**1.Flask-Script介绍：**

Flask-Script的作用是可以通过命令行的形式来操作Flask。

例如通过命令跑一个开发版本的服务器、设置数据库，定时任务等。

**2.Flask-Script安装：**

pip install flask-script

**3.Flask-Script基本使用：**

**1.** 看一个小例子，在flask项目中，建一个manage.py文件，代码如下：

|  |
| --- |
| **from** flask\_script **import** Manager  **from** momo **import** app  manager = Manager(app)  #1.通过命令执行  @manager.command  **def** hello():  print(**'你好,hello'**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  manager.run() |

其中的hello功能函数我们希望通过命令来运行，

然后在dos系统先进入装好Flask-Script的虚拟环境

运行python manage.py hello命令，就可以看hello函数已经执行了。

**2.** 再看一个例子，若执行命令脚本时，想传递某些参数过去，如下：

|  |
| --- |
| **from** flask\_script **import** Manager  **from** momo **import** app  manager = Manager(app)  #2.通过命令执行 并要求传递参数  @manager.option(**"-p"**,**"--province"**,dest=**"province"**)  @manager.option(**"-m"**,**"--month"**,dest=**"month"**)  **def** get\_pm\_data(province,month):  print(**"您请求的省份是：%s，月份是：%s"** % (province,month))  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  manager.run() |

**4.Flask-Script实战场景**

老板要求给某员工快速建立一个后台账号。

分析：可通过命令执行 并传参的方式 将参数添加到数据库中

**数据库连接config.py文件代码为**：

|  |
| --- |
| HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'flask\_script\_demo'**  USERNAME = **'root'**  PASSWORD = **'root'**  DB\_URI =**"mysql+pymysql://{username}:{password}@{host}:{port}/{db}?charset=utf8"**.format(username=USERNAME,password=PASSWORD,host=HOSTNAME,port=PORT,db=DATABASE)  SQLALCHEMY\_DATABASE\_URI = DB\_URI  SQLALCHEMY\_TRACK\_MODIFICATIONS = **False** |

**主运行文件momo.py代码为：**

|  |
| --- |
| **from** flask **import** Flask  #实战  **from** flask\_sqlalchemy **import** SQLAlchemy  **import** config  app = Flask(\_\_name\_\_)  app.config.from\_object(config)  db = SQLAlchemy(app)  **class** BackendUser(db.Model):  \_\_tablename\_\_ = **'backend\_user'**  id = db.Column(db.Integer,primary\_key=**True**,autoincrement=**True**)  uname = db.Column(db.String(50),nullable=**False**)  email = db.Column(db.String(50),nullable=**False**)  db.create\_all()  @app.route(**'/'**)  **def** hello\_world():  **return 'Hello World!'**  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  app.run() |

**存放命令的脚本文件manage.py代码为：**

|  |
| --- |
| **from** flask\_script **import** Manager  **from** momo **import** app,BackendUser,db  manager = Manager(app)  #3.Flask-Script实战场景 ：老板要求给某员工快速建立一个后台账号  @manager.option(**"-u"**,**"--uname"**,dest=**"uname"**)  @manager.option(**"-e"**,**"--email"**,dest=**"email"**)  **def** add\_user(uname,email):  user = BackendUser(uname=uname,email=email)  db.session.add(user)  db.session.commit()  print(**"添加OK"**)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  manager.run() |

**5.写命令脚本技巧**

如果有一些命令是针对某个功能的。

如有一堆命令是针对ORM与表映射的，那么可以将这些命令单独放在一个文件中方便管理。

然后到主脚本文件manage.py中，通过`Manager`的对象名.add\_command`方法来添加。

**如创建一个db\_script.py文件 管理跟操作数据库表相关的东西，代码如：**

|  |
| --- |
| **from** flask\_script **import** Manager  db\_manager = Manager()  @db\_manager.command  **def** init():  print(**'迁移仓库创建OK！'**)  @db\_manager.command  **def** revision():  print(**'迁移脚本文件生成OK！'**)  @db\_manager.command  **def** upgrade():  print(**'迁移脚本文件映射到数据库OK！'**) |

**存放命令的脚本文件manage.py代码为：**

|  |
| --- |
| **from** flask\_script **import** Manager  **from** momo **import** app,BackendUser,db  manager = Manager(app)  #4.如果有一些命令是针对某个功能的。  # 比如有一堆命令是针对ORM与表映射的，那么可以将这些命令单独放在一个文件中方便管理。  # 也是使用`Manager`的对象来添加。然后到主manage文件中，通过`manager.add\_command`来添加。  **from** db\_script **import** db\_manager  manager.add\_command(**"db"**,db\_manager)  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  manager.run() |