TP6 Python Django

Sécurité et site d'administration

https://docs.djangoproject.com/en/5.1/

L3 - IFNTI Sokodé – 5 novembre 2024

Ce TP contient 3 parties indépendantes. D'abord nous nous pencherons sur l'authentification. Ensuite nous explorerons les problèmes de sécurité : CSRF, HTTPS, clé de chiffrement et détournement de clic. Enfin nous verrons comment customiser le site d'administration.

1 Authentification

1. Ouvrez le fichier `settings.py` et trouvez là où l'authentification est configurée. Listez l'ensemble des installations installées et les autres constantes du fichier gérant l'authentification.

2. Recopiez ce dont vous vous souvenez sur les différents utilisateurs qui peuvent accéder au site. Qui devrait pouvoir utiliser la vue de formulaire d'ajout d'une note ?

3. Dans le site d'administration, vous pouvez voir qu'il y a 2 classes que vous n'avez pas ajouté vous-mêmes. Quelles sont ces 2 classes ? Regardez dans la classe `User`. Qu'y a-t-il déjà dedans ? Est-ce normal ? D'où cela vient-il ?

4. Créez 3 groupes correspondants aux 3 groupes de la question 1.2 qui ne sont pas publics. Quelles permissions avez-vous données à chacun de ces groupes ? Une fois ajoutés, insérez une capture d'écran du site d'administration montrant clairement ces 3 groupes.

5. Django fournit une classe `User` qui contient l'ensemble des informations nécessaires à l'authentification et aux autorisations. Quelles sont les 2 manières d'étendre le modèle `User` par défaut sans que cela modifie votre propre modèle ?

6. Utilisez celle qui fait intervenir un `OneToOneField` en ajoutant un attribut au modèle `Personne`. Vous venez de changer le modèle, vous devez donc lancer les commandes `makemigrations` puis `migrate`. Lorsque vous créerez une nouvelle migration, vous devriez avoir une erreur du type :

   ```
   You are trying to add a non-nullable field 'user' to person without a default;
   we can't do that (the database needs something to populate existing rows).
   ```

   Pour corriger cette erreur, nous devons rendre le nouveau champ `nullable`. Tapez `Ctrl-C` et corrigez l'erreur dans le bon fichier. Qu'avez-vous fait ? Recopiez la ligne que vous avez ajoutée.

7. Pour chaque instance existante de `Personne`, créez un utilisateur et associez-le à cette instance. Le nom d'utilisateur est la première lettre du prénom en minuscule concaténée au nom en minuscule et le mot de passe temporaire est `ifnti2023`. Insérez une capture d'écran de la page du site d'administration qui liste toutes les instances de `User`.

8. Annulez la correction "nullable" que vous aviez faite à la question 1.6 dans le modèle puis migrez de nouveau vos données. Qu'avez-vous fait ? Recopiez ce qui est écrit dans le terminal.

9. Nous pouvons maintenant nous pencher sur l'accès aux pages. Dans le site d'administration, donnez un groupe à chaque instance de `User`. Donnez les associations que vous avez faites.

10. Faites en sorte que chaque vue de votre projet sauf celles auxquelles tout le monde peut accéder requièrent une authentification et donnez l'accès aux utilisateurs en fonction de leurs permissions. Comment avez-vous fait ? Recopiez le contenu du fichier.

11. Il faut se déconnecter de votre compte pour pouvoir tester l'accès des utilisateurs. Pourquoi ? Faites-le et expliquez comment vous avez fait. Essayez d'accéder aux vues à accès restreint (toutes sauf la page d'accueil). Vous devriez voir 2 erreurs consécutivement. Que disent-elles ?

12. Que devez-vous faire pour éviter ces erreurs ? Faites-le, testez si cela fonctionne et recopiez votre code. L'accès devrait maintenant être correctement restreint pour les utilisateurs.

Nous n'avons vu qu'une partie de ce qu'il est possible de faire avec l'authentification sur Django. Essayez de vous renseigner sur la déconnexion, le changement et la réinitialisation des mots de passe, le mailing, la redirection, et le reste de tout ce qui peut être fait avec l'authentification.

2 Les problèmes de sécurité

Pour chaque question qui suit, essayez d'être le plus précis possible et de bien tout expliquer avec vos propres mots.

1. Qu'est-ce que le `Cross site scripting (XSS)` ? Comment est-ce que Django s'occupe de ce problème ? Quelles sont ses limites ? Que devez-vous faire pour éviter les attaques XSS ?
2. Qu'est-ce que `CSRF` ? En quoi cela consiste-t-il ? Vous ne vous en souvenez certainement pas, mais vous avez déjà fait quelque chose permettant d'éviter les attaque CSRF. Qu'avez-vous donc déjà fait ? En quoi consiste cette protection ? Dans quel cas devez-vous vous protéger ? Quelles sont les limites de la protection de Django contre les attaques CSRF ?
3. Qu'est-ce qu'une `injection SQL` ? Comment Django se protège de ce type d'attaque ? Avec quelles limites ?
4. Qu'est-ce que le `détournement de clic` ? En quoi cela consiste-t-il ? Donnez un exemple d'attaque par détournement de clic.
5. Qu'est-ce que `SSL` ? Qu'est-ce que `HTTPS` ? Qu'est-ce que `HSTS` ? Expliquez aussi pourquoi ces sigles sont importants.
6. Donnez une explication brève sur les `Sessions` en Django, sur la `Validation de l'en-tête Host` et sur le `Contenu envoyé par les utilisateurs`.

3 Customiser le site d'administration

Le site d'administration par défaut est sympa, mais peut être amélioré, surtout pour les modèles que vous avez importés.

1. Pourquoi est-il intéressant de customiser le site d'administration ? Pour qui sera-t-il amélioré ?
2. Comment pourrions-nous rendre nos modèles accessibles depuis le site d'administration d'une autre manière que celle que nous avons utilisée ?
3. Expliquez précisément les options suivantes :
   - `date_hierarchy`
   - `empty_value_display`
   - `list_filter`
   - `list_per_page`
   - `list_display`, `list_display_links` et `list_editable`
   - `fields` et `exclude`
   - `save_as` et `save_as_continue`
   - `raw_id_fields`
   - `readonly_fields`
   - `show_full_result_count`
   - `save_on_top`
   - `sortable_by`
   - `search_fields`
   - `filter_horizontal` et `filter_vertical`
   - `view_on_site`
4. Utilisez quelques-unes des options de la question précédente comme vous le souhaitez afin que la page d'administration d'un élève soit "belle". Recopiez la classe `EleveAdmin` et insérez une capture d'écran de la page de modification d'une instance d'`Eleve`.

5. Pouvons-nous modifier les `templates` d'administration ? Comment est-il possible de redéfinir les `templates` par défaut ? Pouvons-nous créer de nouveaux `templates` ?

6. Pouvons-nous modifier le `JavaScript` et le `CSS` du site d'administration ? Comment insérez notre propre code ?

7. Quelle est l'entête par défaut des pages du site d'administration ? Comment pouvons-nous la modifier ? Quelle serait la conséquence sur les modèles `User` et `Group` ? Comment pourrions-nous contourner cette conséquence ?