfinition de l'information n'était pas forcément adaptée. Il me semble qu'une libre expression du lecteur (concise ou plus étendue) était de loin plus enrichissante. En outre, certains ne tiennent pas à transmettre leur nom ou leur domicile et souhaitent ne s'en tenir qu'à leur adresse émail.

Voici donc un exemple de livre d'or que vous pouvez adapter :

```
<FORM method="post" action="mailto:votre_adresse_E-mail">
Votre nom :<BR>
<INPUT type="text" name="nom"><BR>
Votre adresse :<BR>
<TEXTAREA name="adresse" ROWS=2 COLS=35>
</TEXTAREA><BR>
<INPUT type="submit" value="Envoyer">
<INPUT type="reset" value="Annuler">
</FORM>
```



Vous recevrez dans notre boite de réception, un truc bizarre du genre :

nom=Van+Lancker+Luc&adresse=Rue+des+Brasseurs+2217OD%OA7700+Mouscron.

- où on retrouve les champs nom= et adresse=, •où les champs sont séparés par le signe &,
- où les espaces sont remplacés par le signe +
- et 17%OD%OA correspond à un passage à la ligne.

Il existe heureusement des programmes comme "mailto: Formatter" pour vous aider à décoder tout ceci.

Van Lancker Luc
Rue des Brasseurs, 22
7700 Mouscron
Vanlancker.Luc@ccim.be