

## 44 | 实战（四）：“画图”程序后端实战

2019-09-24 许式伟

许式伟的架构课

[进入课程 >](#)



**讲述：姚迪迈**

时长 05:59 大小 5.49M



你好，我是七牛云许式伟。

上一讲我们介绍了帐号与授权相关的基础体系，并重点介绍 OAuth 2.0 背后的逻辑。今天我们开始考虑如何让 QPaint 引入帐号与授权体系。

最常规的做法，当然是自己建立一个帐号数据库，做基于用户名 + 密码的登录授权并转为基于 Cookie 的会话（Session）。示例如下：

<https://github.com/qiniu/qpaint/tree/v44-bear>

<https://github.com/qiniu/qpaint/compare/v42...v44-bear>

但我们考虑提供 Open API 的话，就需要考虑遵循 OAuth 2.0 的授权协议规范，以便第三方应用可以快速接入，而不是搞半天去研究我们自己发明的授权是怎么回事。

除此之外，我们也可以考虑基于微信、支付宝等 OpenID 来实现用户的快速登录，而不是让用户在注册环节折腾半天。

所以，比较理想的方式是我们基于 [OpenID Connect](#) 协议来提供帐号系统，基于 OAuth 2.0 协议来实现 [Open API](#) 体系。

这个选择与业务无关。所以很自然地，我们决定评估一下，看看是否有开源项目和我们想得一样。

最后，我们发现 CoreOS 团队搞了一个叫 dex 的项目，如下：

<https://github.com/dexidp/dex>

<https://github.com/xushiwei/dex> (部分依赖库受 GFW 的影响，我们调整 Makefile 改为基于 `go -mod=vendor` 来编译。)

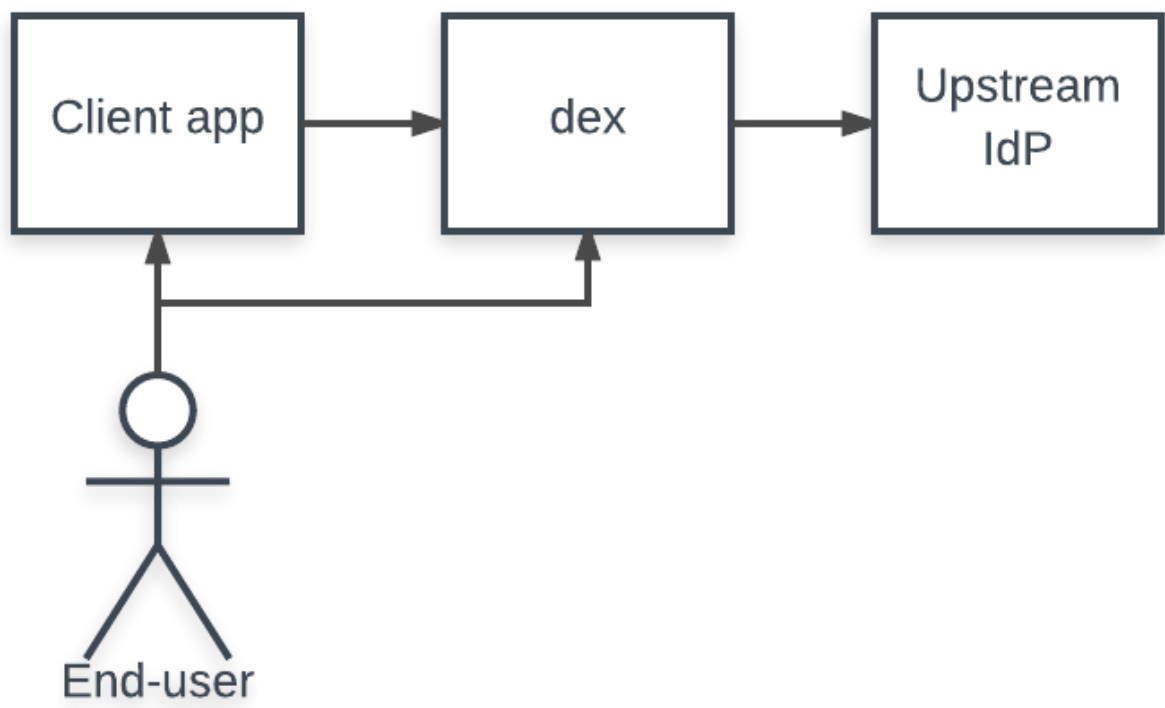
dex 项目的这么描述自己的：

dex - A federated OpenID Connect provider  
OpenID Connect Identity (OIDC) and OAuth 2.0 Provider with  
Pluggable Connectors.

Dex is an identity service that uses OpenID Connect to drive authentication for other apps. Dex acts as a portal to other identity providers through “connectors.” This lets dex defer authentication to LDAP servers, SAML providers, or established identity providers like GitHub, Google, and Active Directory. Clients write their authentication logic once to talk to dex, then dex handles the protocols for a given backend.

概要来说，dex 基于各类主流的 OpenID 来提供帐号系统，上游的 OpenID Provider（即下图中的 Upstream IdP）是以插件方式（Pluggable Connector）提供。这也是为什么把

它叫联邦 OpenID (federated OpenID) 的原因。然后，dex 再通过 OAuth 2.0 协议对客户端（即下图中的 Client app）提供授权服务。



## 联邦 OpenID

我们先看 dex 在联邦 OpenID 这块的支持。当前已经支持的 Pluggable Connector 如下：

Name	supports refresh tokens	supports groups claim	status	notes
<a href="#">LDAP</a>	yes	yes	stable	
<a href="#">GitHub</a>	yes	yes	stable	
<a href="#">SAML 2.0</a>	no	yes	stable	
<a href="#">GitLab</a>	yes	yes	beta	
<a href="#">OpenID Connect</a>	yes	no ( <a href="#">#1065</a> )	beta	Includes Google, Salesforce, Azure, etc.
<a href="#">LinkedIn</a>	yes	no	beta	
<a href="#">Microsoft</a>	yes	yes	beta	
<a href="#">AuthProxy</a>	no	no	alpha	Authentication proxies such as Apache2 mod_auth, etc.
<a href="#">Bitbucket Cloud</a>	yes	yes	alpha	

可以看出，对于那些支持 [OpenID Connect](#) 协议的 OpenID，比如 Google、Salesforce、Azure 等，可以统一用同一个 Connector 来支持。而对于其他的 OpenID，比如 Github，则实现一个独立的 Connector 来支持。

除了 OpenID Connect，我们也可以看到很多耳熟能详的开放帐号授权协议，比如在前面课程中，有人提议讲一讲的单点登录 SAML 2.0 和 LDAP。但这的确不是我们的重点。我们这里提供相关的链接供大家参考。

LDAP 的资料如下：

<https://www.openldap.org/>

SAML 2.0 Web Browser Single-Sign-On 的资料如下：

[https://en.wikipedia.org/wiki/SAML\\_2.0](https://en.wikipedia.org/wiki/SAML_2.0)

<http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html>

不同的 OpenID Provider 作为后端，会导致一些细节上的差异。有的 OpenID Provider 不支持更新令牌（Refresh Token），有的会导致 ID Token 不支持 groups 字段。详细在以上 Connector 列表中有明确说明。

另外，虽然 dex 支持了颇为丰富的 OpenID Provider，但不幸的是，国内的主流 OpenID Provider，比如微信和支付宝，都没有在支持之列。

不过好在，它基于开放的插件机制，我们可以自己依葫芦画瓢实现一个。Pluggable Connector 相关的文档和插件如下：

<https://godoc.org/github.com/dexidp/dex/connector>

国内也会有人想到做类似 dex 这种项目，比如：

<https://github.com/tiantour/union>

看到我们熟悉的微信、支付宝、新浪微博了，所以想到点子并不难，但看架构设计就会看到两者巨大的差距。

当然，如果你看到了其他很好的开源实现，欢迎留言交流。

## 提供 OpenID + OAuth 2.0 服务

尽管 dex 底层所基于的 OpenID Provider 多种多样，但是 dex 对外统一提供了标准的 [OpenID Connect](#) 协议和 [OAuth 2.0](#) 服务。


OpenID Connect 作为 OAuth 2.0 的一个扩展，最重要的一个改进是引入了身份令牌 (ID Token) 概念。

为什么需要扩展 OAuth 2.0?

因为 OAuth 2.0 本身只关心授权，所以它会返回访问令牌 (Access Token) 和更新令牌 (Refresh Token)。但无论是访问令牌还是更新令牌，都并没有包含身份 (Identity) 信息。没有身份信息，就没法作为 OpenID Provider。

身份令牌 (ID Token) 解决了这一问题。ID Token 是一个 [JSON Web Token \(JWT\)](#)，支持你对 Token 进行解码 (decode) 并验证 (verify) 用户身份。关于 JSON Web Token 的详细介绍，请参阅 <https://jwt.io/>。

dex 并不是一个包 (package)，而是一个可执行程序 (application)，它提供了帐号与授权服务。你可以这样运行它：

 复制代码

```
1 dex config.yaml
```

其中 config.yaml 是它的配置文件。其格式可参考以下这些样例：

[examples/config-dev.yaml](#) (开发用途，用 mock 的帐号与授权服务。)

[examples/config-ldap.yaml](#) (基于 LDAP 来做帐号与授权服务。)

## 使用 dex

有了 dex 服务，我们就可以开始回到 QPaint 业务，去支持帐号与授权了。

我们并不需要自己开发太多东西。

OAuth 2.0 的客户端 SDK，Go 语言自己有一个准官方的版本。如下：

包名：[golang.org/x/oauth2](https://golang.org/x/oauth2)

项目地址：<https://github.com/golang/oauth2/>

OpenID Connect 的客户端 SDK，CoreOS 团队也开发了一个。如下：

<https://github.com/coreos/go-oidc>

具体如何对接 dex，CoreOS 团队也写了一个详细的说明文档。如下：

<https://github.com/xushiwei/dex/blob/master/Documentation/using-dex.md>

有了这些 SDK 和 dex 的使用说明，具体 QPaint 业务怎么对接 dex，就比较简单了。我们这里就不详细展开，详细代码请参考：

<https://github.com/qiniu/qpaint/tree/v44>

<https://github.com/qiniu/qpaint/compare/v42...v44>

## 结语

总结一下今天的内容。

今天我们主要讨论如何基于 OAuth 2.0 来改造 QPaint 的帐号与授权机制。实际上这方面业界有非常成熟的实践，所以我们没有太大的必要去自己重新造一个轮子。我们的核心思路是，基于 [OpenID Connect](#) 协议来提供帐号系统，基于 [OAuth 2.0](#) 协议来实现 Open API 体系。

我们不只是用标准的协议，背后的实现也基于开源项目：CoreOS 团队开发的 dex。

这样，我们就可以把关注的重心放在 QPaint 业务本身上。

如果你对今天的内容有什么思考与解读，欢迎给我留言，我们一起讨论。我们服务端程序的实战到这里就要结束了。下一讲聊一聊“架构：怎么做详细设计”这个话题。

如果你觉得有所收获，也欢迎把文章分享给你的朋友。感谢你的收听，我们下期再见。

 极客时间

# 许式伟的架构课

从源头出发, 带你重新理解架构设计

许式伟  
七牛云 CEO



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 43 | 实战（三）：“画图”程序后端实战

下一篇 45 | 架构：怎么做详细设计？

## 精选留言 (2)

 写留言



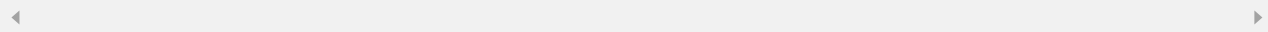
Sentry

2019-09-25

请问老师，怎么把微信，支付宝或者自定义的openid集成到dex？



作者回复: 这个问题, 我倾向于把它作为这个课程的作业。你可以试试自己做做看。



**Aaron Cheung**

2019-09-24

很好的介绍 打卡44

展开 ∨

