Documents dynamiques

1. Types de documents WEB
2. *Common Gateway Interface* (CGI) pour documents dynamiques
   - Principe
   - Utilisation, formulaires HTML
   - Configuration d'Apache
   - Sécurité
3. *Server Side Include* (SSI) pour documents dynamiques
4. Introduction à PHP

1. Types de documents WEB

- Documents statiques :
  - contenu fixé au moment de sa publication
  - écrits en HTML statique
  - chaque requête obtient la même réponse
- Documents dynamiques :
  - contenu crée par le serveur quand il reçoit une requête
  - implique le lancement d'un programme sur le serveur pour générer le document
  - chaque requête peut obtenir une réponse différente
- Documents actifs :
  - le document = copie d'un programme
  - le programme tourne sur la machine client et interagit avec l'utilisateur
  - une seule requête est nécessaire pour avoir plusieurs instances de documents

Chaque type de document a des avantages et des inconvénients !

Common Gateway Interface (CGI)

- = moyen d'exécuter des requêtes complexes sur le serveur.
- Standard de fait maintenu par le NCSA
  `http://hoohoo.ncsa.uiuc.edu/cgi/`
  Ce standard
  - spécifie comment le serveur et le programme communiquent entre eux
  - ne spécifie pas comment écrire des CGIs (p.e. le langage)
- Schéma général :
  - Envoi d'une requête sur une URL spéciale
  - Réception de la requête sur le serveur
  - Exécution du programme désigné par l'URL
  - Réponse du serveur = sortie standard du programme lancé
- Utilisation : compteurs d'accès, consultation bases de données, moteurs de recherche, formulaires, etc.

Envoi d'une requête

- Les requêtes HTTP qui peuvent être utilisées sont : `GET` ou `POST`.
- Chaque CGI est prévu pour l'une ou les deux requêtes.
- Les paramètres sont transmis :
  - pour `GET` : dans l'URL

    `http://www.toto.fr/cgi-bin/mon-cgi?x=1`
  - pour `POST` : dans une chaîne de caractères dans le corps de la requête

```
POST /cgi-bin/mon-cgi HTTP/1.1
        Host: www.toto.fr
Content-Type: application/x-www-form-urlencoded
Content-Length: 3

x=1
```

- Les paramètres sont encodés (*URL-encoding*) :
  - espace $\longrightarrow$ +
  - caractères spéciaux $\longrightarrow$ %code-ascii
  - concaténation de paramètres en utilisant & :

    `format=html+3&type=Confe%e9rence`

  Il existe des librairies pour décoder les paramètres.

Réception de la requête par le serveur

- A la réception de la requête, le serveur transmet les informations au programme à exécuter par des variables d'environnement :
  - Variables non spécifiques à la requête :

    `SERVER_SOFTWARE SERVER_NAME GATEWAY_INTERFACE`
  - Variables spécifiques à chaque requête :

`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`	

  - Variables reçues du client par la requête HTTP :

    `HTTP_ACCEPT HTTP_USER_AGENT`
- Les paramètres sont passés :
  - pour `GET` : uniq. dans `$QUERY_STRING`
  - pour `POST` : sur `stdin` et `$QUERY_STRING`

Exécution du programme sur le serveur

- Le serveur lance le programme CGI par un “exec”, donc le lancement peut prendre du temps.
- Normalement, un CGI doit produire rapidement une réponse.
- Etapes classiques d'un CGI :
  1. récupération des paramètres en fonction de la méthode HTTP
  2. décodage des paramètres
  3. calculs, préparation réponse
  4. écriture sur `stdout` de la réponse
- Outils de décodage des paramètres :
  - Shell Unix : cgiparse, outils CERN,...
  - C : util.c (CERN), cgic (Boutel),...
  - Perl : module CGI, lib-cgi.pl (Cambridge),...

Envoi de la réponse au client

La réponse peut être :

- Un document → obligatoire d'envoyer un en-tête HTTP pour préciser le format du document :

  ```
  Content-Type: text/plain

  Exemple de document texte.
  ```

- Une référence, si au lieu de générer un document, on veut faire apparaître une document existant
  - sur le serveur :

    ```
    Location: /index.html
    ```
  - à l'extérieur :

    ```
    Content-Type: text/plain
    Location: http://www.toto.fr/index.html

    Contenu ou cas où la redirection n'est pas
    comprise par le navigateur
    ```

Le serveur complète avec les autres en-têtes HTTP...

Envoi **directe** de la réponse au client

Si on veut envoyer une réponse sans passer par le serveur : utiliser les “non parsed headers”.

- Exemples d'utilisation : réponse longue, donc envoi de résultats partiels.
- Pour qu'un CGI soit NPH il suffit de le préfixer par “`nph-`”.

Exemple mauvais : `nph-ex.sh`

```
#!/bin/sh

# generation de l'en-tete 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 affiche au vol"
done
```

Exemple

CGI “finger.sh” :

```
#!/bin/sh
echo "Content-Type: text/html"
echo
echo "<HTML><HEAD><TITLE>"
echo $QUERY_STRING
echo "</TITLE></HEAD><BODY><PRE>"
/usr/bin/finger $QUERY_STRING
echo "</PRE></BODY></HTML>"
```

Requête type :

```
http://www.toto.fr/cgi-bin/finger?sighirea
```


Formulaires HTML

Eléments FORM, INPUT, SELECT, TEXTAREA,...

```
<FORM ACTION="http://www.toto.fr/cgi-bin/dem.cgi"
      METHOD="POST">
Text :
<INPUT TYPE="text" SIZE=15 NAME="text1" VALUE="defaut">
<P> Mot secret :
<INPUT TYPE="password" SIZE=15 NAME="passwd">
<P> Votre choix :
<SELECT NAME="menu">
<OPTION> choix 1
<OPTION> choix 2
</SELECT>
<P> Un message :
<TEXTAREA NAME="msg" ROWS=8 COLS=50>
Par defaut...
</TEXTAREA>
<P> Envoyer le formulaire :
<INPUT TYPE="submit" VALUE="SUBMIT FORM">
<P>Anuler le formulaire :
<INPUT TYPE="reset" VALUE="RESET FORM">
</FORM>
```

Configuration Apache

- `ScriptAlias` : correspondance nom CGI et répertoire où se trouvent par défaut les programmes CGI.

  ```
  ScriptAlias /cgi/ /www/cgi-bin/
  ```

  Sur ce répertoire et les CGI contenus il faut donner les bons droits.

- `AddHandler` : peut autoriser l'exécution des CGI en faisant correspondre une extension de fichier au gestionnaire `cgi-script`

  ```
  AddHandler cgi-script pl
  ```

  Pas très sûr !

- `Options ExecCGI` : autoriser l'exécution dans un répertoir particulier. Peut être utilisée dans :
  - `httpd.conf`

    ```
    <Directory /home/~toto/public_html/cgi-bin/>
         Options ExecCGI
    </Directory>
    ```

  - `.htaccess` si `AllowOverride Options`

Sécurité

- Les CGI/SSI/PHP sont les principal trou de sécurité pour les serveurs.
- Les CGIs doivent vérifier les paramètres qu'ils reçoivent afin de ne pas ouvrir de brèche aux pirates.
- Exemple "classique" :

  ```
  #!/usr/bin/perl
  require "cgi-lib.pl"
  print "Content-Type: text/plain  "
  if (&ReadParse)  system("finger $inaddr");
  ```

  Si on passe comme paramètre la chaîne :

  ```
  addr=pierre;cat /etc/passwd
  ```

  le CGI exécute la commande pirate et diffuse un fichier sensible.

- D'où le besoin limiter les endroits où se trouvent les CGI afin de pouvoir vérifier leur comportement avant de les utiliser !

SSI : Server Side Includes

- Méthode simple mais limitée pour envoyer des informations dynamiques au client. Exemples d'utilisation :
  - Insérer un autre fichier HTML.
  - Insérer la date de la dernière modification du fichier.
  - Exécuter des CGIs.
- = Directives directement insérées dans les pages HTML et analysées par le serveur au moment de servir la page au client. Au lieu de transmettre la page directement, le serveur remplace les directives SSI par le contenu qu'elles requiert.

  ```
  <!--#include file="footer.html" -->
  ```
- Méthode supportée (au moins partiellement) par la plupart de serveurs.
- La configuration par défaut d'Apache n'autorise pas les SSI pour des raisons de sécurité.

Configurations du serveur Apache pour les SSI

- Activer dans `httpd.conf` ou .htaccess l'option `Includes`

  ```
  Options Includes
  ```

  pour le répertoire, hÙte virtuel, etc. où les SSI sont autorisés.
- Plusieurs solutions pour indiquer les fichier SSI :
  - Par extension de fichier :

    ```
    AddType text/html .shtml
    AddHandler server-parsed .shtml
    ```
  - Par droits (exécution) associés au fichier, directive `XBitHack`.

    ```
    XBitHack on
    ```

    ou `off` (par défaut), `full`.
  - Par type MIME (non recommandée).

Directives SSI

Forme générale :

```
<!--#element attribut=valeur attribut=valeur ... -->
```

où `element` peut être l'une des commandes suivantes :

- `config` : définit les diverses options d'analyse de la page. Attributs possibles : `errmsg`, `sizefmt`, `timefmt`.

  ```
  <!--#config timefmt="%B %e, %Y" -->
  ```

- `echo` : affiche l'une des variables

  ```
  DATE_GMT      DATE_LOCAL     DOCUMENT_NAME
  DOCUMENT_URI  LAST_MODIFIED
  ```

  ```
  Today's date is <!--#echo var="DATE_LOCAL" -->
  ```

- `exec` : permet d'exécuter une commande shell (attribut `cmd`) ou un programme CGI (attribut `cgi`).

  ```
  <!--#exec cmd="/usr/bin/ls -la /tmp" -->
  ```

  Pour désactiver cette directive : `Options IncludesNOEXEC`

- `fsize` : affiche la taille du fichier définit par les attributs `file` ou `virtual`.

- `flastmod` : affiche la date de la dernière modification du fichier définit par les attributs `file` ou `virtual`.

- `include` : insère le contenu du fichier définit par les attributs `file` ou `virtual`.

- `printenv` : affiche toutes les variables d'environnement existantes.

- `set` : définit et affecte des variables

- `if-elif-else-endif` : contrÙle de flux

  ```
  <!--#set var="agent" value="$HTTP_USER_AGENT" -->
  <!--#if expr="$agent = /Mozilla/" -->
  MOZILLA !
  <!--#elif expr="$agent = /MSIE/" -->
  Internet Explorer
  <!--#else -->
  Autre chose !
  <!--#endif -->
  ```

Introduction à PHP v3

PHP = "*Personal Home Pages*" ou "*Portable Hypertext Programming*"

Référence : `www.php.net`

Objectif : délivrer un contenu dynamique sur le Web

- en intégrant du PHP dans des pages HTML
- en générant des pages HTML
- en se connectant aux BD

Langage interprété ⇒ interpréteur à configurer sur la machine du serveur Web.

Combine du C, C++, Perl.

```
<HTML>
<HEAD><TITLE>Bonjour</TITLE></HEAD>
<BODY>
<?php /* Exemple simple */
    phpinfo(); // Affiche des infos Apache/PHP
?>
</BODY>
</HTML>
```


Eléments du langage

Délimitation d'un script PHP :
`<?php ... ?>` ou `<? ... ?>`

Identificateurs : "C-like", mais le noms des fonctions intégrées insensibles aux maj./min.

Commentaires : "C-like" et "C++-like"

Variables :

- On ne les déclare pas sauf pour changer leur portée (en `global` ou `static`) qui est par défaut locale.
- Leur type est "amorphe" (non fixé).
- Utilisation de variable : $idVariable

Constantes :

- Pré-définies : `TRUE`, `FALSE`, `M_PI`, `PHP_VERSION`, `__FILE__`, `__LINE__`, ...
- Utilisateur : `define("PI", 3.14)`
- Leur portée est globale.
- Utilisation : idConstante

Exemple :

```
<?php
  $my_global = 23;
  $my_float = 36.0;
  function ShowVariables() {
    global $my_float;
    echo("Variable globale (non-importée) " .
         "my_global = $my_global\n");
    // my_global ci-dessous est locale !
    $my_global = 1963;
    echo("Variable locale " .
         "my_global = $my_global\n");
    echo("Variable globale (importée) " .
         "my_float = $my_float\n");
  }
  echo("Demo portée variables");
  ShowVariables();
?>
```


Types de base

- Booléean : `TRUE`, `FALSE`, opérateurs logiques “C-like” et syntqxe explicite `AND`, `OR`, `XOR`
- Entier : 23, opérateurs arithmétiques “C-like”
- Nombres à virgule flottante : 23.0, opérateurs arithmétiques “C-like”
- Chaînes de caractères :
  - non-interprétées : `'chaine fixe'`
  - interprétées : `"chaine $my_global"`

  opérateur de concaténation `.`

Opérateurs de conversion entre types : `intval`, `doubleval`, `strval`, ...

Opérateurs de test pour le type d’une variable : `is_integer`, `is_float`, `is_array`, ...

Instructions

... peu de nouveautés par rapport au C.

Toute instruction est suivie de `;`

- Affectation de variables.
- Instructions conditionnelles :
  `if...else`, `if...elseif...else`, `()?:`
  `switch` sur tout type de valeurs
- Boucles :
  `for`, `while`, `do ... while`
- Renvoi valeur : `return`

Fonctions utilisateur

Comme en C mais :

- Mot-clé : `fonction`
- La liste des paramètres formels est facultative.
- Sans spécification du type du résultat.
- Passage des paramètres : par défaut par valeur ; si préfixage par & **à l'appel**, passage par référence.

```
fonction DoesSomething($param1, $param2) {
  static $remember_it;
  global $external;

  $param1 = $param2 + $remember_it;
  $remember_it = $param1;
  Return $param2;
}
// appel
$result = DoesSomething(&$p1, $p2);
```

Tableaux

Deux types :

- tableau indexé (“C-like”)
- tableau associatif = l’indexation est faite par une clé

```
$colors = array( 'red'  => '#ff0000',
                 'blue' => '#ff0000',
                    3   => '#000003' );
```

Accès :

- “classique” : `$colors[3]`
- par pointeur interne et fonctions : `reset`, `end`, `next`, `prev`, `key`, `current`

Fonctions prédéfinies :

- `join` / `explode` : conversion vers / depuis une chaîne de caractères

```
$words = array( 'Hello', 'world', '!');
$phrase = join($words, ' ');
// $phrase = 'Hello world !'
$words = explode(' ', $phrase);
```


Lien avec les formulaires HTML

- Supporte les deux méthodes (GET/POST) de façon transparente.
- Convertit automatiquement les paires (clé,valeur) du formulaire en variables PHP.
- Pour utiliser une valeur entrée dans le champ :

  ```
  <INPUT TYPE="text" NAME="vlogin" SIZE=23>
  ```

  il suffit d’utiliser la variable `$vlogin`.
- Pour tester si la valeur a été initialisée : `isset`
- Pour les champs avec entrées multiples, le nom doit être un tableau :

  ```
  <INPUT TYPE="checkbox"
         NAME="vserver[]"
         VALUE="amertume">
  ```