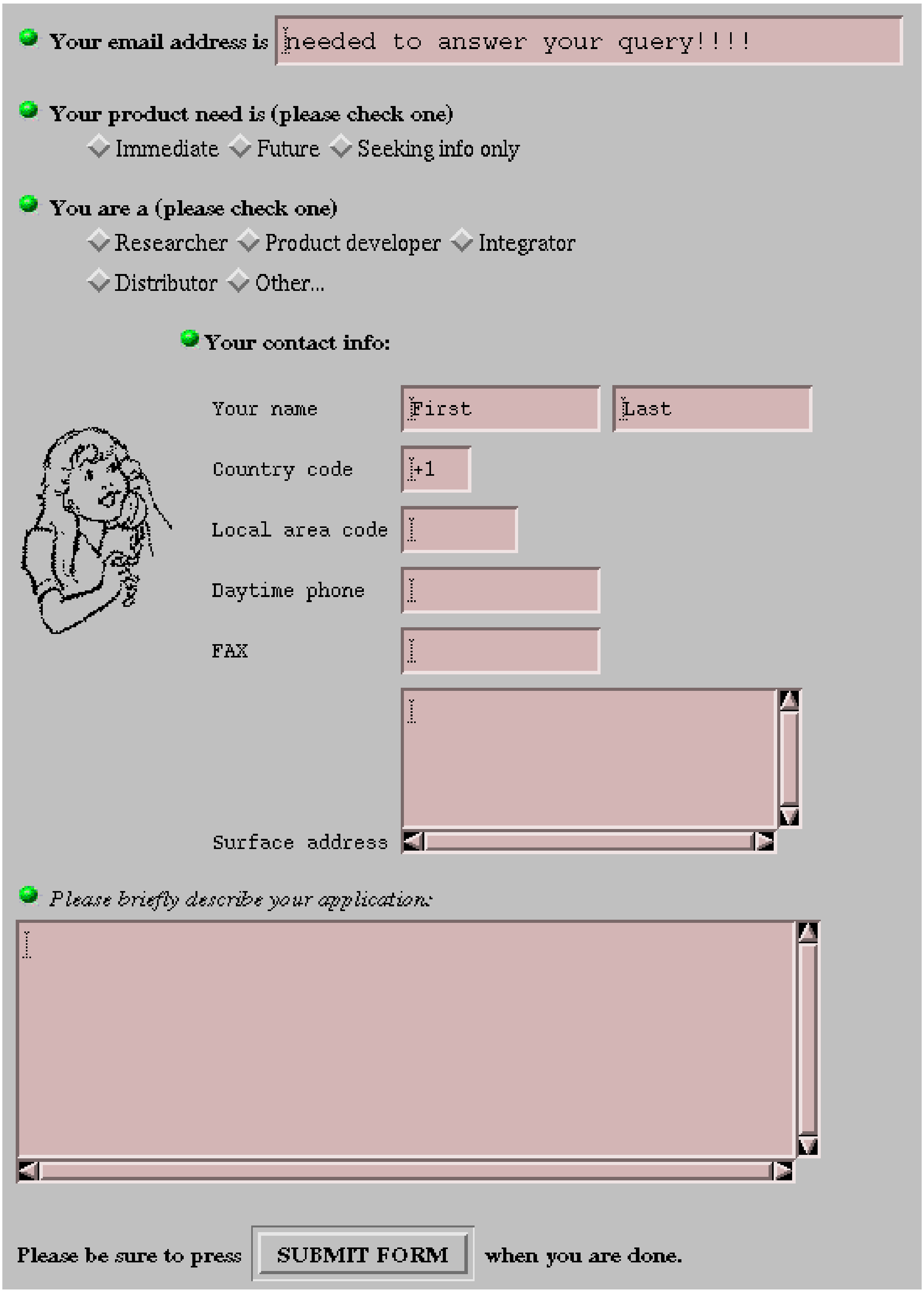
Internet

Pages interactives
Les programmes CGI

Michel Buffa

Université de Nice Sophia Antipolis
UFR Sciences
Parc Valrose, Nice
Téléphone : 92 96 5115 — Télécopie : 92 96 51 55
Email : `buffa@essi.fr`

Vers des pages WWW interactives


Your email address is needed to answer your query!!!!

Your product need is (please check one)
Immediate Future Seeking info only

You are a (please check one)
Researcher Product developer Integrator
Distributor Other...

Your contact info:

Your name First Last
Country code +1
Local area code
Daytime phone
FAX
Surface address

Please briefly describe your application:

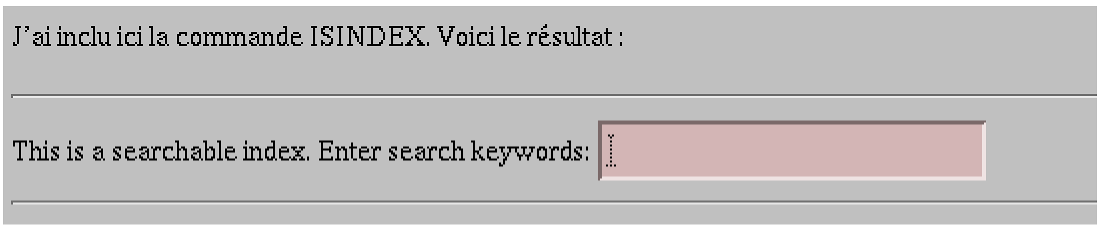
Please be sure to press SUBMIT FORM when you are done.

La plus simple méthode d'interaction
ISINDEX

- **La commande HTML <ISINDEX> est la manière la plus simple pour réaliser une interaction dans une page WWW.**

- **Ajouter la commande <ISINDEX> dans n'importe quelle page va ordonner au programme de consultation (Netscape, Mosaic) d'afficher une boîte de dialogue.**

```
...
J'ai inclu ici la commande ISINDEX.
Voici le résultat : <P>
<isindex>
...
```

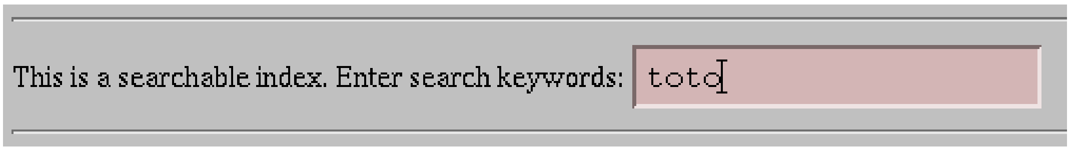


- **Si l'utilisateur entre du texte dans cette boîte, alors le texte sera renvoyé au serveur avec l'URL original.**

La plus simple méthode d'interaction *ISINDEX (suite)*

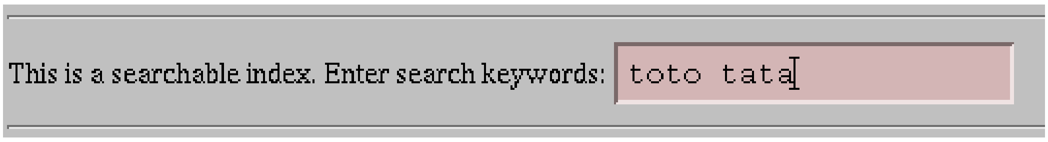
- **Entrer du texte dans la boîte et taper <RETURN> relance la relecture de la page.**

- **Le contenu de la boîte est ajouté à la fin de l'URL de la page.**
  - Exemples :



...rappelle l'URL :

`http://www.essi.fr/~buffa/isindex.html?toto`



...rappelle l'URL :

`http://www.essi.fr/~buffa/isindex.html?toto+tata`

- **Pour traiter les paramètres, un simple code HTML ne suffit plus !**
  - Il faut qu'un *programme* s'en charge.

La plus simple méthode d'interaction *ISINDEX (suite)*

- **On utilise *un programme CGI* (script, langage compilé) pour :**

1. parser les paramètres,
2. déclencher des actions (Exemple : consulter une BDD, etc...),
3. générer le code HTML pour renvoyer au client une page WWW avec les résultats de la recherche.

The Common Gateway Interface (CGI)

Ou comment écrire des pages interactives

- **CGI = un *standard* pour interfacer des applications externes avec un serveur WWW.**
  - Autorise la création de pages *dynamiques*
  - Par exemple: consultation de Base de données
- **Un programme CGI est exécuté par le serveur HTTPD**
- **Les résultats sont mis en forme par le CGI et renvoyés par le serveur au client (Ex: Netscape)**
- **Pas de limites -> puissant mais dangereux**
- **Un CGI doit s'exécuter rapidement**

Comment écrire un programme CGI ?

Attention à la sécurité

- **Le monde entier peut exécuter un programme CGI.**
- **Il faut prendre des précautions !**
- **En général, le webmaster décide de la politique à adopter.**
- **Les serveurs proposent différentes options :**
  - Pas de CGI
  - Les CGI résident dans un ou plusieurs répertoires sous le contrôle du webmaster. En général au moins /cgi-bin
  - Les CGI ont un suffixe spécial (exemple myprog.cgi)
  - Les CGI sont autorisés en exécution (bit x positionné).

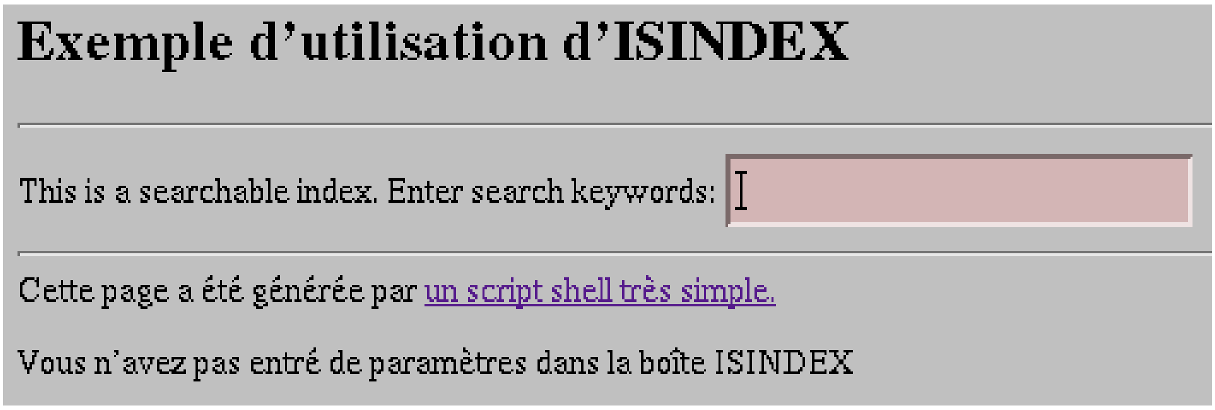
Comment écrire un programme CGI ?

Langages utilisables

- **Peu importe le langage, il existe des programmes CGI en :**
  - C/C++
  - Fortran
  - PERL
  - Shell unix : sh,ksh, csh, zsh, etc...
  - Applescript
  - Tcl
  - ...
- **Recommandés: PERL, C, Shell car il existe des librairies simples et performantes.**
- **Ne pas oublier qu'un script est plus facile à debugger et à maintenir.**
- **...mais PERL est encore 30 fois plus lent que le C !**

Etude d'un exemple simple

Cas de <ISINDEX>



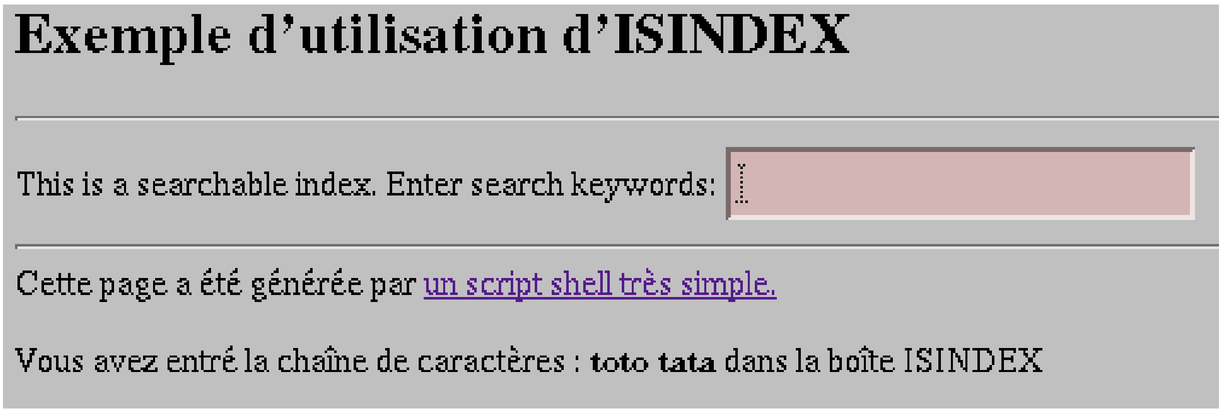

```
#!/bin/sh
echo Content-type: text/html
echo
# 1) Texte saisi = vide ?
if [ $# = 0 ]
then
message="Vous n'avez pas entre de parametres dans la boîte
ISINDEX"
else
message="Vous avez entré la chaine de caractères : $* dans
la boîte ISINDEX"
fi
# 2) On génère la page HTML avec les resultats
cat << EOM
<TITLE>Exemple d'utilisation d'ISINDEX</TITLE>
<H1>Exemple d'utilisation d'ISINDEX</H1>
<ISINDEX>
# affichage du texte saisi ou d'un message si texte vide
$message
EOM
```

Etude d'un exemple simple

Cas de <ISINDEX> (suite)

- **Exemples de résultats après saisie et validation d'une chaîne :**

  Après saisie de “toto tata”, la page est affichée à nouveau :



- **Avant d'aller plus loin avec les CGI, étudions comment développer en HTML des interfaces graphiques plus complexes...**

Les FORMs en HTML 3.0

Le tag FORM

- **Le tag FORM**

```
<FORM ACTION="url" METHOD="POST">
...description de l'interface...
</FORM>
```

- **Plusieurs tags FORM possibles dans un même document, mais séparés.**

- **Les attributs du tag FORM**
  - ACTION: indique l'URL du cgi qui va traiter le contenu des FORMS de la page courante. Si absent = URL de la page.
  - METHOD: choix de la méthode utilisée pour la transmission des données.
    - METHOD = "GET" : valeur par défaut, le contenu des FORMS sera ajouté à la fin de l'URL.
    - METHOD = "POST" : le contenu des FORMS sera envoyé séparément de l'URL du cgi précisé par le tag ACTION.

    La doc officielle recommande l'utilisation de la méthode POST.

  - ENCTYPE : spécifie la méthode d'encodage des données. Inutile pour le moment, la valeur par défaut est la seule admissible avec HTTP/1.0.

Les FORMs en HTML 3.0

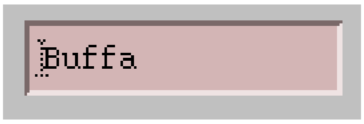
Le tag INPUT

- **Spécifie un élément permettant des entrées dans une FORM.**

```
<FORM METHOD="POST" ACTION="http://www.essi.fr/~buffa/
test.cgi">
...
<INPUT TYPE="text" size=15 NAME="Nom" VALUE="Buffa">
...
</FORM>
```

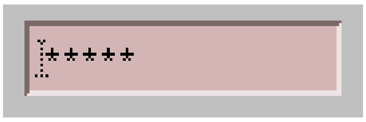
- **Attributs du tag INPUT**
  - attribut TYPE :
    - "text" : valeur par défaut. Entrée de texte.

```
<INPUT TYPE="text" size=15 NAME="Nom" VALUE="Buffa">
```



    - "password" : idem mais le texte enté est masqué.

```
<INPUT TYPE="password" size=15 NAME="passwd">
```



    - "checkbox" : bouton on/off

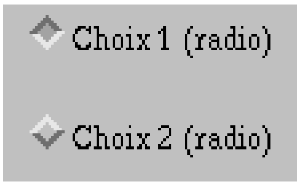
```
<INPUT TYPE="checkbox" size=15 NAME="ExempleCheckbox">
```

Les FORMs en HTML 3.0

Le tag INPUT (suite)

- attribut TYPE (suite) :
  - "radio" : un seul bouton activé parmis les boutons radio ayant le même valeur pour l'attribut NAME.

```
<INPUT TYPE="radio" size=15 NAME="group1" VALUE="1">
<INPUT TYPE="radio" size=15 NAME="group1" VALUE="2">
```

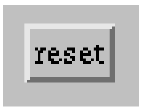


  - "submit" : un bouton poussoir qui déclenche l'appel du cgi précisé dans l'attribut ACTION du tag FORM.

```
Please be sure to press <INPUT TYPE="submit" VALUE="SUBMIT
FORM"> when you are done.
```

  - "reset" : un bouton poussoir qui remet toutes les FORMs avec leurs valeurs par défaut.

```
<INPUT TYPE="reset" size=15 VALUE="reset">
```



- attribut NAME : nom symbolique de l'objet permettant une entrée (boîte, bouton, etc...)
  - N'apparaît pas dans la page!
  - Obligatoire pour tous les types sauf "reset" et "submit".
  - Sert au CGI pour récupèrer les données saisies.

Les FORMs en HTML 3.0

Le tag INPUT (suite)

- attribut VALUE :
  - Avec le type “text” ou “password” : contenu par défaut.
  - Avec le type “checkbox” ou “radio” : spécifie la valeur du bouton lorsqu’il est en position ON (checked). Valeur par défaut = “on”.
  - Avec les types “submit” ou “reset” : spécifie le label du bouton.
- attribut CHECKED :
  - Pas de valeur associée.
  - Pour les types “checkbox” ou “radio” uniquement.
  - Spécifie si un bouton est ON (checked) par défaut.
- attribut SIZE :
  - Spécifie la taille du champs de saisie en caractères.
  - Pour les types “text” et “password” uniquement
  - Valeur par défaut = 20 caractères.
  - Peut servir à spécifier des champs de saisie sur plusieurs lignes : SIZE=80, 24 -> 80 lignes, 24 caractères. Plutôt utiliser le tag TEXTAREA pour faire ça.
- attribut MAXLENGTH :
  - Spécifie le nombre maximal de caractères que l’on peut saisir.
  - Pour les types “text” et “password” uniquement.

Les FORMs en HTML 3.0

Le tag SELECT

- **Permet de créer des menus d'option ou des listes**

```
<SELECT NAME="menu">
<OPTION> choix 1
<OPTION> choix 2
</SELECT>
```

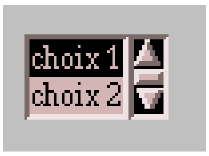
- **A l'intérieur d'une paire <SELECT>...</SELECT> seule une séquence de tags <OPTION> est possible.**

- **Attributs du tag SELECT :**

  - attribut NAME :
    - Nom symbolique du menu. Obligatoire pour que le CGI puisse récupèrer le choix.
  - attribut SIZE :
    - Si SIZE="1" (valeur par défaut) : création d'un menu d'option.

    - Si SIZE="2" ou plus : création d'une liste d'option scrollable. La valeur de SIZE indique le nombre d'items visibles.

```
<SELECT NAME="menu" SIZE="2">
<OPTION> choix 1
<OPTION> choix 2
<OPTION> choix 3
<OPTION> choix 4
</SELECT>
```

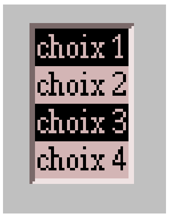


Les FORMs en HTML 3.0

Le tag SELECT (suite)

- **Attributs du tag SELECT (suite):**
  - attribut MULTIPLE :
    - S'il est présent, création d'une liste, même si SIZE n'est pas spécifié (à ce moment là la liste n'est pas scrollable).
    - Indique qu'un choix multiple est possible.

```
<SELECT NAME="menu" MULTIPLE>
<OPTION SELECTED> choix 1
<OPTION> choix 2
<OPTION SELECTED> choix 3
<OPTION> choix 4
</SELECT>
```



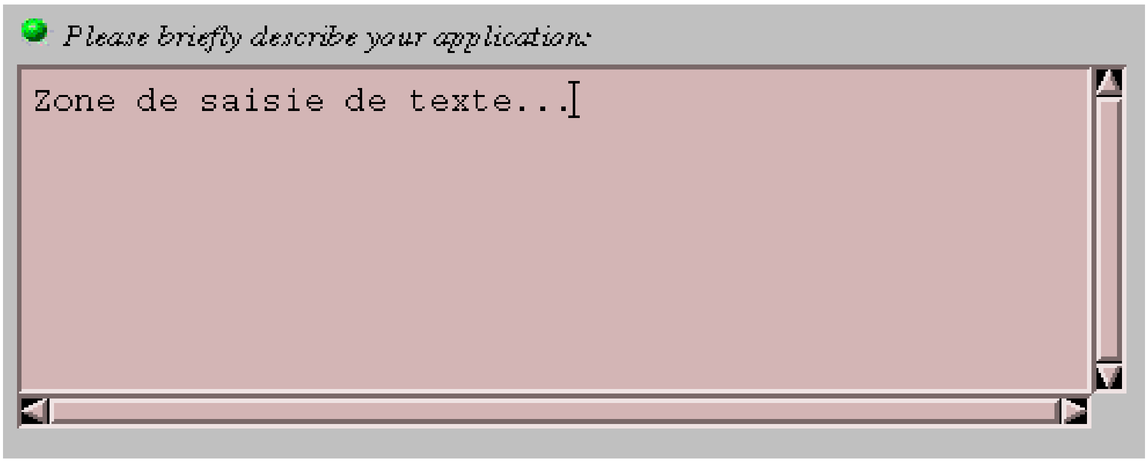
- **Attributs du tag OPTION :**
  - attribut SELECTED :
    - Indique si l'option est séléctionnée par défaut.

Les FORMs en HTML 3.0

Le tag TEXTAREA

- **Crée une zone d'entrée de texte multiligne.**

```
<TEXTAREA NAME="comments" ROWS=8 COLS=50>
</TEXTAREA>
```



- **Possibilité de spécifier un texte affiché par défaut.**

```
<TEXTAREA NAME="comments" ROWS=8 COLS=50>
Le texte par défaut est déclaré ici...
</TEXTAREA>
```

- **Attributs du tag TEXTAREA :**
  - attribut NAME :
    - Nom symbolique de la zone.
  - attribut ROWS :
    - Nombre de lignes
  - attribut COLUMNS :
    - Nombre de colonnes

Envoi du contenu d'une FORM

GET et POST

- **Il existe deux méthodes : GET et POST, fonctions de la valeur de l'attribut METHOD du tag FORM**

```
<FORM METHOD="POST" ACTION="http://
www.essi.fr/buffa/test.cgi">
```

- **Méthode GET :**
  - Lorsque le bouton SUBMIT est enfoncé l'URL précisé dans l'attribut ACTION du tag FORM est appelé, et les données ajoutées à la fin de l'URL

```
http://www.essi.fr/buffa/test.cgi?name=value&name=value
```

  - Méthode non recommandée, certains serveurs limitent le nombre de caractères de l'URL.

- **Méthode POST :**
  - L'URL est appelé simplement :

```
http://www.essi.fr/buffa/test.cgi
```

  - La partie paramètres est envoyée séparément (elle est *POSTée* à l'URL)
  - Même format que pour GET `(name=value&name=value)`
  - Lisible par le CGI sur stdin

Envoi du contenu d'une FORM

Codage des données

- **Le codage est automatique**
- **Exemple de codage de données saisies :**

```
http://www.essi.fr/~buffa/
test.cgi?name1=value1&name2=value2...
```

  - Une paire name=value représente le nom d'un objet dans la FORM et sa valeur.
  - ATTENTION : les caractères bizarres (espaces, caractères réservés, etc...) sont “échappés”.
  - Les valeurs des objets INPUT de type “text” ou “password” ne sont pas “échappées”.
  - Pour les checkboxes ou les boutons radios, l'attribut VALUE spécifie la valeur de l'objet lorsqu'il est ON (checked).
  - Les boutons non activés (OFF) ne sont pas passés en paramètres.
- **Parsing des paramètres peu simple!!!!**
- **...Mais il existe des outils pour faire ce pénible travail !**

Réception des données par le CGI

Dépend de la méthode

- **La récupération du contenu des FORMS par le CGI dépend de la méthode employée.**

- **Méthode GET**
  - Les données se trouvent dans la variable d'environnement QUERY_STRING
  - QUERY_STRING contient tout ce qui suit le "?" dans l'URL du CGI.

- **Méthode POST :**
  - Les données sont disponibles sur stdin.
  - Contenu identique à QUERY_STRING.

- **Appel par ISINDEX**
  - Données sur la ligne de commande ($# et $* en shell, argc, argv en C...)
  - Aussi dans QUERY_STRING

- **Encodage des données :**
  - Les espaces sont remplacés par des "+"
  - Les caractères spéciaux (New Line, Retour Chariot, etc...) sont au format %xx hexadécimal.

Cas du passage forcé de paramètres

Script appelé directement

- **On peut appeler un CGI en passant les paramètres "à la main"**
  - Les paramètres sont passés après le "?" qui suit l'URL, séparés par un "&"
- **Récupération par le CGI :**
  - Dans tous les cas dans QUERY_STRING
  - ...mais aussi dans la ligne de commande si aucun caractère "=" ne figure dans QUERY_STRING

```
http://www.essi.fr/cgi-bin/
test.cgi?toto&tata+titi

Donne :

   $# = 2, $* = toto\&tata titi

   QUERY_STRING = toto&tata+titi

Mais :

http://www.essi.fr/cgi-bin/
test.cgi?toto&tata+titi=tutu

Donne :

   $# = 0, $*

   QUERY_STRING = toto&tata+titi=tutu
```

Décodage des données par le CGI

Existence de nombreux outils

- **Le décodage peut être fastidieux car il existe de nombreux cas particuliers**

- **Heureusement, nombreux outils de parsing des paramètres!**
  - Shell Unix : cgiparse, etc...
    - /ftp.oleane.net:/pub/mirrors/www/cern/bin/osf1/cern_httpd_utils_3.0.tar.gz
  - Perl : (1) lib-cgi.pl, (2) CGI.pm, etc...
    - (1) : http://www.bio.cam.ac.uk/web/form.html
    - (2) : http://www-genome.wi.mit.edu/ftp/pub/software/WWW/
  - C : (1) Librairie cgic, (2) CGILibrary, etc...
    - (1) http://sunsite.unc.edu/boutell/cgic/cgic.html
    - (2) http://wsk.eit.com/wsk/dist/doc/libcgi/libcgi.html
  - TCL :
    - http://www.lbl.gov/~clarsen/projects/htcl/http-proc-args.html

- **Certains de ces outils facilitent également la génération de pages HTML au vol pour envoyer une réponse au client.**

Envoi de la réponse au client

Deux cas

- **La réponse peut être (1) un document ou (2) *une référence* à un document**

- **(1) Envoi d'un document :**
  - Nécessité d'envoyer un en-tête MIME pour décrire le format du document (texte ascii, texte html, gif, mpeg, son, etc...)
  - Exemple :

```
Content-type: text/html
Ici, sauter une ligne
<HTML>
<HEAD>
<TITLE>Doc. Produit par un CGI</TITLE>
</HEAD>
<BODY>
<H1>Coucou!<H1>
</BODY>
</HTML>
```

Envoi de la réponse au client
Référence à un document

- **(2) Référence à un document :**
  - Au lieu de générer le document résultat, on veut faire apparaître chez le client un document existant:

```
Content-type: text/html

Location: http://www.essi.fr

Ici, sauter une ligne

<HTML>

<HEAD>

<TITLE>Resultat au cas où...</TITLE>

</HEAD>

<BODY>

Le document a été déplacé vers <A HREF="http.essi.fr">http.essi.fr</A>

</BODY>

</HTML>
```

- **Le message HTML (facultatif) sert au cas où les redirections automatiques ne sont pas comprises par le browser.**

Envoi de la réponse au client

Référence à un document (suite)

- **Cas où le document référencé est visible par le serveur :**

```
Location : path/file
Ici, sauter une ligne
...
```

- **Le serveur s'occupera de générer l'en-tête approprié.**

Ecriture de programmes CGI

Variables d'environnement

- **Le serveur passe au CGI de nombreuses information le concernant et concernant le client sous la forme de variables d'environnement.**

- **Pour la documentation complete :**

  http://hoohoo.ncsa.uiuc.edu/cgi/env.html

- **En VRAC :**

  - SERVER_SOFTWARE, SERVER_NAME, GATEWAY_INTERFACE, SERVER_PROTOCOL, SERVER_PORT, REQUEST_METHOD, PATH_INFO, PATH_TRANSLATED, SCRIPT_NAME, QUERY_STRING, REMOTE_HOST, REMOTE_ADDR, AUTH_TYPE, REMOTE_USER, REMOTE_IDENT, CONTENT_TYPE, CONTENT_LENGTH, HTTP_ACCEPT, HTTP_USER_AGENT

- **Toutes ne sont pas forcément initialisées (REMOTE_USER par exemple)**

Ecriture de programmes CGI

Variables d’environnement (suite)

- **Exemple :**

`http://www.essi.fr/PubCGI/buffa.testVariables :`

Voici la valeur des variables :

- Nombre paramètres = 0
- Paramètres de la ligne de commande =
- SERVER_SOFTWARE = Netscape–Communications/1.1
- SERVER_NAME = www.essi.fr
- GATEWAY_INTERFACE = CGI/1.1
- SERVER_PROTOCOL = HTTP/1.0
- SERVER_PORT = 80
- REQUEST_METHOD = GET
- PATH_INFO =
- PATH_TRANSLATED =
- SCRIPT_NAME = /PubCGI/buffa.testVariables
- QUERY_STRING =
- REMOTE_HOST = jessica.essi.fr
- REMOTE_ADDR = 157.169.25.100
- REMOTE_USER =
- REMODE_IDENT =
- CONTENT_TYPE =
- CONTENT_LENGTH =
- AUTH_TYPE =
- HTTP_ACCEPT = image/gif, image/x–xbitmap, image/jpeg, image/pjpeg, */*
- HTTP_USER_AGENT = Mozilla/2.0b4 (X11; I; SunOS 5.4 sun4d)

CGI avec "Non Parsed Headers"

Afficher des lignes "au vol"

- **Normalement, lorsqu'in CGI produit un document résultat, l'en-tête (ex : content-type: text-html) est interprété par le serveur avant d'être envoyé au client.**

```
buffa@jessica:> telnet www.essi.fr 80

telnet> GET / HTTP/1.0

HTTP/1.0 200 OK
Server: Netscape-Communications/1.1
Date: Friday, 12-Jan-96 14:53:17 GMT
Last-modified: Friday, 05-Jan-96 15:13:21 GMT
Content-length: 758
Content-type: text/html

<HTML>
<TITLE>ESSI Home Page (France)</TITLE>
```

- **Dans certains cas, on veut pouvoir envoyer des informations "au vol". Par exemple si le calcul de la réponse est long, envoi des résultats partiels dès que possible.**
  - Le CGI doit générer le header
  - Le serveur ne doit pas l'interpréter

CGI avec "Non Parsed Headers"
Conventions de nommage

- **Convention de nommage :**
  - Le script doit commencer par "nph-"
  - Exemple : http://www.essi.fr/PubCGI/nph-buffa1

```
#!/bin/sh

# le header complet sans Content-length
echo HTTP/1.0 200 OK
echo Content-type: text/plain
echo Server: $SERVER_SOFTWARE
echo

echo CGI/1.0 test script report:
echo

while true
do
  echo Ceci est une ligne affichee au vol
done
```

Trucs et astuces !

- **S’assurer que :**
  - Le CGIest bien visible et exécutable par le serveur. Attention aux ScriptAlias et aux ordres ExecCGI de la config du serveur (srm.conf)
  - S’exécutant sous le user nobody en général, vérifier qu’il a bien accès à tout ce qu’il faut.
  - Il a un environnement limité par défaut (celui du user Nobody, fourni par le serveur). Penser à en proposer un si besoin est !
- **Tester les programmes sous shell si possible (syntaxe) avant de les exécuter via le serveur.**
- **En cas d’erreurs serveur, regarder le fichier d’erreurs (tail -f du fichier error_log)**
- **Attention aux alias dans la config du serveur (srm.conf)**
- **Ne pas oublier de générer un header minimal (Content-type)**
- **Ne pas oublier la ligne vide après l’en-tête Content-type**

Trucs et astuces !

(suite)

- **Si malgré tout vous pensez que l'en-tête est reponsable : tester à la main :**

```
telnet www.your-machine 80
telnet> GET /document HTTP/1.0
```

  ... et vérifier l'en-tête

- **Question très fréquemment posée : "qui me consulte ?"**
  - On ne peut pas toujours obtenir la variable HTTP_FROM, ça dépend de la config du client

Les Server Side Includes

Compteurs, includes, etc...

- **Méthode simple mais limitée pour envoyer des informations dynamiques au client**
  - Insérer la date et l'heure dans un document
  - Insérer un compteur de pages
  - Insérer un pied de page, etc...

- **Les pages html sont parsées par le serveur au moment où elles sont envoyée au client**

- **Simple à utiliser, cette méthode évite l'écriture de CGI. Exemple :**

```
Date d'aujourd'hui :<!--#exec cgi="/
cgi-bin/date"--> <P>
```

- **Supporté (au moins partiellement ?) par la plupart des serveurs**

- **Différentes configurations possibles :**
  - Fichiers se terminant par .shtml
    - Addtype text/x-server-parsed-html .shtml (NCSA)
  - Tous les fichiers (à éviter **!**)
    - Addtype text/x-server-parsed-html .html (NCSA)
  - Fichiers se trouvant dans un directory particulier

Les Server Side Includes

Format d'appel

- **Le format d'appel d'un SSI est le suivant :**

```
<!--#commande tag1="value1" tag2="value2" -->
```

- **Chaque commande prend des arguments différents :**
  - Commande "exec" : exécution d'une commande shell ou d'un CGI. Deux tags possibles :
    - cmd : exécute une commande à l'aide de /bin/sh.

```
<!--#exec cmd="who"-->
```

    - cgi : exécute un CGI. La valeur du tag est un URL.

```
<!--#exec cgi="/cgi-bin/date"-->
```

  - Commande "include" : insère le texte d'un document. Deux tags possibles :
    - virtual : URL du document (pas un CGI)

```
<!--#include virtual="/~buffa/toto.html"--> ???
```

    - file : pathname d'un fichier *relatif* au directory courant. "../" est interdit.

```
<!--#include file="toto.html"-->
```

  - Commande "echo" : affiche la valeur d'une variable d'environnement (cf pages suivantes). Un seul tag:
    - var : nom d'une variable d'environnement

```
<!--#echo var="REMOTE_HOST"-->
```

Les Server Side Includes

Format d'appel (suite)

- Commande fsize : renvoit la taille d'un fichier. Un seul tag : file

```
<!--#fsize file="toto.html"-->
```

- Commande flastmod : renvoit la date de dernière modification d'un fichier. Un seul tag : file

```
<!--#flastmod file="toto.html"-->
```

- Commande "config" : permet de customizer le comportement des SSI. Plusieurs tags :
  - errmsg : personalisation du message d'erreur des SSI. Par défaut [an error occurred while processing this directive]

```
<!--#config errmsg="Message d'erreur personalisé"-->
```

  - timfmt : changement du format de la date
  - sizefmt : changement du format des tailles fichiers

Les Server Side Includes

Variables d'environnement

- **L'utilisation de SSI fournit de nouvelles variables d'environnement (affichables avec la commande #echo)**
  - DOCUMENT_NAME
  - DOCUMENT_URI : path virtuel du document (exemple : ~buffa/index.html)
  - QUERY_STRING_UNESCAPED : permet de passer des paramètres à un fichier .html
  - DATE_LOCAL
  - DATE_GMT
  - LAST_MODIFIED

```
DOCUMENT_NAME = <!--#echo var="DOCUMENT_NAME"--> <P>

DOCUMENT_URI = <!--#echo var="DOCUMENT_URI"--> <P>

QUERY_STRING_UNESCAPED = <!--#echo
var="QUERY_STRING_UNESCAPED"--> <P>

DATE_LOCAL = <!--#echo var="DATE_LOCAL"--> <P>

DATE_GMT = <!--#echo var="DATE_GMT"--> <P>

LAST_MODIFIED = <!--#echo var="LAST_MODIFIED"--> <P>
```