Création d'un formulaire Django avec une base de données MySQL

# Introduction

Ce document décrit le processus de création d'un formulaire Django qui enregistre des données dans une base de données MySQL. Nous utilisons XAMPP pour gérer MySQL et Django comme framework web.

# Étape 1: Installer MySQL et configurer Django

Avant de commencer, assurez-vous que MySQL est installé sur votre machine, idéalement via XAMPP. Pour connecter Django à MySQL, installez d'abord le package `mysqlclient` en exécutant la commande suivante :  
```  
pip install mysqlclient  
```  
Ensuite, configurez votre base de données dans le fichier `settings.py` de votre projet Django en ajoutant la configuration suivante :

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'nom\_de\_votre\_base\_de\_donnees',  
 'USER': 'root', # Par défaut, MySQL sur XAMPP utilise 'root' comme utilisateur.  
 'PASSWORD': '', # Laissez vide si aucun mot de passe n'est défini.  
 'HOST': 'localhost',  
 'PORT': '3306',  
 }  
 }

# Étape 2: Créer une application Django et un modèle

Créez une nouvelle application dans votre projet Django en exécutant :  
```  
python manage.py startapp myapp  
```  
Dans `models.py`, définissez un modèle qui correspond aux données que vous souhaitez stocker. Par exemple :

# myapp/models.py  
 from django.db import models  
  
 class Person(models.Model):  
 name = models.CharField(max\_length=100)  
 age = models.IntegerField()  
 email = models.EmailField()

# Étape 3: Créer et appliquer les migrations

Après avoir défini le modèle, créez et appliquez les migrations pour synchroniser le modèle avec la base de données MySQL :  
```  
python manage.py makemigrations  
python manage.py migrate  
```

# Étape 4: Créer un formulaire Django

Dans `forms.py`, créez un formulaire basé sur le modèle que vous venez de créer :

# myapp/forms.py  
 from django import forms  
 from .models import Person  
  
 class PersonForm(forms.ModelForm):  
 class Meta:  
 model = Person  
 fields = ['name', 'age', 'email']

# Étape 5: Créer une vue pour gérer le formulaire

Dans `views.py`, ajoutez une vue qui gérera l'affichage et le traitement du formulaire :

# myapp/views.py  
 from django.shortcuts import render, redirect  
 from .forms import PersonForm  
  
 def person\_form(request):  
 if request.method == 'POST':  
 form = PersonForm(request.POST)  
 if form.is\_valid():  
 form.save()  
 return redirect('person\_success')  
 else:  
 form = PersonForm()  
 return render(request, 'person\_form.html', {'form': form})

# Étape 6: Créer le template pour le formulaire

Dans le dossier `templates`, créez un fichier `person\_form.html` pour afficher le formulaire :

<!-- myapp/templates/person\_form.html -->  
 <h1>Person Form</h1>  
 <form method="POST">  
 {% csrf\_token %}  
 {{ form.as\_p }}  
 <button type="submit">Submit</button>  
 </form>

# Étape 7: Ajouter une URL pour le formulaire

Dans `urls.py`, ajoutez une URL pour la vue du formulaire :

# myapp/urls.py  
 from django.urls import path  
 from . import views  
  
 urlpatterns = [  
 path('person/new/', views.person\_form, name='person\_form'),  
 path('person/success/', views.person\_success, name='person\_success'), # Pour redirection après succès  
 ]

# Étape 8: Créer une page d'accueil

Enfin, vous pouvez créer une page d'accueil qui comprend un bouton redirigeant vers votre formulaire :

<!-- myapp/templates/home.html -->  
 <h1>Welcome to My Website</h1>  
 <a href="{% url 'person\_form' %}">  
 <button type="button">Go to Form</button>  
 </a>

# Conclusion

En suivant ces étapes, vous aurez créé une application Django avec un formulaire qui communique avec une base de données MySQL. Les utilisateurs peuvent ainsi soumettre des informations via le formulaire, et ces données seront enregistrées dans la base de données.