Université Assane SECK de Ziguinchor

Unité de Formation et de Recherche des Sciences et Technologies

Département d'Informatique

Développement d'applications avec Django

Licence 2 en Ingénierie Informatique

Avril 2022

©Papa Alioune CISSE

Papa-alioune.cisse@univ-zig.sn

Résumé : À la différence du Développement Front-End, qui concerne l'aspect visuel et ergonomique d'un site web, le Développement Back-End se penche sur les aspects techniques et fonctionnels du développement d'un site web dynamique. Il s'agit du développement de l'ensemble des fonctionnalités "invisibles" d'un site web (le serveur, le code applicatif, la base de données, etc.) qui sont indispensables au bon fonctionnement d'un site. Ainsi, on peut dire que les Développeurs Back End travaillent sur la partie immergée de l'iceberg, alors que les Développeurs Front End se chargent essentiellement de la partie visible.
Ce chapitre est une continuité du précédent. Il aborde la partie sur les « Templates Django » et la personnalisation du système d'authentification de Django.

1 - IMPLÉMENTATION DES FONCTIONNALITÉS DE « APPTEST »

1.1 - Fonctionnalité de visualisation de la liste des développeurs

Cette fonctionnalité permet aux visiteurs de visualiser la liste de tous les développeurs existants dans notre base de données avec les informations suivantes : prénoms, noms, spécialités/professions, nombre d'année d'expériences, … Pour voir les détails sur un développeur, il suffira d'ouvrir son CV en cliquant sur un bouton.

La première chose à faire consiste à ajouter l'url (dans le fichier « urls.py » de « apptest ») permettant d'accéder à cette fonctionnalité :

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.accueil),
    path('developpeurs', views.developpeurs, name='developpeurs'),
]
```

Il faut ensuite ajouter la vue « developpeurs » dans le fichier « views.py » de « apptest ». Vous pouvez retrouver ce fichier « views.py » dans le dossier (codes sources/apptest/).

Essayons de comprendre le code de la vue « developpeurs » donné ci-dessous :

```
@login_required
def developpeurs(request):
        liste_developpeurs = db_access.getCVs()
        return render(request,'apptest/developpeurs.html', {'developpeurs': liste_
developpeurs})
```

La 1ère ligne de ce code est une restriction faite sur l'accès à la vue « developpeurs » qui veut dire qu'il faut s'authentifier pour y accéder.

Dans cette vue, nous récupérons la liste des développeurs en invoquant la méthode « getCVs() » du module « db_access ». La liste des développeurs est ensuite passée à la page html (le template « developpeurs.html » que vous retrouverez dans le dossier « codes sources/apptest/templates/ ») qui est censée afficher la liste.

Il suffit maintenant d'ajouter la page « developpeurs.html » dans le dossier « templates/apptest/ » de l'application « apptest ».

Tester cela en tapant sur la barre d'adresse de votre navigateur « 127.0.0.1 :8000/apptest/developpeurs »

La page qui s'affiche donne la liste des développeurs dans un tableau où chaque ligne renseigne sur un développeur. La dernière colonne de chaque ligne comporte un bouton qui, si cliqué, renvoie au CV du candidat.

1.2 - Fonctionnalité de visualisation du CV d'un développeur

La page qui présente les CVs des développeurs est accessible depuis la liste des développeurs avec l'url (à définir dans « urls.py » de « apptest ») ci-dessous :

```
from django.urls import path
from django.conf.urls import url
from . import views

urlpatterns = [
    path('', views.accueil),
    path('developpeurs', views.developpeurs, name='developpeurs'),
    url(r'^cv/(?P<idCV>[0-9]+)/$', views.cv),
]
```

L'id du cv à visualiser est donné dans l'URL. Il s'agit d'un chiffre, ce qui explique la nature de cette url.

La vue « cv » censée traitée cette requête doit être aussi ajoutée dans « views.py » de « apptest » : à regarder ensemble pour essayer de comprendre. La page html (cv.html) qu'elle renvoie est dans le dossier (codes sources/apptest/templates/).

1.3 - Formulaire de demande de recrutement d'un développeur

Ce formulaire, accessible depuis la page de visualisation d'un CV, permet aux recruteurs de de demander le recrutement d'un développeur.

L'url d'accès est :

```
from django.urls import path
from django.conf.urls import url
from . import views

urlpatterns = [
    path('', views.accueil),
    path('developpeurs', views.developpeurs, name='developpeurs'),
    url(r'^cv/(?P<idCV>[0-9]+)/$', views.cv),
    url(r'^recruter/(?P<idCV>[0-9]+)/$', views.recruter),
]
```

Dans Django, pour manipuler des formulaires dans une application, il faut ajouter un fichier « forms.py » dans l'application, au même niveau que les fichiers « models.py », « views.py », etc., pour contenir les formulaires de l'application sous formes de classes Python. En outre, un formulaire peut être rattaché à un modèle (une classe du fichier « models.py ») pour que les attributs de la classe correspondent aux champs du formulaire. De cette façon, il est possible d'instancier un objet de la classe rattachée au formulaire à partir des données des champs du formulaire. Inversement, il sera aussi possible de pré remplir les champs du formulaire avec les valeurs des attributs d'un objet de la classe rattaché au formulaire.

Le formulaire de recrutement que nous allons ajouter est appelée « DemandeRecrutementForm » et rattaché au modèle « DemandeRecrutement ». Une implémentation est donnée dans le fichier « forms.py » (à retrouver dans le dossier « codes sources/apptest/ »).

La vue « recruter » qui traite l'url du formulaire de recrutement est ajoutée dans le fichier « views.py » de « apptest ». Puisqu'elle doit traiter le formulaire « DemandeRecrutementForm », un objet de ce formulaire y est instancié et passé à la page html « recruter.html » pour y être inclus. Quand le formulaire sera rempli depuis la page html et soumis, la requête de soumission est encore traitée par la vue « recruter ». Si le formulaire est valide, alors un objet de la classe rattachée au formulaire (il s'agit ici du modèle « DemandeRecrutement ») est généré en correspondance avec les données du formulaire et enregistré dans la base de données. Sinon, on est redirigé vers le formulaire avec les données qui y étaient remplies.

1.4 - Fonctionnalités de gestion des demandes recrutements

Il s'agit ici de permettre aux recruteurs de pouvoir ***visualiser*** toutes les demandes de recrutements qu'ils sont effectuées, de pouvoir les ***modifier*** et les ***supprimer*** au besoin.

1.4.1. Visualisation des demandes recrutements

L'url d'accès à cette fonctionnalité est appelée « mes_recrutements » :

```
from django.urls import path
from django.conf.urls import url
from . import views

urlpatterns = [
    path('', views.accueil),
    path('developpeurs', views.developpeurs, name='developpeurs'),
    url(r'^cv/(?P<idCV>[0-9]+)/$', views.cv),
    url(r'^recruter/(?P<idCV>[0-9]+)/$', views.recruter),
    path('mes_recrutements', views.mes_recrutements, name='mes_recrutements'),
]
```

La vue « mes_recrutements » est ajoutée dans « views.py » de « apptest ». Elle récupère la liste des demandes de recrutement de l'utilisateur connecté depuis la base de données et la renvoie à la page html « mes_recrutements.html » qui affiche la liste dans une table. Dans la dernière colonne de cette table, se figurent 2 boutons sur chaque ligne : une pour modifier la demande et une autre pour la supprimer.

1.4.2. Modification d'une demande de recrutement

L'url de cette fonctionnalité est :

```
from django.urls import path
from django.conf.urls import url
from . import views

urlpatterns = [
    path('', views.accueil),
    path('developpeurs', views.developpeurs, name='developpeurs'),
    url(r'^cv/(?P<idCV>[0-9]+)/$', views.cv),
    url(r'^recruter/(?P<idCV>[0-9]+)/$', views.recruter),
    path('mes_recrutements', views.mes_recrutements, name='mes_recrutements'),
    url(r'^mes_recrutements/modifier/(?P<idRecrutement>[0-9]+)/$', views.modifier_recrutement),
]
```

La vue « modifier_recrutement » est ajoutée à « views.py » de « apptest ». Vous pouvez remarquer qu'on a utilisé le même formulaire « DemandeRecrutementForm » pour le rattacher à l'objet (DemandeRecrutement) qu'on cherche à modifier :

```
form = DemandeRecrutementForm(request.POST or None, instance=theRecrutement)
```

La page html qui contient le formulaire de modification est « modifier_recrutement.html ».

1.4.3. Suppression d'une demande de recrutement

L'url d'accès à cette fonctionnalité est :

```
from django.urls import path
from django.conf.urls import url
from . import views

urlpatterns = [
    path('', views.accueil),
    path('developpeurs', views.developpeurs, name='developpeurs'),
    url(r'^cv/(?P<idCV>[0-9]+)/$', views.cv),
    url(r'^recruter/(?P<idCV>[0-9]+)/$', views.recruter),
    path('mes_recrutements', views.mes_recrutements, name='mes_recrutements'),
    url(r'^mes_recrutements/modifier/(?P<idRecrutement>[0-9]+)/$', views.modifier_recrutement),
    url(r'^mes_recrutements/supprimer/(?P<idRecrutement>[0-9]+)/$', views.supprimer_recrutement),
]
```

La vue « supprimer_recrutement » est ajoutée à « views.py » de « apptest ». Elle récupère la demande de recrutement correspondant à l'id passé en paramètre à la vue et la supprime de la base de données.

2- ADMINISTRATION DES « DEMANDES DE RECRUTEMENTS » DEPUIS LE BACK-OFFICE

La dernière chose à faire consiste à permettre aux administrateurs du système d'administrer (depuis django admin) les demandes de recrutements, effectuées par les recruteurs depuis l'application « apptest ». Pour cela, il suffit d'ajouter la classe « DeamndeRecrutement » dans « models.py » de « apptest » dans le fichier « admin.py » de « apptest » que vous pouvez retrouver dans le dossier (codes sources/apptest/).