中山大学数据科学与计算机学院

移动信息工程专业高级实训

教学班级	1508	专业	软件工程
学号	15352175	姓名	李其宁

实训题目

基于Django的慕课共享平台

开发工具：

Python3.x + Django2.0.1 + mysql

实训内容介绍：

Django 本小组慕课共享平台采用Django Web框架，实现基于网易云课堂为原型的慕课平台。 功能包括基本的视频播放，课程展示，个人用户信息以及个人反馈等功能。

个人负责部分：

- 课程列表展示
- 热门课程列表
- 课程详情展示
- 课程收藏功能
- 课程章节信息
- 课程视频播放
- 课程评论功能
- 讲师列表及排行榜
- 讲师行情页

课程相关urls.py

定义相关页面的url，urls均定义在urls.py中

```
url(r"^course/", include('courses.urls', namespace="course")),
url(r'^list/$', CourseListView.as_view(), name="list"),
re_path('course/(?P<course_id>\d+)/', CourseDetailView.as_view(),
name="course_detail"),
url(r'^info/(?P<course_id>\d+)/$', CourseInfoView.as_view(), name="course_info"),
re_path('comments/(?P<course_id>\d+)/', CommentsView.as_view(),
name="course_comments"),
```

课程列表展示view

在CourseListView中渲染course-list.html页面

```
class CourseListView(View):
    def get(self, request):
        return render(request, "course-list.html", { })
```

分页功能

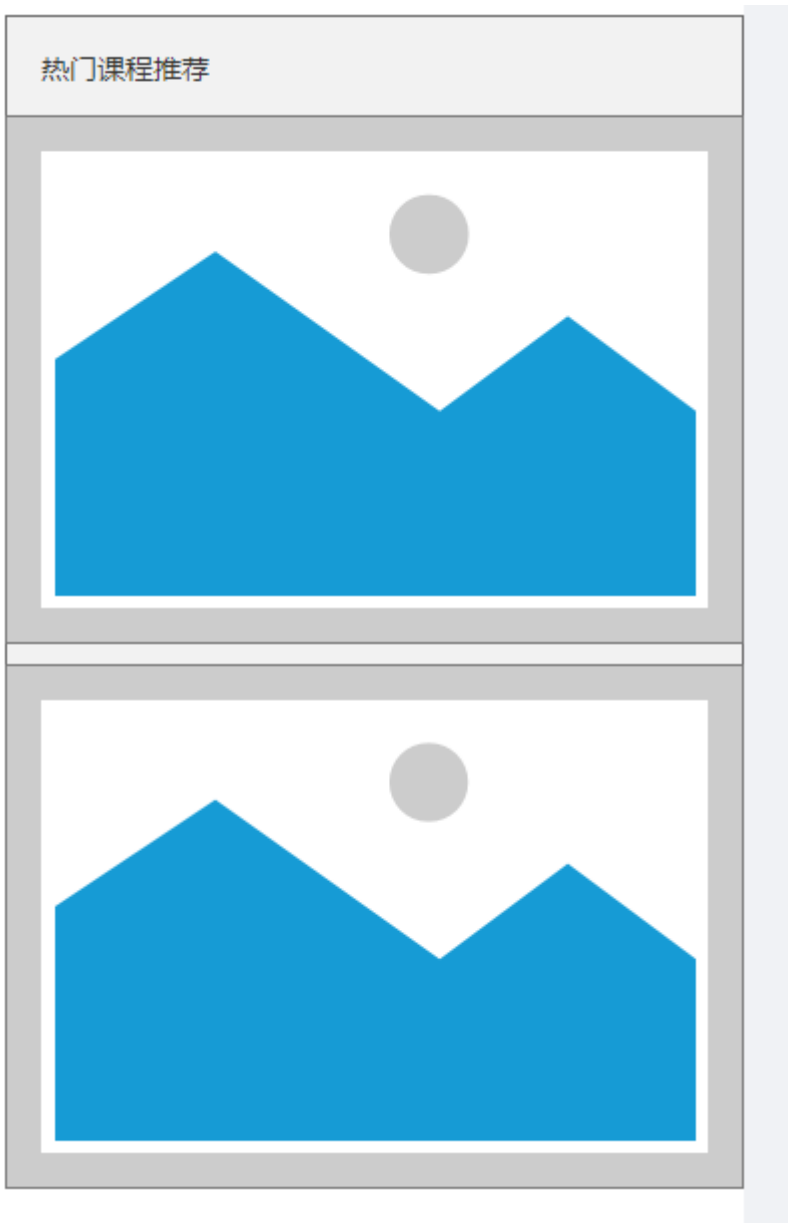
对课程列表分页，首先尝试获取前台get请求传递过来的page参数如果是不合法的配置参数默认返回第一页，然后从purepage对象取出五个列表中的对象，将其渲染至也买你course-list.html中

```
class CourseListView(View):
    def get(self, request):
        all_course = Course.objects.all()
        try:
            page = request.GET.get('page', 1)
        except PageNotAnInteger:
            page = 1
        p = Paginator(all_course,6 , request=request)
        courses = p.page(page)
        return render(request, "course-list.html", {
            "all_course":courses,
        })
```

针对热门课程排序

在分页之前，让分页处理所有筛选过的数据。

```
# 进行排序
        sort = request.GET.get('sort', "")
        if sort:
            if sort == "students":
                all_course = all_course.order_by("-students")
            elif sort == "hot":
                all_course = all_course.order_by("-click_nums")
```



课程详情页

实现方式：首先继承base页面，布局课程详情页面，效果如下：

首先是课程详情view的处理

```
class CourseDetailView(View):
    def get(self, request, course_id):
        return  render(request, "course-detail.html", {

        })
```

然后在列表展示页放入详情url，把对应的参数传进列表展示页中，进行数据填充。

```
course = Course.objects.get(id = int(course_id))

                return  render(request, "course-detail.html", {
            "course":course,
        })
```

课程收藏功能

在课程详情页，添加block js，写到页面底部

```
<script type="text/javascript">
function add_fav(current_elem, fav_id, fav_type){
```

```
    $.ajax({
        cache: false,
        type: "POST",
        url:"{% url "org:add_fav" %}",
        data:{'fav_id':fav_id, 'fav_type':fav_type},
        async: true,
        beforeSend:function(xhr, settings){
            xhr.setRequestHeader("X-CSRFToken", "{{ csrf_token }}");
        },
        success: function(data) {
            if(data.status == 'fail'){
                if(data.msg == 'User logout'){
                    window.location.href="/login/";
                }else{
                    alert(data.msg)
                }

            }else if(data.status == 'success'){
                current_elem.text(data.msg)
            }
        },
    });
}

$('#jsLeftBtn').on('click', function(){
    add_fav($(this), {{ course.id }}, 1);
});

$('#jsRightBtn').on('click', function(){
    add_fav($(this), {{ course.course_org.id }}, 2);
});


</script>
```

最后是添加收藏数据库操作

```
     has_fav_course = False
     has_fav_org = False

     if request.user.is_authenticated:
         if UserFavorite.objects.filter(user=request.user, fav_id=course.id,
fav_type=1):
             has_fav_course = True
         if UserFavorite.objects.filter(user=request.user, fav_id=course.course_org.id,
fav_type=2):
             has_fav_org = True
```

课程章节信息

章节信息View

```
class CourseInfoView(View):
    def get(self, request, course_id):
        course = Course.objects.get(id=int(course_id))
        return render(request, "course-video.html", {
            "course": course,
        })
```

章节视频信息

在video表中添加视频对应的url信息

```
url = models.CharField(max_length=200, default="http://blog.mtianyan.cn/"
,verbose_name=u"访问地址")
```

课程评论页面

配置课程评论的url和view

```
class CommentsView(View):
    def get(self, request, course_id):
        course = Course.objects.get(id=int(course_id))
        all_resources = CourseResource.objects.filter(course=course)
        return render(request, "course-comment.html", {
            "course": course,
            "all_resources": all_resources,
        })
```

对于评论功能，点击video可以跳转到相关评论页面。发表评论采用ajax技术，发布成功后就会刷新页面。添加评论在另一个页面添加，所有新建一个AddCommentView由于添加评论。

```
# ajax方式添加评论
class AddCommentsView(View):
    def post(self, request):
        if not request.user.is_authenticated:
            # 未登录时返回json提示未登录，跳转到登录页面是在ajax中做的
            return HttpResponse('{"status":"fail", "msg":"用户未登录"}',
content_type='application/json')
        course_id = request.POST.get("course_id", 0)
        comments = request.POST.get("comments", "")
        if int(course_id) > 0 and comments:
            course_comments = CourseComments()
            # get只能取出一条数据，如果有多条抛出异常。没有数据也抛异常
            # filter取一个列表出来，queryset。没有数据返回空的queryset不会抛异常
            course = Course.objects.get(id = int(course_id))
            # 外键存入要存入对象
            course_comments.course = course
            course_comments.comments = comments
            course_comments.user = request.user
            course_comments.save()
            return HttpResponse('{"status":"success", "msg":"评论成功"}',
content_type='application/json')
```

```
        else:
            return HttpResponse('{"status":"fail", "msg":"评论失败"}',
content_type='application/json')
```

相应的js代码为：

```
<script type="text/javascript">
    //添加评论
    $('#js-pl-submit').on('click', function(){
        var comments = $("#js-pl-textarea").val()
        if(comments == ""){
            alert("评论不能为空")
            return
        }
        $.ajax({
            cache: false,
            type: "POST",
            url:"{% url 'course:add_comment' %}",
            data:{'course_id':{{ course.id }}, 'comments':comments},
            async: true,
            beforeSend:function(xhr, settings){
                xhr.setRequestHeader("X-CSRFToken", "{{ csrf_token }}");
            },
            success: function(data) {
                if(data.status == 'fail'){
                    if(data.msg == '用户未登录'){
                        window.location.href="/login/";
                    }else{
                        alert(data.msg)
                    }

                }else if(data.status == 'success'){
                    window.location.reload();//刷新当前页面.
                }
            },
        });
    });

</script>
```

课程播放页面

对于课程视频的播放，采用开源库video.js，对应的视频播放view为：

```
# 播放视频的view
class VideoPlayView(LoginRequiredMixin, View):
    login_url = '/login/'
    redirect_field_name = 'next'
    def get(self, request, video_id):
        video = Video.objects.get(id=int(video_id))
        course = video.lesson.course
        user_courses = UserCourse.objects.filter(user=request.user, course=course)
```

```
        if not user_courses:
            user_course = UserCourse(user=request.user, course=course)
            user_course.save()
        all_resources = CourseResource.objects.filter(course=course)
        user_courses = UserCourse.objects.filter(course=course)
        user_ids = [user_course.user_id for user_course in user_courses]
        all_user_courses = UserCourse.objects.filter(user_id__in=user_ids)
        course_ids = [user_course.course_id for user_course in all_user_courses]
        relate_courses = Course.objects.filter(id__in=course_ids).order_by("-
click_nums").exclude(id=course.id)[:4]
        return render(request, "course-play.html", {
            "course": course,
            "all_resources": all_resources,
            "relate_courses": relate_courses,
            "video": video,
        })
```

讲师列表页展示

思路跟课程列表展示一样，首先需要统计有多少老师使用count进行统计，然后对讲师进行分页，尝试获取前台get请求传递过来的page参数，如果参数不合法那么默认返回第一页。取得前台参数之后，从allorg对象中取出五个列表对象，每页显示5个。

```
# 课程讲师列表页
class TeacherListView(View):
        def get(self, request):
            all_teacher = Teacher.objects.all()
            # 总共有多少老师使用count进行统计
            teacher_nums = all_teacher.count()
            # 对讲师进行分页
            # 尝试获取前台get请求传递过来的page参数
            # 如果是不合法的配置参数默认返回第一页
            try:
                page = request.GET.get('page', 1)
            except PageNotAnInteger:
                page = 1
            # 这里指从allorg中取五个出来，每页显示5个
            p = Paginator(all_teacher, 4, request=request)
            teachers = p.page(page)
            return render(request, "teachers-list.html", {
            "all_teacher":teachers,
            "teacher_nums":teacher_nums
            })
```

讲师排行榜

```
sort = request.GET.get("sort", "")
            if sort:
                if sort == "hot":
                    all_teacher = all_teacher.order_by("-click_nums")
```

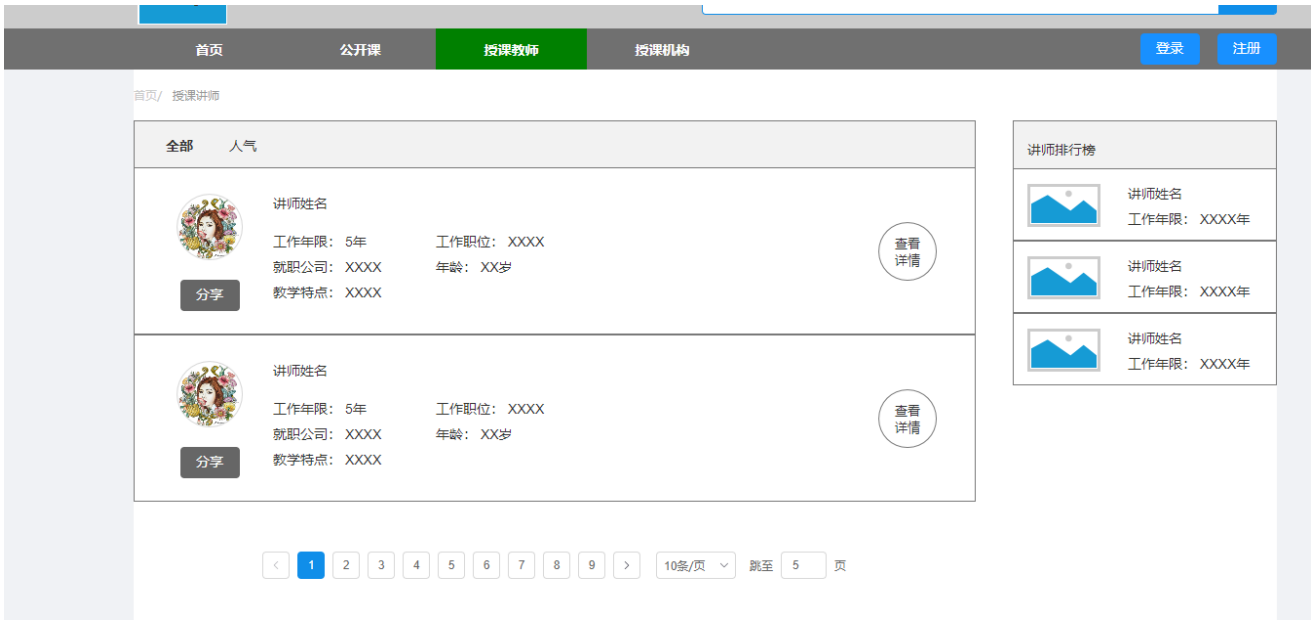
讲师详情页

讲师列表页中配置入口

```
class TeacherDetailView(View):
    def get(self, request, teacher_id):
        teacher = Teacher.objects.get(id = int(teacher_id))
        all_course = teacher.course_set.all()
        rank_teacher = Teacher.objects.all().order_by("-fav_nums")[:5]
        has_fav_teacher = False
        if UserFavorite.objects.filter(user=request.user, fav_type=3, fav_id=
teacher.id):
            has_fav_teacher = True
        has_fav_org = False
        if  UserFavorite.objects.filter(user=request.user, fav_type=2, fav_id=
teacher.org.id):
            has_fav_org = True
        return render(request, "teacher-detail.html", {
            "teacher":teacher,
            "all_course":all_course,
            "rank_teacher":rank_teacher,
```

```
            "has_fav_teacher":has_fav_teacher,
            "has_fav_org":has_fav_org,
        })
```

讲师详情效果：



实训感想

本慕课共享平台采用了Python Django框架，相比主流的SSM框架，它搭建一个网站更加快速方便。Django框架强制让开发人员使用MVC架构思想，这令整个网站各个模块松耦合。 Django前端样式采用面向对象的编程思想，对于具有相同结构的页面，使用类继承、多态、封装性质，以减少代码量和编程复杂度。此外，在样式源码中可以潜入python代码，使得前端样式更加灵活，减少了前端编程的复杂度。后端直接使用python代码对sql数据库操作，增删改均均以面向对象操作来替代mysql语法来对数据库进行操作。数据库设计在后端占比较重要的位置，良好的数据库设计在本次项目中均通过Django数据库映射功能完成数据库表创建和数据增删改，总的来说，Django框架替开发者完成了很多复杂的操作，提高了开发效率。

这次慕课共享网站的开发总体来说是比较顺利，其强大的功能，开源社区，前端样式模板和后台数据库映射为开发者提供了诸多便利。在实训过程中，由于本人负责课程相关模块的开发，属于慕课平台的一大类，所以总体来说是充分了解Django各个组件功能和它们之间的联系，提高了自己的网站开发能力和python语言开发能力。