软件工程基础 第三次实验

后端入门基础——Django

本次实验将带大家从零开始创建一个Django后端项目

创建项目

>django-admin startproject [项目名]

>cd [项目目录]

>python manage.py startapp [应用名]
(别忘加入settings.py)

项目结构

-manage.py
-[项目名]
    -__init__.py
    -settings.py
    -asig.py
    -wsig.py
    -urls.py
-[应用名]
    -migrations
    -__init__.py
    -models.py
    -apps.py
    -views.py
    -tests.py
    -admin.py

项目运行

运行项目三部曲：

>python manage.py makemigrations

>python manage.py migrate

更改models.py
后必须执行

(>python manage.py createsuperuser)

>python manage.py runserver

开始编写

1.models.py

作用：建立数据库表中需求需要的表与属性。

Django的Model数据类型：
1、AutoField
根据 ID 自增长的 IntegerField 字段。通常用于主键ID，无需使用。
2、IntegerField
32位整数，可自定义选项（具体方法见视频）。
3、BooleanField
一个布尔值(true/false)字段。
4、CharField
一个字符串字段，对小字符串和大字符串都适用。对于大量文本建议使用TextField 。
必须参数：max_length,字段的最大字符数。
5、DateField
利用 Python 的 datetime.date 实例来表示日期。
可选参数DateField.auto_now：每一次保存对象时，Django 都会自动将该字段的值设置为当前时间。一般用来表示 "最后修改" 时间。
可选参数DateField.auto_now_add：在第一次创建对象时，Django 自动将该字段的值设置为当前时间，一般用来表示对象创建时间。

作用：建立数据库表中需求需要的表与属性。

Django的Model数据类型：
6、DateTimeField
利用 datetime.datetime 实例表示日期和时间。该字段所按受的参数和 DateField 一样。
7、DecimalField
使用 Decimal 实例表示固定精度的十进制数的字段。
必须参数DecimalField.max_digits：数字允许的最大位数
必须参数DecimalField.decimal_places：小数的最大位数
8、EmailField
可以看做带有 email 合法性检测的CharField 。
默认max_length=75。
9、TextField
超大文本字段。
10、FileField
文件字段
11、ImageField
继承于FileField，确保是有效图片

作用：建立数据库表中需求需要的表与属性。

Django的Model数据的通用参数：

1、null

是否允许为空值，默认为false。

2、default

该属性的默认值。

3、primary_key

该属性是否为主键，默认为false。

4、unique

该属性的值是否唯一，默认为false。

2.admin.py

作用：在admin页面注册model，方便管理自带数据库（不使用自带数据库可忽略）。

创建类的方法类似models
类内部使用list_display定义要管理的属性
类外部使用admin.site.register()函数进行注册

```
1  from django.contrib import admin
2  from student.models import Student
3
4
5  class StudentAdmin(admin.ModelAdmin):
6      list_display = ('id', 'name', 'sex')
7
8
9  admin.site.register(Student, StudentAdmin)
```

3.apps.py

作用：与settings.py有相似作用，作用域为所属应用而非全局，利用config在settings.py中注册使应用在项目中可见。

创建类的方法类似models
类内部使用list_display定义要管理的属性
类外部使用admin.site.register()函数进行注册

apps.py

```python
from django.apps import AppConfig


class StudentConfig(AppConfig):
    name = 'student'
```

settings.py

```python
INSTALLED_APPS = [
    'student',

    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

4.views.py

作用：编写和规划所有的请求处理函数。

HTML8大请求：

序号	方法	描述
1	GET	请求指定的页面信息，并返回实体主体。
2	HEAD	类似于get请求，只不过返回的响应中没有具体的内容，用于获取报头
3	POST	向指定资源提交数据进行处理请求（例如提交表单或者上传文件）。数据被包含在请求体中。POST请求可能会导致新的资源的建立和/或已有资源的修改。
4	PUT	从客户端向服务器传送的数据取代指定的文档的内容。
5	DELETE	请求服务器删除指定的页面。
6	CONNECT	HTTP/1.1协议中预留给能够将连接改为管道方式的代理服务器。
7	OPTIONS	允许客户端查看服务器的性能。
8	TRACE	回显服务器收到的请求，主要用于测试或诊断。

Django获取请求的方法：request.method

```
<form action="/login/" method="post">
    {% csrf_token %}
    请输入用户名<input type="text" name="username"/>
    <br/>
    请输入密码<input type="password" name="password"/>
    <br/>
    <button type="submit">登录</button>
    <br/>
```

```
if request.method == 'POST':
        username = request.POST.get('username')
        password = request.POST.get('password')
```

Django数据库相关操作：

1.增：

方法一：

```
user = User()
...
user.save()
```

方法二：

```
user = User(id=..., name=..., ...)
```

方法三：

```
User.objects.create(id=..., name=..., ...)
```

Django数据库相关操作：

2.查：

查询特定结果（如有相同则返回首个）：

```
user = User.objects.get(id=1)
```

查询多个结果（返回一个列表）：

```
users = User.objects.filter(kind='学生')
```

查询全部结果：

```
users = User.objects.all()
```

模糊查询、比较查询等高级查询方法：

https://blog.csdn.net/yanpenggong/article/details/82316514

Django数据库相关操作：

3.改：

方法一：

```
user = User.objects.get(id=1)
user.name = lkw
user.save()
```

方法二：

```
users = User.objects.filter(kind='学生')
users.update(kind='老师')
```

4.删：

```
user = User.objects.get(id=1)
user.delete()
```

4.views.py

session：临时保存在服务器端的用户数据，本质是键值对
将用户登录时的ID存储在session中以便后续获取，从而知道后续是谁在进行操作

应用示例：

登录时：

```
if user.password == password:
        request.session['id'] = user.id
        request.session['kind'] = "user"
        ...
```

获取当前用户信息时：

```
if request.session.get('kind') == 'user':
        user = User.objects.get(id=request.session.get('id'))
        email = user.email
        real_name = user.real_name
        ID_card = user.ID_card
        ...
```

谢 谢