université
PARIS-SACLAY

Programmation en Python
Master 2 Réseaux Télécoms
TP 10

Dans ce TP on commence la construction d'un site Web dynamique basé sur le framework Django. Ce site sera consacré à la gestion mutualisée des outils d'une association de bricoleurs.

1 Installation du framework Django dans un environnement virtuel

Créez un environnement virtuel dans le repértoire `C:\Temp\envcoop` (voir le TP 9 pour les détails). Installez dans cet environnement le package `Django` avec la commande suivante (à lancer dans un terminal de l'environnement virtuel) :

```
pip install Django
```

La version installée doit être supérieure à 3.2, pour le vérifier exécutez la commande suivante :

```
python -m django --version
```

2 Création d'un projet

Créez un répertoire qui abritera le projet de ce TP, ouvrez un terminal de l'environnement virtuel et déplacez-vous dans ce répertoire. Créez un nouveau projet Django avec la commande suivante :

```
django-admin startproject site_m2rt
```

Cette commande doit créer un sous-répertoire `site_m2rt` avec la structure d'un projet Django :

```
site_m2rt
├── manage.py
└── site_m2rt
    ├── __init__.py
    ├── settings.py
    ├── urls.py
    ├── asgi.py
    └── wsgi.py
```

Lancez le serveur de développement avec la commande suivante :

```
python manage.py runserver
```

Dans le terminal vous allez voir les messages d'avertissement concernant les migrations non-appliquées ; à cette étape on va les ignorer. Avec un navigateur Web, naviguez vers l'adresse `http://localhost:8000`. On doit voir maintenant la page d'accueil du serveur de développement.

3 Configuration de la base de données

Un serveur Django (même fraîchement installé) nécessite pour son fonctionnement une base de données, pour y stocker le modèle d'authentification et le mot de passe d'administrateur. Dans un site réel, cette base est abritée sur un SGBD relationnel indépendant, tel MySQL, PostgreSQL ou Oracle. Mais pour le développement on utilise souvent une base de données embarquée SQLite3, c'est l'option par défaut de Django. Pour instancier cette base de données, lancez la commande suivante :

```
python manage.py migrate
```

La base de données doit apparaitre sous la forme d'un fichier `db.sqlite3` dans la racine du projet.

4 Interface d'administration

Pour créer le compte d'administrateur lancez la commande suivante :

```
python manage.py createsuperuser
```

Saisissez le nom du compte et le mot de passe (on utilisera le nom `admin` et le mot de passe `BAT220ist`), en laissant l'adresse mél vide. Naviguez vers `http://localhost:8000/admin/` et connectez-vous en tant qu'administrateur du site. Créez un groupe d'utilisateurs `Bricoleurs` et quelques utilisateurs fictifs (ils seront tous membres du groupe `Bricoleurs`).

5 Création de l'application Web dynamique

Un project Django peut contenir plusieurs applications ; dans notre cas on se contentera d'une seule application qu'on nommera `coop`. Pour créer cette application, exécutez la commande suivante :

```
python manage.py startapp coop
```

Ça va créer un répertoire `coop` au même niveau que le ficher `manage.py`, avec la structure suivante :

```
coop
├── __init__.py
├── admin.py
├── apps.py
├── migrations
│   └── __init__.py
├── models.py
├── tests.py
└── views.py
```

Ajoutez l'application dans le projet, en modifiant la liste des applications installées dans le fichier `site_m2rt/settings.py` :

```
INSTALLED_APPS = [
    'coop.apps.CoopConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

(Notez la ligne `'coop.apps.CoopConfig'`).

6 Création du modèle des outils

Un modèle Django est une classe Python qui correspond à une table dans une base de données relationnelle. Django gère la correspondance objet-relationnelle par le mécanisme des *migrations*, en créant à la demande la table correspondant à la classe-modèle. Ajoutez le code suivant dans le fichier `coop/models.py` :

```
from django.db import models
from django.contrib.auth.models import User
```

```
class Tool(models.Model):
    owner = models.ForeignKey(User, on_delete=models.CASCADE)
    description = models.CharField(max_length=256)
    price = models.DecimalField(decimal_places=2, max_digits=6)

    def __str__(self):
        return self.description
```

Notez que l'attribut `owner` est une clé étrangère, correspondant à la clé primaire du modèle `User` utilisé par le système d'authentification de Django. Remarquez également la présence de la méthode `__str__`, qui est nécessaire pour l'affichage correct des objets. On peut maintenant préparer la migration avec la commande

```
python manage.py makemigrations coop
```

Vous devez voir le message suivant :

```
Migrations for 'coop':
coop/migrations/0001_initial.py
- Create model Tool
```

On peut visualiser les commandes SQL préparées avec la commande

```
python manage.py sqlmigrate coop 0001
```

Finalement, on lance la migration :

```
python manage.py migrate
```

7 Accès aux outils via le site d'administration

Pour permettre à l'administrateur du site d'ajouter ou d'enlever les outils, ajoutez les lignes suivantes dans le fichier `coop/admin.py` :

```
from django.contrib import admin
from coop.models import Tool

class ToolAdmin(admin.ModelAdmin):
pass

admin.site.register(Tool, ToolAdmin)
```

Notez que le serveur doit se relancer automatiquement dès l'enregistrement des modification dans ce fichier. Rafraichissez la page d'administration, vous devez maintenant avoir l'accès à la table d'outils. Ajoutez en quelques-uns (voir Figure 1).

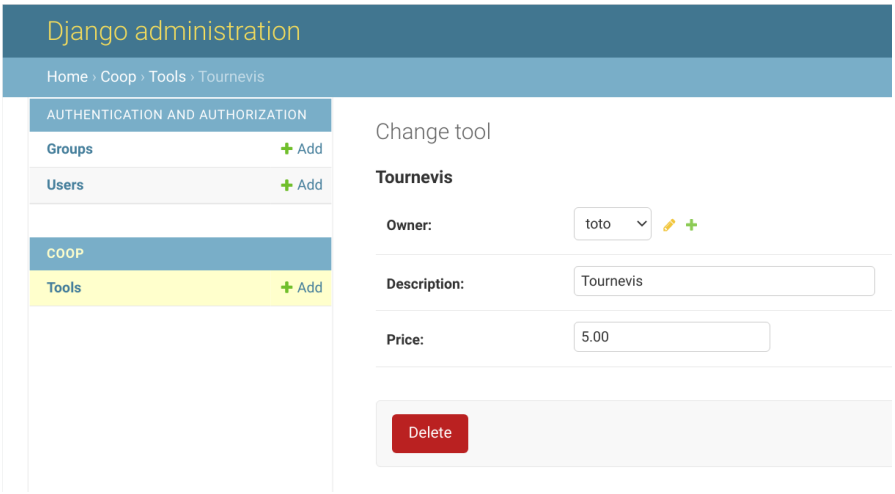

FIGURE 1 – Ajout d'un outil via l'interface d'administration.

8 L'interface Web de l'application

Modifiez le fichier `coop/views.py` en y ajoutant les lignes suivantes :

```
from django.shortcuts import render

def home(request):
    return render(request,"coop/home.html")
```

Ce code définit un *view* `home`, qui correspond à la page principale de notre application. Dans le répertoire `coop` créez le fichier `urls.py` avec le contenu suivant :

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

urlpatterns = [
    path('', views.home, name = "home"),
]
```

La liste `urlpatterns` établit le lien entre les URLs et les *views* (l'interface graphique de l'application WEB). Tous les URLs de notre application seront précédés par le préfixe `coop` ; pour cela on ajoute la ligne correspondante dans la liste `urlpatterns` du fichier `site_m2rt/urls.py` :

```
from django.contrib import admin
from django.urls import include, path
```

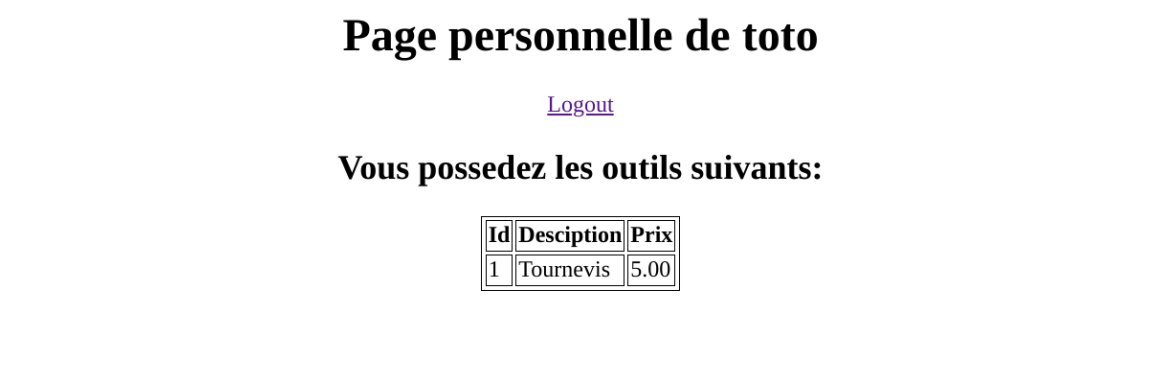

FIGURE 2 – Page principale de l'application.

```
urlpatterns = [
    path('coop/', include('coop.urls')),
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
]
```

Pour simplifier l'expérience de l'utilisateur, on va le diriger directement sur la page principale de notre application immédiatement après le login et vers la page de login immédiatement après la déconnexion. Pour cela ajoutez les lignes suivantes dans le fichier `settings.py` :

```
LOGIN_REDIRECT_URL = "home"

LOGOUT_REDIRECT_URL = "login"
```

Finalement, créez le répertoire `coop/templates` et ajoutez dans ce répertoire et les sous-répertoires `coop` et `registration` les fichiers de gabarits (*templates*) HTML fournis, pour créer l'arborescence suivante :

```
templates
├── coop
│   └── home.html
├── registration
│   └── login.html
└── base.html
```

Naviguez vers `http://localhost:8000/coop/` et vérifiez que vous pouvez vous connecter en tant qu'un utilisateur ordinaire. Après la connexion, on doit observer la page avec la liste des outils appartenant à l'utilisateur connecté (voir Figure 2).