1 Pages web dynamiques

En schématisant on peut considérer que l'accès à une pager web depuis une machine s'effectue selon une **architecture client/serveur**. La machine client émet une requête de page web et la machine serveur lui répond en lui envoyant cette page. Pour les pages web les machines dialoguent selon le protocole HTPP.

La page web retournée par le serveur peut être **statique** : c'est un fichier d'extension .html qui se trouve dans l'arborescence du serveur et contenant du code HTML/CSS qui sera interprété par le navigateur du client pour afficher la page.

De nos jours, les sites contiennent souvent des **pages web dynamiques** dont le contenu peut varier selon le client qui la demande. Dans ce cas le serveur exécute d'abord un script qui récupère les variables envoyées par le client (à travers un formulaire ou un cookie par exemple) puis génère du code HTML/CSS. le serveur retourne la page web au client. Ce dernier reçoit le code HTML/CSS mais pas le code du script générateur.
Les scripts qui servent à générer des pages web sur le serveur sont généralement écrits en langage PHP. C'est la technique la plus rapide, mais l'interface de programmation CGI (Common Gateway Interface) permet d'utiliser n'importe quel langage de script dont l'interpréteur est installé sur le serveur.
En général sur un serveur Apache installé sous Linux, les pages webs en .html et les scripts en .php doivent être placés dans le répertoire `/var/www` et les scripts CGI dans `/var/www/cgi-bin`. Le serveur porte le nom d'utilisateur `www-data`. Il faut penser à rendre exécutables les scripts par tous (other).
Pour notre part, nous installerons le serveur Apache avec PHP avec les extensions réalisées par Jean-Claude Meilland et disponibles sur son site `megamaths.free`. Les fichiers de configuration d'Apache sont modifiés pour permettre l'exécution de scripts PHP ou Python à partir du répertoire `/home/user/apache` qu'il faut créer. Cependant il faudra toujours penser à rendre exécutable les scripts pour tous (clic droit puis permissions).
Le répertoire `/home/user/apache` est alors accessible en tapant l'adresse IP `127.0.0.1` ou son alias `localhost` dans la barre d'adresse d'un navigateur. Pour exécuter le fichier `testcgi.py` placé dans le répertoire `/home/user/apache/python`, on pourra taper `http://localhost/python/testcgi.py` dans la barre d'adresse d'un navigateur.

2 Un exemple

2.1 Objectif

Dans cet exemple nous allons écrire deux scripts Python :

- `formulaire.py` pour générer une page web qui affichera un formulaire où le client saisira ses nom, prénom, catégorie (élève/professeur), le chemin d'un fichier image à transmettre et sélectionnera la nature de l'image (photo, peinture ...).
- `traitement.py` pour générer une page web qui affichera le résultat du traitement du formulaire : nom, prénom, catégorie de l'expéditeur et image envoyée.

Les fichiers `formulaire.py` et `traitement.py` sont disponibles dans la Dropbox.

2.2 Principes d'écriture d'un script CGI

Pour l'instant, je n'arrive pas à utiliset Python3 avec le serveur donc nous programmerons en Python2 (alias python dans notre distribution Linux).
Tous les scripts Python que nous exécuterons devront commencer par le préambule suivant :

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3 print "Content-Type: text/html \n"
```

La ligne 1, dite de *shebang*, indique au serveur où il peut trouver l'interpréteur Python sur sa machine. Pour info `env` est un utilitaire `shell` qui va lancer correctement la commande qu'on lui passe en paramètre (ça devrait marcher avec python3 mais bon ....)
La ligne 2 précise l'encodage du fichier.

La ligne 3 affichera le type MIME du contenu dans l'en-tête du fichier généré. Les types MIME constituent une classification des différents contenus transportés sur Internet. Ils ont été initialement définis pour les contenus des mails. Cette déclaration est indispensable, le serveur pourra ainsi insérer le type MIME dans le header HTPP du fichier envoyé et le navigateur pourra correctement l'interpréter.
Le reste du fichier peut être codé en Python comme d'habitude en gardant à l'esprit que tout ce qui est retourné vers la sortie standard (par `print` par exemple) va correspondre au contenu de la page HTML que l'on souhaite générer.
Exemple d'un script CGI :

```
1  #!/usr/bin/env python
2  # -*- coding: utf-8 -*-
3  print "Content-Type: text/html \n"
4
5  print """<!DOCTYPE html >
6  <html >
7  <head>
8      <title>Premier exemple de scritp CGI </title>
9      <meta charset="utf-8"/>
10 </head>
11 <body>
12     <h1> Ceci est un titre </h1>
13 """
14
15 for i in range(1,6):
16     print '<p> Ceci est un paragraphe. <br/> Bonjour %s fois. </p>'%i
17
18 print """
19 </body>
20 </html>"""
```

exempleCGI.py

Le code HTML reçu par le client est :

```
1  <!DOCTYPE html >
2  <html >
3  <head>
4      <title>Premier exemple de scritp CGI </title>
5      <meta charset="utf-8"/>
6  </head>
7  <body>
8      <h1> Ceci est un titre </h1>
9          <p> Ceci est un paragraphe. <br/> Bonjour 1 fois. </p>
10         <p> Ceci est un paragraphe. <br/> Bonjour 2 fois. </p>
11         <p> Ceci est un paragraphe. <br/> Bonjour 3 fois. </p>
12         <p> Ceci est un paragraphe. <br/> Bonjour 4 fois. </p>
13         <p> Ceci est un paragraphe. <br/> Bonjour 5 fois. </p>
14 </body>
15 </html>
```

Pour plus d'informations, consultez la documentation python :
`http://docs.python.org/2.7/library/cgi.html`

2.3 Un script qui génère un formulaire en HTML

Une page HTML peut contenir un **formulaire** avec différents **champs**. Lorsque le client a rempli tous les champs, il clique sur un bouton envoyer et les données sont transmises au serveur.

Pour écrire un formulaire en HTML on utilise une balise `<form>` où l'on va placer des balises de champ telles que `<input>` ou `<select>`. Voici le code du script `formulaire.py` qui va générer notre formulaire.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print "Content-Type: text/html \n"

print """<!DOCTYPE html >
<html >

<head>
    <title>Page HTML avec formulaire </title>
    <meta charset="utf-8"/>
    <link rel="stylesheet" href="styleformulaire.css"/>
</head>

<body>

    <h1>Formulaire d'envoir d'un fichier image</h1>

    <form action="traitement.py" method="post" enctype="multipart/form-data">

        Nom : <input type="text" name="nom" id="nom"/> <br/>

        Prénom : <input type="text" name="prenom" id="prenom"/> <br/>

        Catégorie : <br/>
        <input type="radio" name="categorie" value="eleve" /> Elève
        <input type="radio" name="categorie" value="professeur" /> Professeur <br/>

         Nature de l'image : <br/>
         <select name="image">
            <option value="peinture"> peinture </option>
            <option value="photo" selected="selected"> photo </option>
            <option value="logo"> logo </option>
            <option value="dessin"> dessin </option>
          </select>
          <br/>

        <input type="file" name="fichier">
        <br/>

        <input type="submit" value="Valider" />

     </form>

</body>

</html>"""
```

formulaire.py

Quelques explications :

- La balise `<form>` comprend deux attributs obligatoires `method="post"` pour préciser la méthode d'envoi et `action="traitement.py"` pour la référence au script qui traitera le formulaire.

Le dernier attribut `enctype="multipart/form-data"` est nécessaire uniquement lorsqu'on veut envoyer des fichiers.
Il existe deux méthodes d'envoi : `"post"` la plus courante et `"get"` la plus ancienne qui transmet les données dans l'URL sous la forme d'une séquence de couples `variable=valeur` placés après un ? et séparés par un & comme ci-dessous :
`http://localhost/python/traitementget.py?nom=Junier&prenom=Frédéric&categorie=professeur`
Attention la méthode `"get"` ne peut être utilisér pour l'envoi de fichiers!!! Pour la tester il faut donc supprimer l'attribut `enctype=multipart/form-data` et le champ `<input type="file" />` dans le formulaire précédent.
Les fichiers `formulaireget.py` et `traitementget.py` sont disponibles sur la Dropbox.

- Un champ de saisie comme `<input type="text" name="prenom" id="prenom"/>` permet de saisir du texte qui sera envoyé au script de traitement comme valeur de la variable `prenom`. L'attribut `id` est facultatif, il permet de référencer la balise dans le code HTML, il peut servir de cible pour un fichier de style en CSS. On n'est pas obligé de donner le même nom aux attributs `name` et `id` mais c'est souvent plus simple.
Il existe d'autres types de champs de saisie `<input />` comme `"textarea"` pour des textes longs, `"password"` pour les mots de passe, `"file"` pour l'envoi de fichiers, `"checkbox"` pour les cases à cocher (plusieurs choix possibles) ou `"radio"` pour les boutons de radio (un seul choix possible).
Enfin pour envoyer les données du formulaire il faut insérer un bouton d'envoi avec un champ `<input type="submit" value="Valider" />` , la valeur de l'attribut `value` étant le texte affiché dans le bouton.

- La balise `<select name="image"> ... </select>` permet d'insérer une liste de choix optionnels sous forme de menu déroulant. Les options sont contenues dans des balises `<option value="valeur de l'option"> intitulé de l'option </option>`.

Pour plus de détails sur les formulaires on pourra consulter les tutoriels du Site du Zéro (désormais OpenClassrooms) : par exemple ceux de Mathieu Nebra sur HTML/CSS ou PHP/MySQL.

- `http://fr.openclassrooms.com/informatique/cours/apprenez-a-creer-votre-site-web-avec-html5-et-les-formulaires-8`
- `http://fr.openclassrooms.com/informatique/cours/concevez-votre-site-web-avec-php-et-mysql/transmettre-des-donnees-avec-les-formulaires`

2.4 Un script qui traite le formulaire précédent

Le script `traitement.py` ci-après réceptionne le formulaire envoyé par la page web générée par `formulaire.py` et génère une page web en réponse.
Les données sont récupérées dans un objet de la classe `cgi.FieldStorage()` du module `cgi`. On assigne cet objet à une variable `form` avec `form = cgi.FieldStorage()`. Les champs du formulaire peuvent être récupérés par le biais de leur attribut `name` : par exemple le champ d'attribut `name="prenom"` est stocké dans l'objet `form['prenom']`. Sa valeur est accessible par `form['prenom'].value`.
Pour les champs qui contiennent des fichiers comme `form['fichier']`, le fichier proprement dit est accessible par `form['fichier'].file`, son nom par `form['fichier'].filename`. On peut obtenir la liste de tous les attributs et méthodes de l'objet `form['fichier']` avec `dir(form['fichier'])`.
Par ailleurs on utilise dans le script :

- La fonction `cgitb.enable()` du module `cgitb` (pour cgi trace back), qui permet, en phase de développement, l'affichage des messages d'erreurs Python dans le navigateur (sinon ce dernier n'affiche que les messages d'erreurs du serveur Apache et ceux-ci ne nous disent rien des erreurs internes au code Python).

- Les fonctions `open()` et `close()` d'ouverture et de fermeture de fichier. Pour sauvegarder sur le serveur le fichier uploadé on a besoin de créer un répertoire `upload` qu'on affecte au groupe du serveur (`www-data`) en lui donnant les droits de créer et supprimer des fichiers. Dans l'extension utilisée sur la clef ISN, le serveur web (`www-data`) appartient au groupe de l'utilisateur `user` (vous) mais en général ce n'est pas le cas et par mesure de sécurité le serveur web n'a pas forcément accès à tous les répertoires sur le serveur physique.

- La fonction `os.path.splitext()` du module `os.path`. Elle retourne un couple formé de la racine puis de l'extension d'un chemin (ou nom) de fichier. On pourrait utiliser la méthode `split` des chaînes de caractères mais elle ne permet pas de traiter correctement les noms de fichier qui comportent des points avant l'extension comme `lena.1.png`.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
print "Content-Type: text/html\n"

import cgi

#pour le débogage (affichage des messages d'erreur dans le navigateur)
import cgitb
cgitb.enable()

#pour traiter les chemins de fichier comme celui du fichier uploadé
import os.path

print """<!DOCTYPE html >
<html>

<head>
    <title>Résultat du traitement du formulaire</title>
    <meta charset="utf-8"/>
</head>

<body>

    <h1> Résultat du traitement du formulaire </h1>
    """
#on crée un objet de type dictionnaire qui récupère le formulaire
#les clefs sont les noms #des différents champs du formulaire (attributs name)
form = cgi.FieldStorage()

if "nom" not in form or "prenom" not in form or "categorie" not in form or "image"
     not in form or "fichier" not in form:
    print """
    <h1> Formulaire mal rempli !!! </h1>

    Retournez sur <a href="formulaire.py"> la page d'inscription </a>.
    """
else:
    #on récupère le nom du fichier uploadé et son extension
    upload_name = form['fichier'].filename
    racine_nom,extension = os.path.splitext(upload_name)
    #si l'extension est acceptable on génère la page de bienvenue
    if extension.lower() in ['.png','.jpg','.bmp','.gif','.svg']:
        print '<p> Bienvenue %s %s.<br/>'%(form['prenom'].value,form['nom'].value)
        print 'Vous êtes un %s.<br/>'%form['categorie'].value
        print 'Vous avez envoyé le fichier image %s d\'extension %s. <br/>'%(
             upload_name,extension)
        #on récupère le contenu du fichier
        upload_content = form['fichier'].file.read()
        #on  sauvegarde le fichier sur le serveur
```

```
48          #dans un répertoire upload qui est en lecture-écriture pour le serveur (
                utilisateur www-data)
49          backup_path = 'upload/'+racine_nom+'-upload'+extension
50          f = open(backup_path,'wb')
51          f.write(upload_content)
52          f.close()
53          print 'Fichier sauvegardé sur le serveur dans %s. </p>'%backup_path
54          print """<p> Image transmise : <br/>
55          <img src= "%s" /> </p>"""%backup_path
56      else:
57          print '<p> Les fichiers d\'extension %s ne sont pas acceptés par mesure de
                sécurité </p>'%extension
58
59  print """
60      </body>
61      </html>"""
```

traitement.py

2.5 Attention à la sécurité

Une gestion fine des droits d'accès des utilisateurs est nécessaire pour sécuriser le serveur et il ne faut pas oublier que le serveur web ne s'exécute pas avec vos droits.
Lorsqu'on manipule des données issues de formulaires il faut être très méfiant : le code HTML du formulaire est visible de tous et rien n'empêche un utilisateur malveillant de fabriquer un faux formulaire pour envoyer des données indésirables. C'est pourquoi il faut toujours s'assurer que les données reçues sont du type attendu.
Par ailleurs, l'utilisateur peut insérer des balises HTML dans le texte transmis : dans une balise `<script>` on peut ainsi injecter du code Javascript malveillant. Pour prévenir ce risque on peut échapper systématiquement les caractères spéciaux HTML dans les textes reçus depuis un formulaire. En Python3 il existe pour celà la fonction `html.escape()` du module `html`.
Enfin, il faut être vigilant avec les fichiers uploadé sur le serveur. Il ne faut pas accepter des fichiers de script en PHP ou Python car ils pourraient servir à un utilisateur malveillant.

3 Un peu d'exercice

Exercice 1

Ecrire deux scripts CGI en Python : un formulaire qui demande à l'utilisateur de saisir un entier n et de sélectionner un entier p parmi $\{5,10,20,50\}$ et un script de traitement qui affiche la table des p premiers multiples de n comme ci-dessous.

```
5 premiers multiples de 11

1 x 11 = 11
.....
```

Exercice 2

1. Ecrire un script `formulaire_passwd.py` qui génère une page HTML demandant à l'utilisateur de saisir un identifiant et un mot de passe.

2. Ecrire le script de traitement `traitement_passwd.py` qui traite les données du formulaire précédent :

- il vérife si l'identifiant et le mot de passe sont corrects (on peut choisir 'ISN' comme identifiant et 'Turing' comme mot de passe) ;
- il ouvre un fichier texte `decompte.txt` pour récupérer le nombre de visiteurs, il l'incrémente et l'enregistre dans `decompte.txt` ;
- il génère une page HTML avec un affichage adéquat comme ci-dessous

Résultat du traitement du formulaire de connexion.
Identifiant ou mot de passe incorrect !!!
Retournez sur la page de connexion.

Résultat du traitement du formulaire de connexion.
Bravo, connexion réussie !!!
Bienvenue ISN. Vous êtes le 4 ème visiteur de notre site.