HuanBlog开发注解-Part1

主题内容

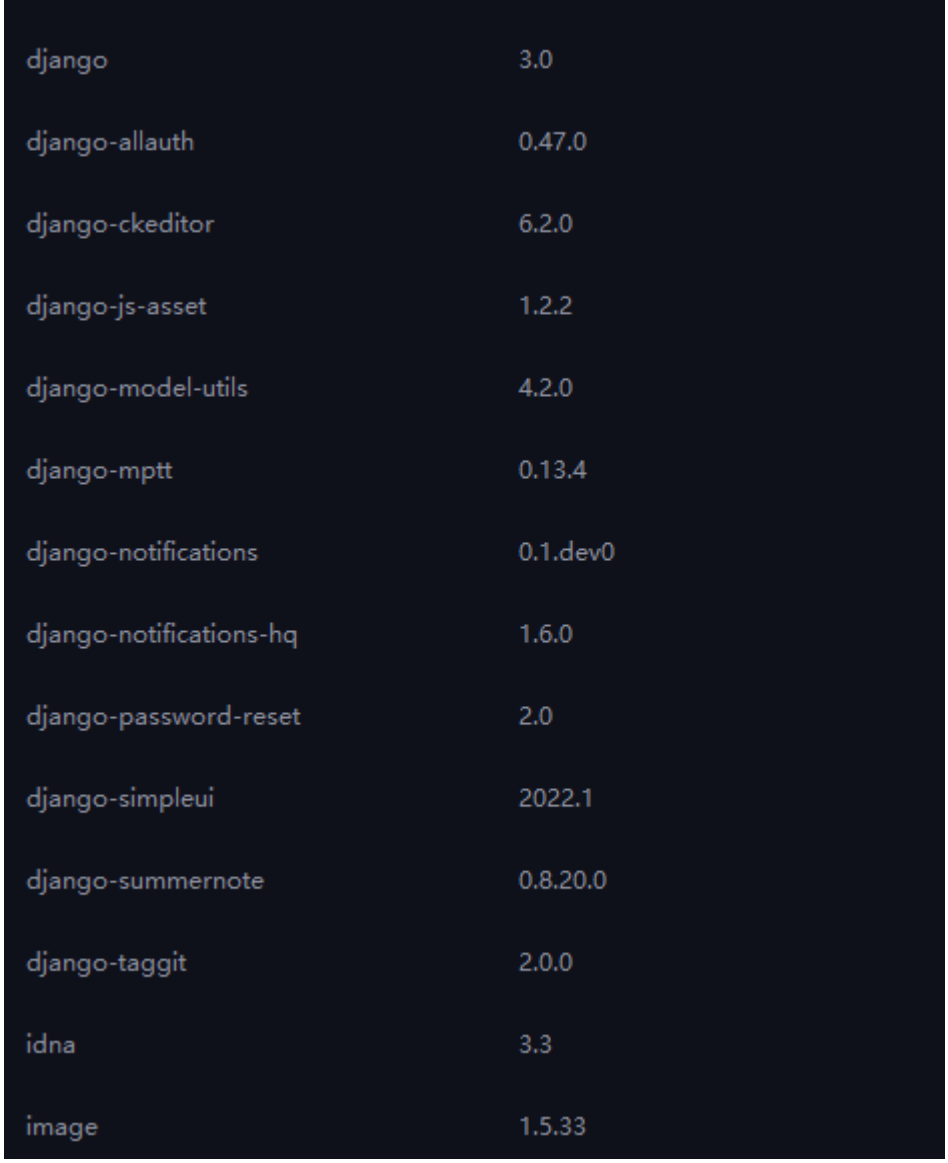
- 数据库原理课程设计
- 主题：基于MySQL与Django的博客管理系统
- 开发者：李欢欢

技术栈

- 前后端半分离架构
- Django为主体Web框架
- 使用的版本为Django3
- 数据库使用MySQL8.0
- 整体采用MVC设计模式
    - MVC模式
    - M-Model 模型 与数据库直接相关
    - V-View 视图 用户看到并与之交互的界面
    - C-Controller 控制器接受用户的输入并调用模型和视图去完成用户的需求

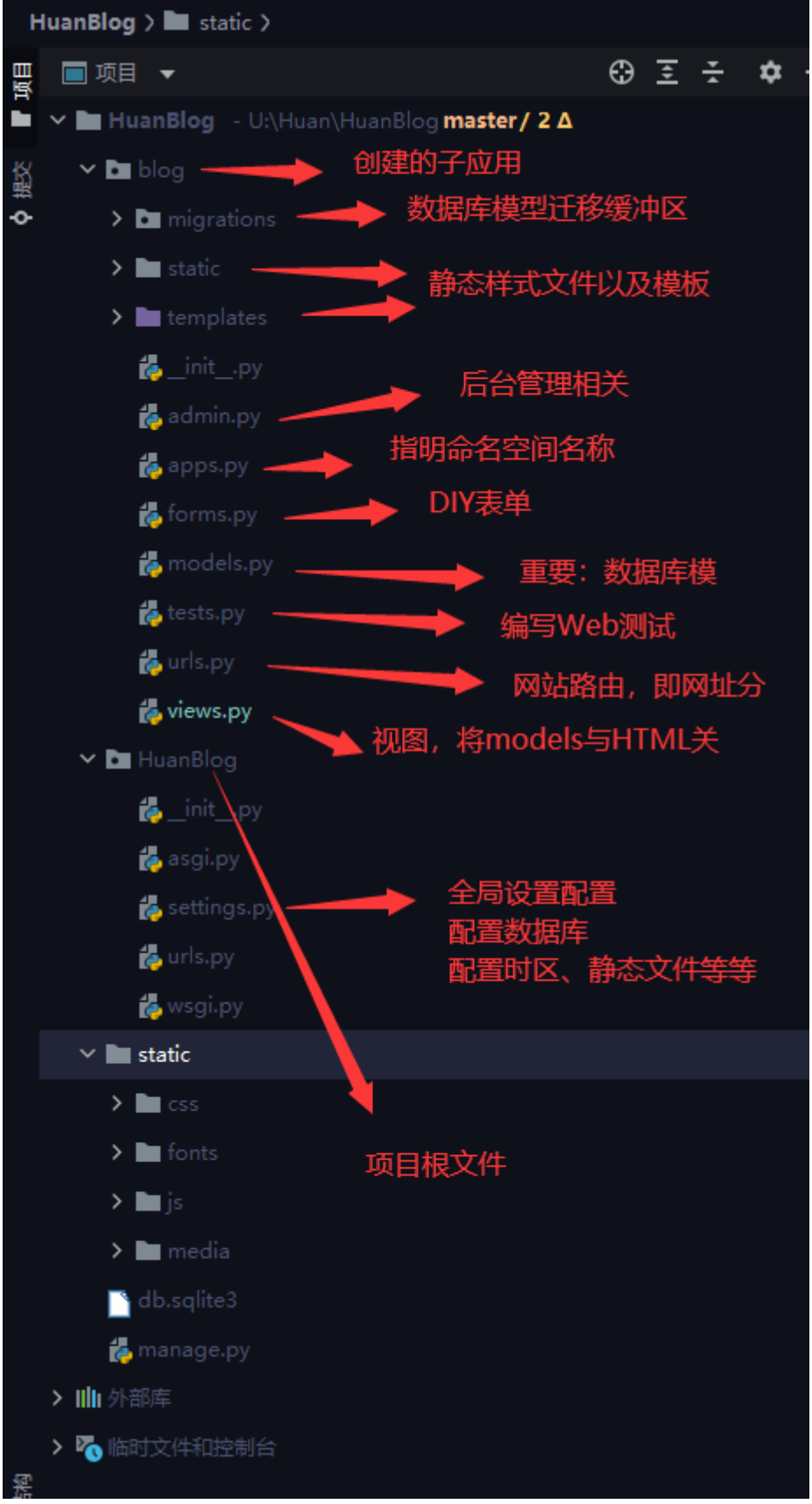
开发流程

- 配置conda环境 安装以下环境库

- 

- 在conda环境下cd进入一个路径，使用django命令行创建项目
  - `django-admin startproject HuanBlog`
- 创建项目子app
  - 子app的创建目的是对一个大的系统进行功能拆分，拆分成一个个的小app以提高代码可读性和编写效率
  - `python manage.py startapp blog`
- STEP1：设计数据库模型，编写models
- STEP2：设计网页界面模板，编写views，视图层完成关联
- STEP3：设计控制器交互以及网站路由，编写urls.py
- STEP4：编写配置文件settings.py
- STEP5：数据模型解码迁移
  - 1、`python manage.py makemigrations`
    - 记录对models.py的所有改动，并且将这个改动迁移到migrations这个文件下生成一个文件
    - 并未迁移到数据库
  - 2、`python manage.py migrate`
    - 把这些改动作用到数据库也就是执行migrations里面新改动的迁移文件更新数据库
    - 比如创建数据表，或者增加字段属性
- STEP6：运行测试 `python manage.py runserver`

文件结构注解（重要）



部分代码注解

models.py

Model (模型) 简而言之即数据模型，是一个Django应用的核心。

模型不是数据本身（比如数据表里的数据), 而是抽象的描述数据的构成和逻辑关系。

每个Django的模型(model)实际上是个类，继承了`models.Model`。

每个Model应该包括属性(字段)，关系（比如单对单，单对多和多对多)和方法。

当定义好Model模型后，Django的接口会自动在数据库生成相应的数据表(table)。

这样就不用自己用SQL语言创建表格或在数据库里操作创建表格了

模型的组成

- 一个标准的Django模型分别由模型字段、META选项和方法三部分组成。
- 定义的模型字段：包括基础字段和关系字段
- 自定义的Manager方法：改变模型
- `class Meta选项`: 包括排序、索引等等(可选)。
- `def __str__()`：定义单个模型实例对象的名字(可选)。
- `def save()`：重写save方法(可选)。
- `def get_absolute_url()`：为单个模型实例对象生成独一无二的url(可选)
- 其它自定义的方法。

模型的字段

`models.Model` 提供的常用模型字段包括基础字段和关系字段。

CharField()

- 字符字段
- 一般需要通过max_length = xxx 设置最大字符长度。
- 如不是必填项，可设置blank = True和default = ‘’。
- 如果用于username, 想使其唯一，可以设置`unique = True`。
- 如果有choice选项，可以设置 choices = XXX_CHOICES

TextField()

- 大量文本字段

DateField()和DateTimeField()

- 可通过default=xx选项设置默认日期和时间。
- 对于DateTimeField: default=timezone.now - 先要`from django.utils import timezone`
- 如果希望自动记录一次修改日期(modified)，可以设置: `auto_now=True`
- 如果希望自动记录创建日期(created),可以设置`auto_now_add=True`

IntegerField(), SlugField(), URLField()，BooleanField()

整型字段、SlugField 本质上相当于存放字符串、URL地址字段、布尔字段

可以设置blank = True or null = True。对于BooleanField一般建议设置`defaut = True or False`

FileField(upload_to=None, max_length=100) - 文件字段

- upload_to = “/some folder/”：上传文件夹路径
- max_length = xxxx：文件最大长度

ImageField (upload_to=None, max_length=100,)- 图片字段

- upload_to = “/some folder/”: 指定上传图片路径

关系字段

OneToOneField(to, on_delete=xxx, options) - 单对单关系

- to必需指向其他模型，比如 Book or ‘self’ .
- 必需指定`on_delete`选项(删除选项): i.e, “`on_delete = models.CASCADE`” or “`on_delete = models.SET_NULL`” .

- 可以设置“`related_name = xxx`”便于反向查询。

ForeignKey(to, on_delete=xxx, options) - 单对多关系

- to必需指向其他模型，比如 Book or ‘self’ .
- 必需指定`on_delete`选项(删除选项): i.e, “`on_delete = models.CASCADE`” or “`on_delete = models.SET_NULL`” .
- 可以设置”default = xxx” or “null = True” ;
- 如果有必要，可以设置“`limit_choices_to =`“,
- 可以设置“`related_name = xxx`”便于反向查询。

ManyToManyField(to, options) - 多对多关系

- to 必需指向其他模型，比如 User or ‘self’ .
- 设置“`symmetrical = False`“表示多对多关系不是对称的，比如A关注B不代表B关注A
- 设置“`through = 'intermediary model'`“如果需要建立中间模型来搜集更多信息。
- 可以设置“`related_name = xxx`”便于反向查询。

on_delete删除选项

了解CASCADE级联删除即可

模型的META选项

模型可以理解成类

- `abstract=True`: 指定该模型为抽象模型
- `proxy=True`: 指定该模型为代理模型
- `verbose_name=xxx`和`verbose_name_plural=xxx`: 为模型设置便于人类阅读的别名
- `db_table= xxx`: 自定义数据表名
- `odering=['-pub-date']`: 自定义按哪个字段排序，`-`代表逆序
- `permissions=[]`: 为模型自定义权限
- `managed=False`: 默认为True，如果为False，Django不会为这个模型生成数据表
- `indexes=[]`: 为数据表设置索引，对于频繁查询的字段，建议设置索引
- `constraints=`: 给数据库中的数据表增加约束。

模型的方法

以下三个方法是Django模型自带的三个标准方法:

- `def __str__()`: 给单个模型对象实例设置人为可读的名字(可选)。
- `def save()`: 重写save方法(可选)。
- `def get_absolute_url()`: 为单个模型实例对象生成独一无二的url(可选)

代码实例讲解

今天来看一下我们的Blog中的User关系模型

```
class User(models.Model):
    username = models.CharField(max_length=50)
    password = models.CharField(max_length=200)
    nickname = models.CharField(max_length=50, default='匿名用户')
    email = models.EmailField
    created_time = models.CharField(max_length=50, default=now)
    comment_num = models.PositiveIntegerField(verbose_name='评论条数', default=0)
    avatar = models.ImageField(upload_to='media', default="media/cat.png")

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

```
    def comment(self):
        self.comment_num += 1
        self.save(update_fields=['comment_num'])

    def comment_del(self):
        self.comment_num -= 1
        self.save(update_fields=['comment_num'])

    class Meta:
        verbose_name = '用户'  # 指定后台显示模型名称
        verbose_name_plural = '用户'  # 指定后台显示模型复数名称
        db_table = "blog_user"
```

姐姐尝试理解上述代码