WWW et les formulaires électroniques . . .

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 vers client . . .

Les clients WWW 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 ou la présentation d'images réactives utilisent ces propriétés.

Depuis l'émission d'un formulaire jusqu'à l'accusé de reception du traitement par un serveur HTTP, les protagonistes sont les suivants :

– **HTML** (niveau 2) qui propose la gestion de formulaires, . . .

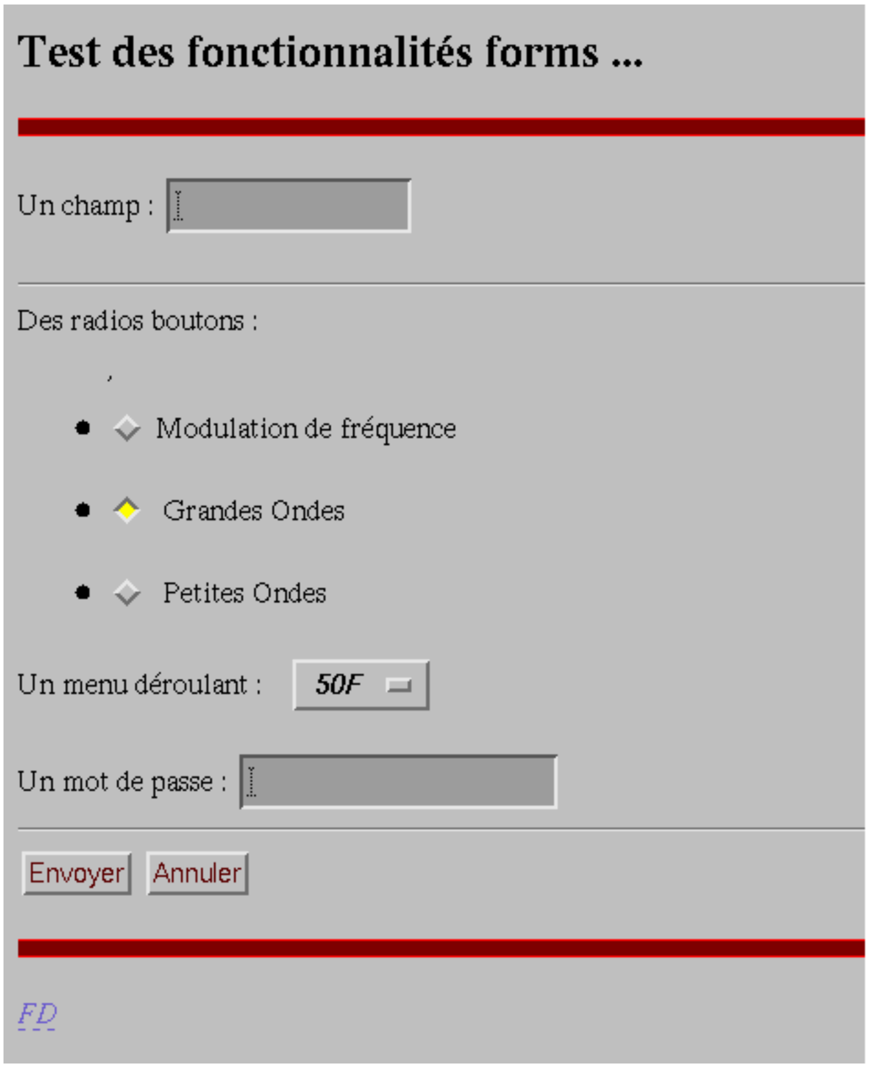
– **HTTP/1.0** qui procure les méthodes *POST* et *GET* pour transmettre des informations à un serveur **W3** ;

– les serveurs **W3** qui permettent l'appel de programmes externes pour traiter les informations transmises par les clients (*les scripts CGI*).

Un exemple de formulaire : le code HTML

```
<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>
```

L'exemple interprété



Test des fonctionnalités forms ...

Un champ :

Des radios boutons :

- Modulation de fréquence
- Grandes Ondes
- Petites Ondes

Un menu déroulant : 50F

Un mot de passe :

Envoyer Annuler

FD

La transmission par HTTP du formulaire

```
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
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 `Content-length`, ils sont encodés. Le serveur passe le contrôle à une procédure CGI.

CGI

Common Gateway Interface est une interface d'écriture/éxécution de programmes qui se déroulent sous le contrôle d'un serveur **W3**.

Les programmes (les *Gateway CGI*) peuvent être écrits dans n'importe quel langage produisant des fichiers exécutables, ils communiquent avec le serveur à l'aide de variables d'environnement.

Le serveur transmet au programme (*Gateway CGI*) des paramètres émis par les clients :

– dans la variable d'environnement **QUERY_STRING** si les paramètres ont été émis par la méthode **GET**;
– sur l'entrée standard si les paramètres ont été émis par la méthode **POST**;

CGI (suite)

Le programme peut émettre différents types de documents qu'il précise au moyen d'un type MIME (Text/Html, Text/Plain, . . . cf. HTTP/1.0) :

```
Content-type: text/html
+ une ligne vide

 .... les résultats ....
```

Le programme peut aussi retourner seulement une référence à un autre document en émettant :

```
Location: URL
```

Les variables d'environnement CGI

Utilisées par les serveurs **W3** pour communiquer avec les programmes :

- **REQUEST_METHOD** indique quel est la méthode utilisée pour émettre la requête (**GET**, **POST** ou **HEAD**) ;
- **QUERY_STRING** contient les paramètres émis par le client dans le cas d'une requête utilisant la méthode **GET**;
- **REMOTE_HOST** indique (si possible) le nom de la machine cliente ;
- **REMOTE_ADDR** indique l'adresse IP du client ;
- **CONTENT_LENGTH** précise le nombre de caractères constituant les paramètres émis par le client;
- . . .

Un programme CGI

```
#!/bin/sh

echo Content-type: text/plain
echo

echo CGI/1.0 test script report:
echo

echo argc is $#. argv is "$*".
echo

echo SERVER_SOFTWARE = $SERVER_SOFTWARE
echo SERVER_NAME = $SERVER_NAME
echo GATEWAY_INTERFACE = $GATEWAY_INTERFACE
echo SERVER_PROTOCOL = $SERVER_PROTOCOL
echo SERVER_PORT = $SERVER_PORT
echo REQUEST_METHOD = $REQUEST_METHOD
echo HTTP_ACCEPT = "$HTTP_ACCEPT"
echo PATH_INFO = "$PATH_INFO"
echo PATH_TRANSLATED = "$PATH_TRANSLATED"
echo SCRIPT_NAME = "$SCRIPT_NAME"
echo QUERY_STRING = "$QUERY_STRING"
echo REMOTE_HOST = $REMOTE_HOST
echo REMOTE_ADDR = $REMOTE_ADDR
echo REMOTE_USER = $REMOTE_USER
echo AUTH_TYPE = $AUTH_TYPE
echo CONTENT_TYPE = $CONTENT_TYPE
echo CONTENT_LENGTH = $CONTENT_LENGTH

if [$REQUEST_METHOD -eq POST] then
   read param
   echo Les parametres : $param
fi
```

Le résultat

```
CGI/1.0 test script report:

argc is 0. argv is .

SERVER_SOFTWARE = NCSA/1.2
SERVER_NAME = sir.univ-rennes1.fr
GATEWAY_INTERFACE = CGI/1.1
SERVER_PROTOCOL = HTTP/1.0
SERVER_PORT = 80
REQUEST_METHOD = POST
HTTP_ACCEPT = text/plain, text/html, audio/*, image/*, video/fli .....
PATH_INFO =
PATH_TRANSLATED =
SCRIPT_NAME = /cgi-bin/test-cgi
QUERY_STRING =
REMOTE_HOST = servix.univ-rennes1.fr
REMOTE_ADDR = 129.20.128.49
REMOTE_USER =
AUTH_TYPE =
CONTENT_TYPE = application/x-www-form-urlencoded
CONTENT_LENGTH = 62
les parametres :
nom=oups&mode=substr&maxhits=10&serveur=archie.univ-rennes1.fr
```

Synoptique des échanges

Client : demande l'acquisition d'un URL par un accès HTTP utilisant la méthode **GET** (la description HTML d'un formulaire) ;

Serveur : émission de l'URL demandé ;

Client : interprète le HTML reçu . . . affiche le formulaire ;

Client : émet (après remplissage) les champs du formulaire par une requête HTTP ciblant un URL par la méthode **POST** ;

Serveur : communique les champs saisis à la procédure référencée par l'URL. La procédure traite les paramètres (les champs du formulaire) et envoie un acquittement au client.

La procédure utilisée dans cet exemple

```
#!/bin/csh -f
set argus = `/usr/local/www/cgi-bin/get-param`

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

switch ($argus[1])
 case Nothing :
  cat <<FINHTML
  <HTML>
  <HEAD>
  <TITLE>Test des fonctionnalités forms ...</TITLE>
  </HEAD>
  <BODY>
  <H1>Test des fonctionnalités forms ...</H1>
  <p>
  <IMG SRC="/icons/baton-rouge.gif">
  <FORM ACTION="http://www.univ-rennes1.fr/cgi-bin/test-cgi-form"
        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>
    <IMG SRC="/icons/baton-rouge.gif">
    <P>
    <i> <A HREF="/fd.html">FD</A></i>
    </BODY>
    </HTML>
FINHTML
   breaksw
 default :
echo '<HTML>'
echo '<TITLE>Vos parametres étaient</TITLE>'
echo '<HR>'
echo '<H1> Vos parametres étaient </H1>'
echo $#argus paramètres : $argus
echo '<HR>'
echo '</HTML>'
endif
```

La procédure get-param

```
#!/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 != POST) 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. On appelle
#       pour cela la procedure unescape qui s'occupe des caracteres
#       d'echappement et des espaces codes en '+'.
#
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 `/usr/local/www/cgi-bin/unescape $val` " "
end
```

Formulaires et authentification

Avant de valider le contenu d'un formulaire, il faut au moins authentifier l'émetteur et si possible certifier le contenu.

Un canevas possible :

- avant d'accepter le formulaire, obliger l'émetteur à fournir une adresse électronique ;
- en receptionnant un formulaire valide, envoyer un courrier à l'émetteur. Il contient une clef qui permet de certifier le contenu du formulaire (**md5**);
- l'émetteur répond (et communique la clef) ;
- une procédure (au vu de la clef) retrouve le message d'origine et traite ensuite le contenu du formulaire.

Formulaires et authentification (suite)

Le mécanisme de base utilisé peut être testé de la façon suivante :

- `mail echo-authentique@univ-rennes1.fr`;
- réception d'une invite de confirmation ;
- à l'issue du *reply*, l'émetteur reçoit une confirmation.

La liste des correspondants sécurité du CRU fonctionne avec ce principe, pour plus de détails `aumont@univ-rennes1.fr`.

Netscape Communication Corporation propose un serveur HTTP ainsi qu'un client WWW sécurisé.

WWW et les images réactives

La présentation d'images réactives (*cliquables*) par les clients WWW facilite la réalisation d'interfaces graphiques conviviales :

– l'image est découpée en régions (sur un serveur HTTP), à chacunes d'entre elles est associée une action (un URL) ;

– l'image est présentée au client avec l'attribut `ISMAP` (par l'intermédiaire d'une ancre) ;

– le client transmet au serveur HTTP les coordonnées de l'endroit *cliqué*.

WWW et les images réactives (suite)

```
<A HREF="/cgi-bin/imagemap/France">
   <IMG SRC="/France/France.gif" ISMAP>
</A>
```

Quand l'utilisateur clique dans une zone de la carte présentée, le client émet la requête suivante :

```
GET /cgi-bin/imagemap/France?323,425
HTTP/1.0
```

WWW et les images réactives (suite)

Les interfaces utilisateurs utilisant des images réactives ne sont pas toujours bien comprises des usagers. 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.

Exemples:

http://web.urec.fr:/france/france.html

http://www.univ-rennes1.fr/ROR/renater.html