SP1-lab1.1&lab1.2

Lab 1.1 Web Environment Setup

Overview

Web programming is a really big topic, which is related to a series of work. But In this lab, the only thing you have to do is: learn how to setup the web environment, include Java, Tomcat and Eclipse. It is the very base of all our further work about Web.

Before you start, you should choose the OS platform first, windows or linux. Actually, both platforms are suitable, just up to you!

Back to the lab, what's our objective:

1. Setup the Java, and make sure it's correct.
2. Setup the tomcat, and check the tomcat welcome page.
3. Setup the Eclipse, and learn how to use it.

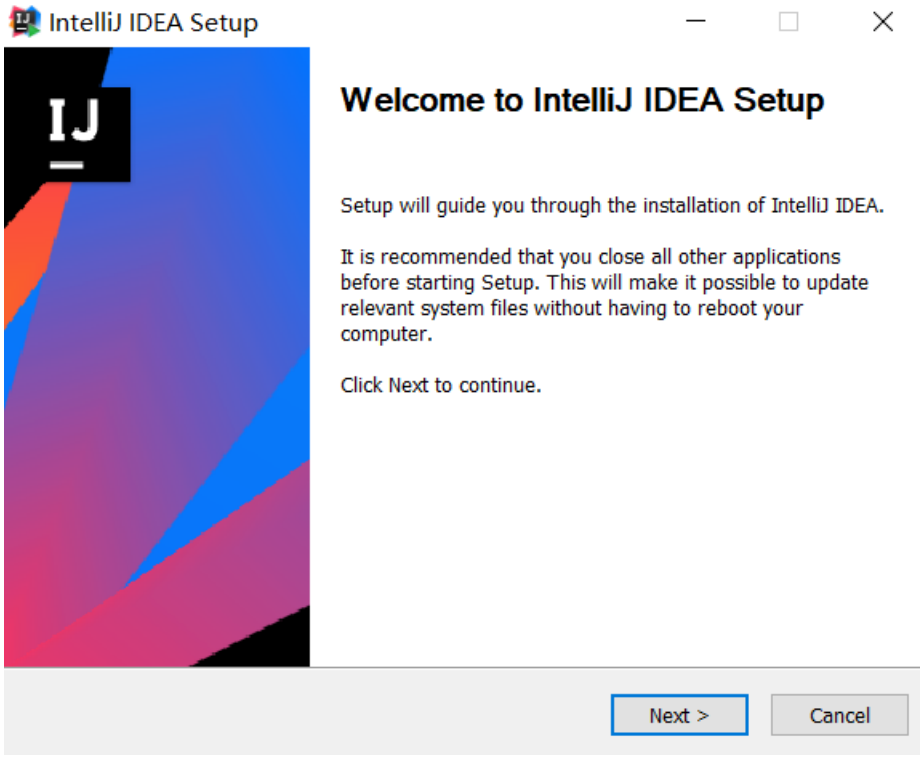
实验过程

1.实验要求的环境安装

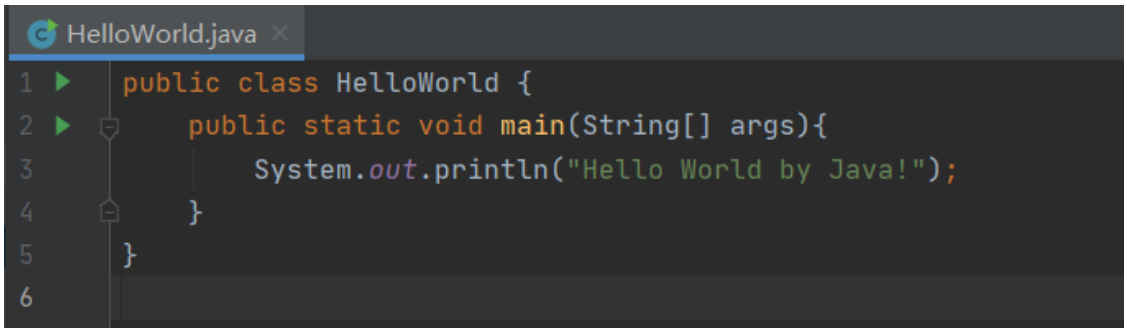
- Java的安装：电脑上之前已经安装过Java，因此安装的过程省略，可以在cmd中查看Java版本

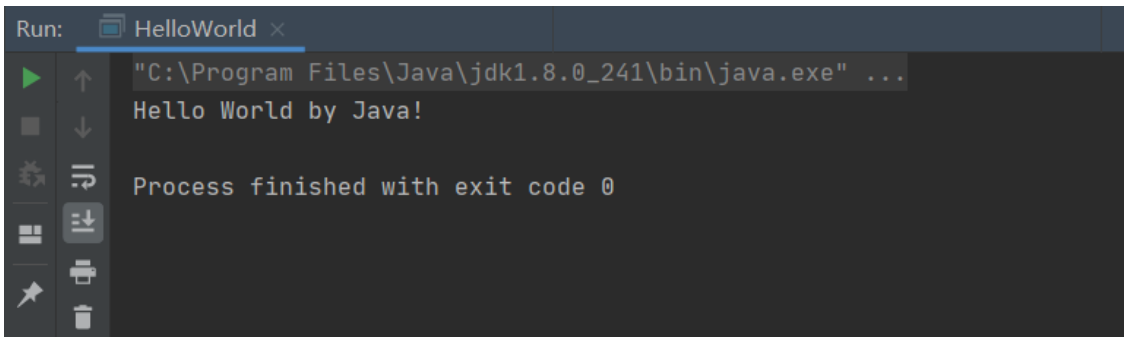
```
C:\Users\74096>java -version
java version "1.8.0_241"
Java(TM) SE Runtime Environment (build 1.8.0_241-b07)
Java HotSpot(TM) 64-Bit Server VM (build 25.241-b07, mixed mode)
```

- Tomcat的安装
  - 在http://tomcat.apache.org/中下载安装包，不过需要下载tomcat7的版本
  - 但是查询之后发现实验1.3中的Webgoat已经自带了Tomcat，不需要另外安装，而1.2的实验我选择了用Python和Django框架来完成，因此Tomcat的安装可以省略(Django的本地部署不需要用到Tomcat)，本实验被我改成了安装Python+Django+Pycharm，具体内容可以看实验1.1的第三部分
- Eclipse的安装
  - 我选择了安装功能更强大的Java集成开发环境IDEA，虽然本次实验用不上了，但是后面的实验还有机会使用，安装过程如下
  - 首先进入JB系列的官网找到IDEA的下载页面进行下载，选择professional版本，因为有学生邮箱可以免费使用，不过安装包比较大，所以下载的过程花费了比较长的一段时间



- 选择安装路径之后成功安装
- 安装成功之后尝试写一个Hello World，成功





- 至此IDEA的安装成功了

2.Web开发环境选择：

- 由于还没有系统学过Java和JSP的相关技术，决定采用Python3.7+Django框架进行Web应用的开发(~~当然Python其实也没系统地学过~~)
- 因此Web开发的环境配置变成了Python3.7+Django框架+Pycharm

3.环境的安装

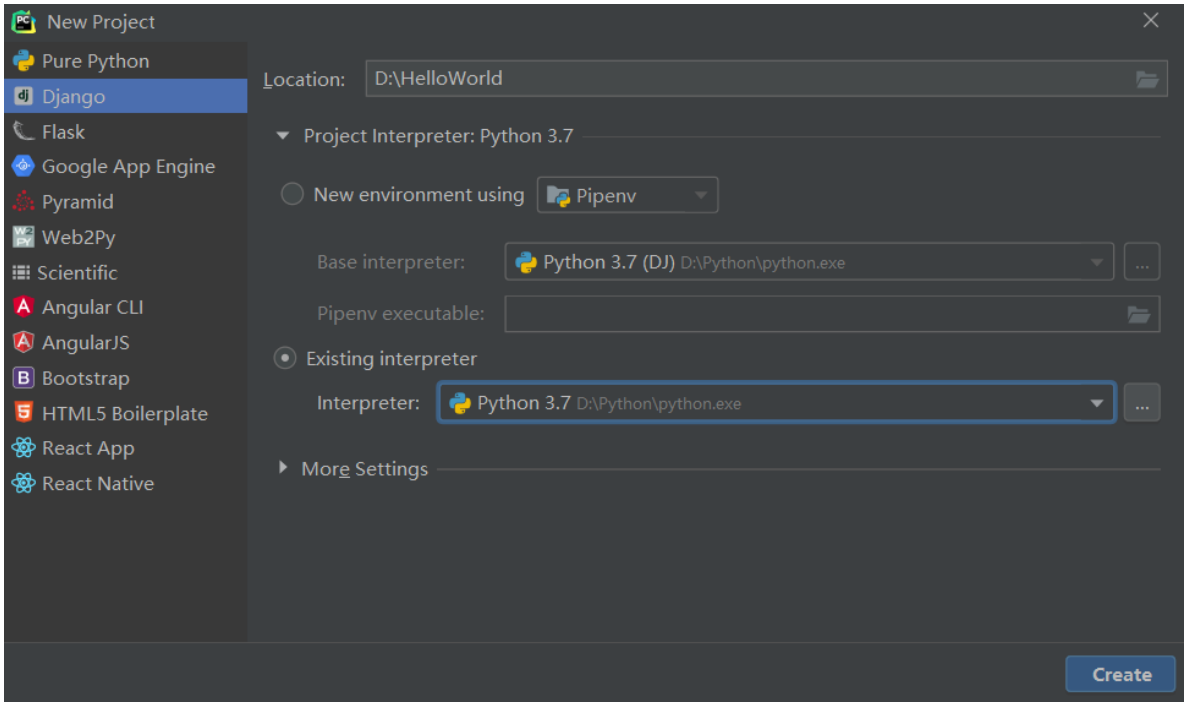
- Pycharm很早之前就已经安装，这里省略安装过程，总之用学生邮箱可以免费使用
- Python的版本是3.7，之前也已经完成安装
- Django相关的Python库可以用pip命令安装

```
C:\Users\74096>pip3.7 install django -i https://pypi.tuna.tsinghua.edu.cn/simple
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple
Collecting django
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/a9/4f/8a247eee2958529a6a805d38fbacd9764fd566462fa0016aa2a2947ab2a6/Django-3.0.5-py3-none-any.whl (7.5MB)
     |████████████████████████████████| 7.5MB 1.7MB/s
Collecting pytz (from django)
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/e7/f9/f0b53f88060247251bf481fa6ea62cd0d25bf1b11a87888e53ce5b7c8ad2/pytz-2019.3-py2.py3-none-any.whl (509kB)
     |████████████████████████████████| 512kB 731kB/s
Collecting sqlparse>=0.2.2 (from django)
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/85/ee/6e821932f413a5c4b76be9c5936e313e4fc626b33f16e027866e1d60f588/sqlparse-0.3.1-py2.py3-none-any.whl (40kB)
     |████████████████████████████████| 40kB 2.7MB/s
Collecting asgiref~=3.2 (from django)
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/68/00/25013f7310a56d17e1ab6fd885d5c1f216b7123b550d295c93f8e29d372a/asgiref-3.2.7-py2.py3-none-any.whl
Installing collected packages: pytz, sqlparse, asgiref, django
Successfully installed asgiref-3.2.7 django-3.0.5 pytz-2019.3 sqlparse-0.3.1
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

- 到此，基于Django框架的Web开放环境已经安装完毕

4.实现“Hello World”的简单网页

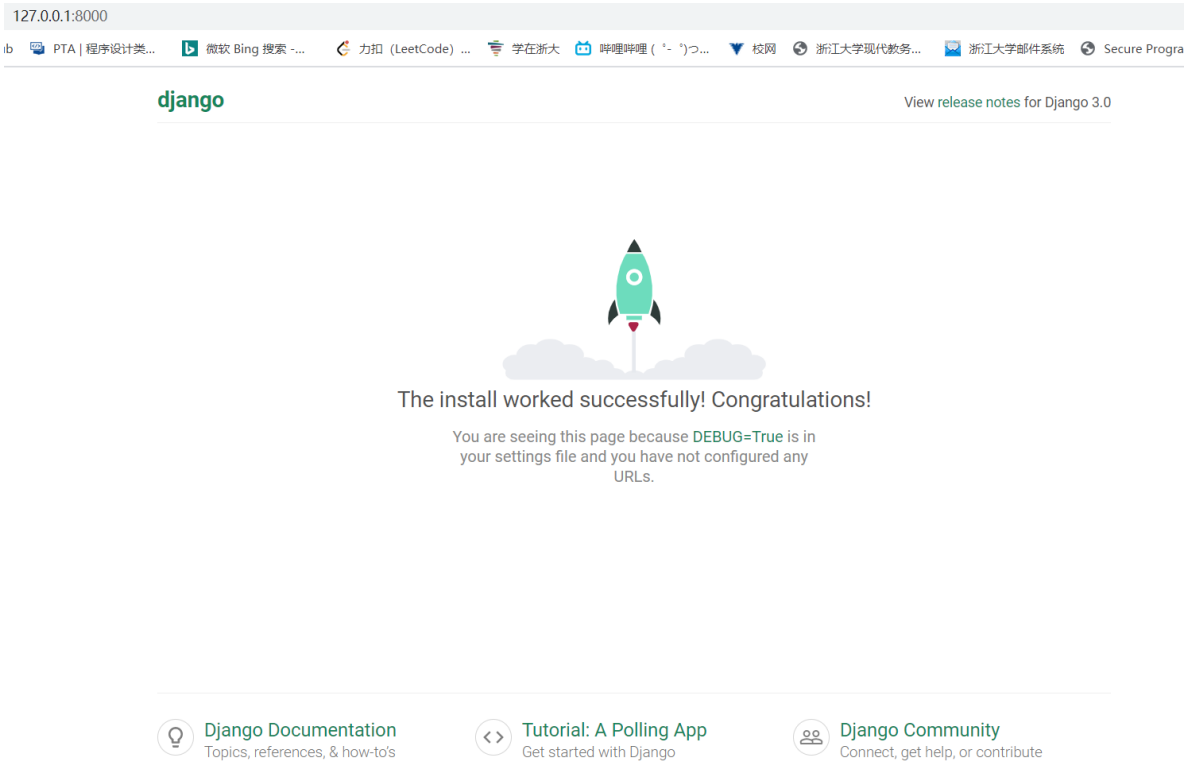
- 在Pycharm中创建一个新的项目



- 在Pycharm中输入启动命令

```
D:\HelloWorld>python3 manage.py startapp Hello
```

- 现在我们运行该项目，看到的Django的默认页面



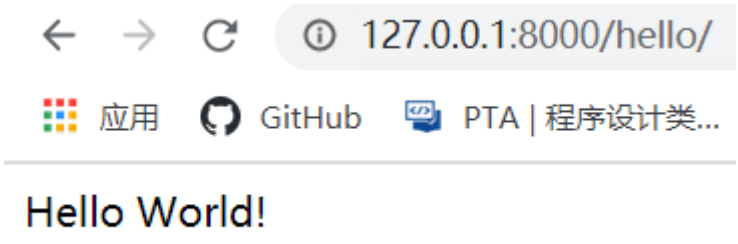
- 下面将这个项目修改为显示Hello World！
  - 创建一个新的文件views.py,写入代码

```
from django.http import HttpResponse


def hello(request):
    return HttpResponse("Hello World!")
```

  - 在urls.py中添加代码

```
from HelloWorld.views import *


urlpatterns = [
    path('admin/', admin.site.urls),
    path('hello/', hello)
]
```

  - 运行项目，查看127.0.0.1:8080/hello可以看到



  - 也可以把Hello World!加上标签<h1>

- 至此实验1.1已经完成

Lab 1.2 Implementation of Web Application

Overview

In lab 1.1, you have learned to setup the Web Programming Environment. It is easy, but very important, which is the base of lab 1.2.

In this lab, you will learn how to implement a web application. Usually, a web application is related with database, web server, data access, service, page design and so on. So there are really much you have to learn before you start the Implementation.

Back to the lab, what's our objective:

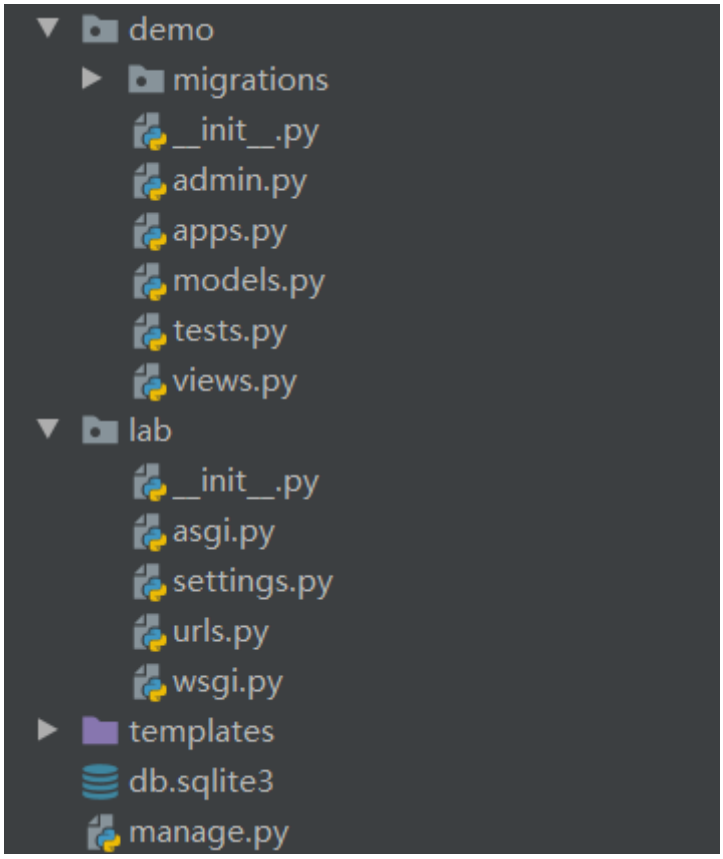
1. Design and access the database (MySQL recommended).
2. Learn to use web server and deploy your web application (Tomcat recommended).
3. Implement the application (Html, JSP recommended).
4. CSS is required to be used for controlling the page display in the “External Style Sheet” way.

实验过程

本次实验用django框架实现了一个简单的学生成绩系统，包含登录页面和主页面两个部分，做的比较简陋，但是已经完成了实验的全部要求，~~虽然让学生来修改自己的成绩有点奇怪~~，但总的来说是一次愉快的web开发的尝试

1. 基本的配置

- 实验选用的相关技术：前端采用原生Html三件套+后端django3.0.3+数据库sqlite3+Python3.6.5
- 打开pycharm，新建一个django的项目取名为lab，打开pycharm中的terminal输入`python manage.py startproject lab`新建一个新的project，然后在输入`python manage.py startapp demo`新建一个名为demo的app，
    - 之后项目中所含的文件如下图



  - demo作为app是整个project lab的一个子模块，lab文件夹中的文件主要是全局相关的一些文件，其中
    - _init_.py 是空文件，目的是声明该项目是一个python项目
    - setting.py是全局的一些配置
    - urls.py是URL声明，包含项目中所含的网站的目录等内容
  - demo中的相关py文件的作用如下
    - admin.py是注册的用户在admin管理网站中能管理的一些内容
    - _init_.py作用和整个project的相同
    - models.py用于创建数据库的模型
    - views.py是视图管理的文件，包含响应Http请求并返回视图的函数
  - templates文件夹用于存储一些页面的模板，通过服务器相应前端的Http请求而发送出去
  - db.sqlite3是django项目自带的sqlite3数据库，sqlite3是一种嵌入式的数据库，具有体量小和开发方便，处理速度快等优点，可以打开terminal通过命令`sqlite3 db.sqlite3`进行操作，而SQL语句的使用和其他数据库基本一致，没有啥区别

- 注册root用户
  - 注册root用户的过程如下

```
(base) D:\lab>python manage.py createsuperuser
用户名 (leave blank to use '74096'): root
电子邮件地址: 740969824@qq.com
Password:
Password (again):
密码长度太短。密码必须包含至少 8 个字符。
这个密码太常见了。
密码只包含数字。
Bypass password validation and create user anyway? [y/N]: n
Password:
Password (again):
Superuser created successfully.
```

  - 注册成功之后可以在网站运行时进入admin页面进行相关的操作

2. 整体系统的设计

- 系统分为登陆页面和操作页面两个部分，整体的设计思路如下
  - 登陆页面的表单包含用户名和密码两个输入框，通过一个按钮返回Http的POST请求将数据传送到服务器端，服务器调取数据库中的用户信息进行判断能否成功登录
  - 操作页面会在登陆成功后跳转，根据登陆人的账号调取成绩显示，用户可以输入成绩进行添加(虽然给自己加成绩这个功能看起来有点奇怪，主要还是为了体验一下前后端的交互和数据库的更新的操作)

3. 数据库的设计和创建

- 本实验一共有两张数据表，表的定义如下(在models.py中用python的class定义)
  - 两张表分别存储用户的账号密码和全部的课程信息，包括课程编号，课程名，成绩和修读该课程的学生姓名

```python
from django.db import models


# Create your models here.
class user_info(models.Model):
    username = models.CharField(max_length=20)
    password = models.CharField(max_length=20)


class course(models.Model):
    course_id = models.CharField(max_length=20)
    course_name = models.CharField(max_length=255)
    course_grade = models.IntegerField()
    student = models.CharField(max_length=20)
```

- 数据库的创建的过程，在models.py中完成两张表的定义之后，进入terminal执行`python manage.py makemigrations`和`python manage.py migrate`两条命令在db.sqlite3数据库文件中生成对应的数据表，过程如下图所示

```
(base) D:\lab>python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying sessions.0001_initial... OK
```

- 完成上述操作之后demo文件夹中会生成migrations文件夹存储一些数据库中的表变更的记录，我们可以在terminal中查看db.sqlite3已有的数据表文件，但是在生成数据表的过程中，sqlite3会给每张表加一个id的属性，这个应该是sqlite3的特性

```
(base) D:\lab>sqlite3 db.sqlite3
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite> select name from sqlite_master;
django_migrations
sqlite_sequence
auth_group_permissions
auth_user_groups
auth_user_user_permissions
auth_group_permissions_group_id_permission_id_0cd325b0_uniq
auth_group_permissions_group_id_b120cbf9
auth_group_permissions_permission_id_84c5c92e
```

```
auth_user_user_permissions
auth_group_permissions_group_id_permission_id_0cd325b0_uniq
auth_group_permissions_group_id_b120cbf9
auth_group_permissions_permission_id_84c5c92e
auth_user_groups_user_id_group_id_94350c0c_uniq
auth_user_groups_user_id_6a12ed8b
auth_user_groups_group_id_97559544
auth_user_user_permissions_user_id_permission_id_14a6b632_uniq
auth_user_user_permissions_user_id_a95ead1b
auth_user_user_permissions_permission_id_1fbb5f2c
django_admin_log
django_admin_log_content_type_id_c4bce8eb
django_admin_log_user_id_c564eba6
django_content_type
django_content_type_app_label_model_76bd3d3b_uniq
auth_permission
auth_permission_content_type_id_codename_01ab375a_uniq
auth_permission_content_type_id_2f476e4b
auth_user
sqlite_autoindex_auth_user_1
auth_group
sqlite_autoindex_auth_group_1
django_session
sqlite_autoindex_django_session_1
django_session_expire_date_a5c62663
demo_user_info
demo_apply
demo_course
```

- 之后在给数据库添加一些初始数据

```
sqlite> insert into demo_course values ('21120491','高级数据结构与算法分析',60,'zyc');
Error: table demo_course has 5 columns but 4 values were supplied
sqlite> insert into demo_course values ('1','21120491','高级数据结构与算法分析',60,'zyc');
sqlite> insert into demo_course values ('2','21121350','数据库系统','61','zyc');
sqlite> insert into demo_course values ('3', '21121290','计算机系统原理','59','zyc');
sqlite> insert into demo_course values ('4','21112345','不知道什么课','zyc1');
Error: table demo_course has 5 columns but 4 values were supplied
sqlite> insert into demo_course values ('4','21112345','不知道什么课','zyc1');
Error: table demo_course has 5 columns but 4 values were supplied
sqlite>  insert into demo_course values ('4','21112345','不知道什么课',0,'zyc1');
sqlite> select * from demo_course;
1|21120491|高级数据结构与算法分析|60|zyc
2|21121350|数据库系统|61|zyc
3|21121290|计算机系统原理|59|zyc
4|21112345|不知道什么课|0|zyc1
```

3. 开发过程

- 登陆界面的开发过程
  - 先编写页面的模板page1.html作为前端，用CSS样式美化一下界面中的内容，通过编写一个表单并执行POST请求实现数据的传输

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
```

```
    <title>登录</title>
    <style>
        input{
            outline-style: none ;
            border: 1px solid #ccc;
            border-radius: 3px;
        }
    </style>
</head>
<body>
<h1 style="text-align: center; color: blue">登陆</h1>
<!--创建一个用于提交的表单-->
<script type="text/javascript">
    function check() {
        if(document.info.username.value.toString().trim().length <= 0){
            alert("您未输入用户名");
            return false;
        }
        else
if(document.info.password.value.toString().trim().length<=0){
            alert("您未输入密码");
            return false;
        }
        return true;
    }
</script>
<form method="post" action="/login_action/" name="info">
    <div style="text-align: center">
        <text style="font-family: 黑体">用户:</text>
        <input name="username" type="text" placeholder="UserName">
    </div><br>
    <div style="text-align: center">
        <text style="font-family: 黑体">密码:</text>
        <input name="password" type="password" placeholder="Password">
    </div><br>
    <div style="text-align: center">
        <text style="color: red">{{ fail }}</text>
    </div><br>
    <div style="text-align: center">
        <button id="button1" type="submit" onclick="return check()">登录
</button>
    </div>
</form>

</body>
</html>
```

- 后端的相应如下，通过在数据库调取用户数据进行比对来判断登陆是否成功，若成功则进入成绩查询页面，若失败则显示失败信息(在前端的fail中)

```
# 初始界面
def hello(request):
    return render(request, 'page1.html')

```

```
5
6   # POST请求的响应
7   def login_action(request):
8       if request.method == 'POST':  # 判断是否为post提交方式
9           user_name = request.POST.get('username', '')  # 通过post.get()方
    法获取输入的用户名及密码
10          pass_word = request.POST.get('password', '')
11          # 连接到数据库中
12          user = models.user_info.objects.all()
13          user_list = {}
14          for i in user:
15              user_list[i.username] = i.password
16          if user_list[user_name] == pass_word:
17              global name
18              name = user_name
19              return HttpResponseRedirect('/success/')
20          else:
21              return render(request, 'page1.html', {'fail': 'username or
    password error'})
```

- 成绩界面的开发过程
  - 编写前端界面，依然是用模板进行渲染形成前端页面，不过成绩界面的渲染的逻辑相比起登陆界面稍微复杂一点，代码有点长就省略一点了，具体的可以去源代码中查看，下面展示一下界面的CSS样式

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>Title</title>
6
7       <style type="text/css">
8           table
9           {
10              border-collapse: collapse;
11              margin: 0 auto;
12              text-align: center;
13          }
14          table td, table th
15          {
16              border: 1px solid #cad9ea;
17              color: #666;
18              height: 30px;
19          }
20          table thead th
21          {
22              background-color: #CCE8EB;
23              width: 100px;
24          }
25          table tr:nth-child(odd)
26          {
27              background: #fff;
28          }
29          table tr:nth-child(even)
```

```
        {
            background: #F5FAFA;
        }
        input{
            outline-style: none ;
            border: 1px solid #ccc;
            border-radius: 3px;
        }
    </style>
</head>
</html>
```

- 后端编写：主要是数据库的增删查改相关操作的相应，不过~~由于Python天下第一~~，所以写起来不是很复杂

```
def success(request):  # 该函数定义的是成功页面的提示页面
    # username =request.COOKIES.get('user', '') #读取浏览器cookie
    # print(name)
    courses = models.course.objects.all()
    # print(courses)
    return render(request, "page2.html", {'student_name': name, 'course': courses})


# 相应insert的请求
def change(request):
    if request.method == 'POST':
        id = request.POST.get('id', '')
        course = request.POST.get('name', '')
        grade = request.POST.get('grade', '')
        stu = name
        print(id, course, grade, stu)
        if id == "" or course == "" or grade == "" or stu == "":
            models.course.objects.create(course_id=id, course_grade=grade, course_name=course, student=stu)
            courses = models.course.objects.all()
        return render(request, "page2.html", {'student_name': name, 'course': courses})
```

- 其实编写的过程中还有一些url配置之类的代码，不过总体的量比较少，写完之后也难以一句句去抠出来，这里就不放出来了

4. 部署和测试

- django框架的封装程度非常高，不需要自己进行服务器的一些配置(推荐使用的JSP开发则需要配置Tomcat)，在terminal中输入命令`python manage.py runserver`就可以在本地服务器部署整个web应用

```
(base) D:\lab>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
May 03, 2020 - 20:13:20
Django version 3.0.3, using settings 'lab.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

- 我们通过本地服务器的8000端口来访问该网站，结果如下

- 输入账号密码后可以登录

- 查看和添加成绩，输入成绩后发现成绩在网页上实时更新了，并且下一次再登陆成绩依然保留，说明成绩的信息被成功写入到数据库中
    - 这里在后端逻辑上增加了防止空值的处理，如果三个属性中存在空值这一条成绩信息就不会被记录，点击添加后仍然显示当前页面

课程成绩

zyc同学,你的成绩是

课程号	课程名	成绩
21120491	高级数据结构与算法分析	60
21121350	数据库系统	61
21121290	计算机系统原理	59
21123232	信息安全原理	91
21190641	数值分析	95

课程编号 课程名称 成绩 添加

课程成绩

zyc同学,你的成绩是

课程号	课程名	成绩
21120491	高级数据结构与算法分析	60
21121350	数据库系统	61
21121290	计算机系统原理	59
21123232	信息安全原理	91
21190641	数值分析	95

21123434 离散数学及其应用 89 添加

课程成绩

zyc同学,你的成绩是

课程号	课程名	成绩
21120491	高级数据结构与算法分析	60
21121350	数据库系统	61
21121290	计算机系统原理	59
21123232	信息安全原理	91
21190641	数值分析	95
21123434	离散数学及其应用	89

课程编号 课程名称 成绩 添加

- 测试
  - 可以在test.py中编写一些测试代码，也可以通过runserver直接运行该web应用通过使用功能进行debug
  - 测试登录的情况：输入错误密码或者不存在的用户

登陆

用户：zyc

密码：•••••

登录

登陆

用户：UserName

密码：Password

username or password error

登录

登陆

用户：zyc123456

密码：•••

登录

- 测试过程中发现如果表单提交的是空值，会发生POST请求的错误导致网站无法正常使用，需要解决，因此解决的办法是在前端加入JavaScript代码，在表单提交之前检查是否有空值

```
<script type="text/javascript">
    function check() {
        if(document.info.username.value.toString().trim().length <= 0){
            alert("您未输入用户名");
            return false;
        }
        else
if(document.info.password.value.toString().trim().length<=0){
            alert("您未输入密码");
            return false;
        }
        return true;
    }
</script>
```

- 另一个界面当然也要加上相应的空值判断，防止POST请求出现错误

```
<script type="text/javascript">
    function check(){
        if(document.info.id.value.toString().trim().length <= 0){
            alert("请填写课程编号！")
            return false;
        }
        else if(document.info.name.value.toString().trim().length <= 0)
{
            alert("请填写课程名！");
            return false;
        }
        else if(document.info.grade.value.toString().trim().length<=0){
            alert("请填写成绩！");
            return false;
        }
```

```
        return true;
    }
</script>
```

- 至此实验1.2的内容全部结束，如果想查看完整的网站可以运行源代码