dd47b3f7e4947beb4c0edb1f303538f~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=477&h=281&s=20040&e=png&b=2c2c2c)

4.2 数据库配置 sqlalchemy

对于数据库的配置，可以通过配置文件的方式来设置，具体的配置如下图所示：

```
# 配置方式
app.config[”SQLALCHEMY_TRACK_MODIFICATIONS”] = True
# 配置文件的方式
app.config.from_object(config)
```


对于数据库的操作，可以使用 pymysql 的方式来操作数据库，对于脚本来说是合适的，但是对于 web 开发来说，还是需要采用面向对象的方式，这里使用的 orm 框架是 SQLAlchemy , 对于 django 框架来说也是同样的。

数据库实体类需要集成db.Model, 在实体类中可以设置数据库的表名称，字段名称，类型，默认值，注释，主键，索引，约束键等信息。数据库采用的是下划线的命名方式，对应的实体是驼峰命名。

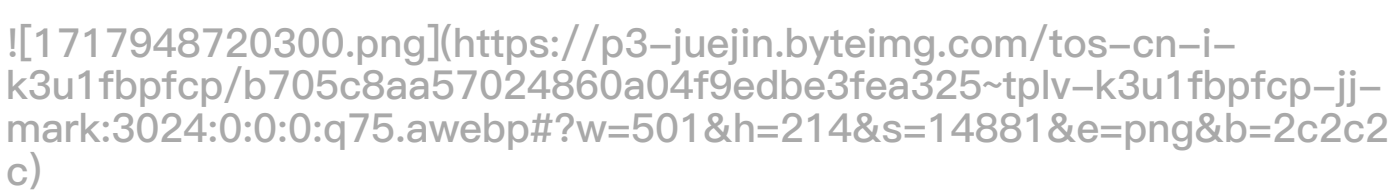
```
class User(db.Model):
```

```
    __tablename__ = "tb_user"
    id = db.Column(db.Integer, primary_key=True, autoincrement=True, comment='主键')
    userId = db.Column("user_id", db.String(32), default="", unique=True, comment='用户Id')
    username = db.Column(db.String(32), default="", comment='用户')
    email = db.Column(db.String(32), default="", nullable=False, comment='邮箱')
    cellphone = db.Column(db.String(32), default="", comment='手机号')
    status = db.Column(db.String(32), default="",  comment='状态')
    seq = db.Column(db.Integer, default="", comment='seq')
    createTime = db.Column("create_time", db.DateTime, default=datetime.now, comment='创建时间')
    updateTime = db.Column("update_time", db.DateTime, default=datetime.now, comment='更新时间')

```

4.3 热加载

在开发模式下，需要开启热加载的模式，方便开发的调试，通过以下的方式可以实现热加载。



5 数据库操作

```
数据库的操作是 t=db_config['port'], charset=db_config['charset'])
        return __pool.connection()

    # 数据插入\更新\删除sql
    def op_update(self, sql, param):
        print('op_insert', sql, param)
        insert_num = self.cur.execute(sql, param)
        # commit 请求
        self.coon.commit()
        return insert_num

    # 数据查询
    def op_query(self, sql, parm):
        print('op_select', sql, parm)
```

```
        self.cur.execute(sql, parm)  # 执行sql
        select_res = self.cur.fetchall()  # 返回结果为字典
        return select_res

    # 批量更新数据
    def op_update_list(self, list):
        sum = 0
        try:
            for tuple in list:
                sql, parm = tuple
                insert_num = self.cur.execute(sql, parm)
                if (insert_num == 0):
                    self.coon.rollback()
                    break
                sum += insert_num
            self.coon.commit()
        except:
            print('事务回滚')
            self.coon.rollback()
        return sum
        # 释放资源

    # 释放资源
    def dispose(self):
        self.coon.close()
        self.cur.close()
if __name__ == '__main__':
    # 数据库连接配置
    db_config = {
        "host": 'localhost',
        "user": 'root',
        "passwd": '123456',
        "db": 'account',
        "port": 3306,
        "charset": 'utf8mb4'
    }

    db = DbPool(db_config)
    dt_list = db.op_query("select * from user where userid > %s order by
userid desc limit 1", (23))
    for nd in dt_list:
        print(nd)

    res = db.op_update("update user set register_date = %s where userid
> %s and userid < %s", ("2024-04-19 16:42:50", 17702, 17705))
    print("res is ", res)

```
```
```

8 总结

在本文中详细介绍了 `flask` 框架搭建 `web` 项目的全部流程以及注意事项，相对而言`flask`比较简单容易上手，主要是前端的配置和数据库的配置以及增删改查等操作。本文中所涉及的代码已经上传至 `github`, 欢迎交流学习。项目地址 [flask_web](http://cxyroad.com/”https://gitee.com/xieyue86/flask_web.git”)。

原文链接: https://juejin.cn/post/7382525876032552998