

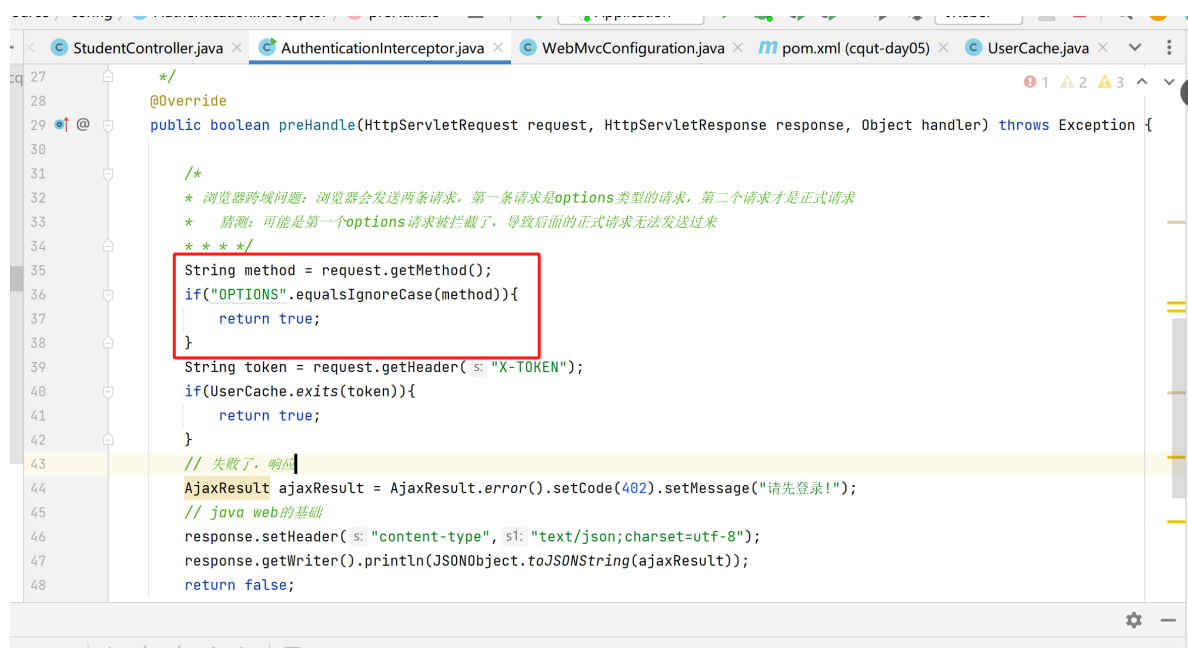
一、自定义拦截器导致跨域处理失败

原因：

自定义拦截器把OPTIONS请求拦截下来了，导致options请求无法正常响应

处理方式：

在自定义拦截器中，把OPTIONS类型的请求放行



```
27  */
28  @Override
29  public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
30
31      /*
32       * 浏览器跨域问题：浏览器会发送两条请求，第一条请求是options类型的请求，第二个请求才是正式请求
33       * 猜测：可能是第一个options请求被拦截了，导致后面的正式请求无法发送过来
34       */
35      String method = request.getMethod();
36      if("OPTIONS".equalsIgnoreCase(method)){
37          return true;
38      }
39      String token = request.getHeader("X-TOKEN");
40      if(UserCache.exists(token)){
41          return true;
42      }
43      // 失败了，响应
44      AjaxResult ajaxResult = AjaxResult.error().setCode(402).setMessage("请先登录!");
45      // java web的基础
46      response.setHeader("content-type", "text/json;charset=utf-8");
47      response.getWriter().println(JSONObject.toJSONString(ajaxResult));
48      return false;
49  }
```

二、Redis

1、安装redis

https://redis.io/docs/latest/operate/oss_and_stack/install/install-redis/

windows版本的redis官网没有，通过其他渠道下载

解压后的目录

<input type="checkbox"/> 名称	修改日期	类型	大小
00-RELEASENOTES	2021/10/17 20:11	文件	129 KB
dump.rdb	2022/12/9 10:23	RDB 文件	1,235 KB
EventLog.dll	2022/2/17 10:57	应用程序扩展	2 KB
README.txt	2020/2/9 13:40	文本文档	1 KB
<input checked="" type="checkbox"/> redis.windows.conf	2022/11/7 14:47	CONF 文件	61 KB
redis.windows-service.conf	2021/8/25 7:32	CONF 文件	61 KB
redis-benchmark.exe	2022/2/17 10:58	应用程序	449 KB
redis-benchmark.pdb	2022/2/17 10:58	PDB 文件	5,572 KB
redis-check-aof.exe	2022/2/17 10:57	应用程序	1,813 KB
redis-check-aof.pdb	2022/2/17 10:57	PDB 文件	9,620 KB
redis-check-rdb.exe	2022/2/17 10:57	应用程序	1,813 KB
redis-check-rdb.pdb	2022/2/17 10:57	PDB 文件	9,620 KB
<input checked="" type="checkbox"/> redis-cli.exe	2022/2/17 10:58	应用程序	614 KB
redis-cli.pdb	2022/2/17 10:58	PDB 文件	5,980 KB
<input checked="" type="checkbox"/> redis-server.exe	2022/2/17 10:57	应用程序	1,813 KB
redis-server.pdb	2022/2/17 10:57	PDB 文件	9,620 KB
RELEASENOTES.txt	2022/2/10 20:16	文本文档	4 KB
server_log.txt	2022/9/28 15:25	文本文档	2 KB
temp-6440.rdb	2022/11/7 15:09	RDB 文件	35,103 KB

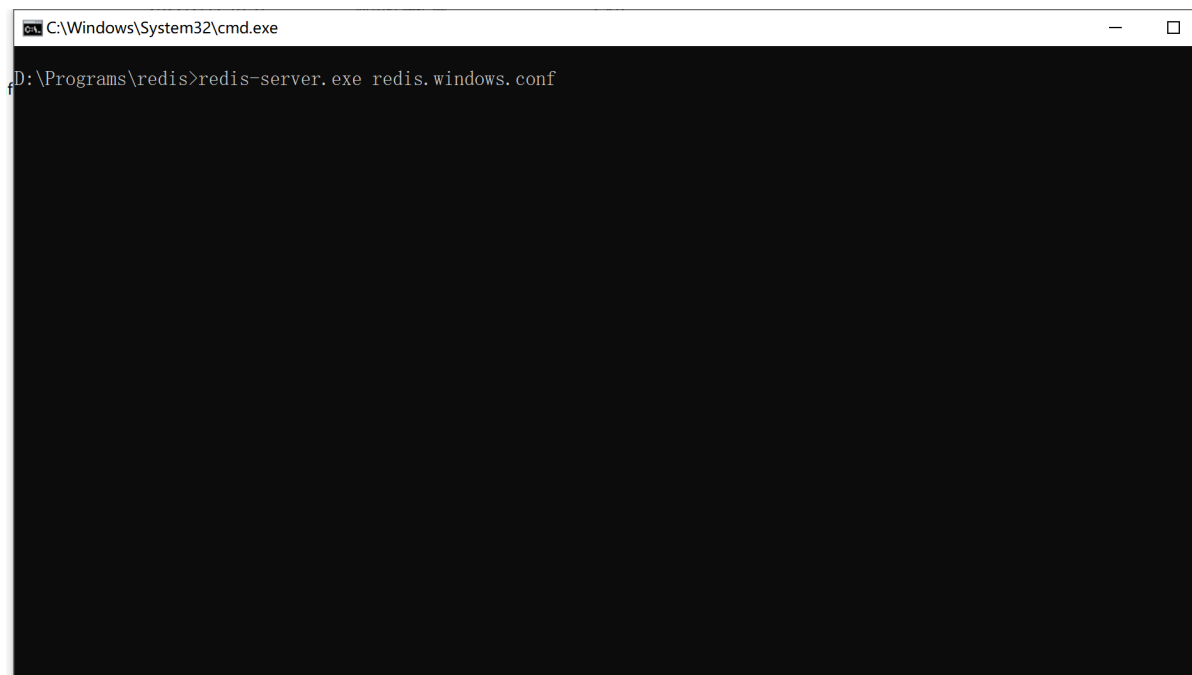
redis-server.exe redis服务端启动的文件

redis-cli.exe redis客户端

redis.windows.conf redis的配置文件

2、启动redis

以配置文件方式启动



3、启动客户端

我们先使用命令行的客户端（大家在网上找图形化的客户端）

```
C:\Windows\System32\cmd.exe - redis-cli.exe
Microsoft Windows [版本 10.0.19045.4529]
(c) Microsoft Corporation。保留所有权利。

D:\Programs\redis>redis-cli.exe
127.0.0.1:6379>
```

4、简单入门

(1) 非关系型数据库 (Nosql)

nosql : not only sql

redis是一种非关系型数据库

特点:

- 内存存储数据（快）
 - 也可以持久化到硬盘上，防止数据丢失
 - 只是把内存中的数据同步到了硬盘上，操作还是在内存数据
- key-value 存储结构的数据库
 - key是字符串
 - value有很多类型
 - 字符串
 - 整数
 - List
 - Set
 - Hash
 -

(2) redis常用命令

auth password

```
auth 123456
```

登录认证

keys *

```
OK
127.0.0.1:6379> keys *
(empty list or set)
127.0.0.1:6379> set name zhangsan
OK
127.0.0.1:6379> keys *
1) "name"
```

- 操作value为字符串类型的值
 - set key value

```
set name zhangsan
```

如果key存在，那么会把之前的key覆盖

- get key

```
get key
```

获取key对应的字符串的值

- 操作List
 - lpush key value1 value2 value3 ...
存储value为List类型的值
 - lrange key start stop
查询List类型的值

```
zhangsan
127.0.0.1:6379> lpush names zhangsan lisi wangwu zhaoliu tianqi wangba
(integer) 6
127.0.0.1:6379> get names
(error) WRONGTYPE Operation against a key holding the wrong kind of value
127.0.0.1:6379> lrange names 0 -1
1) "wangba"
2) "tianqi"
3) "zhaoliu"
4) "wangwu"
5) "lisi"
6) "zhangsan"
127.0.0.1:6379> _
```

- 删除key

del key1 key2 key3

(3)java对象如何保存到redis中?

- 将Java对象序列化保存到redis中【一般用于很大很复杂的Java对象】
- 将Java对象转换成JSON字符串保存到redis中【使用这种方式比较多】

(4) 可以给key设置过期时间

```
set key value ex xx
```

ex指代的是过期的秒数

```
ttl key
```

可以查看这个key还有多久超时

```
expire key 秒数
```

给事先设置好的key设置过期时间

5、Java操作redis

- 添加pom.xml依赖

```
<dependencies>
  <dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>5.1.0</version>
  </dependency>
```

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.28</version>
</dependency>
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>2.0.32</version>
</dependency>
</dependencies>
```

- 测试代码

```
package cn.itsource;

import cn.itsource.domain.User;
import com.alibaba.fastjson.JSONObject;
import redis.clients.jedis.Jedis;

import java.io.*;

public class JedisDemo {

    public static void main(String[] args) throws
    Exception{

        Jedis jedis = new Jedis("127.0.0.1",6379);
        jedis.auth("123456");

        // JAVA对象存储到redis中
        //      User user = new User(1L, "amdin", "admin");
        //      ByteArrayOutputStream outputStream = new
        ByteArrayOutputStream();
        //      ObjectOutputStream objectOutputStream = new
        ObjectOutputStream(outputStream);
        //      objectOutputStream.writeObject(user);
        //
        //      byte[] bytes = outputStream.toByteArray();
        //      jedis.set("student:1".getBytes(),bytes);
        //
```

```

//      outputStream.close();

        // 把之前存储到redis中的java对象反序列化出来
        byte[] bytes =
jedis.get("student:1".getBytes());
        ByteArrayInputStream byteArrayInputStream = new
ByteArrayInputStream(bytes);
        ObjectInputStream objectInputStream = new
ObjectInputStream(byteArrayInputStream);
        User user = (User)
objectInputStream.readObject();

        System.out.println(user);

        byteArrayInputStream.close();
        jedis.close();

    }

    private static void quickStart(){
        Jedis jedis = new Jedis("127.0.0.1",6379);

        jedis.auth("123456");

        // 发送redis命令
//      jedis.set("name","张三");

        String name = jedis.get("name");
        System.out.println(name);

        jedis.close();

    }

    /**
     * java对象存储到redis中
     */

```

```

private static void toJSON(){
    Jedis jedis = new Jedis("127.0.0.1",6379);
    jedis.auth("123456");

    // JAVA对象存储到redis中
    //      User user = new User(1L, "amdin", "admin");
    // 1、将Java对象转为JSON字符串
    //      String userJSON =
JSONObject.toJSONString(user);
    // 扩展一点：key的命名方式，key可以分组，第一级和第二级
可以使用：隔开
    //      jedis.set("user:1",userJSON);

    // 从redis中读取JSON数据转为java对象
    String userJSON = jedis.get("user:1");
    User user =
JSONObject.parseObject(userJSON,User.class);
    System.out.println(user);

    jedis.close();

}

}

```

6、springboot项目中如何操作redis

- 添加pom.xml中的依赖


```

<!--
        spring-data spring提供的数据库操作的规范
        spring-data-redis 是根据spring提供的数据库操作的
        规范，操作redis
        spring-boot-starter-data-redis 将spring-
        data-redis整合到springboot项目中
-->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-
    redis</artifactId>
</dependency>

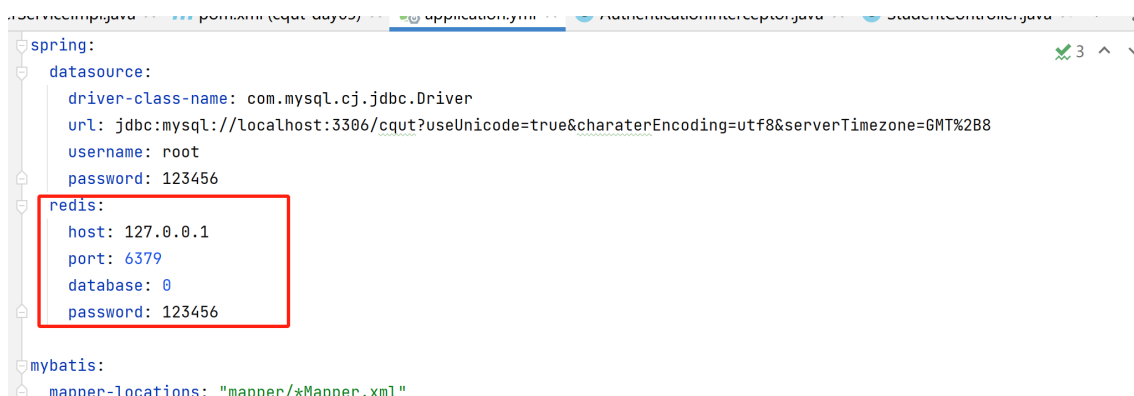
```

- application.yml中配置redis

```

spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/cqut?
    useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8
    username: root
    password: 123456
  redis:
    host: 127.0.0.1
    port: 6379
    database: 0
    password: 123456

```



- 登录的service中UserServiceImpl.java

```
@Service
public class UserServiceImpl implements IUserService {

    1 usage
    @Resource
    private UserMapper userMapper;
    1 usage
    @Autowired
    private RedisTemplate redisTemplate;

    /**
     * 登录
     * @param user
     * @return
     */
    throw new RuntimeException("用户名或密码错误!");
}

// 登录成功!! 生成token
// token和登录的用户信息之间要一一对应
// 项目中一般存到redis中
// JWT
String token = UUID.randomUUID().toString();
// UserCache.putUser(token, resultUser);
// 把 token-user 存入到redis中
redisTemplate.opsForValue().set(token, resultUser, timeout: 10, TimeUnit.SECONDS);

return token;
}
```

其他地方的使用类似

springboot中配置好redis的链接

哪里需要操作redis，那么就把RedisTemplate Autowire进来

由于springboot中redis对java对象的保存使用的是序列化方式

所以要保存到redis中的对象必须实现序列化接口

```
servicesimp.java  User.java  pom.xml (equi-days)  application.yml  AuthenticationInte
package cn.itsource.domain;

import lombok.Data;

import java.io.Serializable;

18 usages
@Data
public class User implements Serializable {

    private Long id;

    private String username;

    private String password;

}
```

三、Maven多模块

略

四、若依框架

1、下载

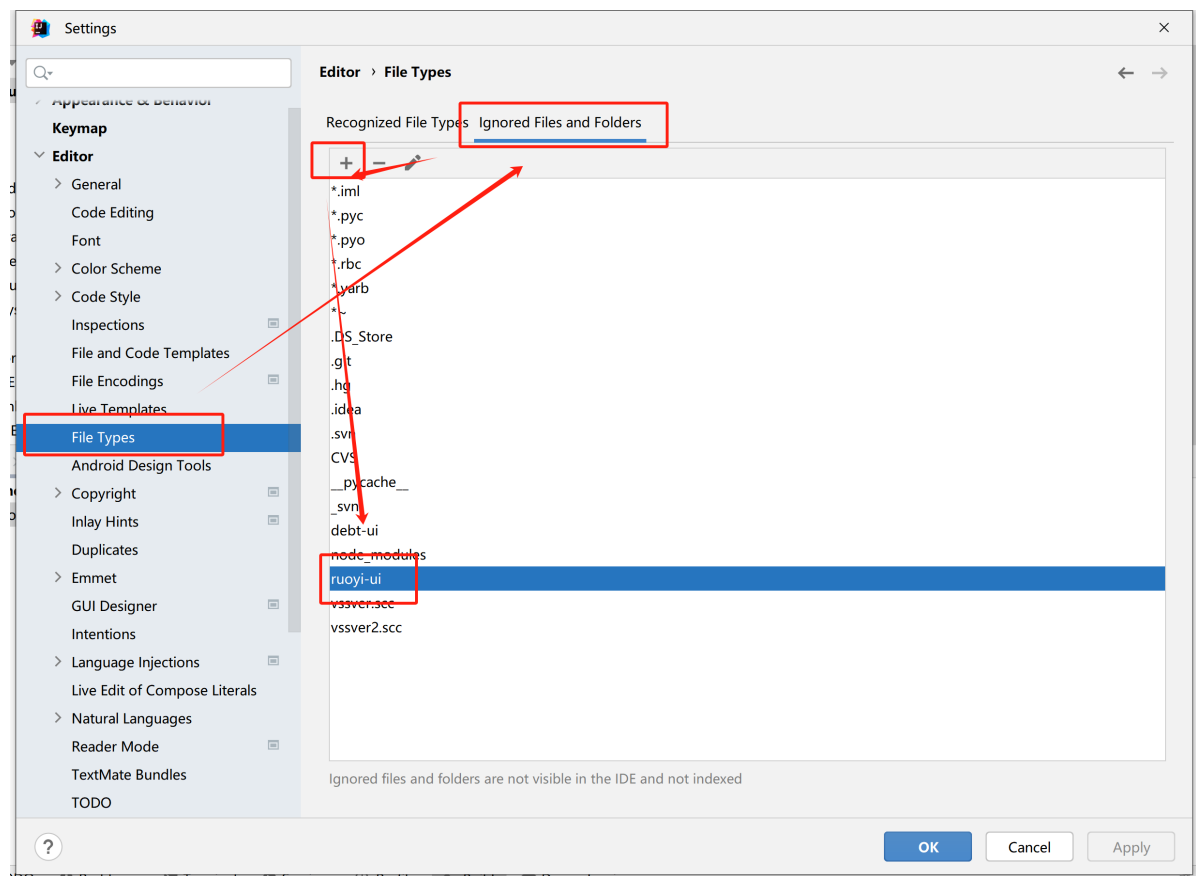
略

https://gitee.com/y_project/RuoYi-Vue

2、导入项目

idea导入项目

隐藏ruo-ui这个目录



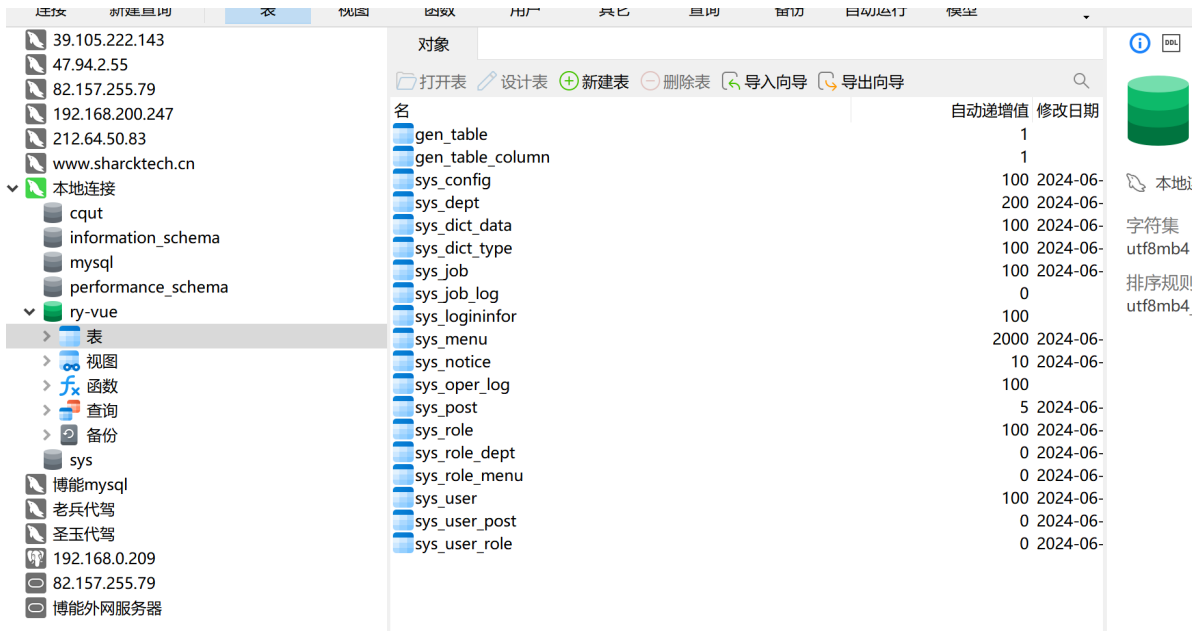
3、vscode导入前端项目(vue-ui)

4、前端安装依赖

```
npm install
```

5、导入sql到数据库

- mysql创建数据库 ry-vue
- 把sql文件导入到ry-vue数据库中



6、启动redis

略

7、修改配置文件

application.yml 中修改redis的配置

application-druid.yml中修改mysql的配置

8、启动项目

9、前端安装依赖

```
npm install
```

10、启动前端项目

```
npm run dev
```

若依管理系统

首页

系统管理

系统监控

系统工具

若依官网

localhost/index

若依管理系统

领取阿里云通用云产品1888优惠券
https://www.aliyun.com/minisite/goods?userCode=brki8iof
领取腾讯云通用云产品2860优惠券
https://cloud.tencent.com/redirect.php?redirect=1025&cps_key=198c8df2ed259157187173bc7f4f32fd&from=console
阿里云服务器折扣区 >点我进入 腾讯云服务器秒杀区 >点我进入
云产品通用红包，可叠加官网常规优惠使用。(仅限新用户)

若依后台管理框架

一直想做一款后台管理系统，看了很多优秀的开源项目但是发现没有适合自己的。于是利用空闲休息时间开始自己写一套后台系统。如此有了若依管理系统。她可以用于所有的Web应用程序，如网站管理后台，网站会员中心，CMS，CRM，OA等等。当然，您也可以对她进行深度定制，以做出更强系统。所有前端后台代码封装过后十分精简易上手，出错概率低。同时支持移动端访问。系统会陆续更新一些实用功能。

当前版本: v3.8.7

v免费开源

访问阿里云访问主页

技术选型

后端技术

SpringBoot
Spring Security
JWT
MyBatis
Druid
Fastjson
...

前端技术

Vue
Vuex
Element-ui
Axios
Sass
Quill
...

联系信息

官网: http://www.ruoyi.vip
QQ群: 满937441 满887144332 满180251782 满104180207 满186866453 满201396349 满101456076 满101539465 满264312783 满167385320 满104748341 满160110482 满170801498 满108482800 满101046199 满136919897 满143961921 满174951577 满161281055 满138988863 151450850

更新日志

v3.8.7 - 2023-12-08
v3.8.6 - 2023-06-30
v3.8.5 - 2023-01-01

捐赠支持

支付宝扫一扫，向我付款
微信扫一扫转账