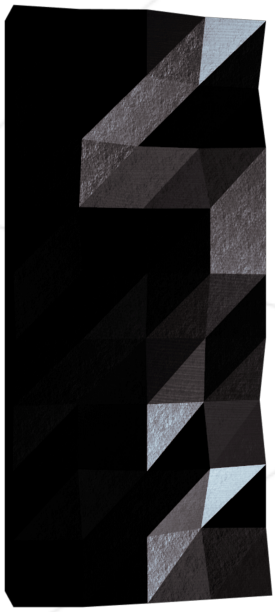

Formation Python-Django - 3

Sessions

Résumé: Où comment devenir ceinture noire des permissions et gestion d'utilisateurs avec un simple projet Django.

Version: 1

Table des matières

Chapitre I

Préambule

Voici quelques lignes d'un ouvrage parlant de fabrication artisanalle d'adresses IP à l'ancienne :

I.1 TRAITÉ De l'art juste de fabriquer des adresses hypées

En cet escript se trouvera l'art véritable & peu souvent exposé, de la fabrication des adresses hypées, comme les anciens l'ont appris dans les ateliers de Vinton-le-Grand. Pleust à Dieu qu'un chascun sut ceste juste & belle matière trop négligée, produisant des adresses si malpropres à l'utilisation dans les échanges de lors.

Souventement, on lira et même on pratiquera la fabrication tant honnie des adresses modernes. De celle-ci n'est pas traictée ycelieu, pour ce qu'elle produit des adresses de peu d'usage, qui s'étiolent et pourrissent en peu de temps comme chiabrenas de pucelles. Celles expliquées en cestes pages, haulte matière et science profonde, dites hypées véritables, sont seules à même de résister au fléau ancien, Faillibus-aulniem, d'Altus-droopium filium.

On se doit adoner songeusment tant à l'estude et vertu parfaicte des ouvriers qu'à la préparation des ingrédients et outils, sans peine épargner pour produire des adresses aigues, subtiles, profondes et seraines, et admirablement proportionnées pour parvenir à degré souverain d'échanges.

I.1.1 Des meurs bellement aornées

Comme disoit le saige Salomon, Sapience n'entrant point en âme malivole, qu'il est une part importante de l'influence du fabricant sur le fabriqué, comme un miroir en représentant la plus haulte vertu. Ce n'estoit donc que les plus fervents et les plus sains ouvriers qui peuvent laquelle entreprise parfaire et consommer.

Vous pouvez trouver la suite de cette saine lecture ici.

Chapitre II

Règles communes

- Votre projet doit être réalisé dans une machine virtuelle.
- Votre machine virtuelle doit avoir tout les logiciels necessaire pour réaliser votre projet. Ces logiciels doivent être configurés et installés.
- Vous êtes libre sur le choix du systmème d'exploitation à utiliser pour votre machine virtuelle.
- Vous devez pouvoir utiliser votre machine virtuelle depuis un ordinateur en cluster.
- Vous devez utiliser un dossier partagé entre votre machine virtuelle et votre machine hote.
- Lors de vos évaluations vous allez utiliser ce dossier partager avec votre dépot de rendu.
- Vos fonctions de doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

Chapitre III

Règles spécifiques de la journée

- Vous devez utiliser la version LTS de django.
- Vous pouvez utiliser la version de booststrap de votre choix. Cette version doit fonctionner avec la version LTS de django.
- Vous devez utiliser Python3.

Chapitre IV

Exercice 00

	Exercice :
Exercice 00 : Sessions anonymes.	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django, random, django-bootstrap*`	

Vous démarrez ici un projet de site pour proposer aux utilisateurs de poster des "Life Pro Tips".

Après avoir créé votre projet, une application et mis en place une page d'accueil accessible à l'URL `/`, implémentez le système de sessions anonymes suivant :

- Un internaute visitant votre site doit se voir attribuer un nom d'utilisateur. Ce nom est choisi aléatoirement dans une liste de 10 noms que vous devez définir dans le fichier `settings.py` de votre projet et a une durée de validité de 42 secondes après sa définition.
- Ce nom doit être persistant et rester identique pendant 42 secondes, une fois ce délai écoulé, ce nom doit être redéfini de la même manière que précédemment.
- Mettez en évidence le comportement ainsi implémenté en ajoutant un message dans un élément `<nav>` en tête de page d'accueil. Ce message sera simplement de la forme "Hello user !" en remplacaçant `user` par le nom aléatoirement attribué au visiteur jusqu'à la fin de validité du nom en question.
- Le nom doit apparaître immédiatement sur les pages visitées : Si vous avez besoin d'actualiser encore une fois la page pour qu'il apparaîsse ou soit changé, c'est que l'exercice n'est pas réussi !

Chapitre V

Exercice 01

	Exercice :
Exercice 01 : Création d'utilisateurs.	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django, django-bootstrap*`	

À présent que le système de sessions anonymes est ajouté, il est temps d'implémenter la gestion des utilisateurs avec la création d'une interface d'authentification.

Vous réaliserez cette interface en plusieurs temps dans cet exercice :

- Commencez par l'ajout d'une page d'inscription, et d'une page de connexion. Ajouter un lien vers chacune de ces pages dans votre élément `nav`. Ajoutez pour cela les urls nécessaires, les templates qu'il vous faut, les vues, etc...

  Ajoutez également le nom du site au début de votre `nav` et faites-en un lien vers la page d'accueil.

- Créez les formulaires à insérer dans ces pages, avec les spécificités suivantes :

  - Formulaire d'inscription :

    — Vous devrez insérer 3 champs différents dont :
    — Un champ pour le nom d'utilisateur.

    — Un champ pour le mot de passe.

    — Un champ pour la vérification du mot de passe.

    — La validation ne devra s'effectuer que si chaque champ est complété, que le nom renseigné n'existe pas déjà au préalable et que les champs de mot de passe aient bien 2 valeurs identiques.

- Formulaire de connexion :
  - Vous devrez insérer 2 champs différents dont :
    - Un champ pour le nom d'utilisateur.
    - Un champ pour le mot de passe.
  - La validation ne devra s'effectuer que si chaque champ est complété, et que le couple utilisateur / mot de passe renseigné existe dans votre base de données et permette une authentification.

- Implémentez maintenant le comportement des pages :
  - En cas de validation des données :

    Inscription : Création et activation d'un nouvel utilisateur avec les informations fournies. Puis connexion avec cet utilisateur et redirection vers la page d'accueil.

    Connexion : Connexion avec l'utilisateur correspondant aux informations entrées, puis redirection vers la page d'accueil.
  - En cas de non-validation des données :
    Dans tous les cas, vous devez réafficher la page avec le formulaire contenant les informations précedemment entrées (Sauf les mots de passe).

    Inscription : Affichage d'un (ou plusieurs) message(s) d'erreur expliquant l'erreur (Champ requis, nom d'utilisateur déjà utilisé, mots de passe différents)

    Connexion : Affichage d'un message d'erreur avec le formulaire (champ requis ou informations de connexion incorrectes).

- Pour finir, lorsque l'utilisateur est connecté :
  - les liens vers les pages d'inscription et de connexion sont remplacés par un lien de déconnexion (évidemment fonctionnel) qui déconnecte le visiteur et le redirige vers la page d'accueil une fois celui-ci cliqué.
  - le nom de la session anonyme dans l'élément `nav` est remplacé par le nom de l'utilisateur.
  - le visiteur ne doit plus avoir accès aux pages d'inscription et de connexion, s'il tente d'y accéder, il est redirigé vers la page d'accueil.

- Bien évidemment, les mots de passe doivent être cachés lorsqu'ils sont tapés.

Dans un autre exercice, vous aurez à faire votre propre classe utilisateur pour remplacer celle fournie par défaut.
Vous seriez donc bien inspirés (mais pas obligés) d'anticiper ce changement dans votre projet pour ne pas avoir à réécrire manuellement chaque appel à votre classe utilisateur dans votre code.

Chapitre VI

Exercice 02

	Exercice :
Exercice 02 : Nos Tips !	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django, django-bootstrap*`	

Il est temps d'implémenter la fonctionnalité principale de ce projet : Un système d'astuces !

Pour cela, vous aurez besoin de concevoir un Model `Tip` avec :

- Un champ `contenu` avec... le contenu du tip (évidemment sous forme de texte).
- Un champ `auteur` (l'utilisateur ayant créé le tip).
- Un champ `date` (la date et l'heure de création).

Les tips devront être listés dans la page d'accueil et tous les attributs cités dans cet énoncé doivent apparaître.

Une fois connecté, l'utilisateur doit avoir la possibilité de créer un Tip avec un formulaire en page d'accueil (utilisez un `modelform` !)

Si l'utilisateur n'est pas connecté, le formulaire ne doit pas apparaître et le visiteur ne doit pas pouvoir créer de tip.

Chapitre VII

Exercice 03

	Exercice :
Exercice 03 : Votes	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django, django-bootstrap*`	

Maintenant que notre système de Tips est en place, il serait bien de pouvoir repérer les meilleurs : Il est donc temps de mettre en place un système d'upvotes et de downvotes !

Pour cela, vous devrez modifier votre Model `Tip` afin d'implémenter ces fonctionnalités.

Vous devrez également implémenter un moyen de supprimer les tips que l'on ne considérera pas comme intéressants.

Vous devez donc ajouter aux tips listés en page d'accueil :

- Un bouton de suppression
- Un bouton d'upvote
- Un bouton de downvote
- Le nombre d'upvotes
- Le nombre de downvotes.

Pour l'instant, la seule restriction à implémenter sera la nécessité d'être connecté pour effectuer des actions de vote ou de suppression. Si l'utilisateur n'est pas connecté, aucune de ces actions ne seront possibles.

Chacunes de ces actions rechargera la page.

Bien évidemment, un `Tip` fraichement créé aura 0 upvotes et 0 downvotes.
Attention :

- Il ne doit pas être possible d'upvoter et de downvoter en même temps un même tip.
- Un utilisateur ne peut upvoter ou downvoter qu'une seule fois un même tip. Recliquer sur le bouton annule simplement le downvote ou l'upvote.
- Si un utilisateur downvote un tip auparavent upvoté, l'upvote est annulé en plus du downvote (et vice-versa).

Soyez particulièrement attentif à l'implémentation de votre système d'upvote. Une incohérence, une imperfection (par exemple un nombre de votes ne baissant pas après annulation du vote) invalidera l'exercice. Envisagez l'utilisation de champs manytomany plutôt que des "compteurs" pour implémenter le système de votes.

À ce stade, chacunes des actions doit pouvoir fonctionner uniquement si le visiteur est connecté.

Vous pouvez ne pas afficher ou désactiver les éléments devant être inutilisables par l'utilisateur.

Chapitre VIII

Exercice 04

	Exercice :
Exercice 04 : Premier usage des permissions.	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django, django-bootstrap*`	

Maintenant que notre projet est un peu plus complet nous nous rendons compte que n'importe-qui peut faire n'importe quoi pourvu qu'il soit connecté. Il est sans doute temps d'ajouter certaines restrictions.

L'utilisateur doit à présent avoir l'autorisation de supprimer des Tips pour... supprimer des tips.

Activez l'interface d'administration incluse par défaut avec Django et utilisez un superutilisateur pour modifier les permissions et ainsi mettre en évidence le comportement de votre projet.

Bien évidemment, supprimer le bouton "Supprimer" de la page n'est pas suffisant (même si vous pouvez le faire). À vous de trouver la bonne manière de restreindre cette fonctionnalité en vérifiant correctement les permissions.

NB : Évidemment, l'auteur d'un Tip peut supprimer son tip même s'il n'a pas l'autorisation d'en supprimer en général. C'est la seule exception à cette règle.

Chapitre IX

Exercice 05

	Exercice :
Exercice 05 : Permissions personnalisées.	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django.`	

Vu le caractère négatif du downvote, il faudrait également restreindre son utilisation à certains utilisateurs bien choisis.

Implémentez le même contrôle que l'exercice précédent, mais pour les downvotes. Comme une telle permission n'existe pas par défaut et qu'il serait moche que vous utilisiez une autre permission non adaptée, faites votre propre permission pour restreindre l'utilisation du downvote !

Tout comme l'exercice précédent, il ne s'agit pas ici simplement de retirer un bouton de la page.

NB : Tout comme pour la suppression d'un tip, l'auteur peut downvoter ses propres tips meme si c'est stupide.

Chapitre X

Exercice 06

	Exercice :
Exercice 06 : Automatisation et réputation	
Dossier de rendu : *ex/*	
Fichiers à rendre : `Votre projet, votre fichier requirements.txt`	
Fonctions Autorisées : `L'ensemble des fonctionnalités de Django, django-bootstrap*`	

À présent que nous avons la plus grande partie des outils nécessaires pour faire tourner ce projet, il serait bien de ne plus avoir à gérer manuellement les permissions. Certains sites ont instauré un gain de privilèges allant avec un gain de réputation. De cette manière, les utilisateurs s'en montrant dignes débloquent de nouvelles actions ! C'est le sujet de cet exercice dans lequel vous devrez instaurer un système de réputation dépendant des votes reçus par les Tips qu'un utilisateur a posté.

Remplacez la classe d'utilisateur standard par votre propre classe utilisateur. Implémentez dans cette classe un moyen de faire dépendre de la réputation d'un utilisateur, les autorisations de downvoter et de supprimer des tips. La réputation de cet utilisateur dépendra elle-même de la somme des upvotes et downvotes reçus par les Tips postés par cet utilisateur.

Chaque nouvel utilisateur commence avec 0 points de réputation. Chaque upvote recu sur un de ses Tips lui rapportera 5 points de réputation alors qu'un downvote lui en enlèvera 2. Si un Tip est supprimé, l'influence de ses votes sur la réputation de son auteur est annulée.

Un utilisateur débloquera l'autorisation de downvoter les Tips autres que les siens à partir de 15 points de réputation, et celle de les supprimer à partir de 30 de réputation. Si la réputation d'un utilisateur baisse, il peut perdre les permissions que sa réputation lui avait apportées.

Modifiez également l'affichage du nom de l'utilisateur en affichant sa réputation à côté dans vos pages, entre parenthèses.

Chapitre XI

Rendu et peer-évaluation

Rendez votre travail dans votre dépôt `Git` comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

`L'évaluation se déroulera sur l'ordinateur du groupe évalué.`