

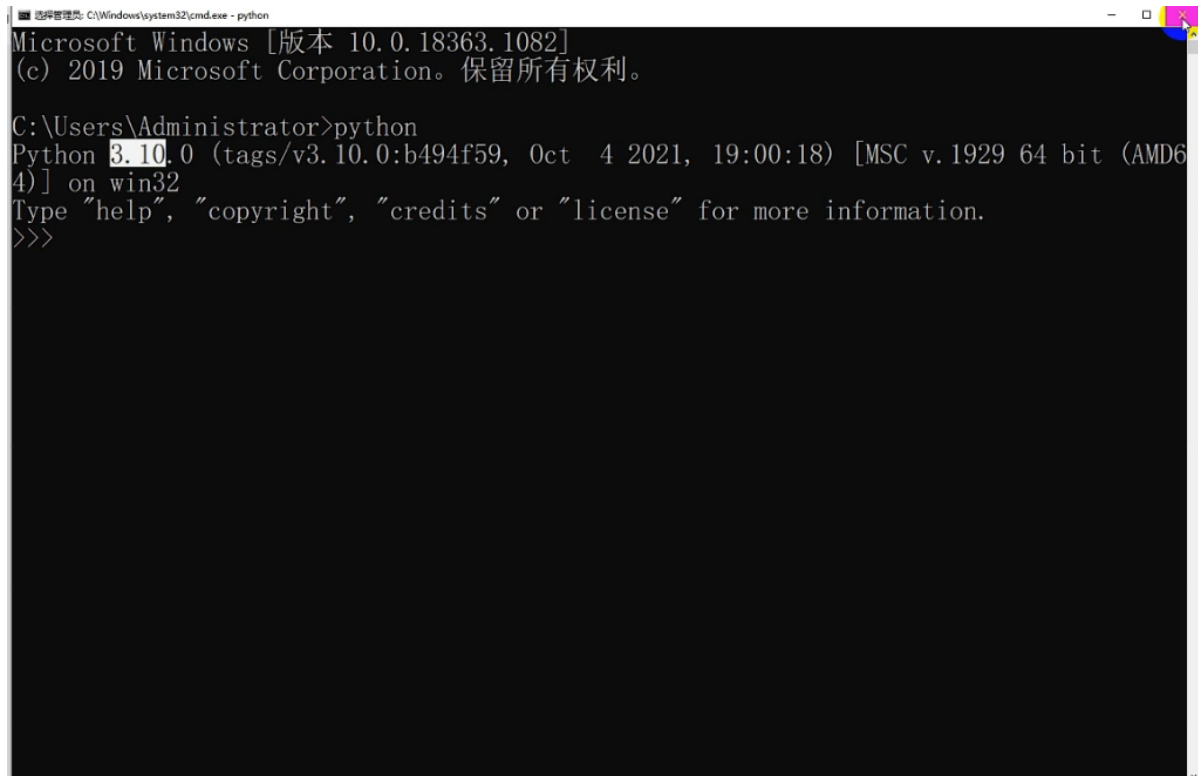
Flask

Flask是一个使用 Python 编写的轻量级 Web 应用框架。其 WSGI 工具箱采用 Werkzeug，模板引擎则使用 Jinja2。Flask使用 BSD 授权。

Flask是一个轻量级的可定制框架，使用Python语言编写，较其他同类型框架更为灵活、轻便、安全且容易上手。它可以很好地结合MVC模式进行开发

一、flask安装

1.先安装python环境和编译器（我使用的是pycharm）



```
Microsoft Windows [版本 10.0.18363.1082]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>python
Python 3.10.0 (tags/v3.10.0:b494f59, Oct 4 2021, 19:00:18) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

python版本尽量在3.9以上

2.安装flask

命令行输入pip install flask

```
管理员: C:\Windows\system32\cmd.exe - pip install flask
Microsoft Windows [版本 10.0.18363.1082]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>pip install flask
```

3.创建flask项目

在pycharm中新建一个flask项目

路由route的创建

- 通过创建路由并关联函数，实现一个基本的网页：

```
1  from flask import Flask
2
3  #使用Flask类创建一个app对象
4  #__name__表示当前这个app.py这个模块
5  app = Flask(__name__)
6
7  #创建路由和函数的映射(绑定视图函数)
8  # 直属的第一个作为视图函数被绑定，第二个就是普通函数
9  # 路由与视图函数需要一一对应
10 @app.route('/')
11 def hello_world(): # put application's code here
12     return 'Hello World!'
13
14
15 ▶ if __name__ == '__main__':
16     # 启动一个本地开发服务器，激活该网页
17     app.run()
18
```

- 通过路由的methods指定url允许的请求格式：

```

52  #methods参数用于指定允许的请求格式
53  #常规输入url的访问就是get方法
54  @app.route(rule: "/hello", methods=['GET', 'POST'])
55  def hello():
56      return "Hello World!"
57  #注意路由路径不要重名，映射的视图函数也不要重名
58  @app.route(rule: "/hi", methods=['POST'])
59  def hi():
60      return "Hi World!"

```

- 通过路由在url内添加参数，其关联的函数可以接收该参数：

```

23  # 可以在路径内以/<参数名>的形式指定参数，默认接收到的参数类型是string
24
25  '''#####
26  以下为框架自带的转换器，可以置于参数前将接收的参数转化为对应类型
27  string 接受任何不包含斜杠的文本
28  int 接受正整数
29  float 接受正浮点数
30  path 接受包含斜杠的文本
31  #####'''
32
33  @app.route("/index/<int:id>")
34  def index(id):
35      if id == 1:
36          return 'first'
37      elif id == 2:
38          return 'second'
39      elif id == 3:
40          return 'thrid'
41      else:
42          return 'hello world!'

```

4.渲染html文件

- render_template(): 可以用于呈现一个我们编写的html文件模板
- request.method用于获取url接收到的请求方式，以此返回不同的响应页面

```
PycharmProject D:\PycharmPro
├── static
├── templates
│   ├── hello.html
│   └── app.py
└── Flask.md
External Libraries
Scratches and Consoles

67 #request: 包含前端发送过来的所有请求数据
68
69 from flask import Flask,render_template,request
70
71 # 用当前脚本名称实例化Flask对象，方便flask从该脚本文件中获取需要的内容
72 app = Flask(__name__)
73
74 @app.route(rule: "/",methods=['GET','POST'])
75 #url映射的函数，要传参则在上述route（路由）中添加参数申明
76 def index():
77     if request.method == 'GET':
78         # 想要html文件被该函数访问到，首先要创建一个templates文件，将html文件放入其中
79         # 该文件夹需要被标记为模板文件夹，且模板语言设置为jinja2
80         return render_template('index.html')
81     # 此处欲发送post请求，需要在对应html文件的form表单中设置method为post
82     elif request.method == 'POST':
83         name = request.form.get('name')
84         password = request.form.get('password')
85         return name+" "+password
86
87 if __name__ == '__main__':
88     app.run()
```

```
115 # 给前端模板传参
116 @app.route("/")
117 def index():
118     data = {
119         'name': '张三',
120         'age': 18,
121         'mylist': [1, 2, 3, 4, 5, 6, 7]
122     }
123     # 以键值对的形式传参给模板index2.html
124     # 左边是我们要在前端调用时使用的变量名称（形参：data）；
125     # 右边是我们给这个变量传的值（实参：字典data）；
126     return render_template(template_name_or_list: 'index2.html', data=data)
```

前端html模板内需要在双括号{{ }}中使用该变量：

5.返回json数据给前端

- jsonify库实现，减少代码行数

```

92     from flask import Flask, jsonify
93
94     app = Flask(__name__)
95     # 在Flask的config是一个存储了各项配置的字典
96     # 该操作是进行等效于ensure_ascii=False的配置
97     app.config['JSON_AS_ASCII'] = False
98
99     @app.route("/index")
100    def index():
101        data = {
102            'name': '张三'
103        }
104        return jsonify(data)
105
106    if __name__ == '__main__':
107        app.run()

```

6.flask访问数据库

- SQLAlchemy是一个基于Python实现的ORM (Object Relational Mapping, 对象关系映射) 框架。该框架建立在DB API (数据库应用程序接口系统) 之上, 使用关系对象映射进行数据库操作。简言之便是将类和对象转换成SQL, 然后使用数据API (接口) 执行SQL 并获取执行结果。
- 它的核心思想在于将关系数据库表中的记录映射成为对象, 以对象的形式展现, 程序员可以把对数据库的操作转化为对对象的操作。
- 安装: `pip install flask-sqlalchemy`
- 如果映射所连接的是MySQL数据库, 还要事先安装好PyMySQL: `pip install pymysql`

初始化数据库配置

要使用SQLAlchemy连接数据库, 必须要进行必要的初始化配置后才能实现, 数据库配置文件一般要求独立成一个文件, 便于管理和移植;

配置文件: config.py

```

USERNAME = 'root' #设置登录账号
PASSWORD = '123456' #设置登录密码
HOST = '127.0.0.1' #设置主机地址
PORT = '3306' #设置端口号
DATABASE = 'flaskdb' #设置访问的数据库

# 创建URI（统一资源标志符）
'''
SQLALCHEMY_DATABASE_URI的固定格式为：
'{数据库管理系统名}://{登录名}:{密码}@{IP地址}:{端口号}/{数据库名}?charset={编码格式}'
'''
DB_URI = 'mysql://{}:{}@{}/{}?charset=utf8'.format(*args: USERNAME,PASSWORD,HOST,PORT,DATABASE)

# 设置数据库的连接URI
SQLALCHEMY_DATABASE_URI = DB_URI
# 设置动态追踪修改,如未设置只会提示警告
SQLALCHEMY_TRACK_MODIFICATIONS = False
# 设置查询时会显示原始SQL语句
SQLALCHEMY_ECHO = True

```

上述配置文件设置完后，在flask程序文件下导入该文件，再用app.config.from_object方法导入到flask对象内即可；

```

from flask import Flask
from flask_sqlalchemy import SQLAlchemy
import flask.config # 导入配置文件

app = Flask(__name__)
# 导入配置文件至flask对象
app.config.from_object(flask.config)

# 初始化一个SQLAlchemy对象
db = SQLAlchemy(app)
# 测试数据库连接是否成功（create_all将我们定义的所有表类映射为数据库下的表）
db.create_all()

```

创建实体类

```

# 创建表模型类对象
1 usage
class Book(db.Model):
    __tablename__ = 'book'
    id = db.Column(db.Integer, primary_key = True, autoincrement = True) #定义id字段
    title = db.Column(db.String(50), nullable = False) #定义title字段
    publishing_office = db.Column(db.String(100), nullable = False) #定义出版社字段

if __name__ == '__main__':
    # 删除数据库下的所有上述定义的表，防止重复创建
    db.drop_all()
    # 将上述定义的所有表对象映射为数据库下的表单（创建表）
    db.create_all()

```

数据的增删改查操作

• 增

```
# 添加数据的路由
@app.route('/add')
def add_record():
    book1 = Book(title='c语言程序设计', publishing_office='人民邮电出版社', price = '68.30 ')
    book2= Book(title='Python游戏编程快速上手第4版', publishing_office = '人民邮电出版社', price = '54.50')
    book3 = Book(title='数据结构', publishing_office = '清华大学出版社', price = '68.30')

    db.session.add(book1)
    db.session.add(book2)
    db.session.add(book3)
    # 需要提交事务给数据库
    db.session.commit()
    return 'add success!'
```

• 查

```
# 查找数据的路由
@app.route('/query')
def query_record():
    # 查找id=1的第一个对象
    result = Book.query.filter(Book.id == '1').first()
    print(result.title)
    # 查找publishing_office=人民邮电出版社的全体对象
    result_list = Book.query.filter(Book.publishing_office == '人民邮电出版社').all()
    for books in result_list:
        print(books.title)
    return 'query success!'
```

• 改

```
# 修改数据的路由
@app.route('/edit')
def edit_record():
    # 查找id=1的第一个对象
    book1 = Book.query.filter(Book.id == '1').first()
    book1.price = 168
    # 需要提交事务给数据库
    db.session.commit()
    return 'edit success!'
```

- 删

```
# 删除数据的路由
@app.route('/delete')
def delete_record():
    # 查找id=9的第一个对象
    book2 = Book.query.filter(Book.id == '9').first()
    db.session.delete(book2)
    # 需要提交事务给数据库
    db.session.commit()
    return 'delete success!'
```