Phuong-Thi Nguyen
Léo de Montard
Elouan Raffray

Choix d'implémentations étape 3

Le point d'entrée dans le code ajouté a cette étape est une fonction `createReponse()` renvoyant la réponse associée a la requete reçue. Cette fonction va appliquer les différentes règles spécifiées dans les RFC afin de determiner le code d'erreur, la ressource a renvoyer, et les différents header à traiter ou créer.

La notion d'un répertoire raçine pour le serveur empêche l'accès aux fichiers en dehors de ce répertoire (pas de ../../../fichier) possible.
Pour renvoyer le header `Content-Type`, on utilise la commande `file` présente sous Linux pour récuperer le mime du fichier désigné. Puis on compare ce résultat avec la liste des mime voulus dans le header `Accept`.

Voici la liste des différents comportements implémentés:

- Un message-body en HEAD ou GET mène à 400 Bad Request
- De multiples header Host mènent à 400 Bad Request
- Aucun header host dans un message en HTTP/1.1 mène à 400 Bad Request
- requete de version HTTP supérieure a 1.1 renvoie 505 HTTP Version Not Supported
- Une méthode inconnue renvoie 501 Not Implemented
- La présence des header Transfer-Encoding et Content-Length dans le même message retourne 400 Bad Request
- Si le fichier designé n'existe pas, on renvoie 404 Not Found.
- Si l'uri designe un dossier, on rajoute 'index.html' derrière.
- Connection-Header
    - Si HTTP/1.1, on ferme la connection si l'option 'close' est présente, sinon la connection persiste.
    - Si HTTP/1.0, on fait persister la connection si l'option 'keep-alive' est présente, sinon la connection est fermée.
- Le percent encoding et le dot segment removal sont implémentés sur le request-target.
- La structure pour décoder/encoder le corps du message est présente (traitement du header accept-encoding/transfer-encoding), mais seul l'encodage chunked peut etre

décodé, les autres n'ont pas été implémentés.

- Content-Length est présent et contient la taille du corps des réponses envoyées.