## 1.Flask的基本使用

## 1.安装flask,使用命令: pip install flask,也可以用pycharm来安装

|  |
| --- |
|  |

## 2.然后用pycharm创建一个flask项目

|  |
| --- |
|  |
|  |

## 3.添加一个index函数,然后在templates文件夹里面添加一个index.html文件

|  |
| --- |
|  |

### 然后我们就可以运行这个flask项目了.

|  |
| --- |
|  |
|  |
|  |

## 4.然后我们结合bootstrap来玩一玩,我们在项目根目录下面新建一个static文件夹

|  |
| --- |
|  |

### 然后我们把boostrap3.4.1和bootstrap-datepicker和css文件夹和js文件夹以及img文件夹粘贴过来

|  |
| --- |
|  |

## 5.然后我们在index.html文件里面引入boostrap样式并且添加一个导航条和一个表格.

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Index</**title**>  <**link rel="stylesheet" href="/static/plugins/bootstrap-3.4.1/css/bootstrap.min.css"**>  </**head**> <**body**>   <**div class="navbar navbar-default"**>  <**div class="container"**>  <**div class="navbar-header"**>  <**a class="navbar-brand" href="#"**>  <**img alt="Brand" width="20" height="20" src="data:image/png;base64,"**>  </**a**>  </**div**>  </**div**>  </**div**>   <**div class="container"**>  <**table class="table table-bordered"**>  <**thead**>  <**tr**>  <**th**>#</**th**>  <**th**>First Name</**th**>  <**th**>Last Name</**th**>  <**th**>Username</**th**>  </**tr**>  </**thead**>  <**tbody**>  <**tr**>  <**th scope="row"**>1</**th**>  <**td**>Mark</**td**>  <**td**>Otto</**td**>  <**td**>@mdo</**td**>  </**tr**>  <**tr**>  <**th scope="row"**>2</**th**>  <**td**>Jacob</**td**>  <**td**>Thornton</**td**>  <**td**>@fat</**td**>  </**tr**>  <**tr**>  <**th scope="row"**>3</**th**>  <**td**>Larry</**td**>  <**td**>the Bird</**td**>  <**td**>@twitter</**td**>  </**tr**>  </**tbody**>  </**table**>  </**div**>    <**script src="/static/js/jquery3.7.1.min.js"**></**script**>  <**script src="/static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"**></**script**>  </**body**> </**html**> |

### 运行程序,效果如下

|  |
| --- |
|  |

### 不过此时数据是写死的,不符合我们的要求,我们需要数据是可以动态改变的.

## 6.我们可以在渲染模板的时候给他传递一个变量(django必须是通过字典),任何它可以在html文件里面用2对花括号来获取变量的值(这就是模板语法,flask的和django的有点不一样)

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

## 7.然后,我们可以创建一个用户列表(就是模拟到数据库查询所有用户),然后把他作为参数传递给模板,然后在模板里面使用循环来渲染数据

|  |
| --- |
|  |

## 8.重新启动程序,访问/index/,效果如下

|  |
| --- |
|  |

# 2.安装MySQL数据库服务,参考: [MySQL的使用](mysql使用.doc)

## 我们可以使用mysql的命令行窗口来测试MySQL数据库是否工作正常.

## 1.我们可以使用mysql.exe来连接MySQL服务器,命令如下:

|  |
| --- |
|  |

## 2.如果是网络上面的另外一台电脑,需要把ip地址改为该电脑的ip,如果是在本机操作可以简化为下面的写法

|  |
| --- |
|  |

### 然后输入密码就可以打开一个命令窗口

## 3>其实我们可以把mysql.exe所在的路径也就是bin目录的路径添加到环境变量,这样子就可以使得命令非常简单

|  |
| --- |
|  |

## 4>可以设置密码,使用下面的命令

|  |
| --- |
|  |

## 5>查看现有的数据库

|  |
| --- |
|  |

## 6>退出mysql命令行工具

|  |
| --- |
|  |

## 7>万一忘记了mysql的密码怎么办

|  |
| --- |
|  |

### 具体做法:

### 1.先停止MySQL服务

|  |
| --- |
|  |

### 2.然后在my.ini文件里面添加一条配置: skip-grant-tables=1

|  |
| --- |
|  |

### 3.设置好后重新启动MySQL服务,打开命令行窗口,输入mysql -uroot -p,还是会提示你输入密码,不过此时你直接回车就可以登录成功

|  |
| --- |
|  |
|  |

#### 此时它是可以正常使用的

|  |
| --- |
|  |

### 4.输入命令use mysql;切换到mysql数据库

|  |
| --- |
|  |

### 5.然后设置密码,命令是: update user set password=password('新密码') where user='root' and host='localhost';注意:新密码是你需要用的密码,比如我们把密码设置为12345,就是:update user set password=password('12345') where user='root' and host='localhost'; (这里老师的命令不对可能是我的版本比较低,我的是5.6.x老师的是5.7.31)

|  |
| --- |
|  |

### 6.重新修改运行配置文件把skip-grant-tables=1注释掉.

|  |
| --- |
|  |

### 7.重新启动MySQL服务

|  |
| --- |
|  |

### 8.然后打开一个命令行窗口,输入mysql -uroot -p,当出现密码提示的时候直接按回车就会报错,

|  |
| --- |
|  |

### 9.然后我们输入正确的密码,有可以正常登录了

|  |
| --- |
|  |

### 10.我们还是把密码改回root好了: update user set password=password('root') where user='root' and host='localhost';

|  |
| --- |
|  |

### 11.然后需要退出mysql登录,然后重新启动MySQL服务

|  |
| --- |
|  |

### 12.然后我们重新登录,发现可以支持登录了

|  |
| --- |
|  |

## 8>有关SQL语句的使用,参考: [mysql数据库SQL语句学习](mysql学习笔记-SQL语句学习.docx)

# 3.安装flask操作MySQL的第三方库

## 1>安装flask-sqlalchemy的扩展

### pip install -U Flask-SQLAlchemy

## 2>python3：要连接mysql数据库，仍需要安装pymysql

### pip install pymysql

### 关于用python+pymysql来操作MySQL数据库,参考” python+pymysql操作MySQL.md”

## 3>常用的SQLAlchemy字段类型

上面看完了如何设置连接数据库，那么来看看，使用SQLAlchemy创建数据模型的时候，基本的字段类型如下：

| **类型名** | **python中类型** | **说明** |
| --- | --- | --- |
| Integer | int | 普通整数，一般是32位 |
| SmallInteger | int | 取值范围小的整数，一般是16位 |
| BigInteger | int或long | 不限制精度的整数 |
| Float | float | 浮点数 |
| Numeric | decimal.Decimal | 普通整数，一般是32位 |
| String | str | 变长字符串 |
| Text | str | 变长字符串，对较长或不限长度的字符串做了优化 |
| Unicode | unicode | 变长Unicode字符串 |
| UnicodeText | unicode | 变长Unicode字符串，对较长或不限长度的字符串做了优化 |
| Boolean | bool | 布尔值 |
| Date | datetime.date | 时间 |
| Time | datetime.datetime | 日期和时间 |
| LargeBinary | str | 二进制文件 |

## 4>常用的SQLAlchemy列选项

| **选项名** | **说明** |
| --- | --- |
| primary\_key | 如果为True，代表表的主键 |
| unique | 如果为True，代表这列不允许出现重复的值 |
| index | 如果为True，为这列创建索引，提高查询效率 |
| nullable | 如果为True，允许有空值，如果为False，不允许有空值 |
| default | 为这列定义默认值 |

## 5>常用的SQLAlchemy关系选项

| **选项名** | **说明** |
| --- | --- |
| backref | 在关系的另一模型中添加反向引用 |
| primary join | 明确指定两个模型之间使用的联结条件 |
| uselist | 如果为False，不使用列表，而使用标量值 |
| order\_by | 指定关系中记录的排序方式 |
| secondary | 指定多对多中记录的排序方式 |
| secondary join | 在SQLAlchemy中无法自行决定时，指定多对多关系中的二级联结条件 |

上面这些有很多基本选项的说明，下面来进行数据库的基本增删改等操作来加强理解。

### flask操作MySQL,参考这篇文章: <https://juejin.cn/post/6844904030892900366>

# 4.flask框架操作MySQL数据库,通过一个案例来学习

## 1>首先我们需要确定我们的flask的web服务,当用户访问一个路由,我们返回一个页面,让他输入用户名,密码和手机号,我们用朋友charm新建一个flask项目,起名:flask\_site\_demo

|  |
| --- |
|  |
|  |

## 2>我们添加一个/add/user路由处理函数

|  |
| --- |
| **from** flask **import** Flask, render\_template  app = Flask(\_\_name\_\_)   @app.route(**'/'**) **def** hello\_world(): *# put application's code here* **return 'Hello World!'** @app.route(**'/add/user'**) **def** add\_user(): *# put application's code here* **return** render\_template(**"add\_user.html"**)   **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run() |

## 3>然后需要在template文件夹里面新建一个add\_user.html模板文件,在里面放置一个表单来收集用户数据

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>添加新用户</**title**> </**head**> <**body**>  <**h2**>添加用户</**h2**>  <**form action="/add/user" method="post"**>  账号:<**input type="text" name="user" placeholder="用户名"**> <**br**>  密码:<**input type="password" name="pwd" placeholder="密码"**> <**br**>  手机:<**input type="text" name="mobile" placeholder="手机号"**> <**br**>  <**input type="submit" value="注册"**>  </**form**> </**body**> </**html**> |

## 4>因为我们get路由和post路由都是/add/user,我们需要在add\_user路由函数中根据访问的方式来做不同处理,注意在flask中,如果你需要一个路由能够接受get和post请求,你需要在methods参数里面添加一个由get和post组成的列表,并且需要使用request对象来获取.

|  |
| --- |
| **from** flask **import** Flask, render\_template,request  app = Flask(\_\_name\_\_)   @app.route(**'/'**) **def** hello\_world(): *# put application's code here* **return 'Hello World!'** @app.route(**'/add/user'**,methods=[**'GET'**,**'POST'**]) **def** add\_user(): *# put application's code here* **if** request.method==**"GET"**: *#如果是get请求* **return** render\_template(**"add\_user.html"**)  *# 如果是post请求* user = request.form.get(**"user"**)  pwd = request.form.get(**"pwd"**)  mobile = request.form.get(**"mobile"**)  print(**f"user:{**user**},password:{**pwd**},Phone:{**mobile**}"**)  **return "提交数据成功"  if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run() |

## 5>运行项目,然后输入/add/user,此时第一次是get请求,就会渲染add\_user.html,返回一个form表单给用户

|  |
| --- |
|  |

## 6>然后我们填写完整资料,然后点击注册按钮

|  |
| --- |
|  |

## 7>此时走的就是post请求,会调用下面的代码(get方式处理代码以外的代码)获取数据并且在控制台输出信息,同时给用户返回提交数据成功的字符串

|  |
| --- |
|  |
|  |

### 现在我们要做的就是吧用户提交的数据保存到数据库,需要先链接数据库,那么,flask然后连接数据库?

## 8>我们先导入pymysql,然后添加链接数据库,保存数据和关闭连接的代码

|  |
| --- |
| **from** flask **import** Flask, render\_template,request **import** pymysql  app = Flask(\_\_name\_\_)   @app.route(**'/'**) **def** hello\_world(): *# put application's code here* **return 'Hello World!'** @app.route(**'/add/user'**,methods=[**'GET'**,**'POST'**]) **def** add\_user(): *# put application's code here* **if** request.method==**"GET"**: *#如果是get请求* **return** render\_template(**"add\_user.html"**)  *# 如果是post请求* user = request.form.get(**"user"**)  pwd = request.form.get(**"pwd"**)  mobile = request.form.get(**"mobile"**)  print(**f"user:{**user**},password:{**pwd**},Phone:{**mobile**}"**)  *# 保存数据到MySQL数据库中  #1.连接MySQL* conn = pymysql.connect(host=**"127.0.0.1"**,port=3306, user=**"root"**,passwd=**"root"** ,charset=**'utf8'**,db=**"unicom"**)  cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)  *#2.保存数据* sql = **"insert into admin(username,password,mobile) values(%s,%s,%s)"** cursor.execute(sql,[user,pwd,mobile])  conn.commit() *# 一定要提交* print(**"插入数据成功"**)  *#3.关闭连接* cursor.close()  conn.close()  **return "提交数据成功"  if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run() |

## 9>然后我们在浏览器中输入/add/user,就会弹出表单,然后我们填写好表单点击提交就会出现一个信息说数据提交成功

|  |
| --- |
|  |
|  |
|  |

### 然后我们用navicat查看,发现用户数据创建成功

|  |
| --- |
|  |

## 10>然后我们给网站添加一个查询所有用户的功能,这个其实不难先添加一个/users路由,在这里获取所有用户数据,然后渲染一个叫做users.thml的模板文件,然后传递数据给模板

|  |
| --- |
|  |

## 11>然后在template文件夹里面新建一个users.html,在这里渲染我们传递进来的数据

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>用户列表</**title**>  <**style**>  **table**{  **width**: 60%;  **border-collapse**: **collapse**;  **margin**: 80**px auto**;  }  **th**,**td**{  **border**: 1**px solid #ccc**;  **padding**: 5**px** 8**px**;  }  </**style**> </**head**> <**body**> <**table**>  <**caption**><**h2**>用户列表</**h2**></**caption**>  <**thead**>  <**tr**>  <**th**>ID</**th**>  <**th**>Name</**th**>  <**th**>Password</**th**>  <**th**>Phone</**th**>  </**tr**>  </**thead**>  <**tbody**>  {% **for user in users** %}  <**tr**>  <**td**>{{ **user**.**id** }}</**td**>  <**td**>{{ **user**.**username** }}</**td**>  <**td**>{{ **user**.**password** }}</**td**>  <**td**>{{ **user**.**mobile** }}</**td**>  </**tr**>  {% **endfor** %}   </**tbody**>  </**table**> </**body**> </**html**> |

## 12>需要重新启动服务器,在浏览器中输入/users就可以显示所有用户数据

|  |
| --- |
|  |

### 其实在真实开发中不应该显示密码,这个只不过是练习,所以无所谓.

## 13>为了让页面更好看,我们可以使用bootstrap来美化表格,比如我们把用户列表美化一下,需要先引入bootstrap的样式然后引入bootsrap的js修改后的代码如下

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>用户列表</**title**> *{# <style>#} {# table{#} {# width: 60%;#} {# border-collapse: collapse;#} {# margin: 80px auto;#} {# }#} {# th,td{#} {# border: 1px solid #ccc;#} {# padding: 5px 8px;#} {# }#} {# </style>#}* <**link rel="stylesheet" href="/static/plugins/bootstrap-3.4.1/css/bootstrap.min.css"**> </**head**> <**body**> <**div class="container"**>  <**table class="table table-bordered"**>  <**caption**><**h2**>用户列表</**h2**></**caption**>  <**thead**>  <**tr**>  <**th**>ID</**th**>  <**th**>Name</**th**>  <**th**>Password</**th**>  <**th**>Phone</**th**>  </**tr**>  </**thead**>  <**tbody**>  {% **for user in users** %}  <**tr**>  <**td**>{{ **user**.**id** }}</**td**>  <**td**>{{ **user**.**username** }}</**td**>  <**td**>{{ **user**.**password** }}</**td**>  <**td**>{{ **user**.**mobile** }}</**td**>  </**tr**>  {% **endfor** %}   </**tbody**>   </**table**> </**div**> <**script src="/static/js/jquery3.7.1.min.js"**></**script**> <**script src="/static/plugins/bootstrap-3.4.1/js/bootstrap.min.js"**></**script**> </**body**> </**html**> |

## 效果如下,感觉还没有我自己写的好

|  |
| --- |
|  |

## 14>把这个项目改名: flask\_site\_demo\_pymysql,然后push到GitHub