day07:

1. `sqlalchemy.exc.OperationalError: (_mysql_exceptions.OperationalError) (2003, "Can't connect to MySQL server on '127.0.0.1' (111)")`: 抱着个错是因为数据库端口错了
2. `python manage.py db migrate` 只是迁移文件的生成, `python manage.py db upgrade` 才是生成表
3. 用一个专门的蓝图提供静态文件的返回: `web_html.py` 中写; 需要用到的自定义转换器定义在utils包中
4. 设置csrf_token的cookie: 后端通过 `current_app.send_static_file()` 向前端发送文件之后会有一个响应返回: `make_response()` 得到这个返回值resp; `resp.set_cookie("csrf_token", csrf_token)`
5. 用一个专门的py文件向前端返回静态页面: 程序中要写**设置csrf_token的部分**: 因为之前补充的 `csrf=CSRFProtect(app)` 做的只是校验cookie和body中的csrf_token是不是相等, 但是这个csrf_token是哪里来的呢? 是我们自己在相应体中自己设置的:

```
from flask import current_app, make_response
from flask_wtf.csrf import generate_csrf

csrf_token = generate_csrf()
resp = make_response(current_app.send_static_file(file_name))
resp.set_cookie("csrf_token", csrf_token)
return resp
```

1. 使用图片验证码的流程:
    a. 浏览器生成验证码编号
    b. 向后端: 携带验证码编号，请求获取一张图片验证码
    c. Flask生成验证码图片，保存验证码真实值和编号进redis, 返回验证码图片
    d. 浏览器请求获取短信验证码, 携带用户填写的图片验证码编号
    e. flask根据编号判断图片验证码的正确性,如果正确发送短信验证码
2. 服务器实际上是对**数据库的增删改查的具象表现**:对资源的操作，所以网址中应该表现出资源; T: POST/DELETE/PUT/GET代表转换"增删改查"; e.g. 对表goods的操作本来是定义为 `/get_goods, /create_goods, /update_goods, /delete_goods`, REST说:都是对goods同一种资源的操作: URL都是 `goods`; 此时的动词由访问方式是GET/POST/PUT/DELETE来表征:
    a. 动词通过前端的请求方式表现出来, 具体来讲是对资源使用状态的转换
    b. 后端只负责数据库资源的表征, 只是名词
    c. 浏览器路径表示API的网址, 对应一种资源/数据库的表名
    d. **所以RESTful特别适合前后端分离的场景**

3. **定义试图函数关键: 分析: 访问路径url&&访问方式(REST风格), 传入参数，返回值**
4. **图片验证码后端做法:
    a. 图片验证码这个: 生成图片验证码
    b. 保存验证码真实值和编号
    c. 返回验证码图片**
5. 编写接口文档: 功能描述, 访问方式, 传入参数, 返回值
6. **BUG**:导包: `ImportError: No module named xxx`: 是因为这个package中没有`__init__.py`, 要加入文件之后这个包才变成了python package, 才能导入
7. 往redis保存KV对要先开启redis: `redis-cli`
8. 浏览器生成图片编号:
    a. 时间戳
    b. UUID
9. 前端触发向后端发送请求; 写前端的register.js文件中的函数 `getImageCode()`: 在
    1. GET请求得到页面;
    2. `onClick`点击图片重新加载register.js文件
    3. 在浏览器中的Sources, Network看
10. **发送短信验证码: 用户-->flask-->云通讯
    a. 用户填写的图片验证码和编号, 用户手机号-->flask
    b. flask验证图片验证码, 成功就发送短信(指令给云通讯)
    c. flask携带参数: 发送短信模版id,模版需要的参数和用户手机号
    d. 云通讯发送短信结果给flask, 短信发送给用户**
11. 发送短信的SDK中的方法改成类的方式:
    a. 发送短信的初始化REST SDK做成单例模式:python中重写new方法: 因为只要创建一次对象就行了

```
class CCP(object):
    def __new__(cls):
        if not hasattr(cls, "instance"):
            obj = super(CCP, cls).__new__(cls)
            # 初始化SDK
            obj.rest = REST(serverIP, serverPORT, softVersion)
            obj.rest.setAccount(accountSid, accountToken)
            obj.rest.setAppId(appId)

            cls.instance = obj # 对象保存到属性中

        return cls.instance
```

1. 创建CCP对象之后，调用发送短信的方法

2. **定义试图函数关键: 分析: 访问路径url&&访问方式(REST风格), 传入参数，返回值**
3. 传递的参数都是资源的一部分，都应该写在对应的路径中; 不过参数也分为路径的一部分和 ? 之后的参数
4. 发送短信验证码之前要校验用户填写的`image_code, image_code_id`KV对是否匹配: **在flask中以get的方式接收到request的参数取出来: request.args.get()**

19'36''