

SAE 2.03 : Programmation d'un serveur Web configurable

Introduction

En TP, vous avez dû programmer un serveur Web dont les fonctionnalités étaient les suivantes :

1. Reçoit des requêtes en TCP depuis un navigateur standard sur le port 80.
2. Analyse la requête pour y trouver un nom de fichier.
3. Retrouve le fichier et le renvoie par la même connexion TCP vers le navigateur, qui l'affiche.

Évolution du serveur

Dans cette SAÉ, on va rendre ce serveur configurable, c'est à dire qu'un certain nombre de fonctionnalités devront être modifiables, sans avoir à recompiler le serveur.

Les fonctionnalités sont :

1. Le port d'écoute sur le réseau
2. La spécification d'un répertoire servant de base au site web
3. La spécification d'adresses IP qui seront acceptées ou refusées par le serveur
4. Enregistrer les accès et les erreurs
5. La possibilité d'afficher l'état de la machine
6. Encoder images, sons ou vidéos en base64
7. Exécuter du code « dans » une page html

Fichier de configuration

Les quatre premières fonctionnalités seront définies dans un fichier au format XML, avec les balises suivantes :

- `<port>xyz</port> !-- Le port d'écoute du serveur`
- `<root>chemin absolu</root> !-- Le répertoire où se situe les fichiers du site web`
- `<accept>@IP/n</accept> !-- accepte toute requête provenant du réseau de l'adresse IP donnée`
- `<reject>@IP/n</reject> !-- refuse toute requête provenant du réseau de l'adresse IP donnée`
- `<accesslog>chemin absolu</accesslog>`
- `<errorlog>chemin absolu</errorlog>`

Encadrées par un balise `<webconf>`. Ci-dessous un exemple de fichier de configuration :

```
<webconf>
  <port>80</port>
  <root>/var/www</root>
  <accept>192.168.0.0/24</accept>
  <reject> 192.168.1.0/24</reject>
```

```
<accesslog>/var/log/myweb/access.log</accesslog>
<errorlog>/var/log/myweb/error.log</errorlog>
</webconf>
```

Le serveur web doit lire ce fichier avant d'accepter des connexions. Si certaines valeurs ne sont pas renseignées, il faut prendre des valeurs par défaut : port 80, répertoire courant, pas d'index et pas de sécurité des accès. Si certaines requêtes ne peuvent aboutir (mauvais nom de fichier, sécurité), prévoir des réponses standard (404...) pour le navigateur.

État du serveur

L'état de la machine sera accessible avec l'URL : `http://@serveur/status`

Il devra contenir :

- La mémoire disponible (non utilisée)
- L'espace disque disponible (idem)
- Le nombre de processus

Encodage

Tout fichier de type image, son, ou vidéo (img, png, jpg, mp3, wav ...) sera encodé en base64 et envoyé au client. Vous devrez utiliser les champs **Content-Encoding** et **Content-Type** du protocole HTTP.

Code dynamique

Votre serveur doit pouvoir prendre en compte la caractéristique suivante :

une balise `<code>` avec un attribut `«interpreteur»` va permettre de demander au serveur d'exécuter du code et de le remplacer dans la page html renvoyée au client.

Par exemple, la page suivante :

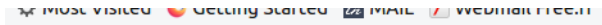
```
<html>
<body>
<h1> Exemple avec la date </h1>

<h2>en bash</h2>
La date est <code interpreteur=«/bin/bash»>date</code>

<h2>En python</h2>
La date est <code interpreteur=«/usr/bin/python»>
import time;
print(time.time())
</code>

</body>
</html>
```

Donne :



Exemple avec la date

en bash

La date est mardi 25 avril 2023, 11:11:05 (UTC+0200)

En python

La date est 1682413868.99

Programmation

Tout doit être programmé en java, avec notamment le package `java.net`. L'accès aux données du fichier de configuration doit se faire avec une API spécialisée dans la manipulation des données au format XML (`javax.xml`, `org.w3c`, ...).

Intégration au système linux

Fichiers

- Les classes nécessaires à votre serveur doivent être dans le répertoire :

`/usr/local/sbin/myweb`

- Le fichier de configuration doit se trouver dans le répertoire :

`/etc/myweb/`

et doit se nommer : `myweb.conf`

Quand le serveur se lance, il doit créer un fichier dans le répertoire :

`/var/run`

qui doit se nommer `myweb.pid`, il contiendra le numéro du processus qui a lancé le serveur (celui de java en fait)

Service

- Utilisez `systemD` pour créer le service « `myweb.service` », qui lancera ou arrêtera votre serveur web. Passez par l'intermédiaire de scripts, à créer dans `/usr/local/sbin`, pour démarrer et arrêter le service.
- Ce service devra aussi être démarré au boot de la machine

Package

Regrouper l'ensemble de votre travail sous la forme d'un package Debian (différent d'un package java). Ce package devra contenir les classes java, le service, les scripts et fichier de configuration.

Travail à rendre :

- Une description de votre programme (diagramme de classes, interactions, algo...)
- Un dossier archivé (tar.gz) des sources java
- Un dossier archivé du package