小回顾

1.Django数据库操作

2.后台管理系统

3.业务逻辑--梳理

4.待实现功能

准备工作

step1.为嘉宾模型添加is_sgin字段，用于记录是否签到

is_sign = models.BooleanField(default=False)\

另外：修改发布会name使其唯一

step2 .同步模型数据库

manage.py makemigrations

manage.py migrate

第二步可能会抛出错误

```
django.db.utils.IntegrityError: UNIQUE constraint failed: new__sgin_event.name
```

原因是之前没有做唯一约束的时候，数据库里面发布会有同名的，所以操作数据库的时候出现了唯一约束失败

进入数据库，把同名的数据删除掉，再执行migrate命令。

表单请求处理

1.详情页增加签到表单，提交嘉宾手机号

知识点：表单基础

```
<form class="form-horizontal" action="/sgin/add_event" method="post"
enctype="application/x-www-form-urlencoded">
  <input type="text" class="form-control" placeholder="发布会名称" name="name">
  <button type="submit" class="btn btn-block btn-info" >提交</button>
</form>
```

三大必填属性

action：提交的URL

method: 请求方法，get或者post

enctype: 三种编码方式

application/x-www-form-urlencoded　#键值对方式

multipart/form-data　#二进制编码 上传文件会用到的方式

text/plain　#文本方式

2.后台逻辑判断

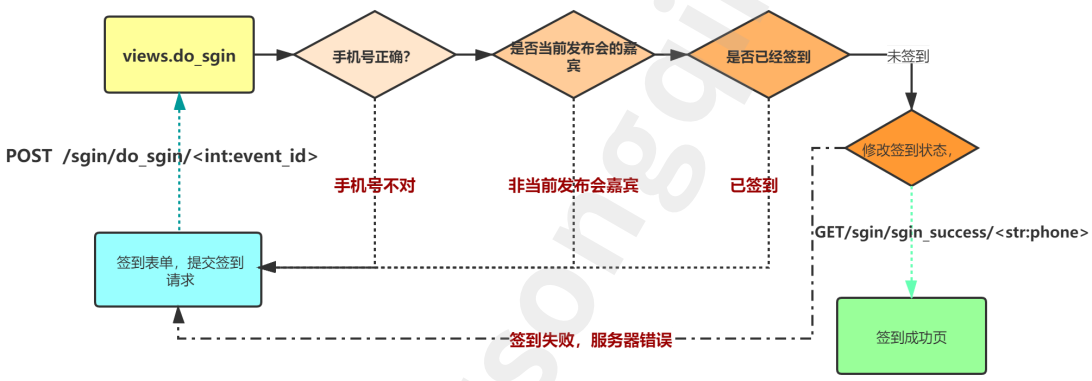
接收表单数据，判断手机号

1.判断是否为有效的手机号码

2.判断手机号对应的用户是否为当前发布会嘉宾

3.判断该嘉宾的签到状态

整个流程参照下图



知识点1. 判断请求方法 request.method　值是大写的POST 或GET PUT DELETE

知识点2. 取post表单数据　request.POST[key] 或request.POST.get() # request.POST返回的是一个字典

参考代码

```
def do_sgin(request,event_id):
    if request.method == 'POST':
        phone = request.POST['mobile']
        # 签到流程处理
```

Django安全策略处理

知识点3. django的csrf防护处理方式（3选1）

1.自废武功

注释django.middleware.csrf.CsrfViewMiddleware

2.开后门

在不需要校验csrf的视图加上装饰器

```
from django.views.decorators.csrf import csrf_exempt

@csrf_exempt
def do_sgin(request,event_id):
    pass
```

3.走正规手续

表单内部加入{% csrf_token %}

```
    <form method="post" action="/sgin/do_sgin/{{ event.id }}"
enctype="application/x-www-form-urlencoded">
      {% csrf_token %}
      <input type="text" name="mobile" required>

      <button type="submit">输入手机号签到</button>
    </form>
```

Tips: **csrf简介**

1. CSRF（Cross-site request forgery）跨站请求伪造，是一种常见的网络攻击手段，具体内容和含义请大家自行百度。

   2.Django默认为我们开启了防范CSRF攻击的机制。

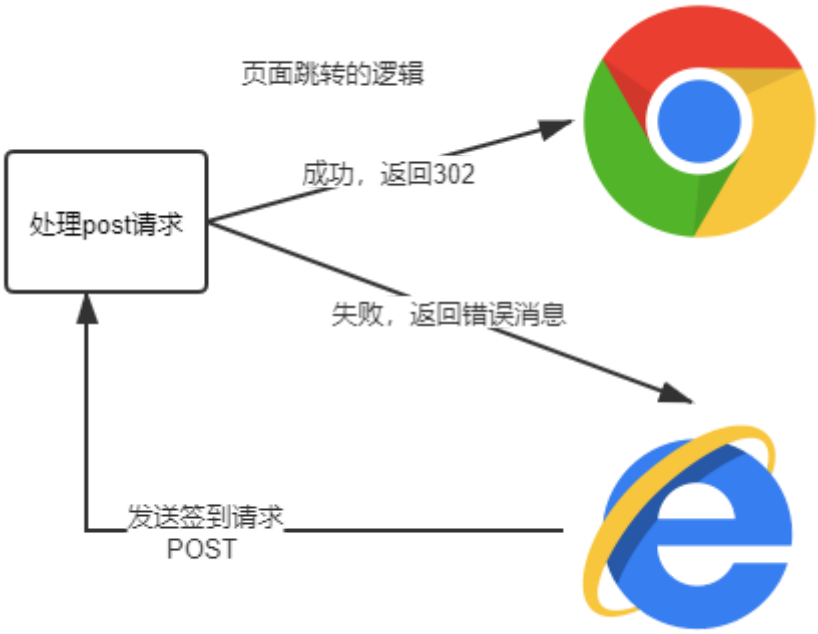
   3.对于GET请求，一般来说没有这个问题，通常是针对POST或PUT方法的

请求的转发

3.页面跳转

手机号正常，状态正常，且签到状态修改成功后，跳转页面

参考



这里我们采用重定向方式来进行页面跳转

```
redirect(url)    # url就是跳转的目标
```

实战练习：签到功能实现

签到视图参考代码v1：

```
def do_sgin(request,event_id):
    # 从post请求获取参数，首先判断请求方法
    if request.method == 'POST':
        phone = request.POST.get('phone')
        # 判断手机号是否正确
        res = Guest.objects.filter(phone=phone)
        if not res:
            return HttpResponse('手机号错误')
        guest = res[0]
        # 是否属于当前发布会
        if guest.event.id != event_id:
            return HttpResponse('非当前发布会嘉宾')
        # 是否已经签到
        if guest.is_sgin:
            return HttpResponse('已签到，不要重复签到')

        #进入签到
        guest.is_sgin=True
        guest.save()
        return HttpResponse('收到手机号： '+phone+'签到成功')
```

签到路由

```
#签到
    path('do_sgin/<int:event_id>',views.do_sgin)
```

签到表单页面---在原发布会详情页上修改

```
    <form method="post" action="/sgin/do_sgin/{{ event.id }}"
enctype="application/x-www-form-urlencoded">
      {% csrf_token %}
      <input type="text" name="mobile" required>

      <button type="submit">输入手机号签到</button>
    </form>
```

优化

当前签到结果返回的页面太简陋，需要优化。

1.制作专属签到成功页面

```
{% extends 'base.html' %}

{% block maintitle %}收到手机号: {{ phone }}， 签到成功! {% endblock %}
```

2.在发布会详情页，留下错误信息显示区

```
        <form method="post" action="/sgin/do_sgin/{{ event.id }}"
enctype="application/x-www-form-urlencoded">
            {% csrf_token %}
            <input type="text" name="phone" required>

            <button type="submit">输入手机号签到</button>
        </form>
        {% if error %}
            <span class="btn btn-danger">{{ error }}</span>
        {% endif %}
        <p><a href="/sgin/events" class="btn btn-info">返回列表</a></p>
```

3.修改视图对应的返回

```
def do_sgin(request,event_id):
    # 从post请求获取参数，首先判断请求方法
    if request.method == 'POST':
        phone = request.POST.get('phone')
        # 判断手机号是否正确
        res = Guest.objects.filter(phone=phone)
        if not res:
            return render(request,'event_detail.html',
{'event':Event.objects.get(pk=event_id),'error':'手机号错误'})
        guest = res[0]
        # 是否属于当前发布会
        if guest.event.id != event_id:
            return render(request,'event_detail.html',
{'event':guest.event,'error':'非当前发布会嘉宾'})
        # 是否已经签到
        if guest.is_sgin:
            return render(request,'event_detail.html',
{'event':guest.event,'error':'已签到，不要重复签到'})
```

```
        #进入签到
        guest.is_sgin=True
        guest.save()
        #return render(request,'sgin_success.html',{'phone':phone})
        return redirect(f'/sgin/sgin_success/{phone}')
```

实战练习：新增嘉宾功能

1.新增嘉宾视图创建

```
@csrf_exempt
def add_guest(request):
    if request.method =='POST':
    #姓名
        name = request.POST['name']
    #手机号
        phone = request.POST['phone']
    #邮箱
        email = request.POST['email']
    #关联发布会
        event_id = request.POST['event_id']
    #创建嘉宾
    try:
        guest =
Guest.objects.create(name=name,phone=phone,email=email,event_id=event_id)
    except Exception as e:
        return render(request,'guest_add.html',{'error': repr(e)}) #返回精简错误信
息
        #保存成功-跳转到嘉宾列表页
    return redirect('/sgin/guests/')
```

```
#添加嘉宾--路由
path('add_guest/',views.add_guest)
```

2.新增嘉宾表单页面guest_add.html

```
{% extends 'base.html' %}
{% block content %}
    <div class="panel panel-info">
        <div class="panel-heading"> 新增嘉宾 </div>
        <div class="panel-body">

            <form method="post" action="/sgin/add_guest/"
enctype="application/x-www-form-urlencoded">
                {% csrf_token %}
                <div class="form-group">
                    <label class="col-sm-2 control-label">名称</label>
                    <div class="col-sm-10">
                        <input class="form-control" type="text" name="name"
required>
```

```
                    </div>
                </div>

                <div class="form-group">
                    <label class="col-sm-2 control-label">手机号</label>
                    <div class="col-sm-10">
                        <input class="form-control" type="text" name="phone"
required>
                    </div>
                </div>
                <div class="form-group">
                    <label class="col-sm-2 control-label">邮箱</label>
                    <div class="col-sm-10">
                        <input class="form-control" type="text" name="email"
required>
                    </div>
                </div>

                <div class="form-group">
                    <label class="col-sm-2 control-label">发布会</label>
                    <div class="col-sm-10">
                        <select class="form-control" name="event_id">
                            {% for event in events %}
                                <option value={{ event.id }}>{{ event.name }}
</option>
                            {% endfor %}

                        </select>
                    </div>
                </div>

                <button class="btn btn-block btn-primary" type="submit">保存
</button>
            </form>
            {% if error %}
                <p class="danger"> {{ error }}</p>
            {% endif %}
            <p><a href="/sgin/guests" class="btn btn-info">返回列表</a></p>
        </div>
    </div>
{% endblock %}
```

3.嘉宾列表增加入口

```
{% extends 'base.html' %}
{% block maintitle %}<a href="/sgin/add_guest_page" class="btn btn-info navbar-
right">添加嘉宾</a>{% endblock %}
{% block content %}
    <ul class="list-group">
        {% for guest in guest_list %}
            <li class="list-group-item text-center"><a
href="/sgin/guest_detail/{{ guest.id }}">{{ guest.name }}</a></li>
        {% endfor %}
    </ul>

{% endblock %}
```

分页功能

分页的三要素

page_size: 每页显示的条数

page_num: 第几页

page_count: 总页数

Paginator对象

方法:

Paginator.page(number)

返回指定页面的Page对象，比如第7页的所有内容，从1开始。如果提供的页码不存在，抛出InvalidPage异常。

Paginator.get_page(number)

上面方法的安全版本，不会弹出异常。如果输入的参数不是数字，返回第一页。如果输入的数字大于最大页码，返回最后一页。

属性:

- Paginator.count：对象总数。
- Paginator.num_pages：页面总数。
- Paginator.page_range：基于1的页数范围迭代器。比如：[1, 2, 3, 4]

Page对象

Paginator.page(num)将返回一个Page对象，我们主要的操作都是基于Page对象的

方法:

- Page.has_next()：如果有下一页，则返回True。
- Page.has_previous()：如果有上一页，返回 True。
- Page.has_other_pages()：如果有上一页或下一页，返回True。
- Page.next_page_number()：返回下一页的页码。如果下一页不存在，抛出InvalidPage异常。
- Page.previous_page_number()：返回上一页的页码。如果上一页不存在，抛出InvalidPage异常。
- `Page.start_index()`：返回当前页上的第一个对象，相对于分页列表的所有对象的序号，从1开始计数。 比如，将五个对象的列表分为每页两个对象，第二页的`start_index()`会返回3。
- `Page.end_index()`:返回当前页上的最后一个对象，相对于分页列表的所有对象的序号，从1开始。 比如，将五个对象的列表分为每页两个对象，第二页的`end_index()`会返回4。

属性:

- Page.object_list:当前页上所有对象的列表。
- Page.number:当前页的序号，从1开始计数。
- Page.paginator：当前Page对象所属的Paginator对象。

使用方法

以嘉宾列表为例,修改嘉宾列表视图，使其返回分页数据

```
# 嘉宾列表
def guests(request):
    # 从数据库获取所有嘉宾
    guest_list = Guest.objects.filter()
    # 页面尺寸 - 每页显示的最大数量
    page_size=3
    # 定义分页器  Paginator(查询集的列表形式,页面尺寸)
    paginator = Paginator(list(guest_list),page_size)
    # 从请求中获取页码
    page_index = request.GET.get('page')
    # 根据页码 显示对应页的数据
    current_page = paginator.get_page(page_index)
    # 返回页面并填充数据
    return render(request,'guests.html',{'guest_list':current_page})
```

修改嘉宾模板页面，添加分页控件

v1简单版分页器--django自带效果

```
    <div class="dataTables_paginate paging_simple_numbers">
    <span class="step-links">
        {% if guest_list.has_previous %}
            <a href="?page=1">&laquo; first</a>
            <a href="?page={{ guest_list.previous_page_number }}">previous</a>
        {% endif %}

        <span class="current">
            Page {{ guest_list.number }} of {{ guest_list.paginator.num_pages
}}.
        </span>

        {% if guest_list.has_next %}
            <a href="?page={{ guest_list.next_page_number }}">next</a>
            <a href="?page={{ guest_list.paginator.num_pages }}">last &raquo;
</a>
        {% endif %}
    </span>
    </div>
```

v2-升级版分页器，采用模板提供样式

```
{% extends 'base.html' %}
{% block maintitle %}<a href="/sgin/add_guest_page" class="btn btn-info navbar-
right">添加嘉宾</a>{% endblock %}
{% block content %}
    <ul class="list-group">
        {% for guest in guest_list %}
            <li class="list-group-item text-center"><a
href="/sgin/guest_detail/{{ guest.id }}">{{ guest.name }}</a></li>
        {% endfor %}
    </ul>

{#    开始分页  #}
```

```
    <div class="dataTables_paginate paging_simple_numbers" id="dataTables-
example_paginate">
        <ul class="pagination">
            {% if guest_list.has_previous %}
                <li class="paginate_button previous " aria-controls="dataTables-
example" tabindex="{{ guest_list.number }}" id="dataTables-example_previous">
                    <a href="?page={{ guest_list.previous_page_number
}}">Previous</a>
                </li>
            {% else %}
                <li class="paginate_button previous disabled" aria-
controls="dataTables-example" tabindex="0" id="dataTables-example_previous">
                    <a href="#">Previous</a>
                </li>
            {% endif %}

            {% for page_num in guest_list.paginator.page_range %}
                {% if page_num == guest_list.number %}
                    <li class="paginate_button active " aria-
controls="dataTables-example" tabindex="0"><a href="#">{{ guest_list.number }}
</a></li>
                {% else %}
                    <li class="paginate_button " aria-controls="dataTables-
example" tabindex="0"><a href="?page={{ page_num }}">{{ page_num }}</a></li>
                {% endif %}
            {% endfor %}

            {% if guest_list.has_next %}
                <li class="paginate_button next" aria-controls="dataTables-
example" tabindex="0" id="dataTables-example_next"><a href="?page={{
guest_list.next_page_number }}">Next</a></li>
            {% else %}
                <li class="paginate_button next disabled" aria-
controls="dataTables-example" tabindex="0" id="dataTables-example_next"><a
href="#">Next</a></li>
            {% endif %}
        </ul>
    </div>
{% endblock %}
```