How to connect your Django website with Database (MySQL), (Django admin panel)

If you have created your website's homepage and contact page already, I recommend you to make a form on your contact page for the users to contact with you. And to do this, you have to connect your contact page with database.

Now what is database?

- A database is an organized collection of structured information, or data, typically stored electronically in a computer system.
- So there are so many database option, but here we will use MySQL database and Django's own database.

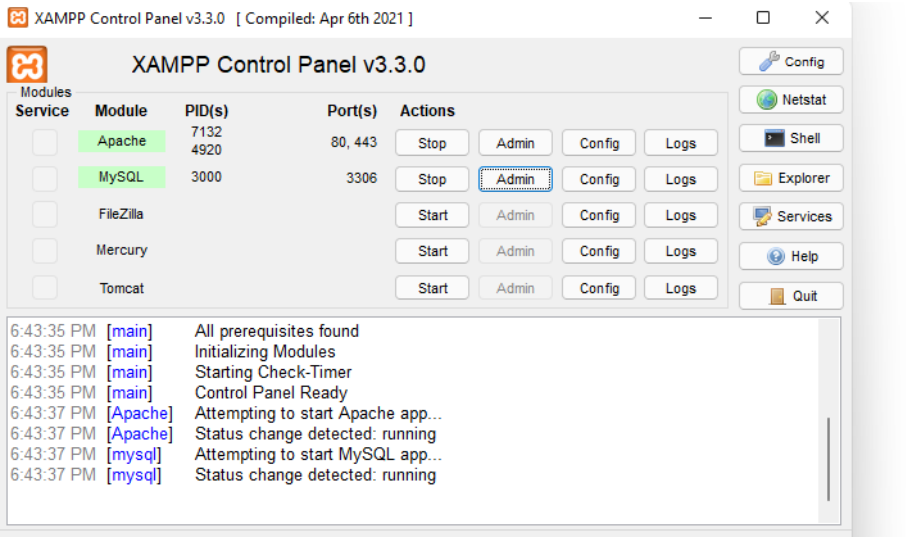
Basically, we will use database to store user's data, so that a user can fill a form by inputting their data and you can see and reply them.

So, today I will discuss about both of the databases and I will connect both of the databases with my website using **VSCODE, XAAMP, LOCALHOST.**

Now, you have to download the XAAAMP,

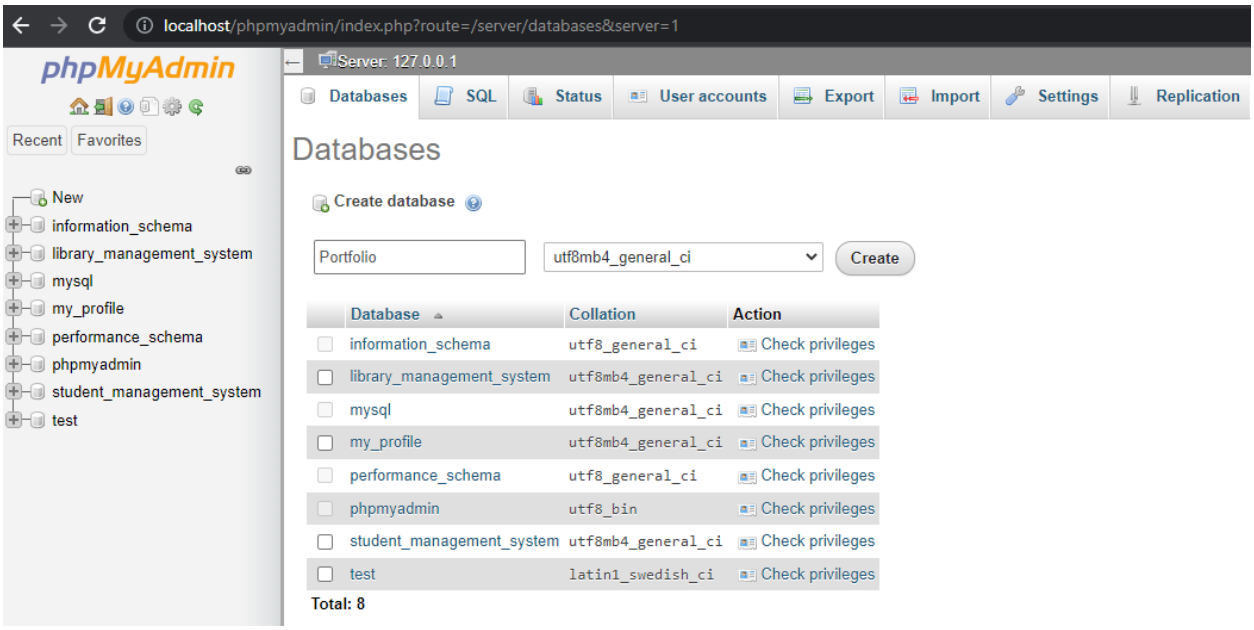
Download link: https://www.apachefriends.org/xampp-files/8.1.2/xampp-windows-x64-8.1.2-0-VS16-installer.exe

After download the XAAMP, press Apache and MySQL start button, like this:



And then, to have access the local host, particularly phpMyAdmin, go to: http://localhost/phpmyadmin/

Now, after getting into phpMyAdmin create a new database by click on new and set a name for your database like this:

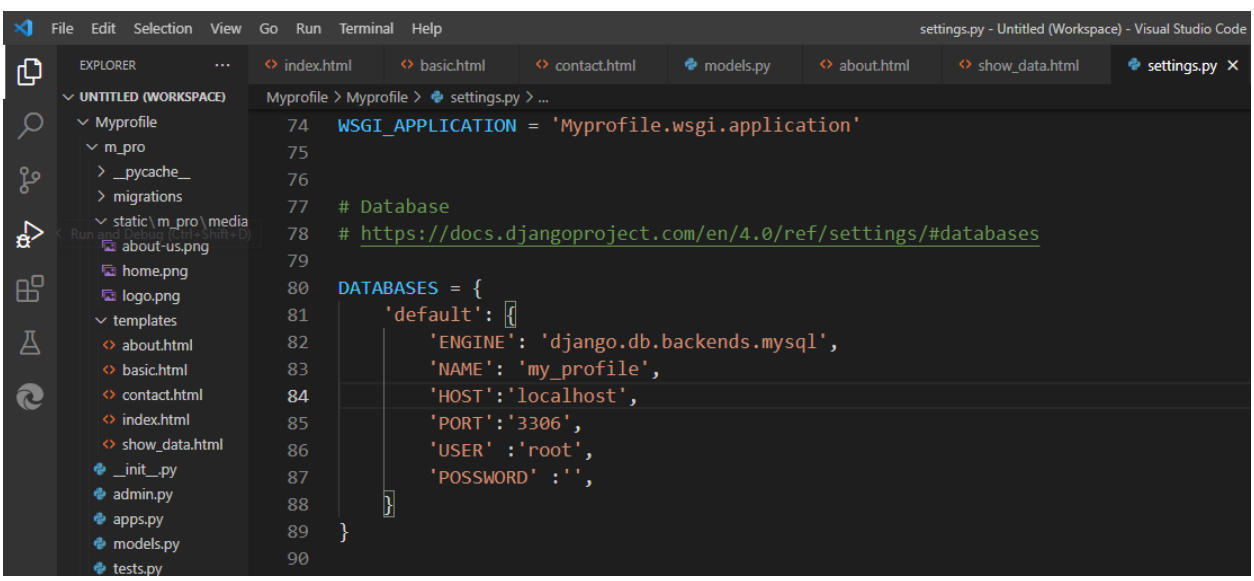


Here, Portfolio is the database name. Then, press click to have your own database.

Now, you have to follow some steps to connect your website with database and to do this go to VSCODE and follow these:

<u>1st step:</u> Go to settings.py under your project to connect your website with your database.

And then, write these codes and set your project name on the 'NAME' and to set the 'PORT', go to XAAMP and then you can see a number beside MySQL as Port(s).



[Note: If you want to work with database, before anything further go to the terminal on VSCODE and write a command pip install mysqlclient to install the MYSQLCLIENT]

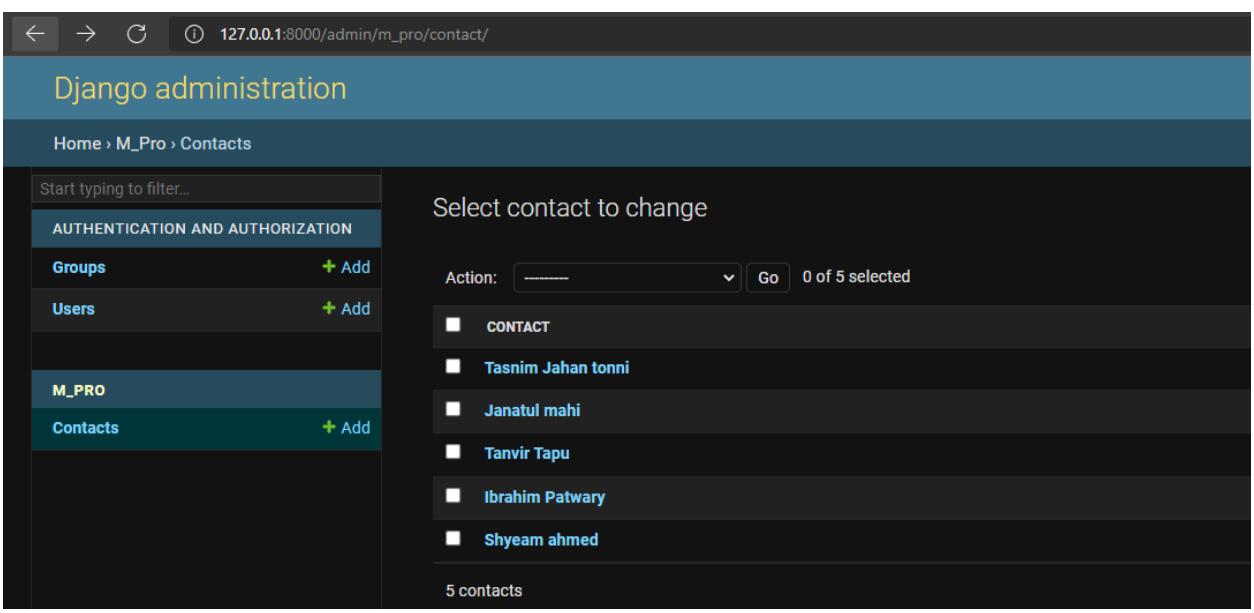
<u>2nd step:</u> Go to models.py under your app to make a table to connect your website with Django's own database or Django admin panel

[Basic: To start the connection, you have to define models. Model is nothing but a collection of classes. Each class will have the definition of a table.]

I. So, firstly open the file models.py under the django web app (In this illustration it is m_pro folder) folder.
II. Add this simple class below the comment # Create your models here. This class is a simple model for a table called Contact with three columns named name, email and message. You don't need to create a unique id field. During the migration process, a unique ID field called id will be added to the table. You will see it later.

```
class Contact(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    message = models.TextField(max_length=500)

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

[Note: Here the last code or the function def __str__(self): used for what you want to see initially on admin panel's like this:



Here, the admin panel shows the name of the users because I return the name on the function (return self.name)

```python
#from unicodedata import name
#from xml.etree.ElementTree import Comment
from django.db import models

# Create your models here.

class Contact(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(max_length=100)
    message = models.TextField(max_length=500)


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

Then, go to the terminal and type these two commands:

python manage.py makemigrations(Enter)

python manage.py migrate(Enter)

3rd step: Go to admin.py to register your model.

First type this code:

```
from django.contrib import admin
from .models import Contact
# Register your models here.
admin.site.register(Contact)
```

```python
from django.contrib import admin
from .models import Contact
# Register your models here.
admin.site.register(Contact)
```

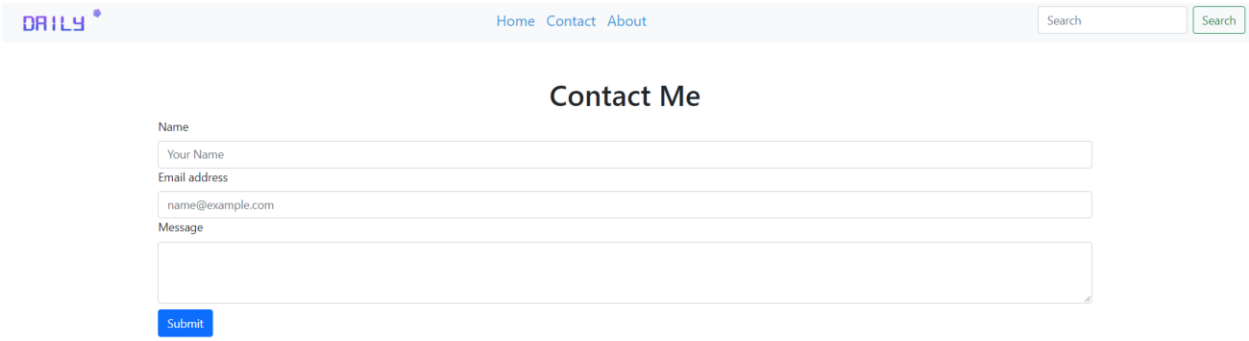
<u>4th step:</u> Go to your contact.html to create a form so that your users can easily contact with you.

The html code:

```
{% extends 'basic.html' %}

{% load static %}

{% block title %}

 Contact

{% endblock title %}

{% block body %}

<h1 class="text-center mt-5">Contact Me</h1>

   <div class="container">

     <form class="" method='post'>

       {% csrf_token %}

       <div class="row">

         <div class="col">

           <label for="exampleFormControlInput1" class="form-label">Name</label>

           <input name='name' class="form-control" placeholder="Your Name" aria-label="Your
Name">

           <label for="exampleFormControlInput1" class="form-label">Email address</label>

           <input name='email' class="form-control" id="exampleFormControlInput1"
placeholder="name@example.com">

           <label for="exampleFormControlTextarea1" class="form-label">Message</label>

           <textarea name='message' class="form-control" id="exampleFormControlTextarea1"
rows="3"></textarea>

           <div class="mt-2">

             <input class="btn btn-primary" type="submit" value="Submit">

           </div>

         </div>

       </div>

     </form>
```

```
  </div>


{% endblock  %}
```

Output (on the browser):



<u>5th step:</u> Go to views.py under your app so that the commands or the buttons can work.

First, import the contact:

```
from m_pro.models import Contact
```

Then, type a function:

```
def contact(request):
    if request.method == 'POST':
        contact = Contact()
        name = request.POST.get('name')
        email = request.POST.get('email')
        message = request.POST.get('message')
        contact.name=name
        contact.email=email
        contact.message=message
        contact.save()
```

return HttpResponse('<h1>Thanks for Contact with us</h1>')

return render(request, 'contact.html')