Django

> **Persona :**
> *En UX (étude de l'expérience utilisateur), un persona est un personnage fictif représentant un archétype utilisateur d'un service (site web, application mobile, etc.).*
> *Les personas peuvent être très utile pour créer des utilisateurs factices afin de tester une application/un service.*

Objectif généraux du projet Django

Créer un service permettant de générer des personas (stockés dans une BDD SQLite) et de les gérer (consultation, suppression).

- La modification/suppression d'un persona **n'est pas demandée** dans la mesure où nous n'avons abordé que récemment ces concepts
- Le système d'héritage de template **n'est pas demandé** pour la même raison

Livrables attendus :

- Un dossier projet contenant :
  - Le projet Django complet (dossier de projet, dossier d'application, BDD sqlite, etc.)
  - Le dossier d'environnement virtuel (!)
  - Le projet doit être prêt à exécuter (`runserver`) sans erreur et doit générer un site web navigable

Spécifications générales

Dans notre projet, un persona sera défini par les attributs suivants :

- `first_name`
- `last_name`
- `address_street`
- `address_number`
- `city`
- `country`
- `postcode`
- `email`
- `username`
- `password`
- `age`
- `picture`

Spécifications détaillées

1. Créer l'environnement Django complet

- Création d'un dossier global pour le projet : `examen_django`
- Environnement virtuel
- Installation de Django
- Installation de la librairie `request` : `pip install requests`
- Création d'un projet Django `persona_project`
- Création d'une application `persona_app`
- Configuration du projet Django
- Migration des tables de base
- Lancement du serveur

Note : A ce stade, l'architecture de votre dossier projet devrait être la suivante :

```
examen_django/
├── db.sqlite3
├── manage.py
├── myenv/
├── persona_app/
└── persona_project/
```

2. Créer le modèle django `Persona`.

- Créez un modèle avec tous les attributs spécifiés en introduction
- Indications pour certains types d'attributs particuliers :
  - `password` : Charfield (est stocké en clair - ne faites **jamais** cela dans la vraie vie 😉)

    - `picture` : Charfield (il s'agit d'une URL, donc une chaine de caractères)
    - Vous aurez besoin du type : `IntegerField()` pour certains attributs du modèle
- Le modèle doit disposer d'une méthode magique `__str__` qui affiche des informations pertinentes sur l'objet par exemple sous la forme : `(id) first_name last_name`

3. Effectuer les migrations Django

- La table `persona_app_persona` correspondante est créée en BDD grâce à une migration réussie
- Remplir manuellement quelques lignes de la table grâce à un logiciel d'exploration des BDD SQLite
- Si vous souhaitez utiliser l'administration de Django pour remplir les lignes :
    - Créer un administrateur (`python manage.py createsuperuser` dans votre terminal)
    - Modifier le fichier `persona_app/admin.py` et rajouter les lignes :

```
from .models import Persona

admin.site.register(Persona)
```

    - Accéder à l'administration : `https://127.0.0.1:8000`
- Pour générer de fausses données, vous pouvez utiliser l'URL : https://randomuser.me/api/?nat=fr et copier les données (rechargez la page pour générer de nouvelles données)

4. Créer les couples vues/routes pour :

- Lister les personas : `persona_list`
- Afficher les détails d'un persona : `persona_details`
- Générer un nouveau persona : `persona_generate`

Note : A ce stade, les vues n'effectuent aucune action, elles sont vides et ne renvoient qu'un objet de type `HttpResponse` avec une chaine de caractère de votre choix pour démontrer leur bon fonctionnement

5. Créer le template et implémenter la vue `persona_list` :

- Affiche l'ensemble des personas stockés en BDD
    - **Par ordre inversé d'attribut** `id`
- La forme d'affichage est laissée libre, pour chaque persona les attributs suivants doivent être affichés : `id` , `first_name` , `last_name` , `email`
- Chaque persona doit disposer d'un lien hypertexte redirigeant vers les détails du persona correspondant
    - Le lien peut-être positionné sur les `first_name` & `last_name`

Note : Vous n'avez pas besoin d'implémenter un template global par héritage (mais vous pouvez le faire si vous le souhaitez, c'est plus propre !)

6. Créer le template et implémenter la vue `persona_details`

- Affiche l'intégralité des attributs du persona concerné
  - **Y compris la photo de profil !**
- Affiche un lien hypertexte de retour vers la liste globale des personas
- La forme d'affichage est laissée libre

Note : Vous n'avez pas besoin d'implémenter un template global par héritage (mais vous pouvez le faire si vous le souhaitez, c'est plus propre !)

7. [Difficulté +] Implémenter le mécanisme de génération de personas par l'API

- Rajouter un lien hypertexte `New persona` sur la liste globale des personas :
- Ce lien redirige vers une nouvelle vue `persona_generate`
- Génère un persona en consommant l'API `https://randomuser.me/api?nat=fr`
- Redirige vers la page de détails du persona qui vient d'être créé

Aides diverses pour le point n°7 :

- Vous n'avez pas besoin de template pour la génération d'un persona par l'API :
  - La vue génère les données grâce à l'API
  - Sauvegarde les données en BDD
  - Récupère l'id de la nouvelle donnée
  - Redirige vers la vue de détails grâce à l'id
- Pour interroger l'API vous pouvez utiliser la librairie Python `requests` (`pip install requests`)
  La documentation de cette librairie comporte **toutes les informations** nécessaires (notamment pour récupérer le Json retourné) : https://docs.python-requests.org/en/latest/user/quickstart
- Pour **afficher, à des fins de debug** (uniquement pour cela) du Json de manière lisible dans votre terminal, vous pouvez utiliser le snippet de code suivant :

```
import json

datas = [{'title':'a cool title','artist':'john doe','year':2020},{'title':'a second music

# Toute source Json ou dictionnaire/list
# peut être affiché ainsi
# ici le contenu de la variable data
print(json.dumps(datas, indent=4))
```

  Cette méthode est très utile pour afficher le contenu json/dictionnaire/liste d'une variable d'une manière visuelle
- Pour sauvegarder une instance d'un modèle, vous pouvez utiliser la méthode `save()`. Pensez au PDF du cours !

    - Bonus : une fois utilisée, cette méthode vous autorise à accéder directement à l'attribut `id` de l'instance sauvée. Pratique !
- Pour effecuter une redirection depuis une vue Django (y compris en passant des paramètres), vous pouvez utiliser la méthode `redirect` dont la documentation est consultable ici : https://docs.djangoproject.com/fr/3.2/topics/http/shortcuts/#redirect

---

Bon courage !
Geoffroy Ladrat - 2021