1.用户进行了三方登陆的绑定之后，用户的登陆方式变的更多，可以方便的使用三方账号进行的密码验证机制进行登陆了。

这样，用户可以不用费心的记着码云的密码。只要记得非常重要且常用的三方账号如微信等的密码即可。

2.如果做得更人性化一点，新建账号时，可以不填写密码。

但是要填一个正确且未注册过的邮箱地址 或者 填写一个正确的且未注册过的手机号（二者至少正确的填写一个），其他都可以不填。

这个邮箱地址默认就是用户的本站账户信息了。相当于用户做了微信和本站邮箱的绑定，但是又不用费心的设置本站的密码。

这样用户就少了一件烦心事，用户心中知道，我在码云是没有密码的，就靠三方的微信密码进行登陆，又少了一个密码，很好。

等将来用户不论出于何种需要，想通过本站的邮箱进行登陆时，发现无论如何登陆都是用户名密码错误，然后就会走忘记密码流程，

进而再次验证邮箱后，设置密码，之后就可以使用本站的邮件加密码进行登陆了。

3.为何会有使用三方登陆的需求存在。

痛点是什么？

（1）站在用户角度：三方登陆从一定程度上解决了用户多站点多密码的烦心事，用户只要记得他最习惯的应用的用户名密码即可，比如喜闻乐见的微信。

如果用户在登陆新的站点的时候都选择使用微信这种第三方登陆方式，那么他只要记得微信的账号和密码即可，并且现在微信一般

在手机上都是长期登陆状态，是要微信扫一下二维码即可。用户再也不用费心的记着那么多站点的密码。

用户记密码是有痛点的，如果用户在不同的站点使用不同的密码，安全度是上去了，但是时间长了，用户会忘记一些站点的密码，

或者会混淆密码，很痛苦，可能要面临不断的忘记密码，找回密码的痛苦过程。

如果用户在不同的站点都使用相同的密码，那么密码的安全度很低，一个站点的密码被破解或泄密，很多其他站点的也就没有密码可言了。

（2）站在用户角度，用户为某一个（账号）绑定了多种三方登陆的方式，那么不同的三方登陆方式登录成功后，可以在本站点对应的显示

其三方账号的用户名和头像等信息，给用户一致的互联网友好体验，当然，这也不能说是什么大的有点，也算是一个特点吧。

反正感觉还不错。

（3）三方登陆情况下，一般现在都做的比较简单，通常是用户在三方账号输入用户名密码正确后，就给站点颁发accessTokn，这个

token只是默认返回用户最基本的信息，所以用户不用在这个过程中去勾选权限。

但是如果要做的复杂一点也是可以的，比如登陆过程中也有授权选择页，

(前提，a.三方账号对外公开的可以供站点访问的接口列表是公开的。b.站点自己在授权过程中根据需要申明要的接口列表，

这个接口列表是三方账号公开的接口列表的子集，也是供用户选择是否授权给站点访问三方接口的列表。

c.加入三方账号对外开放的可对外授权的接口列表相对比较多，同时站点申明要用户确权的接口也比较多，而用户又将这些接口

都勾选确认授权了)，用户勾选了全部的授权接口，那么此时站点，就可以从三方账号拿到更多的与该用户相关的信息，

从而在本站可以给用户更好的一致性体验，用户在没有泄露三方账号密码的情况下，给了站点可以访问的三方账号的数据。

三方账号就可以完成对应的工作了，整个过程都是由用户授权过的，这很好。

当然三方账号通常作为规模比较到的公司，安全性上都是考虑周详的，比较token有效期通常不会太长。过期之后，需要站点再次

调用授权页面，走授权流程，进行授权后，才可以继续访问三方接口，通常不会有什么refreshToken。

当然，如果为了方便，其实也可以给站点refreshToken，方便他一直去换accessToken，但这样会产生一个问题，就是站点如果恶意

一直持有用户的refreshToken和accessToken，那么站点就可以长期访问用户的授权接口，这很恐怖，用户没有结束的地方。

好在三方账号都是大公司，考虑比较周祥，除了用户名头像等最基本的接口外，对于其他对外开放授权访问的相对敏感的接口，三方账号通常会有授权管理功能，可以在这个功能中，删除全部授权或禁用部分用户的授权。

没有了用户的授权，这样站点就不能调用三方接口了，那么如果想访问，那么就乖乖的再弹出授权页面，走授权流程，又是一个循环。

（4）站在新站点企业角度：如果站点比较小，用户一般懒得填写那么多注册信息，而使用三方账号登陆，可以快速引流用户，因为注册流程大大简化了。

另外，用户多一些登陆方式也是比较乐意的，其他站点都支持，你不支持，就low，用户就认为你不行，用户量就上去的慢，所以也没办法，就得支持三方登陆。

4.继续拓展往下说，对于上面的3（3）中，有可能是这样的一种场景，站点本身是只通过自身账号登陆的（三方站点登陆也没关系），但是在站点的功能当中，有那么一项功能，用户如果想使用，就需要访问三方账号的信息才行（非登录的那个三方账号），

比如在一个云重印的站点，你使用用户名密码或者微信登陆进去以后，想在里面使用【冲印我的google相册】这个功能，那么google就是一个三方账号，用户点击这个功能的时候，就需要站点和google之间进行auth授权过程，用户需要参与确认权限，

这个授权过程当然用户也是需要输入他的google账号和密码的，同时勾选同意站点访问google照片授权选项。这样站点就可以临时的访问用户在google中的照片信息了，进而进行冲印。

对于这种场景：

（1）.站点不会将google作为三方用户与自己的用户体系进行账号直接的绑定，可以说只是临时用一下用户授权的三方接口数据而已。

（2）.google的accessToken的有效期很短，并且要么没有使用refreshToken续期的功能，那么就是有授权管理，用户可以在冲印照片完成后，马上登陆google的收取管理将授权进行回收。

（3）.作为三方账号的公司，都是大公司，这种情况下，他们可以针对不同的接口开发不同的授权隔离（项目），

比如专门只用于三方账号登陆的auth2.0授权。

比如对于敏感接口的，需要用户勾选授权，且提供授权管理的auth授权。

不同的授权体系，可能都是使用的auth2.0，但是控制的细节可能有差别，比如有的有refreshToken有的就没有refreshToken，有的accessToken有效期时间较长，有的accessToken有效期时间较短，有的有授权管理，有的没有授权管理。

有的提供可用的授权列表，有的不提供，等等等等，全看业务需要。

5.继续拓展往下说。

所以auth2.0授权机制，怎么用都行，但是要了解她的本质，就是用户授权站点在一段时间内访问三方账号的某些接口信息，这个访问是临时授权的，一旦失效，就要重新确权。

对于auth2.0授权机制，使用到三方账号登陆使用auth2.0的侧重点是不一样的，三方登陆使用auth2.0机制是比较巧妙的，他的最大本意其实并不是经过用户授权后要访问用户的基本信息（当然这个信息他有是最好了，没有他也可以临时生成一个名字也说的过去的），

而是auth2.0的授权过程是要用户输入三方账号密码进行登陆的，进而再确认授权等。对于站点来说，用户能把整个auth2.0的流程走下来，站点能拿到三方的accessToken，本身就已经说明该用户是三方的合法用户，这个合法用户就可以跟我的用户体系进行绑定，

后面用第三方账户登录的时候，利用auth2.0机制，把确认身份的事情交给三方就好了，用户在哪输入密码不是输入密码呢？！就好像是可以使用邮箱登陆，也可以使用手机号登陆，也可以使用微信登陆，都是我的用户，使用微信登陆将登陆过程交给微信而已，

把登陆过程“外包”了，让三方大厂帮我干活，我站点喜闻乐见的多了一个用户的登陆方式，还不用我操心用户名密码的验证流程，还不用关心密码泄密问题。

要知道邮件+密码也好，手机号+验证码也好，微信三方外包登陆也好，本质都是确认你是你，任何过程都要确认你是你，当你在做这几种账号直接的关联的时候，都是在证明了你是你的情况下进行的登陆方式拓展。

所以用户名也好，邮件也好，手机号也好，微信也好，整个绑定过程中都是你是你，不会出现别人是你，或者你是别人的情况，不让每一种绑定都不可能通过，而一旦通过了，从代码的角度看就是你是你了，因为只有你能打开对应的邮箱，

只有你能进一步收到手机号对应短信验证码，只有你能进一步再去绑定微信，那么整个过程下来，程序就会认为这些邮箱，手机号，微信是同一个人的（你跟别人关系好，绑定了别人的手机号，那么那个别人从程序看就是你），

这种情况下，微信的三方登陆通过auth2.0机制，把登陆过程交给三方大厂去做就很容易理解了，只要三方大厂给了accessTokn，就说明他在三方大厂登陆成功了，就说明他是这个微信号的合法主人，进而站点就认为你登陆成功了，

在站点中找到你就行，找到你也就找到你的邮箱了，你的手机号了，等等，要知道你们都是一根绳上的蚂蚱,就是这么个道理。程序一步一步认为你是你，环环相扣，最后把你的用户名，邮箱，手机号，三方账号一步一步绑定到了一根绳上，

都是你是你，这样这根绳上的任何一个蚂蚱通过自己的方式认证成功了，都能说明你是你，这就是根本性的原理。

这里说的你是你，就是表明各种方式绑定到一块的一定是同一个人，不论何种方式登录，他在系统中拥有的权限和资源都是统一的，至于他是现实中的谁，绝大部分系统并不关心，系统关心的是系统中的每一个账号及其关联账号指向的一定是同一个人即可，

即使他在系统中注册两个账号，分别又绑定了多种方式，那系统就认为是两个不同的人。但这两个人中的每一个人的各种认证方式一定都是指向同一个人的。虽然两个账号都是同一个人的，系统不关心，系统关系的是同一个账号管理的所有账号是同一个人的即可。

系统不关系他是现实世界中的谁。

当然，有的系统要求注册用户是实名认证的，那么可能要输入身份证号，并上传照片做实名认证等，这样账号可以再进一步管理身份证号，那么就彻底将同一个账号的各种登陆方式都和这个现实中的身份证号绑定到一块了，系统就才会知道系统中的虚拟的各种账号

对应现实中的这个身份证号，对应现实中的这个人，不过这都是一些少数的系统。大部分系统都不要求实名认证。而只要求每个用户自身的身份统一性即可，他就是这个系统中独立一个合法用户了，登陆后可以享用自己独有的系统权限和系统资源。

只是通常用户名和邮箱是通过密码的方式认证你是你。手机号是通过短信验证码的方式认证你是你，而三方账号登陆是通过auth2.0的机制让三方大厂帮助认证你是你。如此而已。

巧就巧上面那一点，同时利用auth2.0机制，可以访问用户基本信息接口，拿到拿到三方的对外公布的用户的唯一标识，这个标识会被站点作为用户绑定三方账号的依据。

auth2.0授权机制虽然不是为三方账号量身打造的，但是整个过程使用auth2.0机制都十分合适契合，也算是无心插柳柳成荫了。

你是三方的合法用户，我就把你当成我的合法用户，跟我的用户体系做绑定，这叫用户信息共享，最大程度方便用户。其实就这么简单。三方登陆使用了auth2.0，很巧妙，但真不是其出发点和侧重点，那事情往往是这样，

比如一个人没有绝对的好坏，是很负责的，技术也是，有时候不一定要量身打造才行，适合的是最好的，一个技术体系可以准确的解决一件事情的时候，有时候还能举一反三的解决类似的不太“准确”的，但是又特别合适能解决的事情。

所以做技术，任何时候，保持头脑的灵活，善于思考总结，不死板是最重要的。

6.用户是三方登陆进来的，拿到tokekn后只要取一次用户信息进行缓存即可，整个本次会话期间，都使用这个用户信息即可，不考虑用户

在登陆本站期间，再去修改三方账号的信息的情况，意义不大。

7.关于auth2.0站点拿到三方的accessToken后，其实三方的系统已经标注这个站点可以访问这个用户的这些接口资源了，这些会在三方系统中做记录。

另一方面站点这一方，也会记录站点拿到了这个三方系统的accessToken，可以访问这个三方站点的这些资源接口，站点想要进一步获取用户信息只要调用三方的接口，传入clientId，clientSecret，accessToken即可拿到三方系统对外暴露的用户id和用户信息（当然用户id是三方系统经过映射处理的）。

进而站点这一方可以进一步绑定用户id。这是正常的流程，都没有问题。

只是大家真正在做这件事情的时候，进一步又把事情简单了一步，就是站点在拿code换到accessToken的时候，三方系统除了返回accessToken，还同时返回了用户id，如果站点不需要用户的其他信息，其实可以直接将此用户id做记录了，如果进一步需要用户信息可以再调用接口获取。

这样很方便，因为既然站点在三方系统中是合法的（提前有注册认证），用户在三方站点是合法的，是三方站点的用户，然后用户通过auth授权，授权给站点访问三方系统的某些接口资源，这样站点当然就可以通过直接通过

clientId，clientSecret，accessToken去获取用户id和用户信息了，那么刚才整个授权过程都是用户同意的，干嘛不直接将用户id一并返回给站点呢，反正用户都同意授权了，我返回accessToken的时候就将最基本的用户id一并返回给他，在某些场景下，可以减少他的调用不是很好吗，干嘛还必须在再

拿accessToken请求一次才能拿到用户id呢，当然这是正道，直接返回就是省事了一小步，也完全没有问题。返回不返回都行，都没有任何问题。

8.关于三方账号登陆的总结说明：

（1）.毫无疑问三方登陆过程是一个标准的aouth2.0的交互过程。

（2）.对于授权服务器方来说，没有什么特别的，对外开发的接口需要通过合法用户的授权才可以进行访问对应用户的接口数据。

（3）.对于客户端方来说，本身也没有什么特别的，通过oauth交互过程，用户同意授权给客户端访问哪个接口，客户端就可以拿着accessToken去访问对应接口的用户数据了，对于三方登陆来说客户端时是访问用户的基本信息接口数据。

（4）.然后进一步扩展来说，凭什么有了这个过程，客户端就可以做“三方账号”的登陆了，从客户端的角度来思考问题，oauth交互过程没有问题，更重要的是，对于“三方账号”登陆场景，客户端巧妙的利用了oauth交互机制把三方用户当成自己的用户来用。

即换个角度可以理解为客户端将自己的用户登录管理交给了可以信赖的第三方进行管理，用户只要在第三方输入用户名密码等认证成功后，就认为这个用户是在我这个系统是认证成功了，把这个认证过程外延给了第三方。

而oauth在做传统理解的交互过程中，正好是需要第三方验证用户身份的，客户端只要能正常拿到accessToken，就说明用户在第三方输入的用户密码是正确的，用户是三方系统的合法用户了，自然就是我系统中的合法用户了。

通过三方登陆，可以大大提高客户端的注册率，也会增加第三方的影响力，大家都是双赢。Oauth还是原来的配方，还是原来的理解，即用户授权客户端访问用户授权的服务端接口。只是在三方登陆场景，又进一步得到了客户端的巧妙利用，做到了客户端系统用户的外延。

账号密码是用户身份的定海神针。

账号一旦建立，原则上便不可以更改名称。

如果有手机号/邮箱或者第三方账号登陆，本质上都是与账号进行绑定，手机号/邮箱或者第三方账号用户可以随便解除绑定或者绑定新的都可以。因为绑定的时候某种方式的时候，都要通过其他方式来进行验证用户身份，一旦绑定关系产生，他们都是一根绳上的蚂蚱。

解绑同样是这样，要解绑某种方式，都要验证其他任何一种方式即可，就是因为他们是一个用户这一根绳上的蚂蚱，系统认为他们是现实世界中的同一个人。

那么现在常见到的场景有：

1.系统没有注册功能。用户直接通过三方账号登陆，登陆成功后，要求用户必须完善账号密码信息。这样就有了自己的账号密码用户，然后第三方登陆其实是依附在这个账号密码上的一种用户绑定关系，这个关系可以随时让用户验证用户名密码的方式来进行解绑。

2.系统有注册账号功能。用户第一次可以选择注册账号密码。然后登陆时可以选择账号密码方式登录，也可以选择第三方账号登陆，如果是第三方账号登陆，则三方账号身份验证通过后要与本系统的账号进行绑定，通过输入用户名密码的方式。这样他们就是同一个用户绳上的蚂蚱。

3.系统没有注册功能。用户直接使用手机号加验证码的方式快速登陆，此时，用户相当于在用户表中只有手机号。那么这种情况下，用户原则上是不可以更换手机号的，因为更换了，之前的操作相当于无本之木了，另外还有一个风险是用户手机号丢了或者长期不用失效了，不能收验证码，

不能证明自己的身份了，也没有实名认证，然后也不能更换手机号，就比较悲剧了，所以不建议直接手机号加验证码就快速注册登录了。

4. 系统没有注册功能。针对第三点，还是那句话，不做无本之木，系统通过手机号加验证码快速登录后，还是要用户名输入一个用户名和密码来进行账号信息完善，这样手机号就和用户账号这个定海神针进行了绑定，手机号随时可以更换，更换的方式可以通过发手机验证码，也可以账号加密码验证。

5. 系统有注册功能。用户可以使用用户名密码注册并登陆，也可以使用手机号加验证码登陆后进行账号信息的完善绑定或者绑定某个以存在的账号。

6. 邮箱同手机号。

7. 安全设置中提供手机号/邮箱/第三方账号绑定，解绑功能。同时提供账号信息完善，密保问题完善等进一步可以确定用户身份的安全措施，用户安全意识高，补充信息后可以防止账号无法找回的风险，比如用户名密码忘记了，手机号也注销了。

上面说了这么多。

无非强调系统用户要有定海神针，建议是用户账号，这个不变的东西，这样，在此基础上，一切的绑定和解绑都是有系统日志的，不管绑定了哪些方式，操作日志都会记录某账号操作了什么。

而如果没有这个定海神针，比如只有手机号还能换来换去，日志最多记录今天手机号1和主键1做了什么操作，明天手机号2和主键1做了什么操作，就不太容易说明问题了，因为主键是一个一点意义都没有的东西，账号至少还有个名称，将来涉及到外部追溯时，因为账号通常是可以对外公开的，系统其他用户可以指出来某账号如何如何，但不会指出来主键如何如何，这时，我们可以根据账号去进行追溯。