Phase 1 Report: Project Preparation and Basic Architecture Construction

Date: September 19, 2024 - October 8, 2024 Group 18

I.Summary of the Work Done to Date

1.1 Environment Setup and Database Configuration

In the first phase, our main task was to complete the setup of the development environment and configure the database to support subsequent development. The specific steps include:

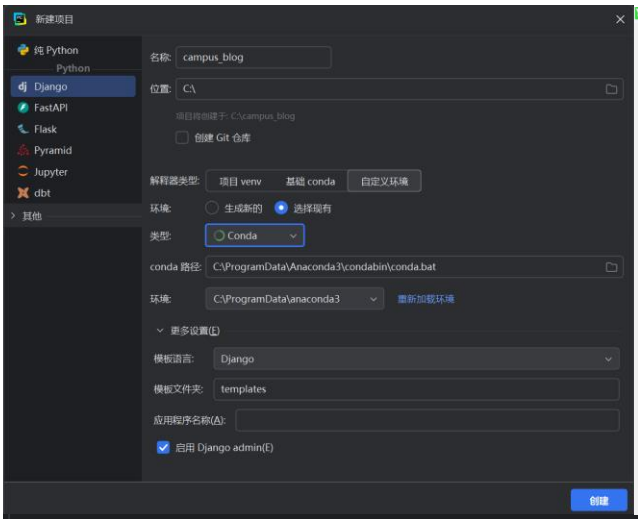
Django Framework Installation: We successfully set up the Django development environment and installed MySQL as the backend database system. To ensure project development efficiency, the team followed a unified environment configuration guide, ensuring consistency across all setups.

At the same time, the team also conducted systematic learning of Django's foundational knowledge, especially the core components such as the ORM class, Form class, and Model class:

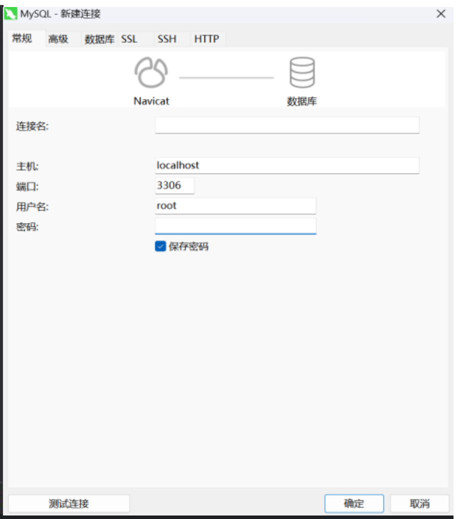
ORM Class (Object-Relational Mapping): Through Django's ORM system, we can map Python objects to database tables, simplifying database operations. We focused on learning how to define model classes and implement CRUD (Create, Read, Update, Delete) operations. Additionally, we learned how to use QuerySets to efficiently manipulate data.

Form Class: The Form class is a key component in Django used for handling form data. We learned how to use the Form class for data validation and how to integrate it with the template system to handle user input. For example, with the ModelForm class, we can quickly generate forms corresponding to models, simplifying form handling logic.

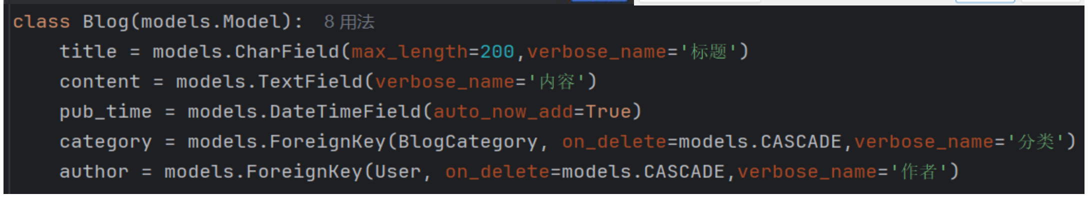
Model Class: The Model class defines the structure of database tables. We designed core data models such as blog posts, user information, and comments using Django's Model class. By defining fields and attributes in the model class, we can describe data structures intuitively and use ORM to operate on the database.



名称: campus_blog
位置: C:\
项目将创建于: C:\campus_blog
创建 Git 仓库
解释器类型: 项目 venv 基础 conda [✓ 自定义环境]
环境: 生成新的 [✓ 选择现有]
类型: Conda
conda 路径: C:\ProgramData\Anaconda3\condabin\conda.bat
环境: C:\ProgramData\anaconda3 重新加载环境
更多设置(E)
模板语言: Django
模板文件夹: templates
应用程序名称(A):
[✓] 启用 Django admin(E)



连接名:
主机: localhost
端口: 3306
用户名: root
密码:
[✓] 保存密码



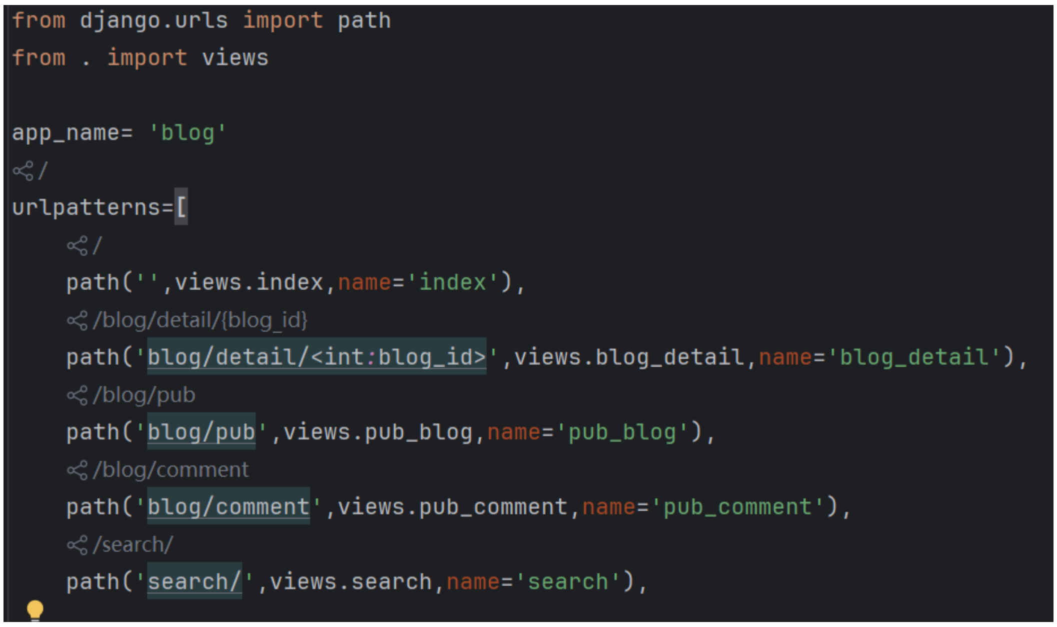
```
class Blog(models.Model):  8 用法
    title = models.CharField(max_length=200,verbose_name='标题')
    content = models.TextField(verbose_name='内容')
    pub_time = models.DateTimeField(auto_now_add=True)
    category = models.ForeignKey(BlogCategory, on_delete=models.CASCADE,verbose_name='分类')
    author = models.ForeignKey(User, on_delete=models.CASCADE,verbose_name='作者')
```

1.2 Basic Configuration of Routing and Template System

We have completed the basic routing configuration of the project, which allows handling page transitions between different views. At the same time, we used Django's template system to build the basic structure of the front-end pages. Through template inheritance and static file loading, we achieved dynamic content rendering. These tasks have laid a solid foundation for subsequent page development.

Routing Configuration: The basic URL routing setup of the project was completed, enabling connections between pages and view functions.

Template System: Using Django Template Language (DTL), we rendered basic HTML pages. By employing the include and extend tags, we ensured the modularity and reusability of page layouts and navigation.



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

app_name= 'blog'
/
urlpatterns=[
    /
    path('',views.index,name='index'),
    /blog/detail/{blog_id}
    path('blog/detail/<int:blog_id>',views.blog_detail,name='blog_detail'),
    /blog/pub
    path('blog/pub',views.pub_blog,name='pub_blog'),
    /blog/comment
    path('blog/comment',views.pub_comment,name='pub_comment'),
    /search/
    path('search/',views.search,name='search'),
```

II.Challenges Faced and Solutions

2.1 **Challenge 1**: Initial Issues with Database Configuration

In the early stages, we encountered issues with database connections, mainly due to compatibility problems between MySQL versions and Django. Some team members were unable to connect to the database smoothly in their local environments.

Solution: We decided to standardize on MySQL 8 and use the same Django version across the team. Team members also tested database connections through local virtual environments to ensure everyone could operate the database smoothly.

2.2 **Challenge 2**: Implementing Dynamic Rendering in the Template System

During the initial setup of the template system, we faced performance issues with page loading, particularly with the loading of static files and the inclusion of CSS and JavaScript resources.

Solution: We optimized the loading paths of static files and ensured that all static resources were correctly loaded by using Django's static template tag. Additionally, we adopted browser caching techniques to improve page loading speeds.

III.Next Steps

Complete User Management and Authentication Functions: In the next phase, we will focus on implementing user registration, login, and authentication systems to ensure data security and accurate user identity verification. Specific features include CAPTCHA functionality and email sending capabilities.

Further Improve Front-end and Back-end Interaction: We will continue to optimize the efficiency of data exchange between the front-end and back-end through the template system, achieving more dynamic data display.

Conduct Initial System Testing: After completing the user management features, we will perform initial testing to ensure that all functional modules operate stably, preparing for further feature expansion in the next phase.