Introduction Web et Services Web : *Serveur de pages Web Dynamiques*

1 Création de pages Web dynamiques

1.1 Introduction

Dans les premiers serveurs Web permettent de lire des pages web dites statiques. En effet les pages renvoyées au client sont celles présentes sur le système de fichiers du serveur et occasionnellement modifiées par leur propriétaire.

Lors de la consultation d'une page Web statique, un serveur HTTP renvoie donc le contenu du fichier où la page est enregistrée. Lors de la consultation d'une page Web dynamique, un serveur HTTP transmet la requête au logiciel pointé dans l'URL, et le logiciel se charge de générer l'équivalent du contenu d'une page Web.

De nombreuses solutions logicielles sont développées pour faciliter et améliorer la génération de pages Web dynamiques. Citons par exemple les langages PHP, JavaServer Pages (JSP) ou Active Server Pages (ASP)…
En fait qu'il s'agisse d'appeler directement un exécutable ou un interpréteur pour lire un script, l'objectif reste identique.

Pour bien comprendre que ce principe ne dépend pas d'une technologie donnée, nous allons mettre en œuvre une des toutes premières approches pour la création de page web dynamique : les cgi-bin qui ne font aucune hypothèse sur les modes d'implémentation (shell script, script python, exécutable binaire …).

Comme nous allons le voir, les seules hypothèses imposées au développeur d'un cgi-bin sont :

Dans le cas d'une requête HTTP/GET,

- Il existe une variable d'environnement `QUERY_STRING` dans laquelle se trouvent l'ensemble des variables et leurs valeurs passées dans l'URL (cas du GET) ou dans le corps de la requête http (cas du POST) ;
- Le cgi-bin renvoie les données à destination du client Web sur sa sortie standard ;

Dans le cas d'une requête http/POST,

- L'ensemble des informations du corps de la requête http (cas du POST) sont envoyées dans le fichier d'entrée standard (stdin) du cgi-bin. Il faudra alors utiliser la variable d'environnement CONTENT_LENGTH qui contient la taille en octets des informations jointes à la requête pour lire exactement l'ensemble des paramètres ;
- Le cgi-bin renvoie les données à destination du client Web sur sa sortie standard ;

A charge donc du serveur de gérer ces hypothèses, en particulier dans le cas d'une requête http/GET en

- affectant correctement la variable d'environnement `QUERY_STRING` avec les paramètres passés dans l'URL ;
- émettant les données retournées par le cgi-bin sur sa sortie standard ;

1.2 Un premier cgi-bin

Nous allons maintenant ajouter l'appel à un cgi-bin (soit l'exécution autorisée d'un binaire qui pourra générer une page Web dynamique). Les paramètres sont alors passés après le "?", séparés par un "&", dans l'URL.

1.2.1 Exemple de mise en œuvre d'un cgi-bin de type http/GET

Pour l'exécution du programme « bonjour jean pascal» dont le fichier bonjour est situé dans $HTTP_ROOT/cgi-bin, l'appel sera :

```
http://localhost:8080/cgi-bin/bonjour?nom=jean&prenom=pascal
```

La page Web retournée sera alors :

```
<HTML>
<HEAD>
<TITLE>Doc. Produit par un CGI</TITLE>
</HEAD>
<BODY>
<H1>Coucou jean pascal !<H1>
</BODY>
</HTML>
```

Attention,

- N'oubliez pas de rajouter la ligne d'introduction dans l'entête de la réponse HTTP
- N'oubliez pas non plus de mentionner en tête de votre fichier cgi-bin si ce dernier n'est pas exécutable mais nécessite un interprète le programme qu'il faut lancer (ex # ! /bin/sh, #!/usr/bin/python, #!/usr/bin/perl …).

  #! s'appelle le « shebang » qui dans l'en-tête d'un fichier texte indique au système d'exploitation (de type Unix) que ce fichier n'est pas un fichier binaire mais un script (ensemble de commandes). Sur la même ligne est précisé l'interpréteur permettant d'exécuter ce script.

1.2.2 Variables d'environnement du cgi-bin et QUERY_STRING pour une requête http/GET

Dans le cas d'un script shell cgi-bin, nombre de variable d'environnement du processus lancé sont positionnées. Exemple : `HTTP_USER_AGENT, SERVER_ADMIN, SERVER_SOFTWARE,…`

La variable d'environnement qui permet la récupération des paramètres de l'URL dans n'importe quel cgi-bin est conventionnellement `QUERY_STRING.`

Exemple : `QUERY_STRING=namex=valuex&namey=valuey&namez=valuez`

Notons qu'un grand nombre de variables d'environnement sont positionnées par le serveur WEB lors de l'appel à un cgi-bin :

Key	Value
DOCUMENT_ROOT	The root directory of your server
HTTP_COOKIE	The visitor's cookie, if one is set
HTTP_HOST	The hostname of the page being attempted
HTTP_REFERER	The URL of the page that called your program

HTTP_USER_AGENT	The browser type of the visitor
HTTPS	"on" if the program is being called through a secure server
PATH	The system path your server is running under
QUERY_STRING	The query string (see GET, below)
REMOTE_ADDR	The IP address of the visitor
REMOTE_HOST	The hostname of the visitor (if your server has reverse-name-lookups on; otherwise this is the IP address again)
REMOTE_PORT	The port the visitor is connected to on the web server
REMOTE_USER	The visitor's username (for .htaccess-protected pages)
REQUEST_METHOD	GET or POST
REQUEST_URI	The interpreted pathname of the requested document or CGI (relative to the document root)
SCRIPT_FILENAME	The full pathname of the current CGI
SCRIPT_NAME	The interpreted pathname of the current CGI (relative to the document root)
SERVER_ADMIN	The email address for your server's webmaster
SERVER_NAME	Your server's fully qualified domain name (e.g. www.cgi101.com)
SERVER_PORT	The port number your server is listening on
SERVER_SOFTWARE	The server oftware you're using (e.g. Apache 1.3)

1.2.3 Extension de votre serveur Web et test sur un cgi-bin

question 1 : Etendez votre serveur Web en python pour qu'il exécute un cgi-bin <nom_exe> dès lors que l'URL correspondante contient le sous-chemin clef /cgi-bin/ comme dans http://localhost:8080/cgi-bin/bonjour?nom=jean&prenom=pascal. Utilisez les informations contenues dans les annexes pour cela.

question 2 : Ecrivez le cgi-bin bonjour détaillé comme exemple ci-dessus. Testez le en entrant l'URL correspondante dans un browser Web.

2 Votre premier serveur M2M over Web

Le client Web est jusque-là un browser web ayant pour objectif d'afficher des informations issues d'un serveur Web. Ce type d'utilisation entre dans la catégorie des applications H2M (Human to Machine) où la vocation du Web est de fournir un grand nombre de sources d'informations dynamiques ou non à un utilisateur. Or aujourd'hui le Web est aussi devenu une technologie de communication entre programmes soient des applications dites Web M2M (Machine to Machine).

C'est ce principe qui sera utilisé pour les Services Web dans la mesure où le programme "client" n'est pas un browser Web pour visualiser les données retournées mais une application logicielle quelconque..

question 3 : A partir des questions précédentes, vous pouvez vous convaincre de la simplicité de ce concept **en mettant en place un programme client qui non seulement enverra des paramètres dans une URL (comme paramètres d'appel d'une fonction) mais récupérera les données de retour dans les données renvoyées par le serveur**.

Vous pouvez par exemple invoquer une méthode incr <val> (qui incrémente la valeur de val) depuis un client TCP/IP qui n'est plus un navigateur WEB. La méthode incr correspondra à l'invocation d'un cgi-bin de même nom (exemple : http://localhost:8080/cgi-bin/incr?val=5). Il vous appartiendra alors de définir le format du contenu du message de réponse du cgi-bin qui ne soit plus de l'HTML mais un format lisible par le client (ex. incr OK val=6).

Nous sommes ainsi devant une technique qui permet d'implémenter de multiples patterns de communication entre applications réparties comme le pattern RPC (Remote Procedure Call) pour l'invocation à distance.

Sachez que vous venez d'implémenter votre premier web Service REST….

Introduction Web et Services Web : *Serveur de pages Web Dynamiques*

3 Annexe : Exécution d'un programme dans un nouveau processus créé en python et redirection des entrées / sorties

Plusieurs techniques sont envisageables pour rediriger la sortie standard du cgi-bin que vous lancez. Si les principes sous-jacents sont les mêmes pour les redirections d'entrées/sorties sur les processus, les API à utiliser pour cela dépendent du langage. Dans le cas de python nous allons utiliser le module subprocess et sa fonction Popen qui possède un ensemble de paramètres gérant la création d'un process, de pipes et les redirections des entrées sorties standards, en un seul appel.

Voici un exemple qui implémente la commande shell ls -l / | sort -k 2 -n.

```
import subprocess
pls = subprocess.Popen(['ls','-l','/'], stdout=subprocess.PIPE)
psort = subprocess.Popen(['sort', '-k','2','-n'], stdin=pls.stdout, stdout=subprocess.PIPE)
res = psort.communicate()[0]
print res
```

Inspirez-vous de ce code pour implémenter le lancement de l'exécution d'un cgi-bin dans votre serveur Web.

4 Annexe : gestion des variables d'environnement en Python

Tout cgi-bin doit pouvoir s'appuyer sur les paramètres qui lui sont destinés et qui ont été collectés dans la variable d'environnement `QUERY_STRING`.

En conséquence le serveur doit être capable de préparer des variables d'environnement avant de lancer le cgi-bin.

Si votre serveur est écrit en Python voici le module à importer et les fonctions à utiliser :

- Pour obtenir une variable d'environnement avec Python vous pouvez utiliser le module os et la variable environ avec par exemple :

```
if os.environ.has_key('nom_de_la_cle'):
print os.environ['nom_de_la_cle']
```