nsd1902_py02_day04

https://down.51cto.com/

正则表达式必知必会：https://down.51cto.com/data/339092

复杂列表的排序

```
>>> patt_dict = {'172.40.58.150': 10, '172.40.58.124': 6, '172.40.58.101': 10, '127.0.0.1':
121, '192.168.4.254': 103, '192.168.2.254': 110, '201.1.1.254': 173, '201.1.2.254': 119,
'172.40.0.54': 391, '172.40.50.116': 244}
>>> patt_list = list(patt_dict.items())
>>> patt_list
[('172.40.58.150', 10), ('172.40.58.124', 6), ('172.40.58.101', 10), ('127.0.0.1', 121),
('192.168.4.254', 103), ('192.168.2.254', 110), ('201.1.1.254', 173), ('201.1.2.254', 119),
('172.40.0.54', 391), ('172.40.50.116', 244)]
# 查看sort的帮助，发现有个参数是key。这个key传递时是函数，用于确定排序依据。
>>> help(patt_list.sort)

# 函数func1用于返回序列中的最后一项
>>> def func1(seq):
...   return seq[-1]
...
# 排序时指定func1的结果作为排序依据。将列表中的每个元组当作func1的参数，返回值就是排序依据
>>> patt_list.sort(key=func1)
>>> patt_list
[('172.40.58.124', 6), ('172.40.58.150', 10), ('172.40.58.101', 10), ('192.168.4.254',
103), ('192.168.2.254', 110), ('201.1.2.254', 119), ('127.0.0.1', 121), ('201.1.1.254',
173), ('172.40.50.116', 244), ('172.40.0.54', 391)]

# 使用lambda匿名函数，并执行降序排列
>>> patt_list.sort(key=lambda seq: seq[-1], reverse=True)
>>> patt_list
[('172.40.0.54', 391), ('172.40.50.116', 244), ('201.1.1.254', 173), ('127.0.0.1', 121),
('201.1.2.254', 119), ('192.168.2.254', 110), ('192.168.4.254', 103), ('172.40.58.150',
10), ('172.40.58.101', 10), ('172.40.58.124', 6)]
```

python软件包管理

- pypi即python package index
- 官方站点：https://pypi.org/

- 通过pip命令可以直接在线安装，但是连接的是国外服务器，速度较慢
- 使用国内镜像站点

```
[root@room8pc16 ~]# mkdir ~/.pip/
[root@room8pc16 ~]# vim ~/.pip/pip.conf
[global]
index-url = http://mirrors.163.com/pypi/simple/
[install]
trusted-host=mirrors.163.com
```

- 在虚拟环境中安装pymysql

```
# 创建虚拟环境
[root@room8pc16 day04]# python3 -m venv ~/mypy
# 激活虚拟环境
[root@room8pc16 day04]# source ~/mypy/bin/activate
# 在线安装pymysql
(mypy) [root@room8pc16 day04]# pip3 install pymysql
# 在线安装pymysql并指定版本
(mypy) [root@room8pc16 day04]# pip3 install pymysql==0.9.2
# 或
# 本地安装
# cd pymysql_pkgs/
(mypy) [root@room8pc16 pymysql_pkgs]# pip3 install *
```

准备数据库

我们正在为一个小公司编写数据库，用于记录公司员工、部门的基本情况，还要记录发工资的情况。

需要的字段：姓名、性别、出生日期、部门、职位、工资日、基本工资、绩效、总工资、联系方式、工号

数据库应该尽量减少冗余：占空间、有可能导致数据不一致、增删改查都不方便

可以考虑，把字段放到不同的数据表中，以减少冗余。

关系型数据库的字段是有要求的，需要符合范式要求。

数据库范式有第一范式、第二范式、第三范式、巴斯-科德范式、第四范式、第五范式六种。第五范式是完美范式，但是，完美的东西不容易存在。一般来说，达到第三范式即可。

所谓第一范式（1NF）是指在关系模型中，对域添加的一个规范要求，所有的域都应该是原子性的，即数据库表的每一列都是不可分割的原子数据项，而不能是集合，数组，记录等非原子数据项。

根据1NF，发现联系方式不具有原子性，它可以再分，应该把它拆成：电话号码、邮箱、QQ、家庭住址、工作地址

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或记录必须可以被唯一地区分。

简单来说2NF就是需要有一个主键。工资表中工号是主键的理想选择；部门表中部门ID是主键；工资表中，用哪个字段作为主键都不合适，所以额外增加一个id字段，用于主键。

第三范式（3NF）是第二范式（2NF）的一个子集，即满足第三范式（3NF）必须满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个关系中不包含已在其它关系已包含的非主关键字信息。

简单来说，3NF要求非主键字段，不能依赖于其他非主键字段。工资总额依赖于基本工资和绩效，所以它就不应该出现在数据库中。工资总额由程序算出。

经过上面的分析，最终需要三张表和它们的字段如下：

员工信息表：工号、姓名、出生日期、电话、email、部门ID

部门表：部门ID、部门名称

工资表：id、工资日、工号、基本工资、绩效

```
[root@room8pc16 ~]# mysql -uroot -ptedu.cn
MariaDB [(none)]> CREATE DATABASE nsd1902 DEFAULT CHARSET utf8;
```

pymysql模块应用

1. 创建到服务器的连接

```
conn = pymysql.connect(
    host='127.0.0.1',
    port=3306,
    user='root',
    passwd='tedu.cn',
    db='nsd1902',
    charset='utf8'
)
```

2. 创建游标。可以理解游标就是打开文件时返回的文件对象。通过游标可以执行sql语句，对数据库进行CRUD（增删改查）。

```
cursor = conn.cursor()
```

3. 编写sql语句

```
create_dep = '''CREATE TABLE departments(
dep_id INT,
dep_name VARCHAR(20),
PRIMARY KEY(dep_id)
)'''
create_emp = '''CREATE TABLE employees(
emp_id INT,
emp_name VARCHAR(20),
birth_date DATE,
phone VARCHAR(11),
email VARCHAR(50),
dep_id INT,
PRIMARY KEY(emp_id),
FOREIGN KEY(dep_id) REFERENCES departments(dep_id)
)'''
create_sal = '''CREATE TABLE salary(
id INT,
date DATE,
emp_id INT,
basic INT,
```

```
awards INT,
PRIMARY KEY(id),
FOREIGN KEY(emp_id) REFERENCES employees(emp_id)
)'''
```

4. 执行SQL语句

```
cursor.execute(create_dep)
cursor.execute(create_emp)
cursor.execute(create_sal)
```

5. 确认

```
conn.commit()
```

6. 关闭游标和连接

```
cursor.close()
conn.close()
```

SQLAlchemy

ORM对象关系映射

- 将数据库表映射成了class类
- 表中的字段映射成类变量
- 数据库的数据类型映射成SQLAlchemy中的类
- 表中的一行记录映射成实体类的一个实例

应用

安装

```
(nsd1902) [root@room8pc16 zzg_pypkgs]# cd sqlalchemy_pkgs/
(nsd1902) [root@room8pc16 sqlalchemy_pkgs]# pip3 install *
```

创建数据库

```
MariaDB [nsd1902]> CREATE DATABASE tedu1902 DEFAULT CHAR SET utf8;
```

使用sqlalchemy的步骤

1. 创建到数据库的引擎
2. 生成实体类的基类
3. 创建实体类
4. 执行创建操作
5. 为了连接数据库，需要创建到达数据库的会话类