目录

[MySQL-mysql 8.0.17安装 1](#_Toc36737702)

[1安装&调试 1](#_Toc36737703)

[正则表达式 2](#_Toc36737704)

[python 3](#_Toc36737705)

[3.1 python基础 3](#_Toc36737706)

[3.2文件操作 4](#_Toc36737707)

[3.3 xlrd 4](#_Toc36737708)

[3.4 selenium 5](#_Toc36737709)

[3.5 urllib 5](#_Toc36737710)

[3.6 Django 6](#_Toc36737711)

1. super().\_\_init\_\_()

super().\_\_init\_\_()的作用就是执行父类的构造函数，使得我们能够调用父类的属性。

## MySQL-mysql 8.0.17安装

### 1安装&调试

1. 下载后是zip包，无exe文
2. 配置高级系统设置→环境变量，添加到bin
3. mysql解压缩文件夹下新建my.ini

[mysqld]

# 设置3306端口

port=3306

# 设置mysql的安装目录

basedir=D:\\mysql-8.0.17-winx64 # 切记此处一定要用双斜杠\\

# 设置mysql数据库的数据的存放目录

datadir=D:\\mysql-8.0.17-winx64\\Data # 此处同上

# 允许最大连接数

max\_connections=200

# 允许连接失败的次数。这是为了防止有人从该主机试图攻击数据库系统

max\_connect\_errors=10

# 服务端使用的字符集默认为UTF8

character-set-server=utf8

# 创建新表时将使用的默认存储引擎

default-storage-engine=INNODB

# 默认使用“mysql\_native\_password”插件认证

default\_authentication\_plugin=mysql\_native\_password

[mysql]

# 设置mysql客户端默认字符集

default-character-set=utf8

[client]

# 设置mysql客户端连接服务端时默认使用的端口

port=3306

default-character-set=utf8

1. 以管理员的身份打开cmd，转到MySQL安装路径，输入如下命令初始化数据库：

mysqld --initialize --console

记住root的密码

1. cd到bin的目录输入“mysqld --install”
2. 启动mysql服务“net start mysql”
3. 更改密码，cd到bin目录后“ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '新密码'; ”
4. show databases;

## 正则表达式

#### 2.1 前瞻、后顾、负前瞻、负后顾

前瞻： exp1(?=exp2) 查找exp2前面的exp1

后顾: (?<=exp2)exp1 查找exp2后面的exp1

负前瞻: exp1(?=exp2) 查找后面不是exp2的exp1

负后顾: (?<=exp2)exp1 查找前面不是exp2的exp1

str = "博客园 顾客 博客 客园"

(?<=博)客(?=园) 匹配字符串里"博客园"的"客"字而不要其他的"客"字，这时就需要用到前瞻，后顾。

(?<!博)客(?!园) 不想要"博客园"的"客"字，但是想要其他"客"字。这时就要用到负前瞻，负后顾

#### 2.2匹配但是不包含

(?<=exp)是以exp开头的字符串, 但不包含本身.

(?=exp)就匹配惟exp结尾的字符串, 但不包含本身

## python

### 3.1 python基础

#### 3.1.1 生成器generator

优点就是节省内存空间

生成器使用方式1：

g = [for x in range(10)]

for y in g:

print y

生成器使用方式2：

依次调用g.next()方法，该方法遇到yield关键词而终止，再使用next()会执行上一个yield之后的程序。

(1)list4 = [2\*n + 1 for n in range(3, 11)]

(2)

L = [3， 7， 11， 14，19， 33， 26， 57， 99]

list6 = [x for x in L if x > 20]

(3)

list8 = [(x, y) for x in L1 for y in L2]

#### 3.1.2 a,b=b,a+b

a,b = b,a+b

该程序首先执行b=，a+b=，然后将b赋值给a，a+b值赋值给b

#### 3.1.3 inistance()方法

isinstance(实例，class) 与 type() 区别：

type() 不会认为子类是一种父类类型，不考虑继承关系。

isinstance() 会认为子类是一种父类类型，考虑继承关系。

判断事物是不是同一个类型推荐使用isintance()，返回boolean

### 3.2文件操作

w 以写方式打开，

a 以追加模式打开

r+ 以读写模式打开

w+ 以读写模式打开

a+ 以读写模式打开

rb 以二进制读模式打开

wb 以二进制写模式打开

ab 以二进制追加模式打开

rb+ 以二进制读写模式打开

wb+ 以二进制读写模式打开

ab+ 以二进制读写模式打开

### 3.3 xlrd

#### 3.3.1 execle之中date处理

xlrd读取出来的时间字段是类似41410的浮点数，在使用时需要转换成对应的datetime类型

1.首先导入 from xlrd import xldate\_as\_tuple

2. xldate\_as\_tuple(d,0)

第二个参数有两种取值，0或者1，0是以1900-01-01为基准的日期，而1是1904-01-01为基准的日期。

该函数返回的是一个元组，他的值类似：(year, month, day, hour, minute, nearest\_second)

#### 3.3.2 execle操作时的数据类型

ctype : 0 empty,1 string, 2 number, 3 date, 4 boolean, 5 error

### 3.4 selenium

#### 3.4.1. find\_element\_by\_css\_selector( )

(1)该函数非常的灵活

可以取name属性，

<a href="http://news.baidu.com" name="tj\_news">新 闻</a>

driver.find\_element\_by\_css\_selector("a[name=\"tj\_news\"]")

也可以取title的属性：

<a onclick="queryTab(this);" mon="col=502&pn=0" title="web" href="http://www.baidu.com/">网页</a>

driver.find\_element\_by\_css\_selector("a[title=\"web\"]")

还可以

<a class="RecycleBin xz" href="javascript:void(0);">

driver.find\_element\_by\_css\_selector("a.RecycleBin")

**3.4.2 browser.find\_element\_by\_link\_text()**

browser.find\_element\_by\_link\_text("贴 吧")

该方法针对的是文字链接

**3.4.3 is\_displayed()**

is\_displayed()该元素是否用户可以见

### 3.5 urllib

#### 3.5.1 urllib.request.urlretrieve()图片的下载

request.urlretrieve(url,filename=None,reporthook=None, data=None)

* url：下载链接地址
* filename：指定保存本地路径
* reporthook：回调函数，可以利用其来显示当前的下载进度。
* data：返回一个包含两个元素的(filename, headers) 元组，filename 表示保存到本地的路径，header表示服务器的响应头
* usually只需要前2个参数

#### 3.5.2 相关的方法

book\_list = soup.find(attrs={"id":"book"})

获取id = “book”标签的中的全部源码

同样的方法：

book\_list\_name = book\_list.find\_all(attrs={"class":"title"})

soup.select('.main-title') 可提出class类名为main-title

未测试的：soup.select('.class span a')这表示在class下的span的a下面

book\_img\_url = book\_one.get('src')

获取book\_one标签中的src属性的内容

### 3.6 Django

#### 3.6.1 django常用命令

1、创建项目命令：django-admin startproject demo

2、创建应用命令：python manage.py startapp demo

3、运行服务器命令：python manage.py runserver 0:8080

(runserver 后面可以跟地址:端口号，如果不写默认本地8000端口启动)

4、生成数据迁移文件：python manage.py makemigrations [AppName]

5、迁移数据：python manage.py migrate

6、创建管理员命令：python manage.py createsuperuser

7. 改成中文LANGUAGE\_CODE = 'zh-hans'

#### 3.6.2 Django的增删查改

1. 增

第一种：

obj = Article(title="My first article", body="My first article body")

obj.save()

如果不save()，对象无法存储

第二种：直接create

Article.objects.create(title="My first article", body="My first article body")

1. 删

第一种：

Article.objects.all().delete()

第二种：

Article.objects.filter(title\_\_icontains="python").delete()

1. 查

第一种：

Article.objects.all()

Article.objects.all().values()

# 只获取title列表-字典形式

Article.objects.all().values('title')

# 只获取title列表- 元组形式，只有value，没有key

Article.objects.all().values\_list('title')

第二种：

Article.objects.get(id=1)

第三种：

Article.objects.filter(title\_\_icontains='python').distinct()

1. 改

第一种：

article = Article.objects.get(id=1)

article.title = "New article title"

article.save()

第二种：

article = Article.objects.get(id=1).update(title='new title')

第三种：

article = Article.objects.filter(title\_\_icontains='python').update(title='Django')

#### 3.6.3 Django之中的坑

1. 在APP模块中admin.py需要注册相应的model，注册方式是

from .models import employee【主要是那个.】

admin.site.register(model名)

1. app的model.py之中的属性，数字类型的不必设置max\_length，但文字类的必须设置，否则启动时会报错。
2. get数据之中的问题
3. django的get方法是从数据库的取得一个匹配的结果，返回一个对象，如果记录不存在的话，它会报错。
4. 用django的get去取得关联表的数据的话，而关键表的数据如果多于2条的话也会报错。
5. django的filter方法是从数据库的取得匹配的结果，返回一个对象列表，如果记录不存在的话，它会返回[]。
6. 用django的get去取得关联表的数据的话，无论关联表有多少记录的都不会报错。
7. e()方法会返回一个结果，没有就报错
8. filter()方法会返回一个结果集，如果没有数据就是一个空集
9. update()与save()

update方法只作用于结果集，搭配filter()方法

save方法针对单个对象，搭配get()方法

1. 更新的三种方法

archivo = archivo.objects.get(archivo\_id=procesar)

archivo.archivo\_registros = sh.nrows

archivo.save()

archivo = archivo.objects.get(archivo\_id=procesar)

archivo.archivo\_registros = sh.nrows

archivo.save(update\_fields=['archivo\_registros'])

archivo = archivo.objects.filter(archivo\_id=procesar).update(archivo\_registros=sh.nrows)

1. 1

3.6.4 Django学习过程

1. 5-1课之中：配置url
   1. 在总url之中

from django.urls import path,include

path('index/',include('employee.urls'))

红色部门为增加和修改的部分，employee为新建的app的名称，其目录下需要新建urls.py的文件，而employee.urls就会按照机制导航到APP下的urls.py，该urls.py可以复制总目录下的同名文件修改但会略有不同：

首先，需要导入同app下的views文件：

from employee import views

其次，导航到相应的方法，employee是导航的关键字，indexEmployee是views.py之中对应的方法

urlpatterns = [

path('employee',views.indexEmployee)

]

按照机制，系统会到views.py之中寻找indexEmployee方法def indexEmployee(request):

return render(request,'index.html',

{'name':employee.name} )

request是固定的参数，

index.html是下一步的显示页面(3.0的特性可能不需要app文件夹名)，

{}是发送的参数，里面可以是一个字典。

前两个参数是必须的。

1. 5-2课，配置Templates
   1. Templates其实就是html文件，只是使用Django之后的DTL模板语言，也可以使用第三方的模板语言。

如果使用第三方的模板语言，需要在setting.py文件之中修改TEMPLATES之中的'BACKEND'的属性为对应的模板引擎文件即可。

* 1. 在app下建立Templates的文件夹，里面放置html文件
  2. 为了解决Templates冲突方案，在App的Templates目录下创建以App名称命名的文件夹，将html文件放入相应的文件夹，同时需要修改相对应的app下的urls之中的路径

1. 6课models
   1. models通常对应数据库之中的一张表，以创建类的形式创建数据表。这也是一种ORM映射。
   2. 首先，在应用App下创建models.py的文件，并引入models模块
   3. 创建类，继承models.Model ，就表明了该类是一张数据表，如

def class employee(models.Model):

* 1. 在类中创建属性

e\_id = models.IntegerField(auto\_created=1)

name = models.CharField(max\_length=10)

#部门编号

g\_id = models.IntegerField(null=True)

* 1. 生成数据表
* cmd下进入manage.py的同级目录
* python manage.py makemigrations [AppName]

不写AppName就生成所有的

* python manage.py migrate

迁移的动作

* 会在相应的app/migrations目录下生成相应的数据
* 0001\_init文件之中里面会增加一个id属性，也就是每个表增加一个id属性，因为没有添加primary key
* python manage.py sqlmigrate 应用名 文件id 查看sql语句

python manage.py sqlmigrate employee 0001

* 数据库会生成在项目根目录下的db.sqlite3文件之中

使用第三方软件SQLite Expert Personal可以查看

* 在相应的app的views.py之中使用：

from . import models

employee = models.employee.objects.get(pk=1)

employees = models.employee.objects.all()[返回所有对象]

* 前端html使用views.py传来的数据必须使用{{model名.属性}}

1. admin的设置：

python manage.py createsuperuser 【创建超级用户】

admin的设置是为了更好的处理业务模型，所以业务上的model需要在网页上显现，需要注册即可

* 1. 在新建的app下建立admin.py文件

from django.contrib import admin

from .models import employee

admin.site.register(employee)

employee是指在models.py之中生成的employee对象

* 1. 此时进入admin的页面就可以看到employee对象的情况，
  2. 需要汉化
  3. 修改数据默认的显示名称:

在app的model类中添加一个方法

\_\_str\_\_(self)/\_\_unicode\_\_(self)[python2.7]

之后返回self.属性

1. 8-1课中：
   1. 从数据库中取出数据显示在前端页面

需要使用到for循环

{% for xx in xxs %}

html语句

{% endfor %}

* 1. 跳转页面附传参数的方法
     1. app的urls.py文件中：

path('department/<emp\_id>',views.set\_dep ),

<emp\_id>即为参数名=emp\_id，set\_dep为views.py之中的方法名

views.py文件之中

def set\_dep(request,emp\_id):

employee = models.employee.objects.get(pk=emp\_id)

return render( request,'employee/department.html',{'employee':employee})

* + 1. 使用？来传递多个参数：

path('index/<pk>/',index,kwargs={'pk':20,'id':30,}),

在views.py中直接定义到，参数的名称要一致

def index(request,pk,id):

即便我们urls是https://localhost:8000/teacher/index/5

,传递过去的参数仍然会是pk=20，id=30

* + 1. 同样是传递多个参数

http://127.0.0.1:8000/plist/?p1=china&p2=2012

在urls.py文件之中：

(r'^plist/$', helloParams1）

在views.py之中：

def helloParams(request):

p1 = request.GET.get('p1')

p2 = request.GET.get('p2')

* + 1. url的跳转

将return render()改成return redirect('/teacher/info')

* + 1. 点击链接附带参数

总的urls.py文件配置

path('employee/',include(('employee.urls'),namespace='employee') )

namespace在include之中，而且不能用name

app的urls.py文件中

app\_name = 'employee'【增加的属性】

urlpatterns的配置之中：

path('department/<emp\_id>',views.set\_dep, name='xx' ),

在html文件之中

<a href="{% url 'employee:xx' employee.e\_id %}">

{{ employee.name }}</a>

在html文件之中生产的页面中就附带有employee的对象id，并且可以在namespace的配置下进行跳转和工作

* + 1. 1
  1. insert的过程演示

首先写好一个form表单，并配置好action：

其次，准备好action对应的方法：

def insert(request):

#两个参数，第二个是设置的默认值

name = request.POST.get('name','11')

g\_id = request.POST.get('g\_id')

level = request.POST.get('level')

#create model,并封装

models.employee.objects.create(emp\_id=3,name=name,g\_id=g\_id,level=level)

employees = models.employee.objects.all()

return render(request,'employee/index.html',{'employees':employees } )

urls.py的文件中配置：

path('insert/action',views.insert, name='insertAction'),出现CSRF验证失败解决：

在action内加入{% csrf\_token %}即可

但此处遇到了一个问题，就是model之中的对象设置了auto\_created，但红色的部分，必须设置不能为空。

* 1. 8-5课

将新增与修改的页面叠加使用

(5.4.1)在views新增判断，在页面上取得的id书否为0，然后跳转不同的page

(5.4.2)在html页面上设置一个判断

{% if employee %}

然后设置不同的value值{{ employee.e\_id }}……

{% else %}

{% end if %}

1. 9-1课 过滤器的使用

格式：{{ value|filter1|filter2|filter3 }}

例如页面之中的设置：

value={{ employee.e\_id|default:'0'}}

0即使默认值，如果不设置此过滤器，e\_id传过去的值是''

1. Django shell

项目目录的cmd输入:

python manage.py shell

1. Django admin增强
2. 更改admin管理页面的显示字段的情况：

admin.py文件中增加

class employeeAdmin(admin.ModelAdmin):

list\_display = ('e\_id','name','g\_id')

#字段必须对应models里面的字段

admin.site.register(employee,employeeAdmin)

#修改，将employeeAdmin进行注册关联上employee

1. new操作

给models里面的对象新增了一个时间的属性;

pub\_time = models.DateTimeField( null = True )

cd目录下执行命令：

python manage.py makemigrations [AppName]

python manage.py migrate

重启即可

1. admin过滤器的用法

admin.py文件中：

list\_filter = ('g\_id',)[]

括号中即便只有一个属性也需要','

1. 1
2. 1