Cahiers GUTenberg

WORLD-WIDE WEB, FORMULAIRES ÉLECTRONIQUES, IMAGES RÉACTIVES, ETC.

François DAGORN

Cahiers GUTenberg, nº 19 (1995), p. 59-66.

<http://cahiers.gutenberg.eu.org/fitem?id=CG_1995___19_59_0>

World-Wide Web, formulaires électroniques, images réactives, etc.*

François Dagorn

Centre de Ressources en Informatique
Université de Rennes 1
E-mail: Francois.Dagorn@univ-rennes1.fr

Résumé. Cet article présente en détail les mécanismes mis en œuvre pour la réalisation de formulaires électroniques et d'images réactives dans un environnement WWW.

Abstract. *This paper details the stuffs used to realize electronic forms or clickable image maps within the World Wide Web.*

1. Introduction

Les « clients WWW » sont en général utilisés pour accéder à des documents diffusés par des serveurs d'informations. En sélectionnant une ancre dans un texte, l'utilisateur déclenche le transfert de données dans le sens serveur-client... Ils peuvent également servir à communiquer des informations dans l'autre direction et être utilisés pour transmettre des paramètres à un serveur HTTP. L'émission de formulaires électroniques et la présentation d'images réactives utilisent ces propriétés.

2. Protagonistes

Depuis l'émission d'un formulaire électronique ou d'une interaction avec une image réactive, jusqu'à l'accusé de réception du traitement adéquat par un serveur HTTP, les protagonistes sont les suivants :

- HTML qui propose les éléments <FORM>, <IMG> et <ISINDEX> respectivement pour le traitement des formulaires, des images réactives et la communication d'un mot-clef à un moteur de recherche ;

*. Présentation réalisée à Nanterre, le 19 janvier 1995 lors de la journée *Diffusion des documents électroniques.*

- HTTP (et plus particulièrement la version HTTP/1.0) qui procure les méthodes `POST` et `GET` pour transmettre des paramètres à un serveur WWW ;
- les serveurs HTTP qui permettent l'appel de programmes externes pour traiter les informations transmises par les clients. Une interface standardisée appelée CGI (*Common Gateway Interface*) précise les règles d'écriture et d'exécution des procédures (les *CGI scripts*). Les principaux serveurs HTTP respectent la convention CGI.

3. Traitement des formulaires : un exemple

Notons que l'exemple ci-dessous suppose l'utilisation d'un serveur HTTP sur une plate-forme Unix.

3.1. Code HTML

Le premier élément conduisant au traitement d'un formulaire est la présentation HTML de ce dernier comme dans l'exemple suivant :

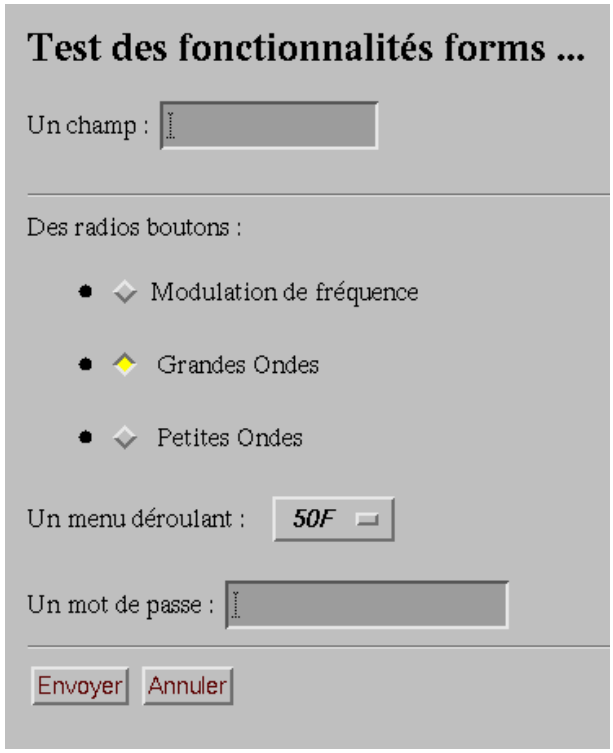
```
<HTML>
<HEAD>
<TITLE>Test des fonctionnalités forms ...</TITLE>
</HEAD>
<BODY>
<H1>Test des fonctionnalités forms ...</H1>
<FORM ACTION="http://www.univ-rennes1.fr/test-cgi-form.cgi"
      METHOD="POST">
      Un champ : <INPUT NAME="nom" SIZE=30>
      <P>
      <HR>
      Des radios boutons : <P>
      <UL>
      <LI> <INPUT TYPE="radio" NAME="mode" VALUE="FM">Modulation de fréquence<P>
      <LI> <INPUT TYPE="radio" NAME="mode" VALUE="GO" CHECKED> Grandes Ondes<P>
      <LI> <INPUT TYPE="radio" NAME="mode" VALUE="PO"> Petites Ondes <P>
      </UL>
      Un menu déroulant :
      <SELECT NAME="menu">
      <OPTION SELECTED>50F
      <OPTION>60F
      <OPTION>70F
      <OPTION>100F
      </SELECT><P>
      Un mot de passe : <INPUT TYPE="password" NAME="passwd" SIZE="40">
      <HR>
      <INPUT TYPE="submit" VALUE="Envoyer">
      <INPUT TYPE="reset" VALUE="Annuler">
</FORM>
</BODY>
</HTML>
```

La description des formulaires fait partie de HTML niveau 2, elle est bornée par `FORM` et `/FORM`. L'attribut `ACTION` donne ici l'URL de la procédure qui va

traiter les champs du formulaire (une fois saisis), par défaut c'est l'URL courant (celui qui a servi pour afficher le formulaire) qui est sollicité.
L'attribut `METHOD` fait référence à HTTP et indique quelle est la méthode qui sera utilisée pour transmettre les champs du formulaire. La méthode `POST` est conseillée, elle transmet les champs dans le corps d'un message MIME. La méthode `GET` transmet les paramètres dans l'URL, divers problèmes liés à la taille et au nombre des champs peuvent survenir.
La description des champs du formulaire est réalisée par des éléments `INPUT`, `SELECT` et `TEXTAREA` qui permettent l'utilisation de champs textuels, radio boutons, menus déroulants ...
Le formulaire est transmis à l'URL indiqué par `ACTION` grâce à l'activation du bouton de commande de `TYPE` submit.

3.2. Formulaire interprété par un client WWW

Un client WWW ayant reçu le code HTML de l'exemple précédent affiche la page suivante :



3.3. Transmission par HTTP du formulaire

Lorsque l'émission du formulaire est demandée, le client effectue une requête HTTP comme dans l'exemple suivant :

```
POST /test-cgi-form.cgi HTTP/1.0
Accept: text/plain
Accept: application/x-html
Accept: application/html
Accept: application/postscript
..........
Accept: */*
User-Agent:  NCSA Mosaic for the X Window System/2.4  libwww/2.12 modified
Content-type: application/x-www-form-urlencoded
Content-length: 61
nom=toto+tutu+titi&mode=GO&menu=50F&passwd=en+clair
```

Les champs sont transmis derrière l'attribut MIME `Content-length`, ils sont séparés entre eux par le caractère &, les espaces sont remplacés par des +, les éventuels &, /, etc. sont codés et remplacés par leur code Ascii précédé du caractère %.

En réceptionnant cette requête `POST` le serveur HTTP ciblé passera le contrôle à la procédure citée en argument (`/test-cgi-form.cgi` ici) en lui transmettant les champs du formulaire sur son entrée standard. Ce procédé est propre à la méthode `POST`, dans le cas de la méthode `GET` les champs sont récupérés dans une variable d'environnement.

3.4. Procédure émettant et réceptionnant le formulaire

Les programmes se déroulant à l'initiative d'un serveur HTTP (les *CGI scripts*) peuvent être écrits dans n'importe quel langage produisant des fichiers exécutables, l'interpréteur de commande `csh` est utilisé dans ce qui suit.

3.4.1. Procédure test-cgi-form.cgi

```
#!/bin/csh -f

#   appel de la procedure get-param, elle transmet les parametres séparés
#   par des espaces.

set argus = `/usr/local/www/cgi-bin/get-param`

echo 'Content-Type: Text/html'
echo ''

switch ($argus[1])
 case Nothing : # affichage du formulaire
  cat <<FINHTML
  <HTML>
  <HEAD>
  <TITLE>Test des fonctionnalités forms ...</TITLE>
  </HEAD>
```

```
  <BODY>
  <H1>Test des fonctionnalités forms ...</H1>
  <FORM ACTION="http://www.univ-rennes1.fr/test-cgi-form.cgi"
     METHOD="POST">
     Un champ : <INPUT NAME="nom" SIZE=30>
     <p>
     <HR>
     Des radios boutons : <P>
     <UL>
       <LI> <INPUT TYPE="radio" NAME="mode" VALUE="FM" >Modulation de
fréquence <P>
       <LI> <INPUT TYPE="radio" NAME="mode" VALUE="GO" CHECKED> Grandes
Ondes<P>
       <LI> <INPUT TYPE="radio" NAME="mode" VALUE="PO"> Petites Ondes <P>
     </UL>
     Un menu déroulant :
     <SELECT NAME="menu">
       <OPTION SELECTED>50F
       <OPTION>60F
       <OPTION>70F
       <OPTION>100F
     </SELECT><P>
     Un mot de passe : <INPUT TYPE="password" NAME="passwd" SIZE="40">
     <HR>
     <INPUT TYPE="submit" VALUE="Envoyer">
     <INPUT TYPE="reset" VALUE="Annuler">
   </FORM>
</BODY>
</HTML>
FINHTML
     breaksw
 default : # traitement des parametres
     echo '<HTML>'
     echo '<TITLE>Vos parametres étaient</TITLE>'
     echo '<HR>'
     echo '<H1> Vos parametres étaient </H1>'
     echo $#argus paramètres : $argus
     echo '<p>'
     echo 'Apres traitement des caracteres d echappement :'
     @ i = 1
     while !($i > $#argus)
       # bien noter l'appel de unescape pour les espaces et caracteres speciaux
       echo '<P>'
       echo ' ===>' `/usr/local/www/cgi-bin/unescape $argus[$i]`
      @ i++
     end
     echo '<HR>'
     echo '</HTML>'
endif
```

La procédure est appelée de deux façons :

- à la suite d'une requête HTTP de type `GET`. Dans ce cas, il s'agit d'une demande d'émission du formulaire vers le client, la procédure get-param retourne la valeur `NOTHING`, l'envoi du texte HTML décrivant le formulaire est effectué ;
- à la suite d'une requête de type `POST`, la variable `argus` contient les champs séparés par des espaces (dépouillés de leurs noms ici), le traitement du formulaire consistant à vérifier le contenu des champs reçus et à faire part au

client de leurs valeurs. Avant d'utiliser les champs contenus dans la variable `argus` il est nécessaire d'effectuer le décodage des caractères d'échappement, c'est le rôle de la procédure `unescape`.

Le document retourné au client WWW peut être de n'importe quel type. On précise ici `Content-Type: Text/html` suivi d'une ligne vide. La présence de la ligne vide est très importante, elle stipule au client que des éléments de HTTP (`Last-modified`, `Content-length`,...) sont omis. Si cette ligne vide est absente, le client ne sait pas où débutent les spécifications HTML du document résultant.

3.4.2. Procédure get-param

Le rôle de `get-param` est de transmettre les champs du formulaire à la procédure `test-cgi-form.cgi`. Les champs sont donc récupérés sur l'entrée standard, dépouillés de leur nom (ce qui n'est pas tout à fait judicieux) et écrits sur la sortie standard. La commande `get-param` telle que donnée ci-dessous ne réalisant qu'un sous-ensemble des fonctionnalités d'une procédure de traitement des paramètres CGI, elle ne doit donc pas être sortie du contexte de cet exemple :

```
#!/bin/csh

#  get-param sert a decoder les champs d'un formulaire. Ils sont rendus sur
#            la sortie standard separes par des espaces.
# FD (9/11/94)

#  Si la methode n'est pas POST, il s'agit du premier appel (GET), il faut
#     simplement afficher le formulaire en indiquant qu'il n'y a pas de
#     parametres.

if ($REQUEST_METHOD == GET) then
        echo Nothing
exit
endif

#
#  Appel sans parametres ?  (methode POST sans arguments ... hum !)
#
if ($CONTENT_LENGTH == "") then
echo Nothing
exit
endif

#
#  Le formulaire (rempli) vient d'etre emis, il faut receptionner les
#      paramètres et les restituer séparés par des espaces.
#
set param=$<
set argus=`echo $param | awk -F'&' '{i=1; while (i < NF) {print $(i); i++} ;
print $(NF)}'`

foreach var ($argus)
set val = `echo $var | awk -F'=' '{print $2}'`
echo -n  $val " "
end
```

Les variables `REQUEST_METHOD` et `CONTENT_LENGTH` sont des variables d'environnement (dites CGI) positionnées par le serveur HTTP. Elles ne sont pas les seules, les suivantes sont également utilisables :

- `QUERY_STRING` qui contient les paramètres émis par le client dans le cas d'une requête utilisant la méthode `GET` ;
- `REMOTE_HOST` qui indique (si possible) le nom de la machine cliente ;
- `REMOTE_ADDR` qui indique l'adresse IP de la machine cliente ;
- ...

3.4.3. Remarques

La mise au point des *CGI scripts* n'est pas aisée, les erreurs ne sont pas facilement décelables car ils se déroulent sous le contrôle d'un serveur HTTP qui redirige ses sorties. La réception et le décodage des valeurs des champs est laissée à la charge du programmeur et peuvent donc être sources de problèmes.
Des utilitaires sont disponibles pour le traitement des paramètres, par exemple les utilisateurs de PERL apprécieront la fonction `ReadParse` fournie par le package `cgi-lib`[1]. Elle restitue les paramètres dans un tableau `$in`, le traitement des caractères d'échappement et des espaces étant effectué au préalable.

4. Principes des images réactives

La présentation d'images réactives (parfois dites *cliquables*) par les clients WWW facilite la réalisation d'interfaces graphiques conviviales. Les principes de fonctionnement sont les suivants :

- une image à diffuser par un serveur HTTP est découpée en régions, à chacunes d'entre elles est associée une action à réaliser (sous la forme d'un URL). Pour découper une image en régions il est possible d'utiliser des outils (sur une plate-forme Unix) tels que `ImageMagick`[2] ou `xv`[3] la configuration des actions associées dépend du serveur HTTP utilisé (s'agissant du NCSA HTTPD, la lecture de *Graphical Information Map Tutorial*[4] est recommandée) ;
- l'image est présentée aux clients WWW par l'intermédiaire d'une ancre ayant la structure suivante :

```
<A HREF="URL d'une procédure"> <IMG SRC="path de l'image" ISMAP> </A>
```

1. ftp://ftp.urec.fr/pub/reseaux/services_inf os/WWW/ncsa/ncsa_httpd/cgi/
2. ftp://ftp.x.org/contrib/applications/ImageMagick
3. http://www.univ-rennes1.fr/pub/X11/contrib/xv-3.00.tar.Z
4. http://hoohoo.ncsa.uiuc.edu/ agoyal/mapping.html

C'est l'attribut `ISMAP` qui indique au client que l'image est réactive, et qu'il convient de noter les coordonnées du pointeur de la souris sur sollicitation de l'utilisateur ;

- les coordonnées de l'endroit pointé sont transmises au serveur HTTP derrière l'URL de la procédure de traitement (en utilisant une requête HTTP de type `GET`) comme dans l'exemple suivant :

```
GET /cgi-bin/imagemap/bretagne?542,228 HTTP/1.0
Accept: text/plain
Accept: application/x-html
Accept: application/html
..........
Accept: */*
User-Agent:  NCSA Mosaic for the X....
```

Dans l'exemple ci-dessus, `/bretagne` est un paramètre transmis à la procédure `/cgi-bin/imagemap`.

Remarques

Les interfaces utilisateurs utilisant des images réactives ne sont pas toujours bien comprises des usagers. En effet le client WWW n'a pas connaissance du découpage en régions, il ne peut donc pas signaler (par un effet vidéo) les zones sensibles de l'image.