Laboratoire

Politique de remise de travail

Méthodes de remise:

Via l'interface git hub https://classroom.github.com/a/IevW4xc2

Attention, la remise doit avoir lieu avant la date inscrite sur le site du cours http://mescours.ca/ . Toute remise après cette date de tombé sera considéré comme remise en retard et soumis à la pénalité de 10%/jour et ne sera pas corrigé après 7 jours de retard. Prévoyez les problèmes et délais possible de transmission internet, n'attendez pas à la dernière minute.

Installation de l'environnement de travail

À partir des instruction du laboratoire précédent, créer un environnement virtuel python avec les librairies nécessaires. Puis faire un "clone" (une copie) de votre dépôt de source de ce laboratoire.

Modifier votre serveur "web" de base

En utilisant les notions vu en cours, vous devez modifier votre serveur "web simple" pour le faire fonctionner en mode sécurisé HTTPS.

Votre serveur devra (différence avec le laboratoire #1 en bleu):
- Recevoir des requêtes HTTPS
- Traiter les requêtes reçues de la manière suivante:
  - Les appels dont l'adresse commence par **/static/*** devront retourner le fichier correspondant dans le répertoire **static** sur le disque
  - Les appels à l'adresse **/** devront être traités par le logiciel et retourner le contenu d'un gabarit nommé **accueil.tpl**
  - Les appels aux autres pages devront retourner une erreur **404** et le contenu d'un gabarit nommé **erreur404.tpl**

Vous aurez besoin de

Créer votre propre clé et certificat SSL avec les **outils OpenSSL** ou un générateur web disponible en ligne.

Installer et utiliser dans votre environnement virtuel les bibliothèques **pyopenssl** et **hashlib** (pas nécessaire pour python3, il est inclus).

Page d'accueil

Composants

Le gabarit de votre page d'accueil devra contenir les éléments suivants:

- Une page HTML 4.01 ou HTML 5 minimale valide
- Un titre <h1> (dont la valeur est mise en place par le logiciel)
- Un bloc de texte <p> ou avec les instructions suivantes: "**Tentez de deviner le code secret `5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8`**"
- Un formulaire <form>
  - ~~un champ nom~~ un champs <select> nommé algorithme avec les <options> suivantes: md5, sha1, sha224, sha256, sha384, sha512.
  - ~~un champ ville~~ un champ reponse
  - un bouton pour transmettre les information
  - doit transmettre les informations avec la méthode POST vers la page d'accueil / du serveur.
- Un logo quelconque <img> (image jpg ou autre)

Comportement

- Si la page est affichée directement (méthode GET sans arguments) elle doit s'afficher avec le formulaire
- Si la page est affichée après avoir remplis le formulaire (POST) vous devez afficher
  - **Vous avez choisi l'algorithme <algorithme choisi>. Votre réponse encodée donne <réponse encodée avec l'algorithme choisi>. Cela <correspond ou pas> avec le code à deviner.** En remplaçant les valeurs <algorithme>, <réponse encodée avec l'algorithme> et <correspond ou pas> par les valeurs appropriées.
  - La chaine de comparaison pour trouver le code est: `'5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8'` ce qui correspond à `'password'`. À vous de déterminer le bon algorithme pour arriver à ce résultat en testant votre système.
  - Vous ne devez plus afficher le formulaire.
  - Vous devez ajouter un lien <a> vers l'accueil original pour revenir au formulaire, intitulé "essayer à nouveau"

Page erreur 404

Composants

Le gabarit retourné devra afficher:

- Une page HTML 4.01 ou HTML 5 minimale valide
- Le nom de la page demandé
- Un message d'erreur pour indiquer qu'elle n'existe pas
- Un lien vers la page d'accueil (/)

Comportement

Tous les appels non prise en charge par le système (autre que /static/* et /) doivent retourner un erreur 404 avec le contenu du gabarit page404.tpl

Vous pouvez faire votre propre @route et fonction ou utiliser les outils fournit par flask pour gérer les pages inexistantes à votre guise.

Code

Votre code doit être lisible (variable avec des noms significatifs, commentaires, …)

Les données reçues (POST) doivent être validées avant d'être retournés au client. (s'assurer qu'ils ne contiennent pas de caractères illégaux, balises HTML, scripts, …)

À remettre

Tous les fichiers qui devraient être dans votre dépôt de source (voir les fichiers en rouge dans la section "arborescence et fichiers".

Arborescence et fichiers

Votre environnement devra dont contenir les répertoires et fichiers suivants:

- **projet**
  - **bin** (fichier du système)
    - …
  - **lib** (bibliothèques du système)
    -
    - …
  - **include** (bibliothèques du système)
    - …
  - **templates** (fichier gabarits)
    - accueil.tpl
    - erreur404.tpl
  - **static** (fichier statiques retournés par le serveur)
    - fichiers statique1
    - fichiers statique2
    - …
  - **scripts** (bibliothèques du système)
    - …
- .**hg** (répertoire du dépôt de source)
- .gitignore (fichier d'exclusion du gestionnaire de source)
- requirements.txt (fichier avec la liste des dépendances installées avec PIP)
- mon_projet.py (votre projet)
- ssl.key
- ssl.crt

Les fichiers en rouge et en bleu sont ceux qui devraient être dans votre gestionnaire de source. Ils sont également ceux à remettre.

Références:

Flask SSL
- https://pypi.org/project/flask-ssl/
- https://blog.miguelgrinberg.com/post/running-your-flask-application-over-https

HTML
- https://www.w3.org

REST (demandé en classe mais pas pour ce lab)
- http://www.pompage.net/traduction/comment-j-ai-explique-rest-a-ma-femme
- https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm