Spring 2022 CS307 Project2

Part 1-Our team

王宇航 SID 12012208 Lab5	**李玉金 SID 12012225 Lab5**
Tasks: Code	Tasks: paperWriting
contribution ratio：50%	50%

Part 2-API Specification

本部分主要介绍每个API的实现原理，对于源代码不做详细说明，需要的用户可查看源码文件.对于每个API的参数和返回值,文档均做出了详细说明.需要注意的是，参数均为网页请求（API1含其它参数），返回值均为网页，这会在非基础部分详细说明

2.0)importData(参数为网页请求，返回值为网页)

本API功能为一键导入初始数据

实现思路：先将初始数据文件通过pandas读入并转化为pandas的dataframe类型，通过pandas的库函数将表直接拷贝到数据库中。初始数据的四张表需放在django_test文件夹根目录下

2.1)APIs for manipulating the original data(参数为网页请求及其它，返回值为网页)

本API功能为对初始数据文件进行增删改查操作。

实现思路：这些需求通过pandas内置的库函数可以优雅地完成

函数说明：

参数名	参数含义
filename	需修改的文件名
funcType	操作类型：add,delete,update,check
addlist	若操作类型为add，则addlist为需要添加的行
deleteIndex deleteFlag deleteJudge	若操作类型为delete，则deleteIndex为需要删除的行的index，deletFlag为需要判断的属性，deleteJudge为判断的条件，仅支持==操作
updateFlag updateJudge updateCol updateData	若操作类型为update，且需按条件删除，则updateFlag为需要判断的属性，updateFlag为判断的条件.若不按条件更新，updateCol为要更新的列，updateFlag为更新后的数据
selectFlag selectJudge	若操作类型为select，且需按条件查询，则select表示需要判断的属性，select表示判断的条件

例子：我们想将enterprise.csv中的name和country这两个字段的值更新为‘Alcatele’和‘CH’，条件是id=146的行

```
1    access_sourceData('enterprise.csv', 'update', None, None, None, None, 'id', 146,
    ['name', 'country'],['ALcatele', 'CH'], None, None)
```

2.2-2.5) stockIn(参数为均网页请求，返回值均为网页)

API2-5的功能是实现进货，下订单，修改订单，删除订单的业务逻辑

实现思路：

1. 利用pandas读取文件，注意如何解析csv，tsv文件
2. 对于进货操作，更新数据库中的product表格，增加当前库存；更新sale_detail表格，增加对应商品的总进货成本，减少该商品的当前总收益
3. 对于下订单操作，将所有订单按照签订时间排序。顺序读取订单，每往数据库写入一条订单，都实时更新库存，同步更新orders表格和contract表格；同时更新sale_detail表格，增加销售量,销售额,利润

4. 对于更新订单操作，与placeOrder基本相同，需注意如销售量为0，需删除订单，无需删除合同；需要更新的可能不仅只有货物数
5. 对于删除订单操作，与updateOrder基本相同，需注意如销售量为0，需删除订单，无需删除合同

2.6-2.14)getInformation(API12与API13含其它参数，其余不含，返回值均为网页)

API6-13的功能是实现查询数据库数据的业务逻辑

实现思路：由于pandas可以支持原生的sql语句，故均采用sql语句查询

API12的其余参数表示需要查询的合同，API13的其余参数表示需要查询的订单，API14的其余参数表需要查询的合同号和订单号

Part3-Further enhancement to the usability of the APIs

3.1)Bill Module

本项目实现了一个简单的账单模块，对应数据库中的sale_detail表格

可以计算出每种类型的商品当前的销售量，销售额，成本，利润。每次进货，下订单，修改订单，删除订单时，该表格便会对应更新。（本项目默认为只要订单签订了便算卖出）

WHERE ORDER BY profit DESC

	model	volume	saleroom	cost	profit ▼ 1
1	MouseG8	797	789030	286888	502142
2	YubaR4	924	924000	426938	497062

如图，我们可以看到当前销售利润最高的商品类型

3.2)A mechanism to change order status according to time and date

本项目同时还实现了商品订单状态的自动更新机制，根据实际到货日期判断订单的状态

代码逻辑如下

```
###每次登录web端时，便会执行以下程序，利用python的datetime库快速完成日期比较
cur_time= datetime.date.today()
cur_time=np.datetime64(cur_time)
print('go!!!!!')
for i in range(len(order_fresh)):
        if order_fresh.iloc[i]['lodgement_d']<=cur_time:
            order_fresh.loc[i,'order_type']='Finished'
        else:
            order_fresh.loc[i,'order_type']='UnFinish'
```

同时，我们在下订单模块时，也完成了订单状态的判断逻辑，此处不再赘述

Part4-A real back-end server

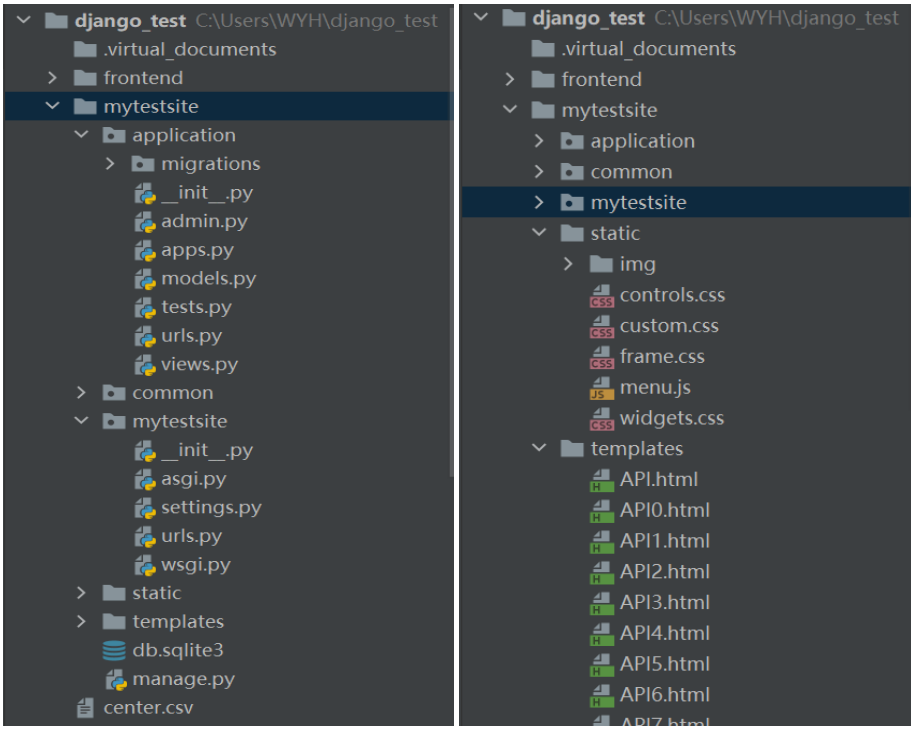
本项目采用了python的Django框架，实现了web后端并配置了数据库，采用css,js等前端技术美化了我们的前端html界面，支持用户在web端进行API的操作

下面详细说明我们采用Django框架实现后端服务器的流程

4.1）项目中主要的python包版本号(数据库为postgresql数据库，django默认为sqlite3，需做更改)

Django	django-db-connection-pool	numpy	pandas	SQLAlchemy
4.04	1.10	1.22.2	1.41	1.4.36

4.2）Django框架目录解析



如上图，左边表示的是Django框架后端的相关文件，右边表示的是Django框架前端的相关文件，下面做整体解析

django_test目录为根目录，其下的mytestsite目录为django文件目录，mytestsite下存在四个文件夹：分别是application，mytestsite，static，templates，以及框架的管理文件：manage.py

文件夹名	文件夹功能
mytestsite	项目的主框架. (此项目主要使用了setting.py与urls.py文件) settings.py表示项目的配置文件，例如选用的数据库，项目所包含的APP，均在此设置 urls.py表示路由设置文件，需要将各个app中view.py所用到的视图函数指定url路径
application	项目的功能模块.(此项目主要使用了views.py文件，由于并未将数据库中的表格映射成django支持的对象，故并未用到model.py文件) view.py表示视图函数，可接受用户的请求，解析请求后返回给用户对应的网页 urls.py表子路由文件，对应的父路由文件为mytestsite中的urls.py
static	包含项目网页所用到的图片，渲染网页所用到的一系列组件，脚本等，这些均为静态文件 须在mytestsite中的setting.py指定静态文件的路径
templates	项目的网页模板，包含了项目各功能对应的html模板,test.html为主页面，其余API.html由它跳转

4.3）视图函数详解

这里采用API13的例子，详细描述视图函数的参数与返回值

```
def API13(request):
    if request.method == 'POST':
        name = request.POST.get('name13')
        contract = api13(name)
        return render(request, 'API13.html', {'contract_num': contract[0], 'con_final':
        contract[1]})
```

如上图，所有API的参数均为request网页请求，返回值为对应的网页（若为静态网页，则无需返回字典，若为动态网页，网页内容参数由返回值中字典决定），本次项目所有API均采用该种设计，其余API不再赘述

4.4）如何与用户进行交互

在html模板中，我们可通过添加按钮组件，文本框组件的形式与用户进行交互

这里采用主页面test.html中API13作为例子，采用注释的方式说明

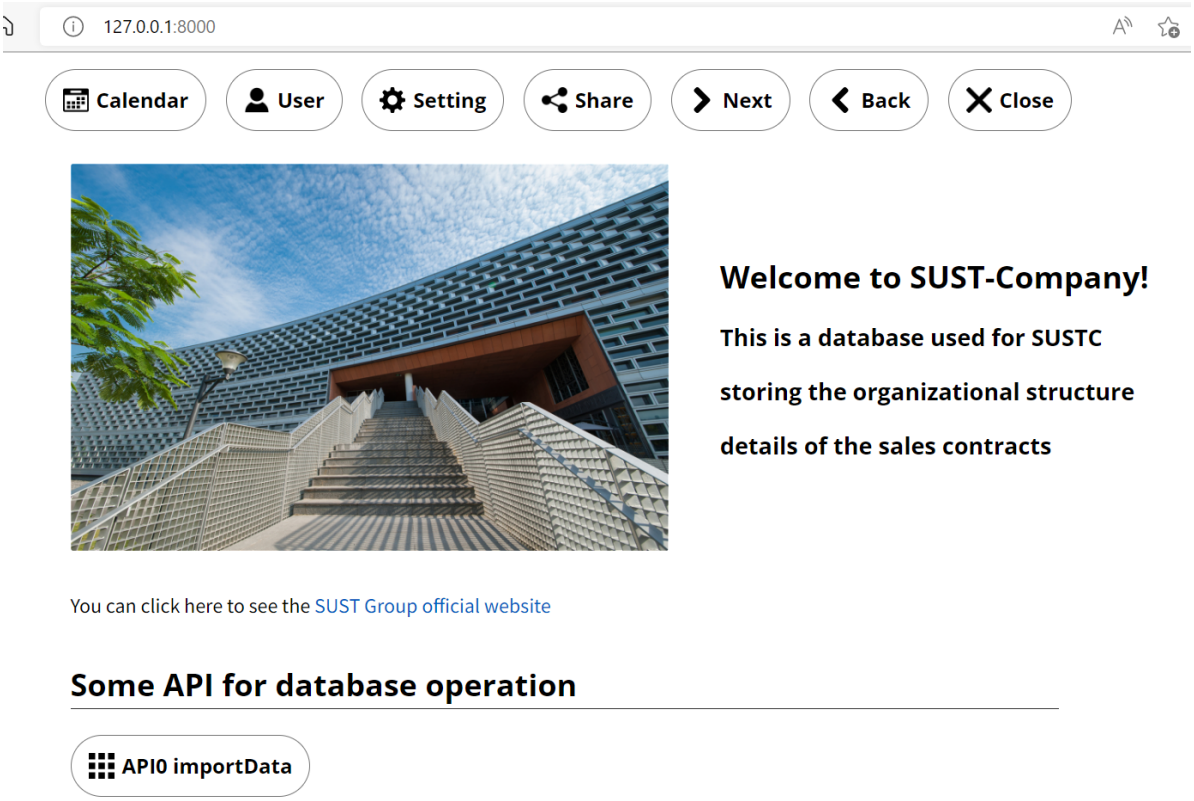
```
</div>
    <!-- 渲染组件，按钮名称为API13 -->
    <div class="control-group">
    <button class="custom-button-flat "><img src="../static/img/collection.png"><span>API13
    getContractInfo</span></button>
         <!-- 请求为post类型，对应网页url为API13 -->
        <form method="post" action="{% url 'API13' %}">
            <!--防止csrf攻击,发送客户端cookie中的token信息给服务器,若请求不携带token,则请求非法-->
            {% csrf_token %}
            <!-- 提供用户输入的文本框，定义提交按钮，跳转至API13. -->
              <input class="custom-textbox stretch-on-mobile" type="text" name="name13"
                placeholder="CSE0000219">
              <input type="submit"  value='commit' style='font-size:5px'
                onclick="window.location.href='../API13'"/>
```

```
15          </form>
16    </div>
```

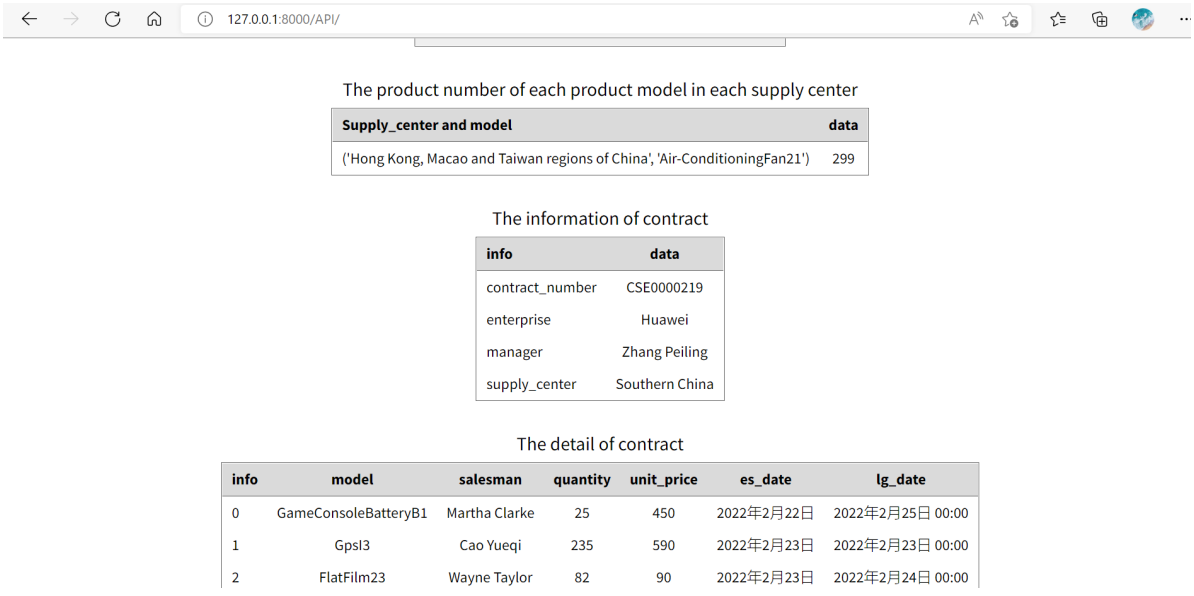
Part5-A usable and GUI

本次项目我们采用了CSS样式，JS脚本语言来美化我们的html模板，同时，我们对于查询的结果，用表格的形式给予显示，让数据更加直观

- 网站的首页：



- 用表格的形式展现数据，更加直观清晰



Part6-Database connection pools

在频繁创建、清除数据库连接的情况下，对数据库的资源的消耗无疑是巨大的。解决方案是在项目新建时同时创建一个数据库连接池，并创建指定数目的数据库连接。之后将之前创建数据库连接的操作换成从连接池中获取一个连接；将清除连接的操作换成归连接到连接池。从而降低了数据库资源的消耗。

由于django原生是不支持数据库连接池，因此需借助第三方库来配置数据库连接池，本次Porject使用的库是django-db-connection-pool，可通过如下命令安装

```
pip install django-db-connection-pool
```

然后在Django的设置文件中，修改数据库对应参数即可

```
DATABASES = {
        # 'ENGINE': 'django.db.backends.postgresql',
        'ENGINE': 'dj_db_conn_pool.backends.postgresql',
        'POOL_OPTIONS': {
            'POOL_SIZE': 100,
            'MAX_OVERFLOW': 50
            #最大连接数量为POOL_SIZE+MAX_OVERFLOW=150
        }
```

Part7-Database index

为数据库中的表格相应的字段创建索引，有助于加速其查找速度，相应的，作为加速查找的代价，表格的插入，更新，删除速度会变慢，于是我们在统计了所有API对表格的增删改查频率后，为以下表格创建了对应的索引

```
create index product_index on product using btree(model);
create index staff_index on staff using btree(type);
create index order_index on orders using btree(contract_number,salemans_id);
create index sale_detail_index on sale_detail using btree(volume);
create index contract_index on contract using  btree(c_number);
```

Part8-Preliminary exploration of defending against CSRF attack

在我们制作网页html模板时，如果需要获得用户输入，我们使用文本框接收的方法。但是当我们点击提交时，网站却提示了403Forbidden的错误信息，经过我们的排查，最终发现这是Django防范CSRF攻击的一种机制，我们需要在对应组件的代码中插入这样一行，才能通过Django的验证

```
{% csrf_token %}
```

于是，我们对CSRF攻击的攻击方式与防范措施做了一个初步的探究

8.1）什么是CSRF攻击?

CSRF的全称是跨站点请求伪造(Cross—Site Request Forgery)，我们可以这样理解：攻击者盗用了你的身份，以你的名义发送恶意请求，但是对服务器来说这个请求是完全合法的。比如以你的名义发送邮件，盗取你的账号，甚至转账等等，具有很大的危害性

8.2） CSRF攻击的机制?

首先我们需要明白什么是cookies，简单来说，cookies就是服务器存放在客户端的有关用户的信息，用于识别用户。用户每次访问网站都会携带用户客户端的cookies，服务器可以根据该信息判断用户是否有权限进行某些操作。若用户权限不够，则请求将会被服务器拦截。CSRF攻击是在攻击者无权限的情况下，利用他人的cookies信息来完成权限验证，使攻击者能够进行非法操作，而被攻击者对此并不知情。

8.3）如何抵御CSRF攻击?

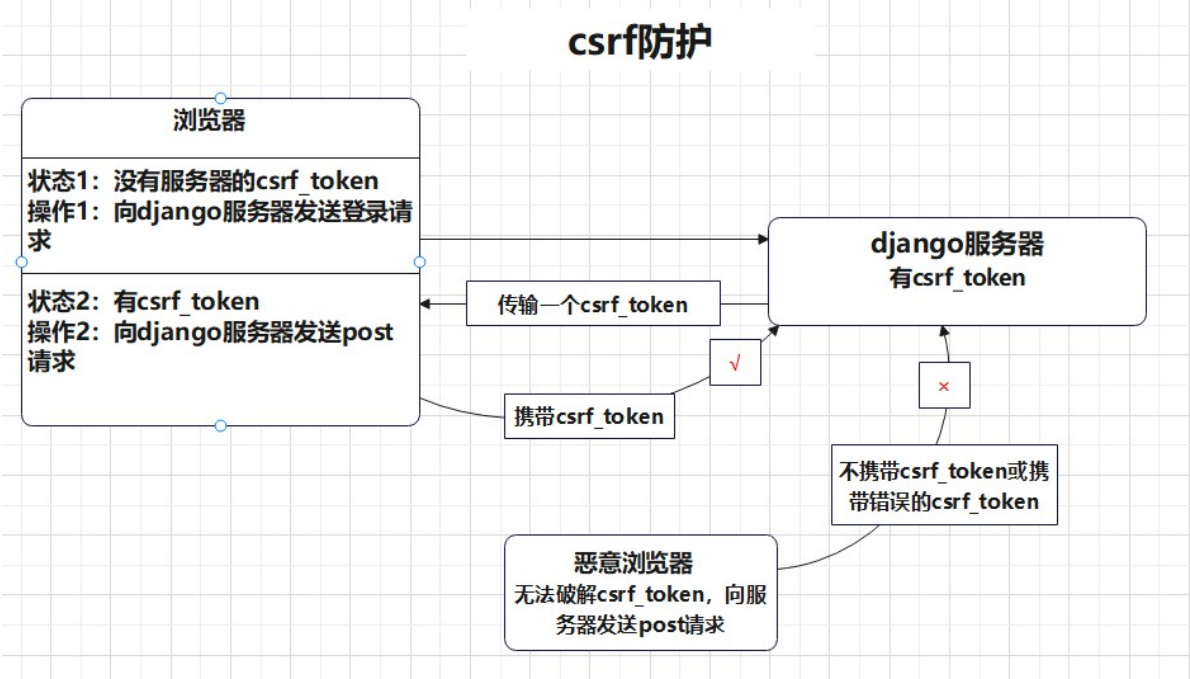
CSRF 攻击之所以能够成功，是因为黑客可以完全伪造用户的请求，该请求中所有的用户验证信息都是存在于 cookie 中，因此黑客可以在不知道这些验证信息的情况下直接利用用户自己的 cookie 来通过安全验证。要抵御 CSRF，关键在于在请求中放入黑客所不能伪造的信息。

Django采用的方式是在 HTTP 请求中以参数的形式加入一个随机产生的 token，用户每次发起请求时，不仅要带上自身1的cookies，还必须携带token。而服务器端则会建立一个拦截器来验证这个 token，如果请求中没有 token 或者 token 内容不正确（与服务器端的token不同），则认为可能是 CSRF 攻击而拒绝该请求。

官方文档提及：检验csrftoken时，只比较secret而不是比较整个token，token字符串的前32位是salt， 后面是加密后的token， 通过salt能解密出唯一的secret。这样就使得表单的token可以不断刷新，而客户端的token则可较长时间不变

```
###在对网页的检查中，我们发现了原本的{% csrf_token %}变成了如下字符串，value即为token
<input type="hidden" name="csrfmiddlewaretoken"
value="xdjfCeW13mozq3VhrTyyZ5AtzC13etB0ipsnfpp2vfChk5Fpp1LjBQIRAIFj9Yc4">
```

这里我们画了一张流程图直观反应csf防护的流程



Part9-Conclusion

在本次project中，我们小组精诚合作。我们完成了13个API设计与一步读入或输出数据设计。然后，我们设计账单模块和根据时间日期更改订单状态的机制来进一步完善API设计，可以让其接受更为灵活参数的请求。接着，我们封装并实现一个真正的后端服务器，使用数据库连接池，构造一个用于呈现数据的实用又美化的界面，还使用了索引来加快操作表格的速度。最后，我们初步探究了CSRF攻击的攻击方式与防范措施。

同时，本次项目中我们也遇到了不少困难

- django环境的配置（这里推荐在安装包的时候，最好指定一个比较稳定的版本，如果不指定版本默认是最新版，可能会与其他库无法兼容）。
- 在我用pandas对数据进行操作时，一开始算出来的答案怎么也不对，最后经过逐步排查，我才发现原来是因为pandas merge函数的原因。merge函数功能基本与sql语句中的join相同，但是需要注意的是，merge后的表格并不会按照原来的顺序排序，顺序会被打乱。由于本次的订单模块对时间字段十分敏感（需要根据订单实时更新库存）但是本次数据的时间字段不够精确，最小单位为天。导致在进行一系列操作时，同一天的订单之间的相对顺序不能改变。如果时间字段精确度足够的话，我们可以转化为时间戳进行排序。所以最后我将对时间敏感的表格，join操作均采用pandas 的 join 函数，不使用merge。
- 同时需要注意的是，在按时间排序时，我们需要指定排序类型为mergesort，pandas默认的排序方式quicksort是不稳定的
- 对于html，css，web框架这些东西，由于是初次接触，所以一定要多学多问，不会的就上网搜，只要有耐心，问题都能得到解决。命令行操作时，千万不能害怕红色的报错，先读懂报错含义，再查阅对应的解决方法。

最后，感谢github用户 **yenchiah** 的网页模板设计思路，感谢老师与助教一学期的付出，感谢在本次项目中给予过我们任何帮助的人！