UQAC
Université du Québec
à Chicoutimi

Laboratoire #3

Politique de remise de travail

À remettre via l'interface "web" http://uqac.adninformatique.com ou le dépôt mercurial http://source.mescours.ca/ avant la date indiquée à 23h59. Toute remise après cette heure de tombé sera considéré comme remise en retard. Prévoyez les problèmes et délais possible de transmission internet, n'attendez pas à la dernière minute.

Les fichiers devraient être regroupés dans un fichier "zip" ou "tar" afin de préserver l'arborescence tel qu'indiquée.

Installation de l'environnement de travail

À partir des instruction de laboratoire #1, créer un environnement virtuel python avec les librairies nécessaires. Puis faire un "clone" (une copie) de votre dépôt de source du laboratoire #2.

Modifier votre serveur "web" de base

VOIR ADDENDUM À LA FIN DE CE DOCUMENT

NOTE: uwsgi ne s'installe pas correctement sous windows avec PIP. Vous pouvez le faire sous unix/linux avec une machine virtuelle ou téléchanger la version binaire pour windows à l'adresse http://nginx.org/en/download.html

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

Pour installer uwsgi dans votre environnement virtuel faire la commande: **pip install uwsgi.** Instructions complètes à http://uwsgi-docs.readthedocs.org/en/latest/WSGIquickstart.html

Configurer UWSGI comme serveur WEB pour utiliser vos clé et certificat SSL générés dans le laboratoire #2. Commande de base: **uwsgi --http :8080** pour démarrer le serveur. Lire la documentation pour lancer votre application derrière ce serveur.

Le logiciel de serveur web recevra les requêtes HTTPS et les transmettra à votre logiciel de livraison du contenu à l'aide du protocole uWSGI. Vous devrez donc modifier votre application pour répondre en uWSGI et non plus en HTTP ou HTTPS.

Modification du logiciel qui délivre le contenu

différence avec le laboratoire #1 en bleu:

- Traiter les requêtes reçu du serveur HTTPS via le protocole uWSGI.

- 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

Installer et utiliser dans votre environnement virtuel (répertoire static) les bibliothèques jquery et jqueryui.

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 champ nom et un champ prénom.
- un champ date (modifiable seulement par l'affichage d'un calendrier "date picker" de jqueriui)
- un bouton voir mon horoscope
- une zone (div) qui affichera l'horoscope.
- Un logo quelconque <img> (image jpg ou autre)

Comportement

- Lorsque la page racine (/) est invoquée elle doit afficher le formulaire (sans horoscope)
- Si les champs sont remplis et que le bouton "voir mon horoscope" est pressé:
  - Vous devez transmettre les information à votre serveur à l'aide d'une requête AJAX (POST) avec les valeurs de nom, prénom et la date. L'URL utilisé pour la requête doit être: /horoscope
  - Vous devez afficher le résultat de la requête AJAX dans la zone destiné à afficher l'horoscope.
- Lorsque la page du service "horoscope" (/horoscope) est appelée avec la méthode POST
  - Vous devez valider que vous avez reçu un nom, prénom et une date
    - Si un paramètre est manquant ou vide, vous devez retourner une erreur avec le message paramètre manquant.
  - Vous devez valider la date reçue
    - Si elle est invalide, retourner un message "date invalide"
    - Si elle est valide, retourner un message contenant "Le nom, prénom, le signe du zodiaque (idéalement accompagné d'un lien <a> vers son image) correspondant à la date accompagné d'un court texte de votre choix prédisant l'horoscope."

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/* , / , /horoscope) 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 donc contenir les instruction nécessaires au démarrage de votre système (démarrer uwsgi) et 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
    - …
- .**hg** (répertoire du dépôt de source)
- .hgignore (fichier d'exclusion du gestionnaire de source)

- require.txt (fichier avec la liste des dépendances installées avec PIP)
- mon_projet.py (votre projet)
- readme.txt (instruction de démarrage)
- uwsgi.cfg (fichier de configuration de uwsgi si besoin)
- 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.

ADDENDUM

Utiliser NGINX sous windows de la manière suivante:

Client -> HTTPS/443 -> NGINX -> HTTP/5000 -> VotreProgrammePython

Pour délivrer les pages dynamiques.

Client -> HTTPS/443 -> NGINX -> Disque

Pour délivrer les fichiers statiques (images, css, javascript, …)

Pour ce faire vous devez installer vos certificats SSL dans la configuration de NGINX et remettre votre code python en HTTP.
Utiliser le mode proxy d'NGINX pour appeler votre application.

Remettre votre configuration de NGINX avec votre projet.

Ce puisque l'utilisation de UWSGI est irréalisable pour le moment sous windows avec les outils à notre disposition. À noter que la configuration serait presque identique pour NGINX si l'on utilisait uwsgi. Il y aurait dépendant une étape de plus nécessaire pour la configuration de uwsgi.