原业务线：以手机号作为唯一登录手段

登录/注册相关两张表

tblUserRegister：以手机号为key做分表，存储用户password，session,uid,封禁状态等信息

tblAppUcloud：已uid为key做分表，存储用户头像、昵称、性别、财富值等才用户信息

**Sso实现**

1. 某一域名下登录
2. 用户前端页面输入手机号账号登录
3. Server接收到数据进行基础校验检查手机号是否注册，若未注册返回，若已注册则读取用户的session,password等信息,(session中存储的通行证zybuss)
4. 对用户的密码做校验操作，若用户密码校验失败则在redis中将用户尝试次数+1，若尝试次数超过限制则对用户账号进行短时间封禁操作并清空尝试次数
5. 密码校验成功，生成临时通行证zybuss。然后用以zybuss为key，用户uid,phone,登录时间,登录方式等信息为value存储在redis中。然后将第二步遍历的session,将这些zybuss为key的redis对象的value置为相应的踢出状态，踢出失败则进行异步删除操作，然后将当前的zybuss写入表中的session字段
6. 最后将当前生成的zybuss放入cookie返回给客户端
7. 如果是浏览器登录成功，还会进行跨域共享session，前端会发送请求到后端获得需要同步的cookie的域名。（也可以前端做）
8. 然后以jsonp的格式跨域分别请求上一步获得的域名，服务端收到请求校验后cookie，（根据redis）正确校验后在不同的域名setcookie，并根据请求的callback参数拼接jsonp数据返回给前端

Session存在db中，而zybuss存储在redis中，因为session只有在用户登录、注册、修改密码等操作时需要读取，不需要存在redis中

其他的登录接口还有tokenlogin（验证码登录，逻辑也类似）

注册：1）新增用户信息

2）新增session

问题：步骤1是事物的，但1和2不耦合，但这么处理挺好

事物1：IdAlloc分配一个uid。

2：开启事物

3：register表新增数据，uid与phone映射关系。

4: uloud新增用户信息

作业帮内所有的请求在action层基类会调用Saf\_SmartMain::getUserInfo();读取session中的信息

新增逻辑

oauth2.0：(oauth2.0分为三种角色—客户端、用户和资源服务器，WeChat和qq是不同的授权模式。授权码模式客户端分为客户端的客户端和客户端的server，简化模式则没有)

1. 端吊起第三方sdk
2. 用户授权（qq返回access\_token，ouid,wechat返回code（临时token））
3. 客户端发送上一步返回的数据给服务端
4. 如果是wechat根据code和服务端存储的secret请求微信server端得到token和ouid
5. Server端根据ouid获取用户的注册状态
6. 若已注册生成session返回，若未注册则请求获取用户信息，然后以一个随机生成的key，在redis中存储ouid，然后将这个key和获取的用户信息返回给客户端
7. 客户端进入绑定手机页面，用户可选择绑定新手机号，老手机号和不绑定手机号，用户操作完将信息（手机号）post后端
8. 后端若用户不绑定手机号则执行事物A，绑定手机则确认手机注册状态。若绑定新手机执行事物B，绑定老手机执行事物c
9. 生成session返回

事物A：

Idalloc分配uid,

新增uid和ouid映射关系到oauth表中，

新增ouid与uid反映射关系到map表中

新增 以uid来分表，存储uid与session，forbien信息的新表password中

新增 用户信息到以uid分表的ucloud表中

事物B:

Idalloc分配uid,

新增uid和ouid映射关系到map表中，

新增ouid与uid反映射关系到oauth表中

新增 以uid来分表，存储uid与session，forbien信息的新表password中

新增 用户信息到以uid分表的ucloud表中

新增 phone与uid的映射关系register表中

事物C：

得到用户手机注册的uid，

根据uid从map表中查询用户已绑定的第三方类型，若已绑定展示已经绑定过了。

若未绑定，

新增uid和ouid映射关系到map表中，

新增ouid与uid反映射关系到oauth表中

拆分数据表：

原本的register（phone,uid,session）

拆分为password(uid,session)+register(phone+uid),需要数据迁移

修改业务接口：手机号登录、注册、手机号修改绑定、修改密码、第三方登录注册绑定，各类验证码发送时的关系验证

问题：

1.老用户第三方注册若不绑定手机号，之后会有两个uid，账号无法合并

2.许多业务线强制需要手机号，需跳转入绑定手机号页面（与其他业务线确认）

数据迁移：

条件：一主多重，凌晨，user独立的库，dbProxy

1. 编写数据迁移脚本(提交qa测试)
2. 排查所有接口修改代码（提给qa测试），12点开始db写操作，变成双重操作，直接写原库与生成新表sql发送至消息队列（待定）存储。
3. 线上建好新表
4. 上线双重逻辑的代码，消息队列发送到proxy但不推送到server
5. 12点停掉一个从库
6. 找到与nmq匹配点，记录
7. 用脚本迁移从库中的存量数据到现在还在运行的主库
8. 存量数据处理，Nmq打开发送至任务A执行sql。
9. 待增量sql消耗完
10. 查看消耗脚本日志
11. 日志执行正确，再次上线代码将队列请求转为同步请求
12. 测试账号测试读写
13. 正常开始切流量看log
14. 全量

迁移脚本：

每次1000读出，拼接成新sql，执行，sleep(0.1)

内网sql执行任务：

一个php脚本，参数是一个token和相应的sql去执行。

问题：

执行出错怎么办？

sql记录出错的sql，再根据uid从原库迁移到现库。

存量数据出错会打warning日志，一直在监控的。

nmq出错怎么办？(线上均限制为4000-5000qps，此时稳定性为99.9999)

队列有稳定性的

nmq挂了怎么办？

改天做这件事

迁移花了多久？

存量两个小时左右，增量和测试一个多小时

为了保证主从延迟做了哪些控制？

作业帮dbuser表在高峰期每秒读写报警数（值不告诉）来控制

怎么控制执行速度的？

增量nmq可以控制速度，并且量不大。存量数据脚本迁移，边度边写，一次读写

Nmq介绍

我的工作：业务重写，迁移脚本，增量处理，错误问题处理。

哪些优化：

注释：idalloc(‘name’,’id’),name主键

Alloc实现：select id from idalloc where name=xx for update

Id++

Insert into idalloc values(name,newid);