*PETIT PROJET DE REALISATION DE CRUD SUR UNE TABLE EMPLOYEE*

# Titre : Réaliser un CRUD (Create, Retrieve, Update, Delete) avec le framework Django

Dans cet article, nous allons apprendre à créer un CRUD avec Django. En termes simples, un CRUD inclut les opérations de base suivantes :

**Create** : signifie "créer" et permet d'ajouter un nouvel élément dans la base de données.
**Read** : signifie "lire" et permet d'afficher un élément enregistré dans la base de données.
**Update** : signifie "modifier" et permet de mettre à jour un champ existant.
**Delete** : signifie "supprimer" et permet de supprimer un champ existant.
Passons en revue les étapes pour réaliser ces opérations.

**Étapes de Création du Projet**

**1. Créer un environnement virtuel**
Dans notre terminal, entrons la commande :
```
python3 -m venv mon_environnement
```
Puis, activons l'environnement virtuel avec :
```
source mon_environnement/bin/activate
```

**2. Créer un projet Django**
Nous allons créer un projet Django en utilisant la commande :
```
django-admin startproject Employee
```
Cela crée un projet nommé Employee.

3. **Créer une application**
Accédons au dossier du projet créé précédemment :
```
cd Employee
```
Puis, créons une application avec :
```
python manage.py startapp CRUDAPP
```

**4. Créer le dossier de templates**
Dans le dossier de notre projet, créons un dossier nommé templates. À l'intérieur, créons les fichiers suivants :

**show.html**
**insert.html**
**edit.html**
**delete.html**

**5. Configuration de l'application**

Dans le fichier settings.py, nous allons inclure l'application CRUDAPP :

---

***Rédigé par :*** *PRISCILLE MANDE CHABUELLE*

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'CRUDAPP',  # Ajout de l'application
]
```

Ensuite, au niveau de la configuration des templates, ajoutons le chemin vers le dossier templates :

```
import os  # N'oubliez pas d'importer 'os'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],  # Ajout du dossier templates
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Nous utiliserons SQLite comme base de données, qui est déjà configurée par défaut dans Django :

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': BASE_DIR / 'db.sqlite3',
    }
}
```

Configuration des URLs
Dans le fichier urls.py du projet, ajoutons le fichier urls.py de notre application :

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

urlpatterns = [
```

---

***Rédigé par :*** *PRISCILLE MANDE CHABUELLE*

```
    path('admin/', admin.site.urls),
    path('', include('CRUDAPP.urls')),
]
```

Dans urls.py de CRUDAPP, ajoutons les différentes URL pour notre CRUD :

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

urlpatterns = [
    path('insert/', views.Insert_emp, name='insert'),
    path('', views.Show_emp, name='show'),
    path('edit/<int:pk>', views.Edit_emp, name='edit'),
    path('remove/<int:pk>', views.Delete_emp, name='remove')
]
```

----Modèle

Dans le fichier models.py, créons la classe qui représentera les champs de notre base de données :

```
class Employe(models.Model):
    nom = models.CharField(max_length=100)
    prenom = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    telephone = models.CharField(max_length=15, blank=True)
    date_naissance = models.DateField()
    date_embauche = models.DateField()
    poste = models.CharField(max_length=100)
    salaire = models.DecimalField(max_digits=10, decimal_places=2)
    adresse = models.TextField(blank=True)
```

----Migrations

Lions cette classe à notre base de données :

```
python manage.py makemigrations
python manage.py migrate
```

----Les Views

Dans views.py, créons les différentes vues pour gérer notre CRUD :

```
from django.shortcuts import render, redirect, get_object_or_404
from CRUDAPP.models import Employe

# Create (Insertion)
def Insert_emp(request):
    if request.method == 'POST':
        data = Employe(
            nom=request.POST['nom'],
```

---

***Rédigé par :*** *PRISCILLE MANDE CHABUELLE*

```
            prenom=request.POST['prenom'],
            email=request.POST['email'],
            telephone=request.POST['telephone'],
            date_naissance=request.POST['date_naissance'],
            date_embauche=request.POST['date_embauche'],
            poste=request.POST['poste'],
            salaire=request.POST['salaire'],
            adresse=request.POST['adresse']
        )
        data.save()
        return redirect('show')
    else:
        return render(request, 'insert.html')

# Read (Affichage)
def Show_emp(request):
    show = Employe.objects.all()
    context = {'show': show}
    return render(request, 'show.html', context)

# Update (Modification)
def Edit_emp(request, pk):
    employe = get_object_or_404(Employe, id=pk)
    if request.method == 'POST':
        employe.nom = request.POST['nom']
        employe.prenom = request.POST['prenom']
        employe.email = request.POST['email']
        employe.telephone = request.POST['telephone']
        employe.date_naissance = request.POST['date_naissance']
        employe.date_embauche = request.POST['date_embauche']
        employe.poste = request.POST['poste']
        employe.salaire = request.POST['salaire']
        employe.adresse = request.POST['adresse']
        employe.save()
        return redirect('show')
    context = {'employe': employe}
    return render(request, 'edit.html', context)

# Delete (Suppression)
def Delete_emp(request, pk):
    employe_delete = get_object_or_404(Employe, id=pk)
    if request.method == 'POST':
        employe_delete.delete()
        return redirect('show')
    context = {'employe_delete': employe_delete}
```

---

***Rédigé par :*** *PRISCILLE MANDE CHABUELLE*

```
    return render(request, 'delete.html', context)
```

Remarque : get_object_or_404 vérifie si l'ID existe, sinon, il retourne une erreur 404.

-----Templates
Téléchargez tous les fichiers nécessaires aux templates ici (https://github.com/MandePriscille/crud-app/tree/master).

-----Lancer le projet
Assurez-vous que l'environnement virtuel est activé, puis lancez le serveur avec :

```
python manage.py runserver
```

Conclusion
En résumé, ces étapes montrent comment réaliser un CRUD avec Django. Pour accéder au projet complet, cliquez ici https://github.com/MandePriscille/crud-app/tree/master.

---

***Rédigé par :*** *PRISCILLE MANDE CHABUELLE*