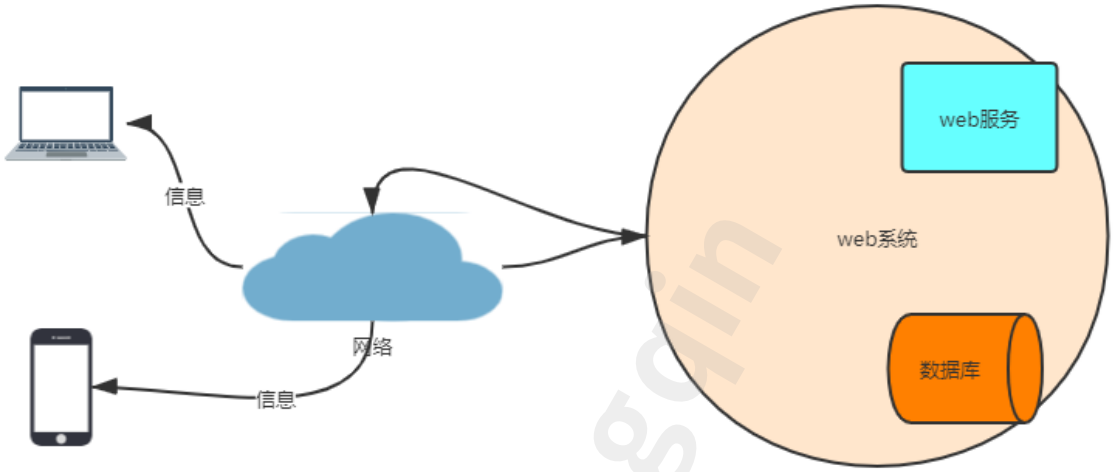
web开发世界观

web系统的基础逻辑---道

web系统本质

提供服务

提供接口或界面供调用



底层逻辑

请求与响应

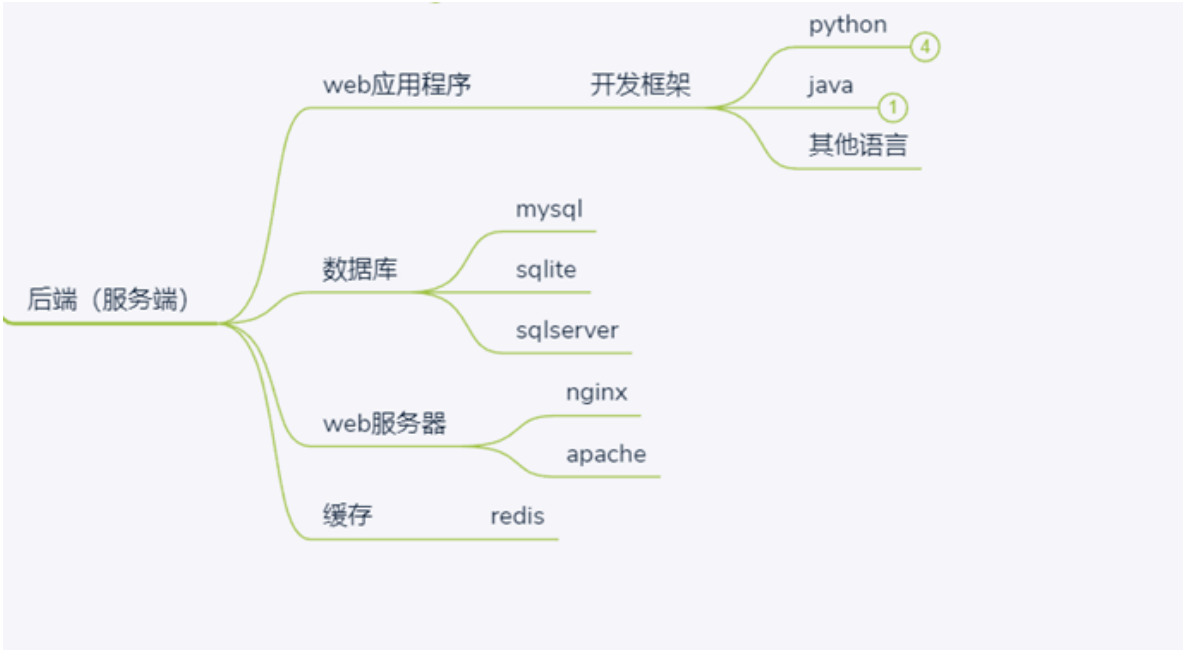
协议支撑

信息交换的标准

web开发的技术支持---术

前端

后端



以术窥道，而不止于术

通过具体的技术结合项目来逐步理解web开发，后续通过不断的练习加深对该技术的理解，定制化自己项目需要的系统。

课程项目介绍

签到系统---Django开发 前后端不分离

测试管理平台---- Django开发后端，前后端分离， 我们聚焦后端的开发

Django开发环境搭建

以Django作为开发框架作为大家的起点，基于大家拥有良好的python基础

开发环境搭建

1.虚拟环境+库 安装django

```
python -m venv myvenv
myvenv\Scripts\activate.bat
pip install Django
```

2.start project 项目创建

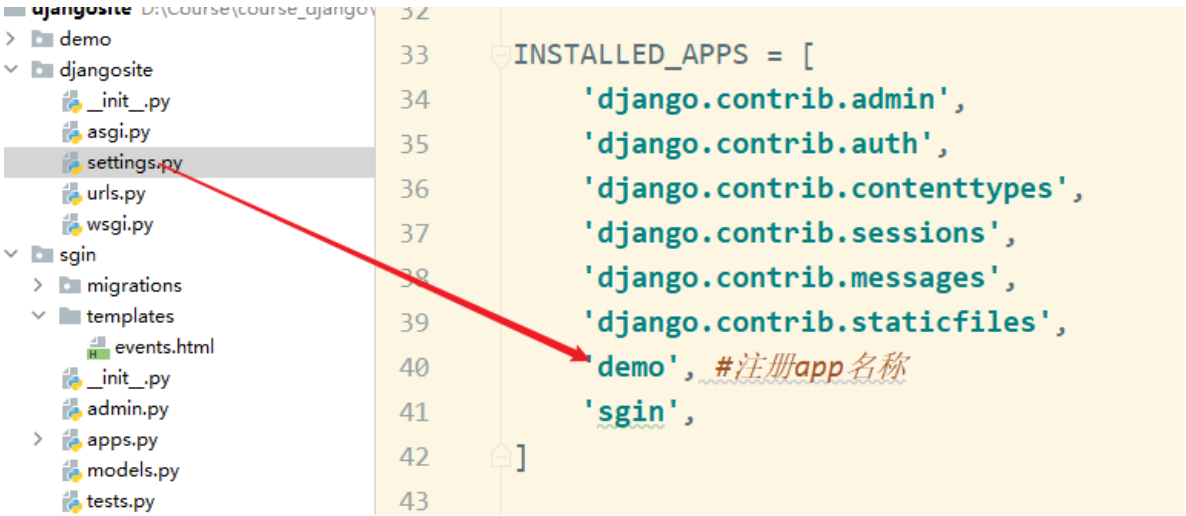
```
python -m django startproject djangosite
```

3.start app 应用程序创建

```
cd djangosite  （外面的djangosite）
python manage.py startapp demo
```

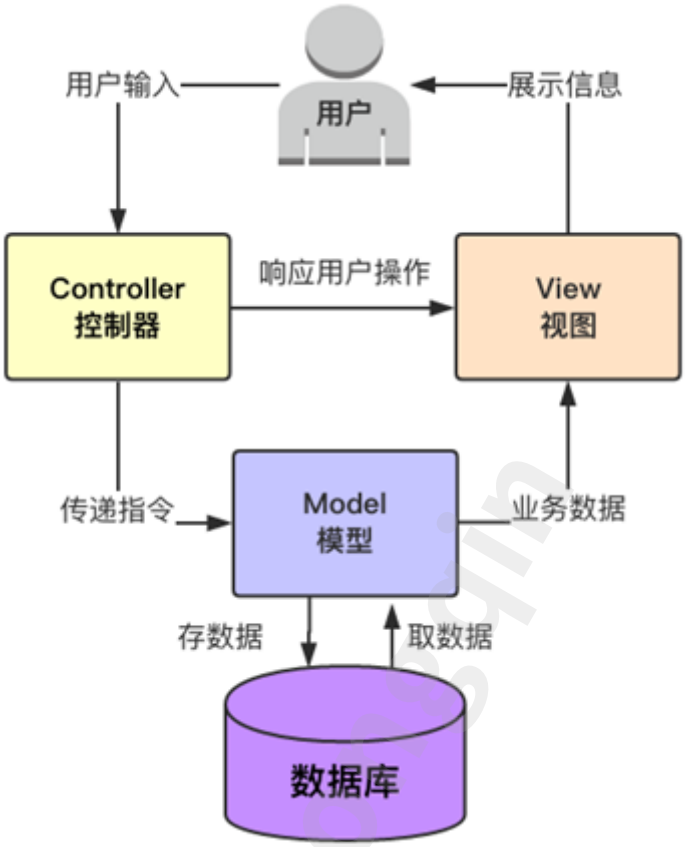
4.注册app

目的是被Django发现,以便可以扫描到数据模型和模板

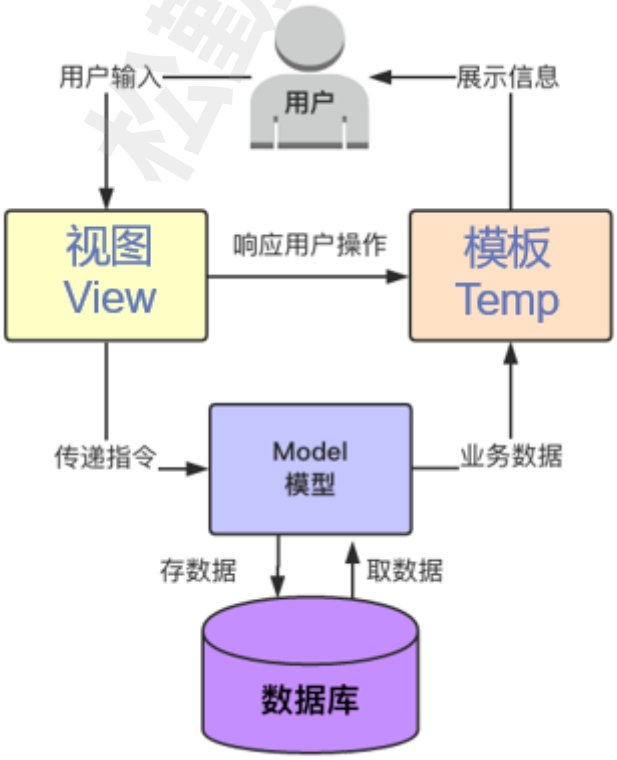


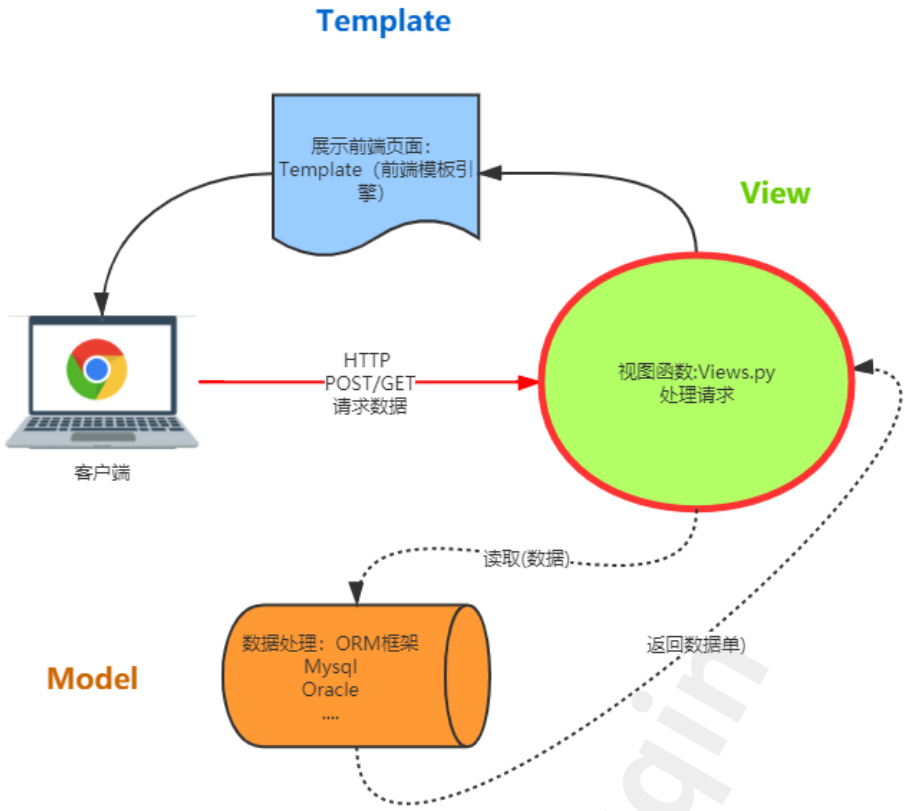
Djanog基本架构

传统MVC



django改进的MVC---MTV





Modle:和数据库交互，通过Model可以不用直接操作数据库，提高了代码编写效率，重点

View: 处理用户的请求，和返回数据响应，可以直达用户，重点

Template: django独特的网页渲染方式，将内容配合模板渲染成网页展现给用户，需要配合前端基本知识使用（html css js）如果前后端分离则用不到此模块 了解即可

Django视图

视图定义

处理http请求的部分，返回http响应

Django如何定义视图

```
# demo/views.py
#定义视图--用户能看到的内容
def index(request):
    return HttpResponse('冲击年薪40w')
```

如何访问该视图?

通过路由系统

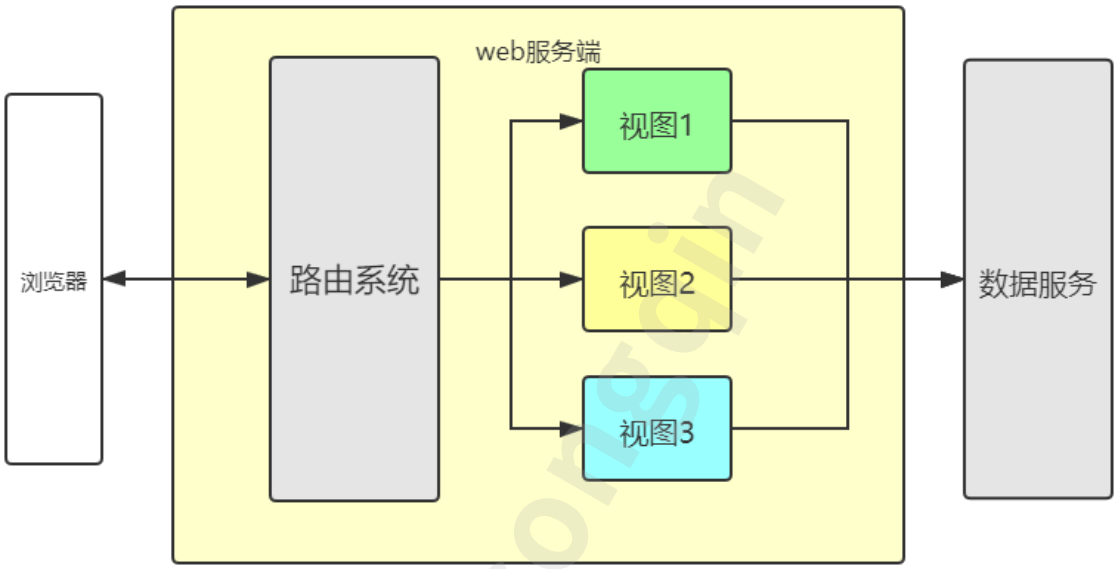
Django的路由系统

什么是路由？

请求的向导，如用户访问 http://host/path/foo/bar 被系统转到foo视图函数，处理后返回结果

路由原理

通过解析传递过来的url，来分配具体执行的视图函数。可以理解为视图的调度器。

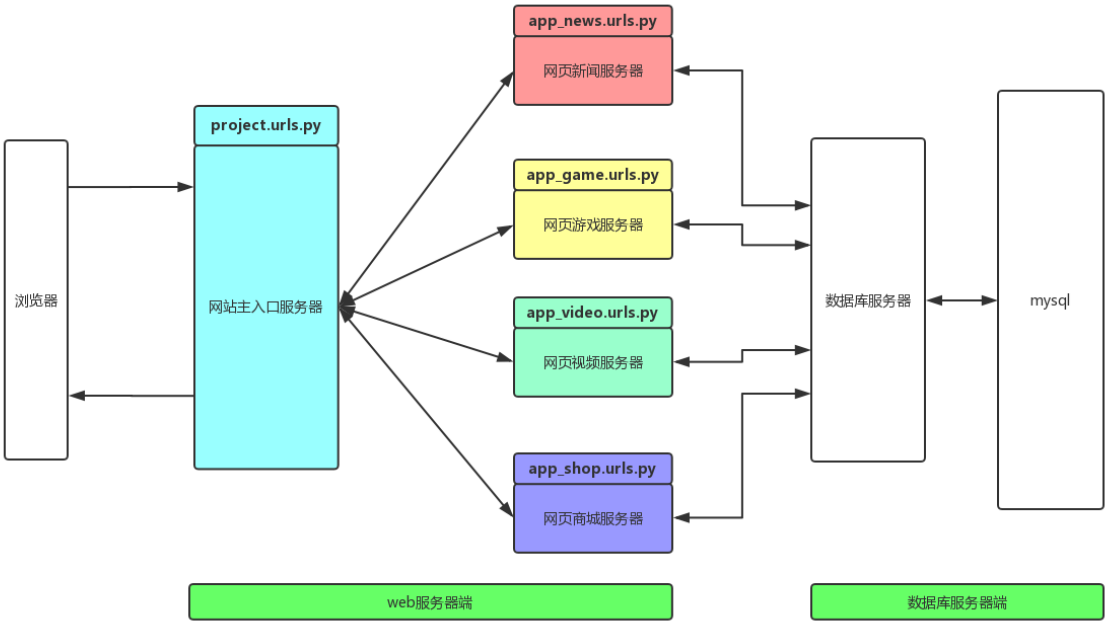


Django的路由定义

```
# djangosite/urls.py
from login import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/',views.index),
]
```

Django路由转发

当系统比较复杂时，一层路由会显得很臃肿，且后期不好维护，所以我们可以把路由层层分解，这个叫做路由的转发。



蓝色的是总路由，统一处理用户的请求，后面五颜六色的是分路由，分别处理总路由发过来的请求。这个就好比之前的小店老板1个人处理业务就够了，但是现在公司做大了，老板接到活以后要分配给下面的员工来处理。

路由的注意点

末尾/的问题

一般我们在定义路由的时候习惯末尾加/(斜杠)

如果定义了斜杠

那么浏览器访问的时候可以不带（会自动为你补全）

但是请求工具，比如代码request必须带（不会自动补全）

如果末尾没有定义/(斜杠)

那么浏览器访问的时候不带/（不会自动帮你删/）

同样请求工具，比如代码request也不能带/（不会自动补全也不会自动帮你删/）

路由与视图组合练习

松勤课程发布会签到系统 sq_course_event_sginsystem

创建应用程序 ---sgin

```
python -m django startapp sgin  # 或者django-admin startapp demo
```

定义URL

```
#djangosite/urls,py
from sgin import views
urlpatterns = [
    path('admin/', admin.site.urls),
    path('events/',views.events)   #列出发布会
]
```

发布会管理页面视图

```
#sgin/views.py
from django.http import HttpResponse
def events(request):
    return HttpResponse('测开课程发布会')
```

启动python开发服务器

```
python manage.py runserver 9090    #最后一个表示端口号，随意，只要不和当前系统程序冲突即可
```

访问http://127.0.0.1:9090/events/

看到返回：测开课程发布会

升级视图-增加发布会数量

```
def events(request):
    event_list=['测开课程发布会',
                '自动化发布会',
                '性能发布会',
                '安全发布会',
                '全栈发布会',
                'ISTQB',
                'TM项目管理',
                'PMP考证']
    return HttpResponse('|'.join(event_list))
```

视图再升级-加入html标签

```
def events(request):
    event_list=['测开课程发布会',
                '自动化发布会',
                '性能发布会',
                '安全发布会',
                '全栈发布会',
                'ISTQB',
                'TM项目管理',
                'PMP考证']
    #套入标签<li></li>中
    res=''.join([f'<li>{event}</li>' for event in event_list])
    return HttpResponse(res)
```

这时候发现在代码里面修改html很麻烦，且不好维护，这个时候我们需要将html的内容外包给模板系统来处理，他负责对外展示的样式。

Django模板--初识

模板的作用

发现页面太简陋了?

来看下演示的登录页面--HTML长什么样？---查看网页源码

虽然不是太复杂，但是相对我们这种极度简陋的HTML已经是很复杂了

要如何替换这段呢?

直接把HTML写在文本中呢？？？ 看下啊

代码太丑陋了，显然不是年薪40W该有的风格~

正确的做法应该是用HMTL文件来保存待显示的内容，然后我们代码直接返回HTML文件即可

Django早就帮你想到了这点

我们只需利用Django的模板机制就可以实现

用法--template

首先在应用程序Login的目录下新建一个templates的目录 目录名必须相同

然后新建HTML文件，里面塞入如下内容

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>发布会</title>
</head>
<body>
<h1>测试开发发布会</h1>
</body>
</html>
```

模板实践---修改发布会视图返回，改为返回html文件

```
return render(request,'events.html')
```

刷新页面，展示的内容

将视图内容（发布会信息）填充到页面中

第一个模板技术-变量

作用：

将视图函数的变量返回到模板中，django将其渲染

用法：

视图返回变量：{'模板中用到的变量名':变量}

```
return render(request,'events.html',{'events':res})
```

模板使用变量：{{变量名}}

```
<body>
{{ events }}   #标签中加入
</body>
```

刷新页面

测开课程发布会
自动化发布会
性能发布会
安全发布会
全栈发布会
ISTQB
TM项目管理
PMP考证

发现变量自带的标签没有被渲染，对的，默认情况下django返回的变量中若带html标签是不会被渲染的，当然如果你本就不想渲染html标签，或者变量内容没有Html标签那么可以忽略，但是如果想的话，请看下面：

第二个模板技术-for循环控制器

作用：

将列表类型的变量挨个展现出来

用法：

视图返回变量列表：

修改下原视图函数的返回，之前是字符串，现在改为列表并且去掉标签

```
#套入标签<li></li>中
    res=[event for event in event_list]

    return render(request,'events.html',{'events':res},)
```

模板使用for:

```
<ul>
{% for event in events %}
    <li>{{ event }}</li>
{% endfor %}
</ul>
```

发现没有？语法和 python几乎一样！不要忘记For循环结束要加{% endfor %}，因为Html里是不认你的缩进的。。

刷新页面，边正常了

第三个模板技术-if控制器

作用：

条件控制，可以根据条件选择渲染哪些元素

用法：

视图返回变量同上

模板使用变量:

```
{% if 'ISTQB' in events %}
    <h3>8个课程发布会</h3>
{% endif %}
```

注意：在标签内使用变量不用再加{{}}了，否则会报错

再换一个条件

```
{% if events.length == 9 %}
    <h3>9个课程发布会</h3>
{% endif %}
```

发现没有生效，原因是在模板中.length这种列表自带的属性失效了，需要使用模板自带的过滤器来实现

模板技术4-过滤器的用法（选修）

作用：转换变量和标签参数的值

写法：

```
{{变量|过滤器}}
```

上面的案例可以改成

```
{% if events|length == 9 %}
    <h3>9个课程发布会</h3>
{% endif %}
```

刷新页面发现可以了，内置的模拟器参考：：https://docs.djangoproject.com/zh-hans/3.1/ref/templates/builtins/#ref-templates-builtins-tags

Tips:当哪天你发现内置的模拟器也不能满足需求了，你可以自定义过滤器，不过如果你不专注前端开发可能很少用到了，参考：https://docs.djangoproject.com/zh-hans/3.1/howto/custom-template-tags/#writing-custom-template-filters

页面的美化-更新模板加入静态文件

现在的页面，开发练习练习还可以，但是作为产品展示就显得简陋了，因此需要升级界面。

但是从头写一套页面会花费我们大量的精力，而且工作中，这部分属于前端的战场，所以我们只需要专注后端即可。

现在我们从网上下载了现成的漂亮模板，并且稍作一些改动去掉多余的元素和引入文件，做些定制化修改

temp_v1.html

```
<!DOCTYPE html>
<html lang="cn">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta content="haiwen" name="author" />
    <!-- Bootstrap Styles-->
    <link href="assets/css/bootstrap.css " rel="stylesheet" />
    <!-- FontAwesome Styles-->
    <link href="assets/css/font-awesome.css" rel="stylesheet" />
    <!-- Custom Styles-->
    <link href="assets/css/custom-styles.css" rel="stylesheet" />
    <title></title>
</head>

<body>
<div id="wrapper">
    <nav class="navbar navbar-default top-navbar" role="navigation">
        <div class="navbar-header">
            <a class="navbar-brand"><i class="icon fa fa-plane"></i> 发布会签到系统</a>

            <div id="sideNav" >
                <i class="fa fa-bars icon"></i>
            </div>
        </div>

        <ul class="nav navbar-top-links navbar-right">
            <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown" href="#" aria-expanded="false">
                    <i class="fa fa-user fa-fw"></i> <i class="fa fa-caret-down"></i>
                </a>
                <ul class="dropdown-menu dropdown-user">
                    <li><a href="#"><i class="fa fa-user fa-fw"></i> User Profile</a>
                    </li>
                    <li><a href="#"><i class="fa fa-gear fa-fw"></i> Settings</a>
                    </li>
                    <li class="divider"></li>
                    <li><a href="#"><i class="fa fa-sign-out fa-fw"></i> Logout</a>
                    </li>
                </ul>
                <!-- /.dropdown-user -->
            </li>
        </ul>
    </nav>
    <!--/. NAV TOP  -->
```

```
    <nav class="navbar-default navbar-side" role="navigation">
        <div class="sidebar-collapse">
            <ul class="nav" id="main-menu">
                <li>
                    <a class="active-menu" href="/sgin/events"><i class="fa fa-
dashboard"></i> 发布会</a>
                </li>
                <li>
                    <a href="/sgin/guests"><i class="fa fa-desktop"></i> 嘉宾</a>
                </li>

            </ul>

        </div>

    </nav>
    <!-- /. NAV SIDE  -->

    <div id="page-wrapper">
        <div class="header">
            <div class="page-header">

            </div>

        </div>
        <div id="page-inner" class="panel-body">


            <footer><p>Author:haiwen.  <a
href="https://ke.qq.com/course/3135766" target="_blank">松勤测试开发课程</a></p>
            </footer>
        </div>
        <!-- /. PAGE INNER  -->

    </div>
    <!-- /. PAGE WRAPPER  -->
</div>


<!-- jQuery Js -->
<script src="assets/js/jquery-1.10.2.js"></script>
<!-- Bootstrap Js -->
<script src="assets/js/bootstrap.min.js"></script>

<!-- Custom Js -->
<script src="assets/js/custom-scripts.js"></script>
<script>
  $(document).ready(
    $('#main-menu>li>a').each(function (){
      $(this).attr('class',''); //先取消选中
      let current_href = window.location.pathname
      if(current_href === $(this).attr('href')){
        $(this).attr('class','active-menu');
      }
    }),
  )
</script>
```

```
</body>

</html>
```

但是直接打开是没有样式的，原因是我们还没有引入静态文件-css和js

加入静态文件

将提供的静态文件按照如上目录结构存放。

由于我们要使用django的模板系统来渲染页面，所以引入静态文件需要遵循django的规则

```
{% load static %}    #开头加上该标签
<!DOCTYPE html>
<html lang="cn">
```

同时修改外部文件的引入格式

原来的引入方式

```
<link href="assets/css/bootstrap.css " rel="stylesheet" />   #css

<script src="assets/js/jquery-1.10.2.js"></script>  #js
```

修改后

```
<link href="{% static 'assets/css/bootstrap.css' %} " rel="stylesheet" /> #css

<script src="{% static 'assets/js/jquery-1.10.2.js' %}"></script>   #js
```

依次类推，修改模板文件

temp_v2.html

```
{% load static %}
<!DOCTYPE html>
<html lang="cn">

<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta content="haiwen" name="author" />
    <!-- Bootstrap Styles-->
    <link href="{% static 'assets/css/bootstrap.css' %} " rel="stylesheet" />
    <!-- FontAwesome Styles-->
```

```
    <link href="{% static 'assets/css/font-awesome.css' %}" rel="stylesheet" />
    <!-- Custom Styles-->
    <link href="{% static 'assets/css/custom-styles.css' %}" rel="stylesheet" />
    <title>{% block title%}{% endblock %}</title>
</head>

<body>
<div id="wrapper">
    <nav class="navbar navbar-default top-navbar" role="navigation">
        <div class="navbar-header">
            <a class="navbar-brand"><i class="icon fa fa-plane"></i> 发布会签到系
统</a>

            <div id="sideNav" >
                <i class="fa fa-bars icon"></i>
            </div>
        </div>

        <ul class="nav navbar-top-links navbar-right">
            <li class="dropdown">
                <a class="dropdown-toggle" data-toggle="dropdown" href="#" aria-
expanded="false">
                    <i class="fa fa-user fa-fw"></i> <i class="fa fa-caret-
down"></i>
                </a>
                <ul class="dropdown-menu dropdown-user">
                    <li><a href="#"><i class="fa fa-user fa-fw"></i> User
Profile</a>
                    </li>
                    <li><a href="#"><i class="fa fa-gear fa-fw"></i>
Settings</a>
                    </li>
                    <li class="divider"></li>
                    <li><a href="#"><i class="fa fa-sign-out fa-fw"></i>
Logout</a>
                    </li>
                </ul>
                <!-- /.dropdown-user -->
            </li>
        </ul>
    </nav>
    <!--/. NAV TOP  -->
    <nav class="navbar-default navbar-side" role="navigation">
        <div class="sidebar-collapse">
            <ul class="nav" id="main-menu">
                <li>
                    <a class="active-menu" href="/sgin/events"><i class="fa fa-
dashboard"></i> 发布会</a>
                </li>
                <li>
                    <a href="/sgin/guests"><i class="fa fa-desktop"></i> 嘉宾</a>
                </li>

            </ul>

        </div>

    </nav>
```

```
    <!-- /. NAV SIDE  -->

    <div id="page-wrapper">
        <div class="header">
            <div class="page-header">
                {% block maintitle %}{% endblock %} <small></small>

            </div>

        </div>
        <div id="page-inner" class="panel-body">

            {% block content %}

            {% endblock %}

            <footer><p>Author:haiwen.  <a
href="https://ke.qq.com/course/3135766" target="_blank">松勤测试开发课程</a></p>
            </footer>
        </div>
        <!-- /. PAGE INNER  -->

    </div>
    <!-- /. PAGE WRAPPER  -->
</div>


<!-- jQuery Js -->
<script src="{% static 'assets/js/jquery-1.10.2.js' %}"></script>
<!-- Bootstrap Js -->
<script src="{% static 'assets/js/bootstrap.min.js' %}"></script>

<!-- Custom Js -->
<script src="{% static 'assets/js/custom-scripts.js' %}"></script>
<script>
  $(document).ready(
    $('#main-menu>li>a').each(function (){
      $(this).attr('class',''); //先取消选中
      let current_href = window.location.pathname
      if(current_href === $(this).attr('href')){
        $(this).attr('class','active-menu');
      }
    }),
  )
</script>

</body>

</html>
```

拷贝到events.html中，然后重启服务，再刷新页面

填充模板页面

将发布会信息显示到页面上

```
{% for event in events %}
    <li>{{ event }}</li>
{% endfor %}
```

发现不好看，再选择性的加入一些样式

```
<ul class="list-group">
    {% for event in events %}
        <li class="list-group-item text-center">{{ event }}</li>
    {% endfor %}
</ul>
```

> Tips: class="list-group" class="list-group-item text-center" 表示使用该css文件时，class对应的样式，bootstrap为我们定义了现成的样式，只需要引入类即可，不用从头写css了

完美呈现

继续页面的开发

u展示发布会具体信息：名称，时间，地点 等

u从发布会管理页面链接到详情页

u增加返回

此时发现一个问题，两个页面的基本样式都差不多---测边栏和页首菜单，只是中间区域不同

如果只有两个页面还好，多个页面就会出现难以维护的情况，因为可能你改动一个样式就要改N个文件，非常不方便!

所以以我们丰富的编程思路该如何解决问题?

对了，定义一个相对不变的公共模板，其他页面继承这个模板就可以了，模板留出接口，继承的页面只需要改动这个接口就可以。

模板的继承

细心的你一定发现了我们改动的最后一版html文件中有大量

```
{% block title%}{% endblock %}
```

类似这种{% block 标签%}{% endblock %}格式的标签

这个就是模板提供的接口，预留了空间让继承的子页面用于改动的

我们要做的就是继承这个模板页面，改动需要改动的即可

定义基础模板base.html

将temp_v2.html的文件内容拷贝进base.html　表示根模板的意思

子页面event.html改写

```
{% extends "sgin/base.html" %}
{% block content %}
            <ul class="list-group">
                {% for event in events %}
                    <li class="list-group-item text-center">{{ event }}</li>
                {% endfor %}
            </ul>
{% endblock %}
```

开头的{% extends "sgin/base.html" %} 表示继承sgin/base.html 页面

{% block content %}{% endblock %} 这之间的内容就填写子页面特有的个性内容。

刷新页面，依旧美丽，但是我们的Html文件简洁了很多，有木有！

依次类推，我们可以在模板想要改动的地方留出接口，子页面继承的时候可以改动这些接口，这里并不是所有的接口都要子页面填充，你可以只填充你想填充的地方。

原则就是，模板提供的接口，子页面可以填充，没有提供的接口，自然就填充不了哦

依次类推，直接继承父模板，定义子页面

event_detail.html

```
{% extends "base.html" %}
{% block title%}发布会详情{% endblock %}
{% block content %}
    <h1>发布会详情</h1>
{% endblock %}
```

定义路由

```
path('event_detail',views.event_detail)  #发布会详情
```

定义视图

```
def event_detail(request):
    return render(request,'event_detail.html')
```

访问http://127.0.0.1:9090/event_detail

增加链接

events.html

#增加超链接元素

```
<li class="list-group-item text-center"><a href="/event_detail">{{ event }}</a>
</li>
```

全文

```
{% extends "base.html" %}
{% block content %}
    <ul class="list-group">
        {% for event in events %}
            <li class="list-group-item text-center"><a href="/event_detail">{{
event }}</a></li>
        {% endfor %}
    </ul>
{% endblock %}
```

增加返回

event_detail.html

```
<p><a href="/events" class="btn btn-info">返回发布会列表</a></p>
```

全文

```
{% extends "base.html" %}
{% block title%}发布会详情{% endblock %}
{% block content %}
    <h1>发布会详情</h1>
    <p><a href="/events" class="btn btn-info">返回发布会列表</a></p>
{% endblock %}
```


附录--知识点官方文档

路由：https://docs.djangoproject.com/zh-hans/3.1/topics/http/urls/

视图:https://docs.djangoproject.com/zh-hans/3.1/topics/http/views/

模板：https://docs.djangoproject.com/zh-hans/3.1/topics/templates/

内置模板标签和过滤器：https://docs.djangoproject.com/zh-hans/3.1/ref/templates/builtins/#ref-templates-builtins-tags

django通用术语：https://docs.djangoproject.com/zh-hans/3.1/glossary/