新加的大部分内容都有简单注释，可以直接搜关键字“巨”或“yida”

用到的AI部分如下

如果不喜欢看文字可以打开文件夹里的HTML文件

（Django 购物平台数据库搭建 (2025\_3\_12 13：08：07).html）翻到比较靠下的地方

你已经写了 MySQL 的配置，基本上是正确的！接下来确认几个关键步骤，确保你的 Django 项目可以正常连接到 MySQL。以下是完整的配置和排查步骤：

**🚦 步骤 1：安装 MySQL 驱动**

你的 settings.py 已经用 django.db.backends.mysql 指定了 MySQL，但还需要安装 MySQL 驱动：

**使用 mysqlclient（推荐）：**

sh

复制编辑

pip install mysqlclient

如果安装失败，可以尝试使用 pymysql（兼容性更好，但性能略差）：

sh

复制编辑

pip install pymysql

如果用的是 pymysql，还需要在 tango\_with\_django\_project/\_\_init\_\_.py 文件中加上：

python

复制编辑

import pymysql

pymysql.install\_as\_MySQLdb()

**⚙ 步骤 2：确认 MySQL 服务运行并创建数据库**

1. 确保 MySQL 服务已启动：

sh

复制编辑

mysql -u root -p

1. 创建数据库（如果还没创建）：

sql

复制编辑

CREATE DATABASE myshop CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;

**📝 步骤 3：修改 settings.py（优化细节）**

你已有的配置基本没问题，但建议补充编码选项，避免中文存储问题：

python

复制编辑

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'myshop',

'USER': 'root',

'PASSWORD': '123456',

'HOST': '127.0.0.1',

'PORT': '3306',

'OPTIONS': {

'charset': 'utf8mb4',

},

}

}

**🔧 步骤 4：迁移数据库**

1. 先让 Django 确认数据库连接正常：

sh

复制编辑

python manage.py inspectdb

如果能打印出数据库的模型结构，说明连接正常。

1. 如果一切正常，运行迁移命令，创建表结构：

sh

复制编辑

python manage.py makemigrations

python manage.py migrate

**✅ 步骤 5：测试**

1. 启动服务器：

sh

复制编辑

python manage.py runserver

1. 访问：

* http://127.0.0.1:8000/admin/：确认能正常操作 MySQL 数据。
* http://127.0.0.1:8000/api/products/：如果你用 DRF，确认能拉取数据。

**🎉 总结：**

1. 安装 mysqlclient 或 pymysql。
2. 创建 MySQL 数据库，确认连接正常。
3. 修改 settings.py 增加 charset。
4. 运行迁移命令，启动项目，测试接口。

如果还有问题，随时告诉我！🚀

**您说：**

(base) PS C:\Users\15893\Desktop\IT\tango\_with\_django\_project-main\tango\_with\_django\_project-main> pip install mysqlclient Requirement already satisfied: mysqlclient in c:\users\15893\anaconda3\lib\site-packages (2.2.4) (base) PS C:\Users\15893\Desktop\IT\tango\_with\_django\_project-main\tango\_with\_django\_project-main> mysql -u root -p mysql : 无法将“mysql”项识别为 cmdlet、函数、脚本文件或可运行程序的名称。请检查名称的拼写，如果包括路径，请确保路径正确，然后再试一次。 所在位置 行:1 字符: 1 + mysql -u root -p + ~~~~~ + CategoryInfo : ObjectNotFound: (mysql:String) [], CommandNotFoundException + FullyQualifiedErrorId : CommandNotFoundException (base) PS C:\Users\15893\Desktop\IT\tango\_with\_django\_project-main\tango\_with\_django\_project-main>

**ChatGPT 说：**

看起来你的电脑已经安装了 mysqlclient，但命令行找不到 mysql 命令。这通常意味着 MySQL 没有正确安装，或者它的路径没有加入系统的环境变量。让我们一步步排查！✨

**🏗 步骤 1：确认 MySQL 是否已安装**

1. 在 PowerShell 中运行下面的命令，看看 MySQL 是否已经安装：

powershell

复制编辑

Get-Command mysql

如果返回“找不到命令”，说明 MySQL 还没有安装，或者安装路径没有添加到系统的 PATH 环境变量。

1. 如果没装，请去 [MySQL 官网下载](https://dev.mysql.com/downloads/installer/) 并安装 **MySQL Community Server**。安装时勾选 "Add MySQL to PATH"。

**⚙ 步骤 2：手动添加 MySQL 到环境变量**

如果已经安装但无法识别 mysql 命令，你需要手动配置环境变量：

1. **找到 MySQL 安装路径**（一般在 C:\Program Files\MySQL\MySQL Server 8.0\bin）。
2. **添加到环境变量**：
   * 打开“系统属性”（sysdm.cpl）。
   * 点击“环境变量”。
   * 找到“Path”，双击进入。
   * 点击“新建”，粘贴 MySQL 的 bin 目录路径（例如 C:\Program Files\MySQL\MySQL Server 8.0\bin）。
3. **验证路径**： 关闭 PowerShell，再重新打开，输入：

powershell

复制编辑

mysql --version

如果能看到版本信息，说明配置成功。

**🔍 步骤 3：确认连接**

如果一切正常，你可以用这个命令连接 MySQL：

powershell

复制编辑

mysql -u root -p

然后输入密码 123456（根据你的 settings.py）。

如果不想每次都输密码，可以直接写在命令里（不安全，慎用）：

powershell

复制编辑

mysql -u root -p123456

**🚀 总结：**

1. 确认 MySQL 安装（如果没装，去官网下载安装）。
2. 如果装了但无法识别 mysql 命令，手动把 bin 目录加到环境变量。
3. 配置好后，重新打开终端，用 mysql -u root -p 测试连接。

如果还有问题，随时告诉我！💪

**您说：**

Using the URLconf defined in tango\_with\_django\_project.urls, Django tried these URL patterns, in this order: index/ user/list/ The empty path didn’t match any of these. You’re seeing this error because you have DEBUG = True in your Django settings file. Change that to False, and Django will display a standard 404 page.

**ChatGPT 说：**

你遇到的这个错误表明 Django 没有找到匹配的 URL 路径。让我们一步步排查，确保你的 URL 配置正常！✨

**🏗 步骤 1：检查项目结构**

确认项目结构大致如下（路径可能不同，但重点是 urls.py 的位置）：

markdown

复制编辑

tango\_with\_django\_project/

├── tango\_with\_django\_project/

│ ├── urls.py

│ ├── settings.py

│ ├── wsgi.py

│ └── \_\_init\_\_.py

└── app01/

├── urls.py

├── views.py

└── models.py

**⚙ 步骤 2：配置 urls.py**

**📌 2.1 主 URL 配置**

打开 tango\_with\_django\_project/urls.py 文件，确认它引入了 app01 的路由：

python

复制编辑

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('app01/', include('app01.urls')), # 确保这行存在

]

如果 include 没导入，别忘了加上它！include 让 Django 知道去 app01 里找更多的 URL 配置。

**📌 2.2 app01 的 URL 配置**

确保 app01 目录下有 urls.py（如果没有，创建一个）：

python

复制编辑

from django.urls import path

from app01 import views

urlpatterns = [

path('index/', views.index, name='index'),

path('user/list/', views.user\_list, name='user\_list'),

]

**解释：**

* path('index/', views.index, name='index') 表示访问 /app01/index/ 时调用 index 视图。
* path('user/list/', views.user\_list, name='user\_list') 表示访问 /app01/user/list/ 时调用 user\_list 视图。

**📌 2.3 配置视图**

确保 app01/views.py 里有对应的视图函数：

python

复制编辑

from django.http import HttpResponse

def index(request):

return HttpResponse("欢迎来到首页！")

def user\_list(request):

return HttpResponse("这是用户列表页面！")

**🚀 步骤 3：测试**

1. 启动服务器：

bash

复制编辑

python manage.py runserver

1. 访问以下地址，确认是否正常工作：

* http://127.0.0.1:8000/app01/index/
* http://127.0.0.1:8000/app01/user/list/

1. 如果希望直接访问根目录 /，可以在 tango\_with\_django\_project/urls.py 里加一个空路径：

python

复制编辑

urlpatterns = [

path('', views.index, name='home'), # 让根目录指向 index 视图

path('admin/', admin.site.urls),

path('app01/', include('app01.urls')),

]

**总结：**

1. 确保 urls.py 里 include 了 app01。
2. 确保 app01/urls.py 里定义了 URL 路由。
3. 确保 views.py 里有对应的视图函数。
4. 重新启动服务器测试。

如果还有问题，把你的 urls.py 和 views.py 发给我，我帮你一起检查！🌟

**您说：**

我希望在此Django项目中完成数据库的增删改查等操作

**ChatGPT 说：**

好的，我们来实现数据库的增删改查（CRUD：Create、Read、Update、Delete）！✨ 我会用 MySQL 数据库，同时使用 Django 的模型（ORM）来操作数据。以下步骤涵盖了模型创建、视图编写、URL 配置和测试。

**🏗 步骤 1：创建模型**

在 app01/models.py 中定义你的数据库模型。比如，我们创建一个商品（Product）模型：

python

复制编辑

from django.db import models

class Product(models.Model):

name = models.CharField(max\_length=100)

description = models.TextField(blank=True, null=True)

price = models.DecimalField(max\_digits=10, decimal\_places=2)

stock = models.PositiveIntegerField()

created\_at = models.DateTimeField(auto\_now\_add=True)

updated\_at = models.DateTimeField(auto\_now=True)

def \_\_str\_\_(self):

return self.name

保存后，运行下面的命令来生成迁移文件并迁移数据库：

bash

复制编辑

python manage.py makemigrations app01

python manage.py migrate

**⚙ 步骤 2：创建序列化器（如果用 DRF）**

如果你打算用 Django REST Framework（推荐），创建一个 app01/serializers.py 文件：

python

复制编辑

from rest\_framework import serializers

from .models import Product

class ProductSerializer(serializers.ModelSerializer):

class Meta:

model = Product

fields = '\_\_all\_\_'

**📝 步骤 3：编写视图**

在 app01/views.py 中，使用基于类的视图或函数视图来处理 CRUD 操作：

python

复制编辑

from django.shortcuts import get\_object\_or\_404

from django.http import JsonResponse

from .models import Product

from .serializers import ProductSerializer

from rest\_framework.decorators import api\_view

from rest\_framework.response import Response

from rest\_framework import status

# 创建（Create）

@api\_view(['POST'])

def create\_product(request):

serializer = ProductSerializer(data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data, status=status.HTTP\_201\_CREATED)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

# 读取（Read）

@api\_view(['GET'])

def list\_products(request):

products = Product.objects.all()

serializer = ProductSerializer(products, many=True)

return Response(serializer.data)

# 更新（Update）

@api\_view(['PUT'])

def update\_product(request, pk):

product = get\_object\_or\_404(Product, pk=pk)

serializer = ProductSerializer(product, data=request.data)

if serializer.is\_valid():

serializer.save()

return Response(serializer.data)

return Response(serializer.errors, status=status.HTTP\_400\_BAD\_REQUEST)

# 删除（Delete）

@api\_view(['DELETE'])

def delete\_product(request, pk):

product = get\_object\_or\_404(Product, pk=pk)

product.delete()

return Response({"message": "Product deleted"}, status=status.HTTP\_204\_NO\_CONTENT)

**🔗 步骤 4：配置 URL**

在 app01/urls.py 里配置 URL 路由（如果没有这个文件，新建一个）：

python

复制编辑

from django.urls import path

from .views import create\_product, list\_products, update\_product, delete\_product

urlpatterns = [

path('products/', list\_products, name='product-list'),

path('products/create/', create\_product, name='product-create'),

path('products/<int:pk>/update/', update\_product, name='product-update'),

path('products/<int:pk>/delete/', delete\_product, name='product-delete'),

]

然后别忘了在 tango\_with\_django\_project/urls.py 里引入 app01.urls：

python

复制编辑

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('api/', include('app01.urls')),

]

**🚀 步骤 5：测试 CRUD**

1. 启动服务器：

bash

复制编辑

python manage.py runserver

1. 使用工具测试接口（推荐 Postman、Insomnia，或者直接用浏览器）：

* **创建商品（POST）：**  
  URL: http://127.0.0.1:8000/api/products/create/  
  Body（JSON）：

json

复制编辑

{

"name": "Laptop",

"description": "Gaming laptop",

"price": 1299.99,

"stock": 10

}

* **查询所有商品（GET）：**  
  URL: http://127.0.0.1:8000/api/products/
* **更新商品（PUT）：**  
  URL: http://127.0.0.1:8000/api/products/1/update/  
  Body（JSON）：

json

复制编辑

{

"name": "Updated Laptop",

"description": "Updated description",

"price": 1399.99,

"stock": 5

}

* **删除商品（DELETE）：**  
  URL: http://127.0.0.1:8000/api/products/1/delete/

**🎉 总结：**

1. 创建模型并迁移数据库。
2. 用 Django REST Framework 编写序列化器和视图。
3. 配置 URL，确保前端能调用 CRUD 接口。
4. 使用 Postman 或浏览器进行测试。

如果有问题，随时告诉我！🚀

4o

窗体顶端

O

窗体底端