UQAC
UNIVERSITÉ DU QUÉBEC
À CHICOUTIMI

Université du Québec à Chicoutimi
Département d'informatique et de mathématique

8INF349 – Technologies Web avancées

TP1 - Client et serveur HTTP

Professeur : Hamid Mcheick	Groupe : 2 étudiants au max
Chargé du TD (labo): Tchaksim LANDROU	Date de distribution : 15 Mai 2024
Trimestre : ÉTÉ2024	Date de remise : 30 Mai 2024 (09:00)
Pondération : 15 points	

Vous devez réaliser **un seul choix**.

Prérequis:

Le protocole HTTP (HyperText Transport Protocol) est le protocole Web utilisé pour la communication de documents pour le World Wide Web. Il permet la transmission de documents HTML, d'images ou autres entre un serveur web et un navigateur. HTTP est défini dans la RFC 2616.

Format des requêtes:

Une requête HTTP minimale est un message textuel de lignes ayant la forme suivante :

```
GET Test/texte.html HTTP/1.0
Accept: text/plain , image/*
If-Modified-Since: Wed, 10 Sep 2018 14:20:21 GMT
Referer: http://www.uqac.ca/
User-Agent: Mozilla/2.0
Ligne blanche
```

Cette requête demande le fichier racine / du serveur contacté, en utilisant la variante 1.0 ou 1.1 de la norme HTTP. Attention, cette norme spécifie que les retours à la ligne qui doivent être marqués par des "\r\n" (encore que de simples "\n" semblent être tolérés par certains serveurs).

Plus généralement :

- La méthode GET de l'exemple peut être remplacée par d'autres méthodes. La principale autre méthode est POST, qui ne diffère de GET que par la façon de transmettre les arguments des formulaires, dont nous ne nous occuperons pas ici.
- A la place du premier /, peut se trouver le chemin menant au fichier voulu sur le serveur, du genre /c/est/ici/truc.html, voire même l'URL complète du fichier voulu, comme http://www.foo.fr/c/est/ici/truc.html.
- Entre la ligne initiale et la ligne vide, peuvent se trouver des lignes facultatives précisant par exemple le navigateur utilisé (User-Agent: XXX).

Format des réponses:

La réponse du serveur est constituée d'un entête et d'un corps, séparés par une ligne vide :

- L'entête commence par une ligne de statut, du genre HTTP/1.1 200 OK.
- Après cette ligne de statut se trouver un certain nombre de lignes indiquant des informations telles que l'heure, le type et la version du serveur, le type du fichier retourné (Content-Type : ...), sa taille (Content-Length : ...), etc.
- En cas de succès de la requête, après la première ligne vide, on trouve le corps du message de réponse, constitué du fichier demandé.

HTTP/1.0 200 OK
Date: Wed, 12 Sep 2018 14:20:21 GMT
Server: NCSA/1.5.2
Mime-Vesion: 1.0
Content-Type: text/html
Last-Modified: Wed, 12 Sep 2018 14:20:23 GMT
Content-Length: 139
Ligne blanche
<html> ……</html>

Enoncé du TP:

Dans ce TP, vous allez créer un client et un serveur HTTP avancés en utilisant Java, Python, PHP, etc. Les deux parties du TP devront être capables de gérer des fonctionnalités suivantes : les méthodes (*GET, POST, PUT et DELETE*), les redirections, les cookies, la génération dynamique de contenu HTML, l'authentification et la prise en charge du protocole HTTPS.

Partie 1 : Client HTTP

Objectif : Écrire un programme en Java, Python ou PHP qui agit comme un client HTTP avec des fonctionnalités avancées. Le client doit être capable de se connecter à un serveur web, demander une page HTML via la méthode GET, suivre les redirections, afficher les en-têtes de réponse, gérer les codes de statut HTTP et afficher le contenu HTML reçu sur la sortie standard.

Étapes :

1. Choisir un langage de programmation : Java, Python ou PHP....
2. Créer une connexion TCP avec un serveur web en utilisant les sockets.
3. Envoyer une requête GET pour demander un fichier HTML (par exemple, index.html) depuis le serveur web.
4. Recevoir la réponse du serveur, qui devrait inclure un code de statut HTTP, des en-têtes et le contenu HTML.
5. Analyser la réponse pour extraire le code de statut HTTP, les en-têtes et le contenu HTML.

6. Gérer les codes de statut HTTP, tels que 401 pour les fichiers inexistants. Si le code est 401, affichez le message d'erreur correspondant.
7. Si le code de statut est un code de redirection (3xx), suivez l'URL de redirection fournie dans l'en-tête "Location" et répétez les étapes 3 à 6 avec la nouvelle URL.
8. Afficher les en-têtes de réponse du serveur, y compris les cookies et autres informations pertinentes.
9. Si le code de statut est 200 (succès), affichez le contenu HTML sur la sortie standard.
10. Fermer la connexion TCP avec le serveur web.
11. Ajouter des options en ligne de commande pour permettre à l'utilisateur de spécifier l'URL, la méthode de requête (GET, POST, etc.), les en-têtes personnalisés et les paramètres de requête.
12. Tester le programme client en le connectant à différents serveurs web (par exemple, google.ca, amazon.com) et en demandant des fichiers HTML existants et inexistants. Vérifiez également le suivi des redirections et l'affichage des en-têtes de réponse.

Exercice 2 - Serveur HTTP

Objectif : Écrire un programme en Java, Python ou PHP qui agit comme un serveur HTTP simple avec des fonctionnalités avancées. Le serveur doit être capable d'écouter les connexions sur un port donné (80), de recevoir des requêtes HTTP (GET et POST), de mettre à jour les pages web HTML (telles que index.html) et de renvoyer une réponse HTTP minimale pour toutes les requêtes possibles. Le serveur doit également gérer les codes de statut HTTP, tels que 401 pour les fichiers inexistants. Testez votre serveur en utilisant un navigateur web.

Étapes :

1. Choisir un langage de programmation : Java, Python, PHP, etc.
2. Créer un socket serveur pour écouter les connexions sur un port donné (80).
3. Accepter les connexions entrantes des clients et créer un nouveau thread ou une coroutine pour gérer chaque connexion individuellement.
4. Lire la requête HTTP du client, qui devrait inclure la méthode (GET ou POST), l'URI, les en-têtes et, éventuellement, les données du formulaire pour les requêtes POST.
5. Analyser la requête pour extraire la méthode, l'URI, les en-têtes et les données du formulaire.
6. Si la méthode est GET, vérifiez si le fichier demandé (par exemple, index.html) existe. Si le fichier existe, préparez une réponse HTTP avec un code de statut 200 (OK), les en-têtes appropriés et le contenu du fichier. Sinon, préparez une réponse avec un code de statut 401 (Non autorisé) et un message d'erreur.
7. Si la méthode est POST, mettez à jour le fichier demandé (par exemple, index.html) avec les données du formulaire et préparez une réponse HTTP avec un code de statut 200 (OK), les en-têtes appropriés et le contenu du fichier mis à jour. Si le fichier n'existe pas, préparez une réponse avec un code de statut 401 (Non autorisé) et un message d'erreur.
8. Envoyer la réponse HTTP au client.

9. Fermer la connexion avec le client et répéter les étapes 3 à 9 pour les nouvelles connexions entrantes.
10. Tester le serveur en utilisant un navigateur web pour envoyer des requêtes GET et POST. Vérifiez que le serveur répond correctement avec des codes de statut 200 (OK) ou 401 (Non autorisé) et que les fichiers sont mis à jour en conséquence.

Livrables:

Vous devez remettre une copie complète de deux parties du TP à la page web Moodle, contenant :

- Le code complet de deux parties qui fonctionnent sur Eclipse/IntelliJ/… ou autre outil 70%
- Code documenté et bien structuré 10%
- Test : un scénario (trace-écran) d'exécution de chaque partie 10%
- Un manuel d'utilisation pour chaque partie. 10%

Pénalité de retard par jour : -10%

Vous devez montrer l'exécution de ces parties pour évaluer ce TP.