静态资源和Ajax请求

基于前面两个章节讲解的知识，我们已经可以使用Django框架来完成Web应用的开发了。接下来我们就尝试实现一个投票应用，具体的需求是用户进入应用首先查看到“学科介绍”页面，该页面显示了一个学校所开设的所有学科；通过点击某个学科，可以进入“老师介绍”页面，该页面展示了该学科所有老师的详细情况，可以在该页面上给老师点击“好评”或“差评”；如果用户没有登录，在投票时会先跳转到“登录页”要求用户登录，登录成功才能投票；对于未注册的用户，可以在“登录页”点击“新用户注册”进入“注册页”完成用户注册操作，注册成功后会跳转到“登录页”，注册失败会获得相应的提示信息。

准备工作

由于之前已经详细的讲解了如何创建Django项目以及项目的相关配置，因此我们略过这部分内容，唯一需要说明的是，从上面对投票应用需求的描述中我们可以分析出三个业务实体：学科、老师和用户。学科和老师之间通常是一对多关联关系（一个学科有多个老师，一个老师通常只属于一个学科），用户因为要给老师投票，所以跟老师之间是多对多关联关系（一个用户可以给多个老师投票，一个老师也可以收到多个用户的投票）。首先修改应用下的models.py文件来定义数据模型，先给出学科和老师的模型。

```
from django.db import models


class Subject(models.Model):
    """学科"""
    no = models.IntegerField(primary_key=True, verbose_name='编号')
    name = models.CharField(max_length=20, verbose_name='名称')
    intro = models.CharField(max_length=511, default='', verbose_name='介绍')
    create_date = models.DateField(null=True, verbose_name='成立日期')
    is_hot = models.BooleanField(default=False, verbose_name='是否热门')

    def __str__(self):
        return self.name

    class Meta:
        db_table = 'tb_subject'
        verbose_name = '学科'
        verbose_name_plural = '学科'


class Teacher(models.Model):
    """老师"""
    no = models.AutoField(primary_key=True, verbose_name='编号')
    name = models.CharField(max_length=20, verbose_name='姓名')
    detail = models.CharField(max_length=1023, default='', blank=True,
verbose_name='详情')
    photo = models.CharField(max_length=1023, default='', verbose_name='照片')
    good_count = models.IntegerField(default=0, verbose_name='好评数')
    bad_count = models.IntegerField(default=0, verbose_name='差评数')
    subject = models.ForeignKey(to=Subject, on_delete=models.PROTECT,
db_column='sno', verbose_name='所属学科')

    class Meta:
        db_table = 'tb_teacher'
        verbose_name = '老师'
```

```
        verbose_name_plural = '老师'
```

模型定义完成后，可以通过“生成迁移”和“执行迁移”来完成关系型数据库中二维表的创建，当然这需要提前启动数据库服务器并创建好对应的数据库，同时我们在项目中已经安装了PyMySQL而且完成了相应的配置，这些内容此处不再赘述。

```
(venv)$ python manage.py makemigrations vote
...
(venv)$ python manage.py migrate
...
```

> 注意：为了给vote应用生成迁移文件，需要修改Django项目settings.py文件，在INSTALLED_APPS中添加vote应用。

完成模型迁移之后，我们可以直接使用Django提供的后台管理来添加学科和老师信息，这需要先注册模型类和模型管理类。

```
from django.contrib import admin

from poll2.forms import UserForm
from poll2.models import Subject, Teacher


class SubjectAdmin(admin.ModelAdmin):
    list_display = ('no', 'name', 'create_date', 'is_hot')
    ordering = ('no', )


class TeacherAdmin(admin.ModelAdmin):
    list_display = ('no', 'name', 'detail', 'good_count', 'bad_count',
'subject')
    ordering = ('subject', 'no')


admin.site.register(Subject, SubjectAdmin)
admin.site.register(Teacher, TeacherAdmin)
```

接下来，我们就可以修改views.py文件，通过编写视图函数先实现“学科介绍”页面。

```
def show_subjects(request):
    """查看所有学科"""
    subjects = Subject.objects.all()
    return render(request, 'subject.html', {'subjects': subjects})
```

至此，我们还需要一个模板页，模板的配置以及模板页中模板语言的用法在之前已经进行过简要的介绍，如果不熟悉可以看看下面的代码，相信这并不是一件困难的事情。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>所有学科信息</title>
    <style>/* 此处略去了层叠样式表的选择器 */</style>
</head>
<body>
```

```
    <h1>所有学科</h1>
    <hr>
    {% for subject in subjects %}
    <div>
        <h3>
            <a href="/teachers/?sno={{ subject.no }}">{{ subject.name }}</a>
            {% if subject.is_hot %}
            <img src="/static/images/hot.png" width="32" alt="">
            {% endif %}
        </h3>
        <p>{{ subject.intro }}</p>
    </div>
    {% endfor %}
</body>
</html>
```

在上面的模板中，我们为每个学科添加了一个超链接，点击超链接可以查看该学科的讲师信息，为此需要再编写一个视图函数来处理查看指定学科老师信息。

```
def show_teachers(request):
    """显示指定学科的老师"""
    try:
        sno = int(request.GET['sno'])
        subject = Subject.objects.get(no=sno)
        teachers = subject.teacher_set.all()
        return render(request, 'teachers.html', {'subject': subject, 'teachers':
teachers})
    except (KeyError, ValueError, Subject.DoesNotExist):
        return redirect('/')
```

显示老师信息的模板页。

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>老师</title>
    <style>/* 此处略去了层叠样式表的选择器 */</style>
</head>
<body>
    <h1>{{ subject.name }}学科老师信息</h1>
    <hr>
    {% if teachers %}
    {% for teacher in teachers %}
    <div>
        <div>
            <img src="{% static teacher.photo %}" alt="">
        </div>
        <div>
            <h3>{{ teacher.name }}</h3>
            <p>{{ teacher.detail }}</p>
            <p class="comment">
                <a href="">好评</a>
                (<span>{{ teacher.good_count }}</span>)
                <a href="">差评</a>
```

```
                (<span>{{ teacher.bad_count }}</span>)
            </p>
        </div>
    </div>
    {% endfor %}
    {% else %}
    <h3>暂时没有该学科的老师信息</h3>
    {% endif %}
    <p>
        <a href="/">返回首页</a>
    </p>
</body>
</html>
```

加载静态资源

在上面的模板页面中，我们使用了`<img>`标签来加载老师的照片，其中使用了引用静态资源的模板指令`{% static %}`，要使用该指令，首先要使用`{% load static %}`指令来加载静态资源，我们将这段代码放在了页码开始的位置。在上面的项目中，我们将静态资源置于名为static的文件夹中，在该文件夹下又创建了三个文件夹：css、js和images，分别用来保存外部层叠样式表、外部JavaScript文件和图片资源。为了能够找到保存静态资源的文件夹，我们还需要修改Django项目的配置文件settings.py，如下所示：

```
# 此处省略上面的代码

STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'), ]
STATIC_URL = '/static/'

# 此处省略下面的代码
```

接下来修改urls.py文件，配置用户请求的URL和视图函数的对应关系。

```
from django.contrib import admin
from django.urls import path

from vote import views

urlpatterns = [
    path('', views.show_subjects),
    path('teachers/', views.show_teachers),
    path('admin/', admin.site.urls),
]
```

启动服务器运行项目，进入首页查看学科信息。

所有学科

Python全栈+人工智能

Python是一种计算机程序设计语言。是一种面向对象的动态类型语言，最初被设计用于编写自动化脚本(shell)，随着版本的不断更新和语言新功能的添加，越来越多被用于独立的、大型项目的开发。

HTML大前端

万维网的核心语言、标准通用标记语言下的一个应用超文本标记语言（HTML）的第五次重大修改

JavaEE+大数据

a随着WEB和EJB容器概念诞生，使得软件应用业开始担心SUN的伙伴们是否还在Java平台上不断推出翻新的标准框架，致使软件应用业的业务核心组件架构无所适从，从一直以来是否需要EJB的讨论声中说明了这种彷徨。

软件测试

软件测试（英语：Software Testing），描述一种用来促进鉴定软件的正确性、完整性、安全性和质量的过程。换句话说，软件测试是一种实际输出与预期输出之间的审核或者比较过程。软件测试的经典定义是：在规定的条件下对程序进行操作，以发现程序错误，衡量软件质量，并对其是否能满足设计要求进行评估的过程。

点击学科查看老师信息。

Python全栈+人工智能学科老师信息

骆昊

10年以上软硬件产品设计、研发、架构和管理经验，2003年毕业于四川大学，四川大学Java技术俱乐部创始人，四川省优秀大学毕业生，在四川省网络通信技术重点实验室工作期间，参与了2项国家自然科学基金项目、1项中国科学院中长期研究项目和多项四川省科技攻关项目，在国际会议和国内顶级期刊上发表多篇论文（1篇被SCI收录，3篇被EI收录），大规模网络性能测量系统DMC-TS的设计者和开发者，perf-TTCN语言的发明者。国内最大程序员社区CSDN的博客专家，在Github上参与和维护了多个高质量开源项目，精通C/C++、Java、Python、R、Swift、JavaScript等编程语言，擅长OOAD、系统架构、算法设计、协议分析和网络测量，主持和参与过电子政务系统、KPI考核系统、P2P借贷平台等产品的研发，一直践行“用知识创造快乐”的教学理念，善于总结，乐于分享。

好评（ 108 ） 差评（ 10 ）

王海飞

5年以上Python开发经验，先后参与了O2O商城、CRM系统、CMS平台、ERP系统等项目的设计与研发，曾在全国最大最专业的汽车领域相关服务网站担任Python高级研发工程师、项目经理等职务，擅长基于Python、Java、PHP等开发语言的企业级应用开发，全程参与了多个企业级应用从需求到上线所涉及的各种工作，精通Django、Flask等框架，熟悉基于微服务的企业级项目开发，拥有丰富的项目实战经验。善于用浅显易懂的方式在课堂上传授知识点，在授课过程中经常穿插企业开发的实际案例并分析其中的重点和难点，通过这种互动性极强的

Ajax请求

接下来就可以实现“好评”和“差评”的功能了，很明显如果能够在不刷新页面的情况下实现这两个功能会带来更好的用户体验，因此我们考虑使用Ajax技术来实现“好评”和“差评”，Ajax技术我们在Web前端部分已经介绍过了，此处不再赘述。

首先修改项目的urls.py文件，为“好评”和“差评”功能映射对应的URL。

```
from django.contrib import admin
from django.urls import path

from vote import views

urlpatterns = [
    path('', views.show_subjects),
    path('teachers/', views.show_teachers),
    path('praise/', views.prise_or_criticize),
    path('criticize/', views.prise_or_criticize),
    path('admin/', admin.site.urls),
]
```

设计视图函数`praise_or_criticize`来支持“好评”和“差评”功能，该视图函数通过Django封装的JsonResponse类将字典序列化成JSON字符串作为返回给浏览器的响应内容。

```
def praise_or_criticize(request):
    """好评"""
    try:
        tno = int(request.GET['tno'])
        teacher = Teacher.objects.get(no=tno)
        if request.path.startswith('/praise'):
            teacher.good_count += 1
        else:
            teacher.bad_count += 1
        teacher.save()
        data = {'code': 200, 'hint': '操作成功'}
    except (KeyError, ValueError, Teacher.DoseNotExist):
        data = {'code': 404, 'hint': '操作失败'}
    return JsonResponse(data)
```

修改显示老师信息的模板页，引入jQuery库来实现事件处理、Ajax请求和DOM操作。

```
<!DOCTYPE html>
{% load static %}
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>老师</title>
    <style>/* 此处略去了层叠样式表的选择器 */</style>
</head>
<body>
    <h1>{{ subject.name }}学科老师信息</h1>
    <hr>
    {% if teachers %}
    {% for teacher in teachers %}
    <div class="teacher">
        <div class="photo">
            <img src="{% static teacher.photo %}" height="140" alt="">
        </div>
        <div class="info">
            <h3>{{ teacher.name }}</h3>
            <p>{{ teacher.detail }}</p>
            <p class="comment">
                <a href="/praise/?tno={{ teacher.no }}">好评</a>
                (<span>{{ teacher.good_count }}</span>)
```

```
                  
                <a href="/criticize/?tno={{ teacher.no }}">差评</a>
                (<span>{{ teacher.bad_count }}</span>)
            </p>
        </div>
    </div>
    {% endfor %}
    {% else %}
    <h3>暂时没有该学科的老师信息</h3>
    {% endif %}
    <p>
        <a href="/">返回首页</a>
    </p>
    <script src="https://cdn.bootcss.com/jquery/3.4.1/jquery.min.js"></script>
    <script>
        $(() => {
            $('.comment>a').on('click', (evt) => {
                evt.preventDefault()
                let anchor = $(evt.target)
                let url = anchor.attr('href')
                $.getJSON(url, (json) => {
                    if (json.code == 10001) {
                        let span = anchor.next()
                        span.text(parseInt(span.text()) + 1)
                    } else {
                        alert(json.hint)
                    }
                })
            })
        })
    </script>
</body>
</html>
```

小结

到此为止，这个投票项目的核心功能已然完成，在下面的章节中我们会要求用户必须登录才能投票，没有账号的用户可以通过注册功能注册一个账号。