

# Web 项目2-day01-课堂笔记

## Web 项目2-day01-课堂笔记

### 用户模块-用户注册功能

- 1.1 【理解掌握】图片验证码数据生成接口实现
- 1.2 【理解掌握】短信验证码生成接口实现
- 1.3 【理解掌握】Celery 异步任务和短信异步发送
- 1.4 【理解掌握】注册用户信息保存接口实现

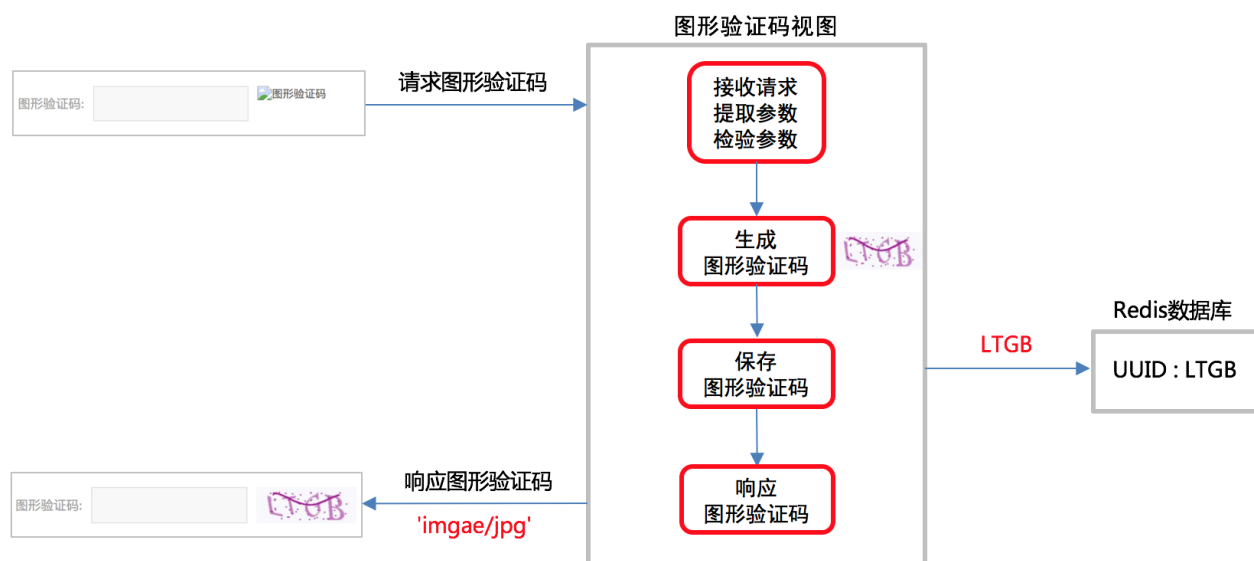
## 用户模块-用户注册功能

### 1.1 【理解掌握】图片验证码数据生成接口实现

思考问题：

- 1) 图片验证码文本内容应该存储到哪里？需不需要设置有效期？

业务逻辑：



API接口设计：

API: GET /image\_codes/(?P<uuid>[\w-]+)/

请求参数：

通过url地址传递唯一标识uuid

响应数据：

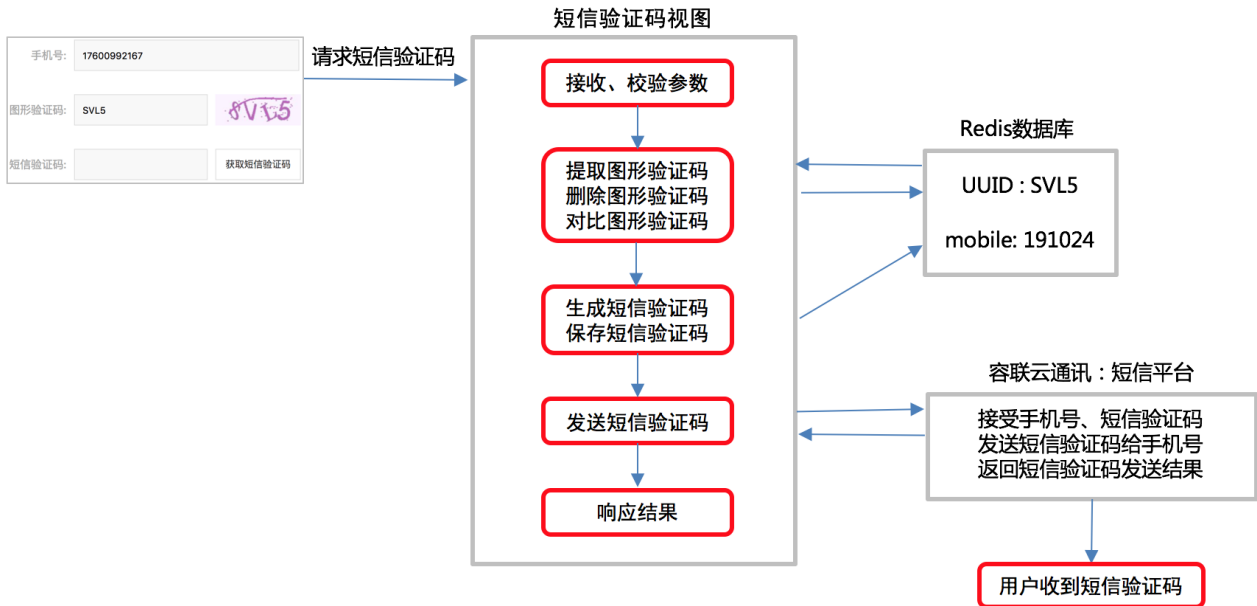
验证码图片数据

### 1.2 【理解掌握】短信验证码生成接口实现

思考问题：

- 1) 短信验证码发送接口设计时，需要有哪些参数？
- 2) 短信验证码发送接口实现的业务逻辑是什么？
- 3) 为什么要避免同一手机号短信60s内重复发送？

业务逻辑：



容联云通讯平台配置和 SDK 集成：

具体过程参考讲义

API接口设计：

```

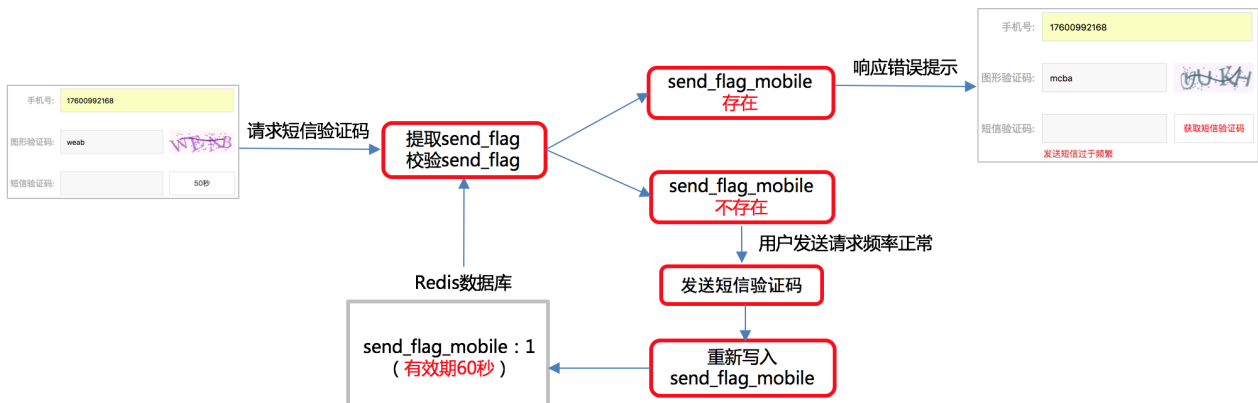
=====
API: GET /sms_codes/{?P<mobile>1[3-9]\d{9}}/
=====

请求参数：
    通过url地址传递mobile手机号
    通过查询字符串传递：
        "image_code": "图片验证码",
        "image_code_id": "图片验证码标识"
参数举例：
    /sms_codes/13566667788/?image_code=VU5M&image_code_id=5b3efb25-f1e3-482f-b8c9-507e72e315a3
=====

响应数据：
{
    "code": "响应码",
    "message": "提示信息",
    "count": "手机号数量"
}
响应举例：
{
    "code": 0,
    "message": "发送短信成功"
}
  
```

}

短信验证码60s内重复发送：



pipeline操作Redis数据库：

利用redis的pipeline管道机制，可以一次性发送多条命令，并在执行完后一次性将结果返回。pipeline通过减少客户端与Redis的通信次数来实现降低往返延时时间。

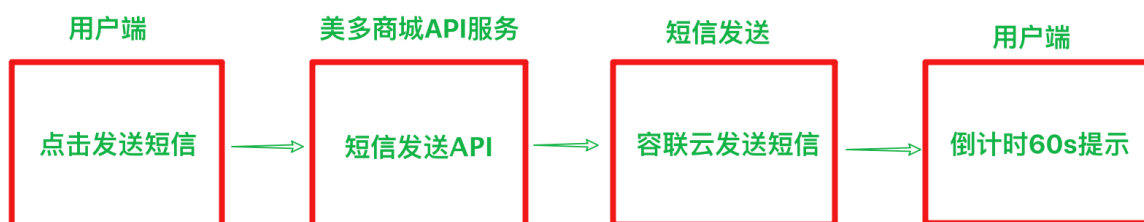
## 1.3 【理解掌握】Celery 异步任务和短信异步发送

思考问题：

- 1) 什么是短信同步发送，可能造成的问题是什么？
- 2) 什么是短信异步发送，异步发送怎么实现？

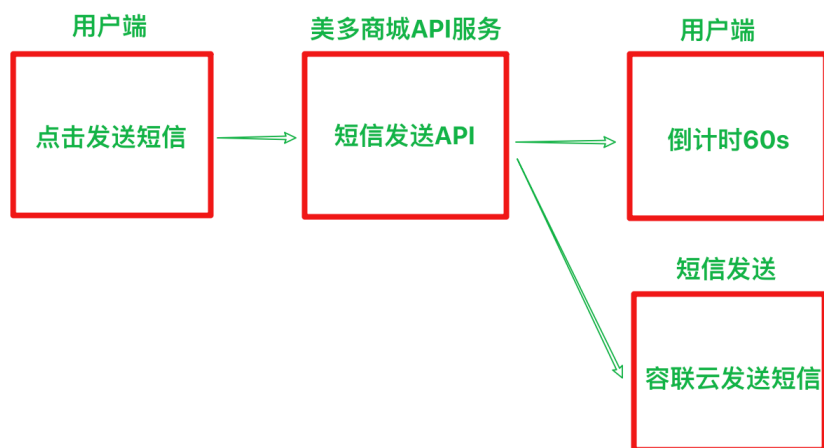
短信验证码同步发送的问题：

若短信发送时间过长，会造成用户长时间的等待，非常不好的用户体验。

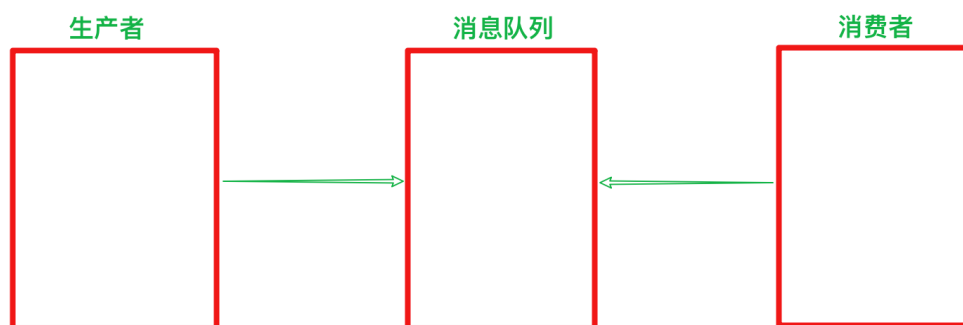


短信异步发送：

将短信发送从API的过程中剥离出来，实现短信发送和API主业务的解耦。

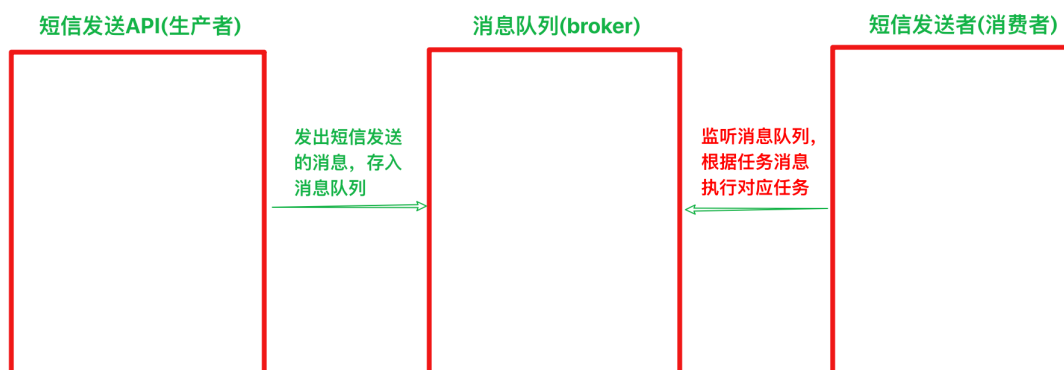


生产者和消费者模式：



生产者一方发出任务消息(通知)，任务消息存入消息队列，生产者继续做自己的事情  
消费者一方监听消息队列，取出对应的任务消息(通知)，根据消息去执行对应的任务

短信验证码发送的异步解决方案：



短信发送API一方(生产者)发出发送短信的任务消息,任务消息存入消息队列,然后短信发送API给客户端返回响应  
短发发送工作者一方(消费者)监听消息队列,取出对应的任务消息,根据消息进行短信验证码的实际发送

## Celery-分布式消息队列系统:

Celery是一个简单、灵活的分布式消息队列系统,使用python语言开发。Celery的本质就是生产者-消费者设计模式,使用Celery可以非常方便的实现异步任务的执行。

核心要素	作用说明
client	生产者,发出任务消息,存储到broker
broker	中间人,消息队列,存储任务消息,官方推荐使用rabbit-mq或redis
worker	消费者,任务执行者,根据任务消息执行对应的任务

## Celery基本使用和短信异步发送实现:

具体过程参考讲义

The screenshot illustrates the implementation of asynchronous SMS sending using Celery. It is divided into three main sections:

- User Registration Form (Left):** A web form with fields for '用户名' (Username), '密码' (Password), '确认密码' (Confirm Password), '手机号' (Mobile Number: 13155667788), '图形验证码' (Image Captcha: JHXP), and '短信验证码' (SMS Captcha: 33秒). A '登录' (Login) button is also present.
- Python Code (Middle):** A snippet from `views.py` showing the logic for sending an SMS:

```
# 这里短信发送的标识,有效则为1,0为0(添加到消息队列)
pl.set('send_flag_%s' % mobile, 1, 60)

# 执行redis pipeline请求
pl.execute()

# 发送短信验证码
# CCP().send_template_sms(mobile, [sms_code, 5], 1)
# 发出发送短信的任务消息, SMSCodeView这里的作用就是生产者
from celery_tasks.sms.tasks import send_sms_code
send_sms_code.delay(mobile, sms_code)

# ④ 返回响应结果 代码继续向下执行
return JsonResponse({'code': 0, ...})
```
- Terminal Log (Bottom):** Shows the Celery worker's execution:

```
[2020-09-13 13:08:38,495: INFO/MainProcess] Connected to redis://192.168.19.131:6379/3
[2020-09-13 13:08:38,505: INFO/MainProcess] mingle: searching for neighbors
[2020-09-13 13:08:39,534: INFO/MainProcess] mingle: all alone
[2020-09-13 13:08:39,552: INFO/MainProcess] celery@smart ready.

[2020-09-13 13:11:47,729: INFO/MainProcess] Received task: send_sms_code[32d0a76a-4629-426c-8091-1db1ba3524ce]
[2020-09-13 13:11:47,731: WARNING/ForkPoolWorker-8] 发送短信任务函数执行...
[2020-09-13 13:11:47,732: WARNING/ForkPoolWorker-8] [mobile: 13155667788] [sms_code: 939466]
[2020-09-13 13:11:47,898: INFO/ForkPoolWorker-8] Task send_sms_code[32d0a76a-4629-426c-8091-1db1ba3524ce] succeeded in 0.167541213999982s
```

Annotations and Diagrams:

- Red arrows point from the '点击发送短信' (Click to send SMS) button in the form to the `send_sms_code.delay()` call in the code.
- Green arrows point from the `send_sms_code.delay()` call to the 'broker' (消息队列) box, and then to the 'worker' box.
- Text labels indicate: '① 点击请发送短信', '② 发出任务消息之后, 代码继续向下执行', '③ 接收到响应, 前端倒计时', and '根据任务消息执行send\_sms\_code发短信函数'.

## 1.4 【理解掌握】注册用户信息保存接口实现

思考问题:

- 1) 注册用户信息保存API接口设计时,需要有哪些参数?
- 2) 注册用户信息保存的业务逻辑是什么?

业务逻辑:



## API接口设计:

API: POST /register/

请求参数:

```
json
{
  "username": "用户名",
  "password": "密码",
  "password2": "重复密码",
  "mobile": "手机号",
  "sms_code": "短信验证码",
  "allow": "是否同意协议"
}
```

参数举例:

```
{
  "username": "smart",
  "password": "123456abc",
  "password2": "123456abc",
  "mobile": "13155667788",
  "sms_code": "135678",
  "allow": "true"
}
```

响应数据:

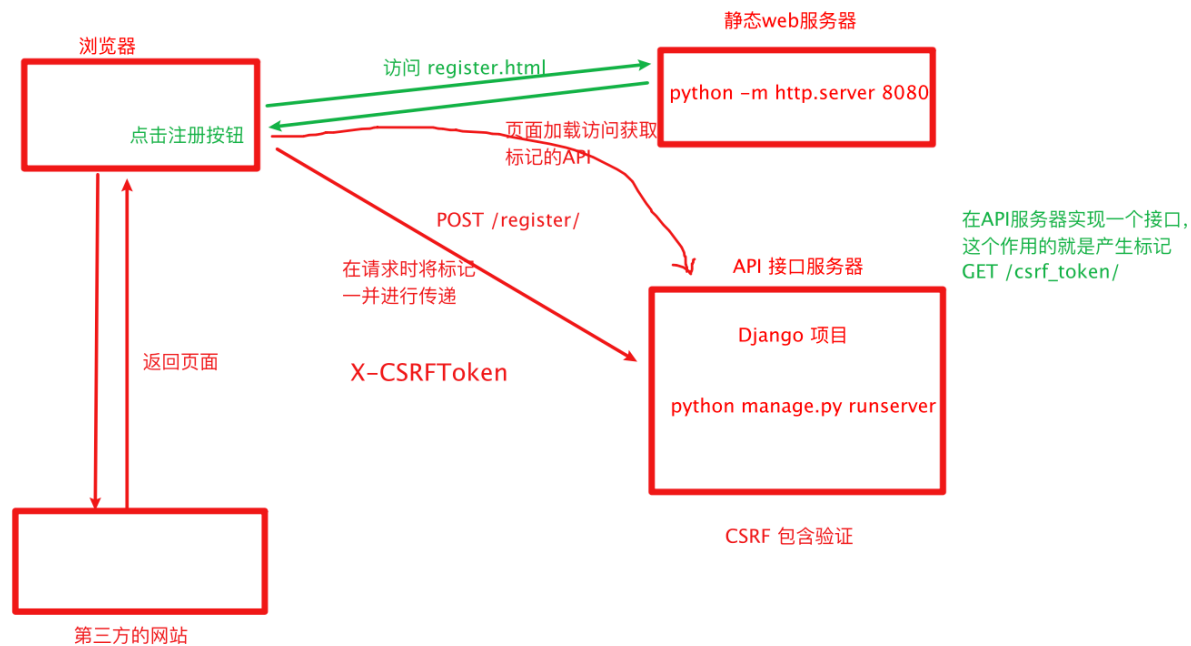
```
{
  "code": "响应码",
  "message": "提示信息"
}
```

响应举例:

```
{
  "code": 0,
  "message": "注册成功"
}
```

User.objects.create\_user(...): 使用注册用户信息保存时密码的加密保存

前后端分离CSRF验证问题:



前后端分离开发中, 后端需要给前端提供一个获取 csrf\_token 的接口。

```
=====
API: GET /csrf_token/
=====
请求参数:
  无
=====
响应数据:
{
  "code": "响应码",
  "message": "提示信息",
  "csrf_token": "csrftoken的值"
}
响应举例:
{
  "code": 0,
  "message": "OK",
  "csrf_token": "891ge98njaw2ebn1yydpwphv0y8qhfn"
}
=====
```

前端在页面加载时, 请求后端API获取csrf\_token, 并且在请求后端的API接口时, 需要通过请求头主动将 csrftoken 传递给后端服务器, 这样才能通过 csrf 保护验证。