Création d'un Formulaire Django avec une Base de Données MySQL

GANAME Abdoulaye Idrissa
élève Analyste Statisticien 3 année

October 7, 2024

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. Alors il faut s'assurer d'avoir au préalable installé XAMPP et avoir son Apache et son MySQL fonctionnel.

Part I

DJANGO, BASE DE DONNEES, PROJET

A Installation de Django et Initialisation du projet

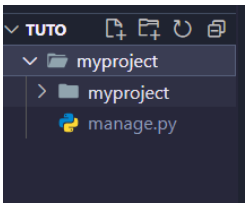
Pour commencer nous allons installer Django directement dans python, pour eviter de créer des environnements à chaque fois.Pour cele nous utilisons la commande suivante dans notre invite de commande :

```
python -m pip install django
```

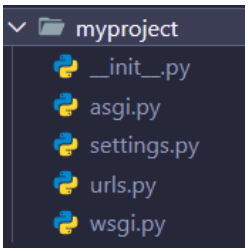
Nous allons créer un dossier dans lequel nous allons mettre tous ce qui sera en rapport avec notre django (l'emplacement importe peu pourvu que vous puissiez le retrouver) ici j'ai décidé de l'appeler "Tuto".
Nous ouvrons ensuite ce dossier sur VScode. Dans votre VScode accedez au terminal (Vous pouvez utiliser la comande CTRL+ù).
Nous allons ici créer notre projet django, qui va comporter tous fichier relatif à notre site, notre base de donnée etc. Pour cela dans le terminal VScode mettez :

```
django-admin startproject myproject
```

Le projet se nommera "myproject" selon ce code.
Après cela vous devrez constater la creation d'un dossier nommé "myproject", contenant un autre dossier du même nom et un fichier "manage.py".



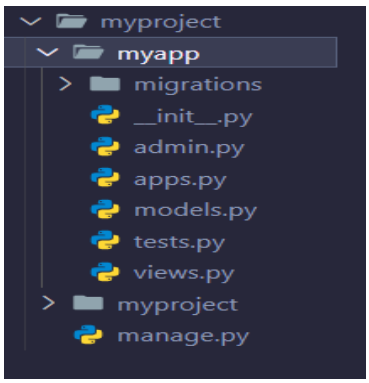
Et dans le dossier vous trouverez plusieurs fichiers ".py"



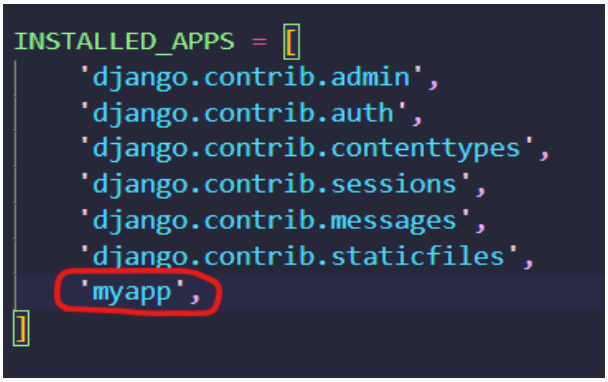
Créez une nouvelle application dans votre projet Django avec :

```
cd myproject
python manage.py startapp myapp
```

Cela créait un dossier "myapp" contenu lui aussi plusiers autres fichiers ".py" par defaut.

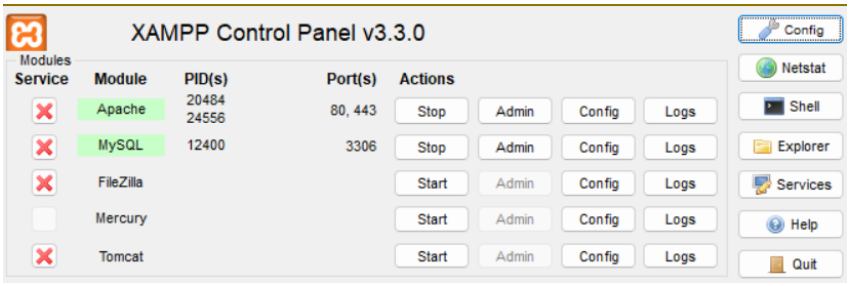


NB : Pour que notre application soit installé dans notre site nous devons aller dans le fichier `setting.py` de notre projet, et dans la section INSTALLED_APPS=[] ajouter notre application :



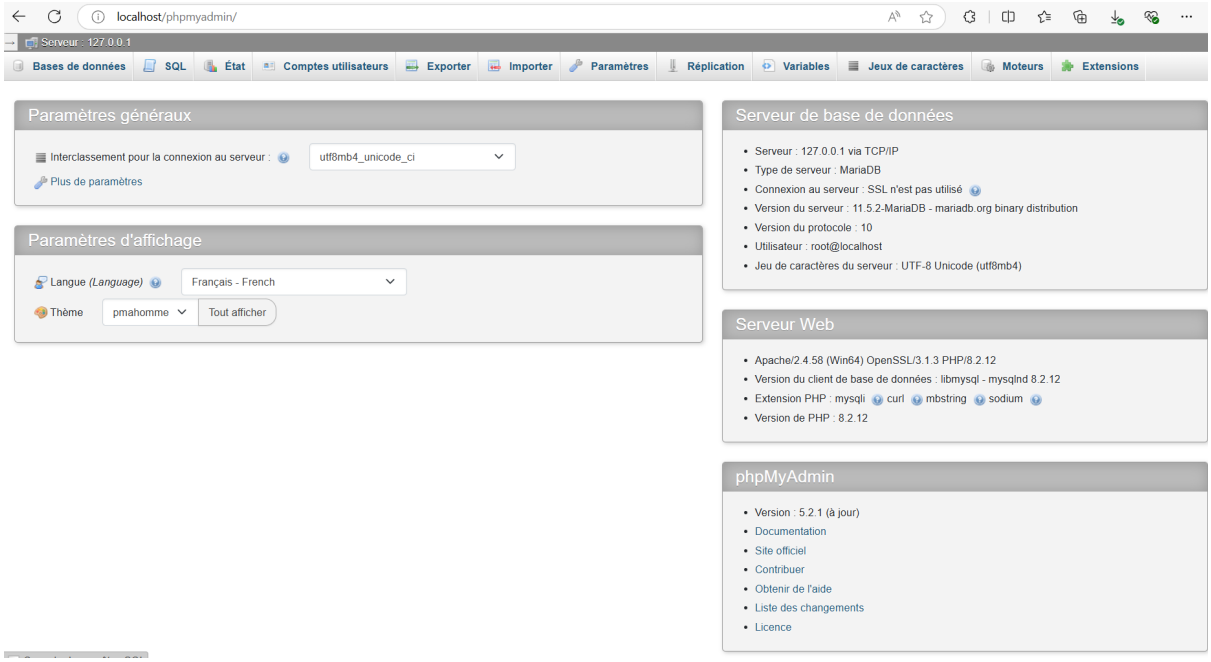
B Creation d'une Base de donnée et d'un utilisateur

Ici nous allons créer une base de donnée MySQL ou sera deversé les informations de notre formulaire, avec un utilisateur dedié.
Pour cela ouvrez XAMPP, activez "Apache" et "MySQL". Vous avez donc :



Après cela cliquez sur le button "Admin" sur la ligne de MySQL, vous serez dirigé sur le site :

```
http://localhost/phpmyadmin/
```



Dans l'onglet ***"compte utilisateur"*** créez un utilisateur.
Et dans l'onglet ***Base de données*** créez une Base de donnée.
Dans ce tuto notre base va s'appeler "Tuto_BD" et notre utilisateur "GanItachi" son code "password"

C Liaison MySQL et notre projet Django

Pour permettre à Django de se connecter à Mysql vous pouvez lancer la commande :

```
pip install mysqlclient
```

Nous allons ensuite configurer notre base de données dans le fichier *settings.py* de notre projet afin de les lier.
De base vous verrez :

Remplacez par ce code, en prenant le soin de mettre les bonnes informations le nom de la base dans 'Name' celui de l'utilisateur dans 'User' et le mot de passe dans 'Password' :

```
DATABASES = {
 'default': {
  'ENGINE': 'django.db.backends.mysql',
  'NAME': 'Tuto_BD',
  'USER': 'GanItachi',
  'PASSWORD': 'password',
  'HOST': 'localhost',
  'PORT': '3306', # Mettez sur le port où se trouve votre SQL
  }
}
```

D Creation d'un modèle de données et deployement

Ensuite, dans le fichier `models.py` de votre application, définissez un modèle `Person` :

```
from django.db import models
class Person(models.Model):
    name = models.CharField(max_length=100)
    age = models.IntegerField()
    email = models.EmailField()
```

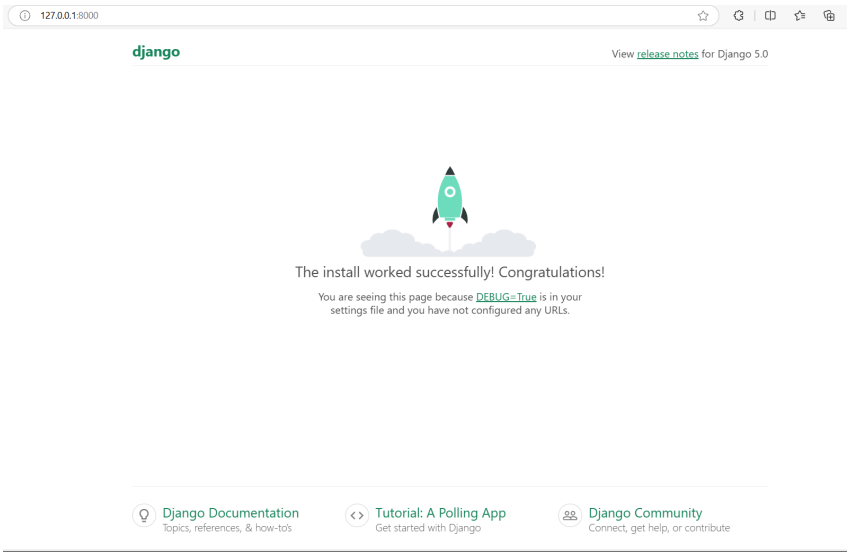
Cela signifie que l'on creait une classe de personne qui est une sous classe de modèle, et une personne est enregistré dans la base avec un nom maximum 100 caractère, un âge, et un mail. Après avoir défini le modèle, effectuez une migration pour synchroniser votre modèle avec la base de données :

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

NB :Assurez vous d'être dans le dossier contenant "manage.py" pour ne pas produire d'erreur.
Tout etant conforme on peut déjà lancer le site avec :

```
python manage.py runserver
```

Et en cliquant sur le lien produit ***http://127.0.0.1:8000***.



Vous aurez remarqué que rien n'y est présent, c'est normal nous n'avons pas encore crée de formulaire, nous n'avons fait que configurer notre base de données.
Nous allons donc passer à la seconde partie consistant en la création d'un formulaire, qui enregistre directement dans notre base, et d'une page d'accueil nous dirigeant vers le formulaire.

Part II

FORMULAIRE

A Créons un formulaire Django

Nous allons créer un fichier nommé `forms.py` dans notre application, et dans ce fichier nous créons un formulaire basé sur le modèle que vous venez de créer :

```
from django import forms
from .models import Person

class PersonForm(forms.ModelForm):
    class Meta:
        model = Person
        fields = ['name', 'age', 'email']
```

Explication

Importations :

• *forms:* Django fournit le module forms qui contient des classes et des outils pour la création et la gestion de formulaires web.

• *Person:* Le modèle défini dans models.py. Ce modèle contient les champs name, age, et email, et représente une personne dans la base de données.

Classe PersonForm :

• Cette classe hérite de ***forms.ModelForm***. En héritant de cette classe, Django génère automatiquement un formulaire basé sur le modèle Person.

Classe interne Meta :

• La classe Meta est utilisée pour spécifier des informations supplémentaires concernant le formulaire.

• model = Person : Cela indique que ce formulaire est lié au modèle Person. Django va automatiquement créer les champs du formulaire en fonction de ceux du modèle Person.

• fields = ['name', 'age', 'email'] : Ici, on spécifie explicitement les champs qui seront inclus dans le formulaire. Dans ce cas, seuls les champs name, age, et email du modèle Person seront inclus. Si on voulait inclure tous les champs, on pouvait utiliser l'option `__all__` à la place.

B Créons une vue pour gérer le formulaire

Toujours dans notre application accédons au fichiers `views.py`, ajoutons une vue pour gérer l'affichage et le traitement du formulaire :

```
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})
```

Explication

Importations :

- `render` : Fonction qui rend une page HTML avec le contexte donné.
- `redirect` : Fonction qui redirige l'utilisateur vers une autre URL.
- `PersonForm` : Le formulaire lié au modèle `Person` (généralement défini dans `forms.py`), qui permet de recueillir des données utilisateurs.

Vue `person_form` :

La vue gère deux types de requêtes HTTP :

- **Si la méthode est `POST` :**
  - Cela signifie que l'utilisateur soumet le formulaire.
  - `form = PersonForm(request.POST)` : Un objet `PersonForm` est instancié avec les données envoyées via le formulaire.
  - `if form.is_valid()` : Le formulaire est vérifié pour s'assurer que les données fournies sont valides (par exemple, respecter les types et contraintes définis dans le modèle).
  - `form.save()` : Si les données sont valides, elles sont enregistrées dans la base de données.
  - `return redirect('person_success')` : Après l'enregistrement réussi, l'utilisateur est redirigé vers une autre vue ou page, identifiée par le nom d'URL `'person_success'`.
- **Si la méthode est autre que `POST`** (par exemple, `GET`) :

– Cela signifie que l'utilisateur accède à la page du formulaire sans encore soumettre de données.

– `form = PersonForm()` : Un formulaire vierge est créé.

– `return render(request, 'person_form.html', {'form': form})` : La page HTML `person_form.html` est rendue avec le formulaire vierge inclus dans le contexte ({`'form': form`}). Cela permet de l'afficher sur la page.

C Créer le template pour le formulaire

Créez un dossier nommé "templates" dans votre dossier d'application, dans ce répertoire `templates`, créez un fichier `person _ form.html` pour afficher le formulaire :

```
<h1>Person Form</h1>
<form method="POST">
    {% csrf_token %}
    {{ form.as_p }}
    <button type="submit">Submit</button>
</form>
```

EXPLICATION :

1. `<h1>Person Form</h1>` :

   - Ceci est un titre de niveau 1 (en-tête) qui apparaît en haut de la page pour indiquer à l'utilisateur qu'il s'agit d'un formulaire à remplir. Le texte affiché sera `"Person Form"`.

2. `<form method="POST">` :

   - Il s'agit de la balise HTML ouvrant un formulaire.
   - L'attribut `method="POST"` indique que lorsque l'utilisateur soumettra ce formulaire, les données seront envoyées au serveur via la méthode HTTP POST. Cela signifie que les données ne seront pas visibles dans l'URL, ce qui est souvent utilisé pour les formulaires qui modifient les données (comme l'ajout ou la mise à jour dans une base de données).

   - Ceci est une balise de template Django. Elle ajoute un jeton CSRF (Cross-Site Request Forgery) dans le formulaire pour protéger contre les attaques CSRF.
   - Django exige que chaque formulaire POST contienne ce jeton pour valider que la requête provient d'une session utilisateur authentique et non d'une source externe malveillante.

3. **{{ form.as_p }}** :

   - Cette balise de template Django permet de rendre le formulaire dans le fichier HTML.

- Ici, `{{ form.as_p }}` va afficher chaque champ du formulaire sous forme de paragraphe HTML (`<p>`). Cela rend le formulaire plus facile à lire et bien structuré.
- Cette balise est utilisée pour afficher automatiquement les champs définis dans le formulaire `PersonForm` que vous avez créé. Ainsi, les champs pour `name`, `age`, et `email` (du modèle Person) apparaîtront sous forme de champs de saisie dans le navigateur.

4. `<button type="submit">Submit</button>` :
   - Ce bouton permet de soumettre le formulaire.
   - Lorsqu'un utilisateur clique sur le bouton `"Submit"`, les données saisies dans les champs de formulaire sont envoyées au serveur pour être traitées.

D Créer la page de succès

Souvenez vous plus haut nous avons défini une page person_success sur lequel on devait se diriger si la personne était effectivement enregistré. Nous devons créer sa page html. Alors dans notre dossier templates nous allons créer un fichier nommé **person_success.html**. On y affichera juste un texte de success(1) et le lien pour créer un autre formulaire (2) avec :

```
<h2>Person created successfully!</h2>
<a href="{%url 'person_form'%}">Create another person</a>
```

On va ensuite l'ajouter dans les views de notre application pour gerer son affichage :

```
  def person_success(request):
    return render(request, 'person_success.html')
```

E Ajouter les URLs pour le formulaire

Dans `urls.py` de notre projet,on doit ajouter les pages pour qu'il puisse s'afficher avec le code :

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

urlpatterns = [
    path('creation/', views.person_form, name='person_form'),
    path('succes/', views.person_success,
       name='person_success'),  # Pour redirection après
       succès
]
```

ce code signifie que nous allons appeler le views de notre application "myapp" et que si on ajoute "creation/" a notre lien de base (http://localhost:8000) sur notre navigateur c'est à dire "http://localhost:8000/Creation/" ça envois à views qui nous amène sur notre page html "person_form" donc notre formulaire.
La même chose pour success.

Alors ce niveau nous avons déjà fini la creation et on peut accéder à notre formulaire. On fait d'abord migrer tous nos codes pour que la base s'adapte et ensuite on lance dans notre terminal :

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

On utilise le lin http://localhost:8000/Creation/ dans notre navigateur.

Vous aurez remarqué que tout fonctionne. Mais que nous n'avons pas de page d'acceuil que la page formulaire et la page de succes. Nous allons donc faire ce dernier element.

Part III

PAGE D'ACCEUIL

A Créer une page d'accueil

On peut créer une page d'accueil avec un bouton qui redirige vers le formulaire en ajoutant une nouvelle page donc un nouveau fichier dans templates avec le nom `home.html` et on y inscris ce code :

```
<h1>Welcome to My Website</h1>
<a href="{% url 'person_form' %}">
    <button type="button">Go to Form</button>
</a>
```

On va ensuite lui créer une vue dans le views.py de notre application :

```
  def home(request):
  return render(request, 'home.html')
```

Maintenant on doit inscrire dans L'url de notre projet qu'il doit se fier à notre views qui renvoi l'aceuil si on met http://localhost:8000 sur le navigateur ou si on lance l'adresse :

```
  path('', views.home, name='home'),
```

B 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 soumettre des informations via le formulaire, et ces données seront enregistrées dans la base de données. Pour regarder que c'est bel et bien enregistré dans la base accedez y par l'interface ou par cmd et de chercher la table portant le nom de l'application et vous y trouverez les données enregistrées.