



公众号系统介绍及其中间件分析

课程报告

课程名称：中间件实现技术

指导老师：胡鹤

学 生：马晓臣

学 号：2019104253

时 间：2020年10月31日

1.系统总体介绍	3
2.所做工作.....	3
3.如何运行.....	3
4.系统功能实现	4
5.中间件——session.....	7
6.中间件——sqlalchemy-ORM&dbSession	9

1.系统总体介绍

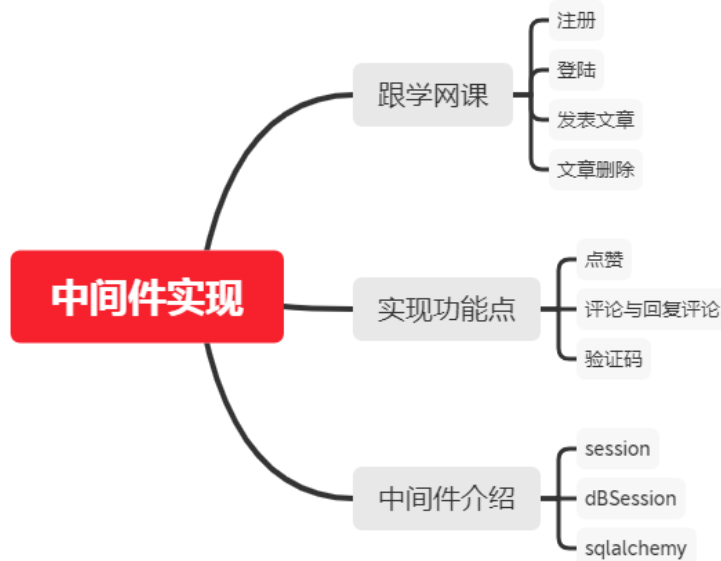
系统基于 Flask 框架，目前实现的功能主要包括注册、登陆、发表文章、文章删除、点赞、评论与回复评论、验证码。

模块：公众号模块和普通用户模块，用户模块只能点赞、评论、浏览，公众号模块可以有全部功能。

2.所做工作

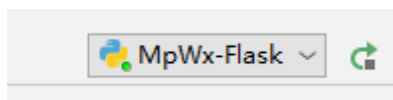
因为自己跨专业，基础太弱，不太会开发，需要一些系统性学习。所以主要展开了以下的工作：跟着网课敲代码，实现一些基本的功能；自己尝试实现一些功能点；根据课上所学，研究该项目用到了什么中间件，展开介绍，如下图所示：

注：“跟学网课”的部分是网课现成的；“实现功能点”这部分是自己写的几个函数；“中间件介绍”是结合课程所讲进行的分析，也是该报告的重点。



3.如何运行

先安装相关的包，具体在 requirements.txt 中，使用 Python3.6 版本运行。本地运行，打开系统后，单击此处：



可以在 <http://127.0.0.1:5000/>查看系统。

4.系统功能实现

4.1 注册，注册分为普通用户和公众号注册，代码如下：

```
# 注册 添加用户
def add_user(form_data):
    datas = {"code": "0", "msg": ""}
    if form_data['utype'] == "0": # 这是普通用户注册
        form_data.pop("utype", None) #pop() 方法删除字典给定键 key 及对应的值，返回值为被删除的值。
        form_data.pop("f_head", None) #f_head: 头像
        res = Users().add(User, form_data) #增加用户
        if res:
            datas["code"] = "1"
        else:
            datas["msg"] = "注册失败，昵称或者邮箱已被注册!"
    else:
        form_data["utype"] = 1 # 这是公众号注册
        if form_data['f_head']:
            f_len = len(form_data['f_head'].read())
            if f_len <= 1048576: #头像大小小于1M
                img = Image.open(form_data['f_head'])
                if img.format not in ("JPEG", "JPG", "PNG"):
                    datas['msg'] = "上传的头像图片文件类型不正确!"
                elif img.size[0] > 100 or img.size[1] > 100:
                    datas['msg'] = "上传的头像图片像素超过100*100px!"
                else:
                    f_name = next_id() #生成图片文件名
                    head = f_name + "." + img.format
                    img.save(baseDir + "/static/head/" + head)
                    form_data.pop("f_head", None)
                    form_data["head"] = head #更名
                    res = Users().add(User, form_data)
                    if res:
                        datas["code"] = "1"
                    else:
                        datas["msg"] = "注册失败，昵称或者邮箱已被注册!"
            else:
                datas['msg'] = "上传的头像图片文件大小超过1MB!"
        else:
            form_data.pop("f_head", None)
            form_data["head"] = ""
            res = Users().add(User, form_data)
            if res:
                datas["code"] = "1"
            else:
                datas["msg"] = "注册失败，昵称或者邮箱已被注册!"
    return datas
```

基本的注释已经写在代码中，再提一句的是头像图片不能大于 1MB，一个邮箱只能注册一个账号。

4.2 登陆，对应代码在 index.py 中 247-259 行，比较简单，不去赘述。

4.3 发表文章，特殊说明的是，如果没有摘要，会自动提取文章前 60 个字符作为摘要。

```
def add_article(form_data):
    # {"title", 't_img', 'b_img', 'template', 'source', "b_desc", "b_time", "body", "b_uid"}
    if not form_data['b_desc']: # 没有描述 取文章前60个字符
        #b_desc = 文章描述
        form_data['b_desc'] = form_data['body'][:60] #文章内容
    if form_data['t_img']: # 检查"标题"图片
        if len(form_data['t_img'].read()) <= 1048576: #图片大小必须小于特定值
            img = Image.open(form_data['t_img'])
            if img.format in ("JPEG", "JPG", "PNG") and img.size[0] < 200 and img.size[1] < 200: #文件大小类型进行限制
                f_name = next_id() #生成图片文件名
                t_img = f_name + "." + img.format #文件名赋值
                img.save(baseDir + "/static/img/" + t_img)
                form_data["t_img"] = t_img
    if form_data['b_img']: # 检查"文章"图片
        if len(form_data['b_img'].read()) <= 5242880:
            img = Image.open(form_data['b_img'])
            if img.format in ("JPEG", "JPG", "PNG") and img.size[0] < 2000 and img.size[1] < 2000:
                f_name = next_id()
                b_img = f_name + "." + img.format
                img.save(baseDir + "/static/img/" + b_img)
                form_data["b_img"] = b_img
    res = Articles().add(Article, form_data)
    return res
```

4.4 文章删除，这里要先核实是不是作者本人，再就是 likequery.is_del 是进行逻辑删除，避免很多麻烦。

```
@app.route("/delarticle", methods=["post", ])
def delarticle(): # 文章删除
    datas = {"code": "0", "msg": "文章没找到"}
    aid = request.form.get('id') #获取文章id 跟前端交互
    if aid.isnumeric():
        likequery = Articles().get(aid)
        if likequery:
            if likequery.b_uid == session['id']: # session['id']是加密的cookie 这个是存在
                #b_uid在Model中有定义 上一句是先核实一下是不是作者本人
                likequery.is_del = 1 #逻辑删除 并不是物理删除 只是查询的时候 带个这个字
                dbSession.commit()
                datas['code'] = "1"
                datas['msg'] = ""
            else: #不是作者本人
                datas['msg'] = "非法访问，您不是该文章的作者,无法删除!"
    return jsonify(datas)
```

4.5 点赞，对应代码在 index.py 中 102-115 行，比较简单，不去赘述。

4.6 评论与回复评论，评论要核实是否登陆以及验证码是否正确，回复评论是公众号用户特有的功能，所以要核实是不是作者。代码在 index.py 中 137-184 行，比较长，就不截图了。

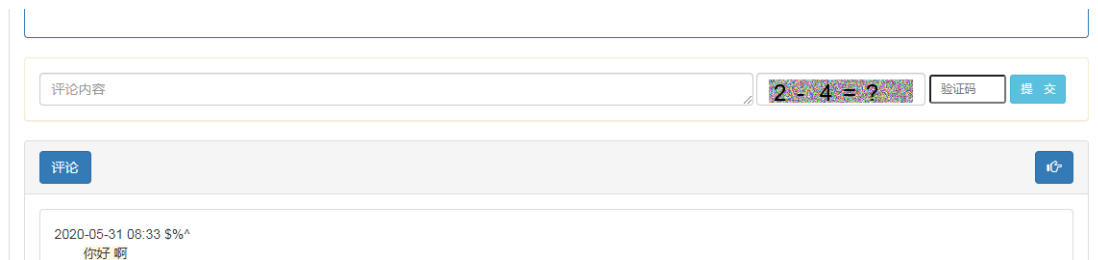
4.7 数据库，数据存储在本地的 Mysql 数据库中，截图如下：



The screenshot shows a MySQL database interface. On the left, a sidebar lists the database structure: 'test' (selected), 'mp', 'main', and '表' (Tables). Under '表', there are 'article', 'comments', and 'user'. The 'user' table is selected, and its data is displayed in a table on the right. The table has 5 columns: an ID, an email address, a username, a password hash, and a status. The data rows are numbered 6 to 11.

ID	Email	Username	Password Hash	Status
6	942198882@qq.com	飞火流星123456	pbkdf2:sha256:150000\$by	0
7	2019104253@ruc.edu.cn	飞火流星Messi	pbkdf2:sha256:150000\$Yl	1
8	2019104254@ruc.edu.cn	!@\$	pbkdf2:sha256:150000\$U	0
9	2019104255@ruc.edu.cn	\$%^	pbkdf2:sha256:150000\$C	1
10	2019104256@ruc.edu.cn	123	pbkdf2:sha256:150000\$e	1
11	2019104246@ruc.edu.cn	123456	pbkdf2:sha256:150000\$B	1

4.8 验证码与评论，代码在 index.py 中 344-393 行。需要填写验证码才能进行评论，确保评论是人进行的操作。把评论和验证码结合了起来，但是这样每一次评论会很麻烦，同学建议直接在登陆的时候填写验证码就可以。



The screenshot shows a web form for comments. It has a text input field for '评论内容' (Comment Content). To the right of the input field is a verification code box displaying '2 - 4 = ?'. Below the input field is a '评论' (Comment) button. Below the '评论' button is a timestamp '2020-05-31 08:33 \$%^' and the text '你好啊'.

验证码实现主要包括以下步骤：选择颜色，数学运算，创建 Draw 对象，填充元素，输出文字。

5.中间件——session

5.1 中间件定义：

中间件是一种独立的系统软件服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算资源和网络通信。

中间件是介于应用系统和系统软件之间的一类软件，它使用系统软件所提供的基础服务（功能），衔接网络上应用系统的各个部分或不同的应用，能够达到资源共享、功能共享的目的。

中间件=平台+通信。

5.2 Flask-session：

session 可以看做是在不同的请求之间保存数据的方法，因为 HTTP 是无状态的协议，但是在业务应用上我们希望知道不同请求是否是同一个人发起的。

5.3 原理：

当客户端进行第一次请求时，客户端的 HTTP request（cookie 为空）到服务端，服务端创建 session，视图函数根据 form 表单填写 session，请求结束时，session 内容填写入 response 的 cookie 中并返回给客户端，客户端的 cookie 中便保存了用户的数据。

公众号平台 登录和注册

登录 注册

邮箱

2019104246@ruc.edu.cn

- 2019104246@ruc.edu.cn
- 2019104256@ruc.edu.cn
- 2019104253@ruc.edu.cn
- 2019104254@ruc.edu.cn
- 942198882@qq.com

登录

当同一客户端再次请求时，客户端的 HTTP request 中 cookie 已经携带数据，视图函数根据 cookie 中值做相应操作（如已经携带用户名和密码就可以直接登陆）。

5.4 代码实现：

该系统通过用户访问 POST /login 来实现登陆，如下图所示：

```
#6
# 登录和注册视图
@app.route("/login/<act>", methods=["post", "get"])
def login(act): #
    if session.get('logged_in'):
        if session.get('utype') == 1:
            return redirect(url_for('articlemanage'))
        return redirect(url_for('mplist'))
    else:
        if request.method == 'POST':
            if act == "login": # 登录验证
```

5.5 存储与删除：

用户登陆的信息在本地的存储如下：

id	email	nick	pwd	uty
11	2019104246@ruc.edu.cn	123456	pbkdf2:sha256:150000\$B	
10	2019104256@ruc.edu.cn	123	pbkdf2:sha256:150000\$e	
9	2019104255@ruc.edu.cn	\$%^	pbkdf2:sha256:150000\$C	
8	2019104254@ruc.edu.cn	!@\$	pbkdf2:sha256:150000\$U	
7	2019104253@ruc.edu.cn	飞火流星Messi	pbkdf2:sha256:150000\$Y	

session 中的值可以通过以下三种方式删除：

- `session.pop(key)`。
- `del session[key]`。
- `session.clear()`：删除 session 中所有的值。

6.中间件——sqlalchemy-ORM&dbSession

6.1 定义：

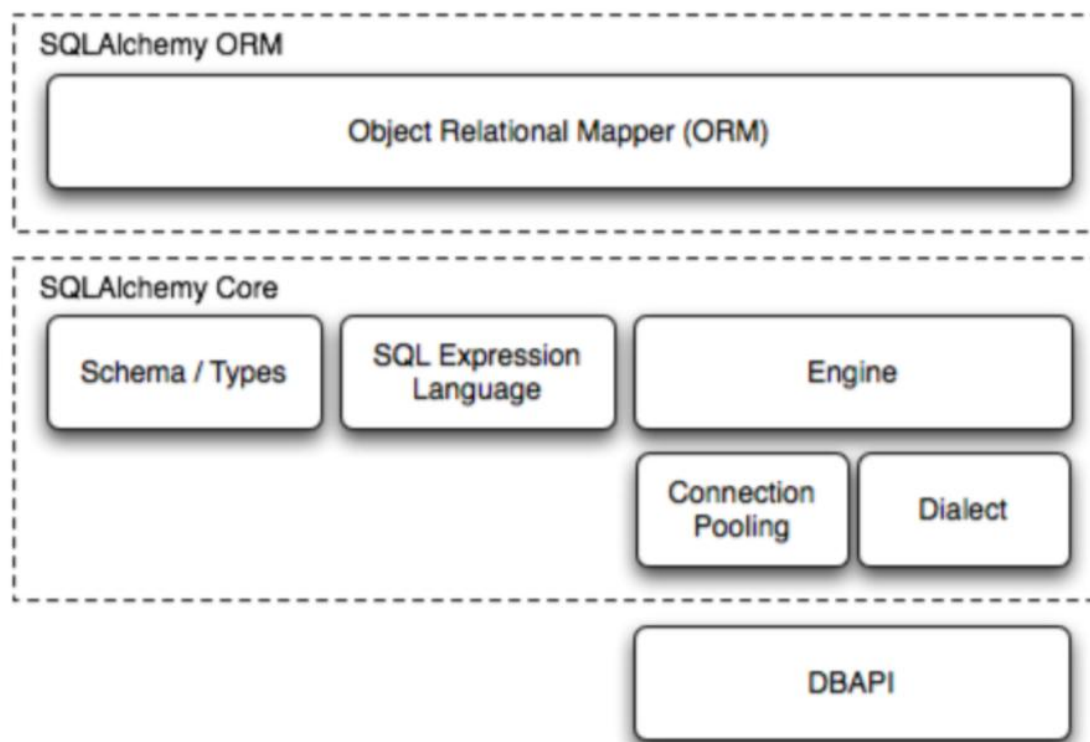
SQLAlchemy 是 Python 中一个通过 ORM 操作数据库的框架。他的作用就是能将数据库里面的实体类数据表映射成 python 里面的实体类，与 Django 的 ORM 框架各有优缺点。

SQLAlchemy 对象关系映射器提供了一种方法，用于将用户定义的 Python 类与数据库表相关联，并将这些类（对象）的实例与其对应表中的行相关联。它包括一个透明地同步对象及其相关行之间状态的所有变化的系统，称为工作单元，以及根据用户定义的类及其定义的彼此之间的关系表达数据库查询的系统。

ORM：一般指对象关系映射。

6.2 架构：

架构图如下图所示，其中 Schema/Types 定义了类到表之间的映射框架（规则），SQL Expression Language 封装好 SQL 语句，Engine 是操作者，Connection Pooling 是连接池，Dialect 根据用户的配置，调用不同的数据库 API(Oracle, postgresql, Mysql) 并执行对应的 SQL 语句。



6.3 代码实现：

安装，略。连接数据库，初始化，如下图：

```
# 创建数据库及连接
engine = create_engine(SQLALCHEMY_DATABASE_URI, echo=True)
# 创建DBSession类型:
DBSession = sessionmaker(bind=engine)
dBSession = DBSession()
```

添加记录操作：

```
def add(self, cls_, data: dict) -> int:
    """
    添加
    @param object cls_ 数据库模型实体类
    @param dict data 数据
    @return bool
    """
    users = cls_(**data)
    dBSession.add(users)
    dBSession.flush()
    return users.id
```

查询操作：

```
def getOne(self, cls_: object, filters: set, order: str = 'id desc', field: tuple = ()):
    """
    获取一条
    @param object cls_ 数据库模型实体类
    @param set filters 查询条件
    @param str order 排序
    @param tuple field 字段
    @return dict
    """
    res = dBSession.query(cls_).filter(*filters)
```

删除、更新操作类似，在此不赘述。

创建表，主要创建的表有三张：用户表，文章表，评论表。以评论表为例，建表需要用到的属性，以及效果如下图所示：

创建表：

```
# 评论表ORM模型
class Comment(Base):
    __tablename__ = 'comments'

    id = Column(Integer, primary_key=True)
    c_nick = Column(String(64), nullable=False)
    c_time = Column(String(32), nullable=False) # 发表时间
    c_retime = Column(String(32), nullable=False) # 回复时间
    c_body = Column(String(255), nullable=False) # 评论内容
    c_rebody = Column(String(255), nullable=False) # 回复内容
    is_del = Column(Integer, default=0) # 是否删除
    c_aid = Column(Integer, ForeignKey('article.id'))
    article = relationship("Article", backref='a2c')
```

数据库中存储评论的内容：

	id	c_nick	c_time	c_retime	c_body
▶	1	喜多钱	2020-05-05 16:00	(Null)	我就看看不说
	2	喜多钱	2020-05-05 17:00	2020-05-05 17:01	没人吗
	3	大王叫我来巡山	2020-05-08 18:06	2020-05-08 18:41	我是吃瓜群众
	4	喜多多	2020-05-08 19:32	(Null)	哈 哈
	5	飞火流星123456	2020-05-25 11:52	(Null)	233
	6	飞火流星123456	2020-05-27 21:54	(Null)	哈哈
	7	\$\$%^	2020-05-31 08:33	2020-05-31 08:34	你好 啊
	8	飞火流星123456	2020-05-31 08:35	(Null)	真好看
	9	123	2020-05-31 08:52	(Null)	打击好
	10	123	2020-05-31 08:54	(Null)	12345