



Authenticating Users

Google App Engine

Agenda

1

Authentication

2

Authorization

3

Lab Exercise



Authentication

Google App Engine



Google Cloud Platform

Authentication Options

Option 1: Use built-in support

- Google Accounts (default)
 - Such as Google Mail, Plus, Docs, YouTube, and Calendar.
 - Google Apps for Work, Education, custom domains
- Federated Login / OpenID (“relying provider”) *Experimental*

Option 2: Build your own!

Built-in Authentication

Defined when creating an application

Authentication Options (Advanced): [Learn more](#)

Google App Engine provides an API for authenticating your users, including **Google Accounts**, **Google Apps**, and **OpenID**. If you choose to use this feature for some parts of your site, you'll need to specify now what type of users can sign in to your application:

☒ Open to all Google Accounts users (default)

If your application uses authentication, anyone with a valid Google Account may sign in.

☐ Restricted to the following [Google Apps](#) domain:

e.g. foo.com

If your application uses authentication, **only members of this Google Apps domain may sign in**. (e.g. an HR tracking tool) that is only accessible to accounts on your Google Apps domain. This option can

☐ (Experimental) Open to all users with an OpenID Provider

If your application uses authentication, anyone who has an account with an OpenID Provider may sign in. In the application's app.yaml or appengine-web.xml, set the `OpenID.Provider` to `ver.appspot.com`.

title:

authenticate to use your application.

entifier:

in the application's app.yaml or appengine-web.xml, set the `OpenID.Provider` to `ver.appspot.com`.

Cookie Expiration:

Default (1 Day) ▾

App Engine uses a cookie to keep users logged in to your application.

Authentication Options:

(Experimental) Federated Login ▾

The [Google Accounts API](#) includes all Gmail Accounts, but does not include accounts on Google Apps domains. [Learn more](#)

Can also be changed in Admin Console
<http://appengine.google.com>

Restricting Access to Your Application

By default, users do not need to login unless you set up restrictions

You can restrict *any URL* to require:

- User to be logged in, or
- User to be an admin

You can set restrictions in two ways

1. App configuration file
2. Users Service APIs

Restricting Access - Configuration Files

Java

```
<security-constraint>
  <web-resource-collection>
    <url-pattern>/profile/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>
```

web.xml

Python & Java

```
- url: /profile/*
  script: user_profile.py
  login: required

- url: /admin/*
  script: admin.py
  login: admin
```

app.yaml

Valid Role Names

- "*" (asterisk) for any valid user
- "admin" role for App Engine application viewers, developers, and owners

Restricting Access - Users Service API

Functions

- Get current login user
- Is current user admin?
- Create login or logout URL

Identify basic details about the logged in user

- Nickname
- Email
- User ID (unique for Google Accounts, could be empty for federated user)

Environment variable

- AUTH_DOMAIN - the domain your application is restricted to
 - gmail.com if open to all Google accounts

User Service API

java

```
UserService userService =
    UserServiceFactory.getUserService();

User user = userService.getCurrentUser();

if (user == null) { // not logged in
    String loginUrl = userService.createLoginURL(target);
    // redirect the user to the loginUrl.
} else { // user is logged in

    String email = user.getName();
    if (userService.isUserAdmin()) {
        // logged in user is an Admin
    }
}
```

User Service API

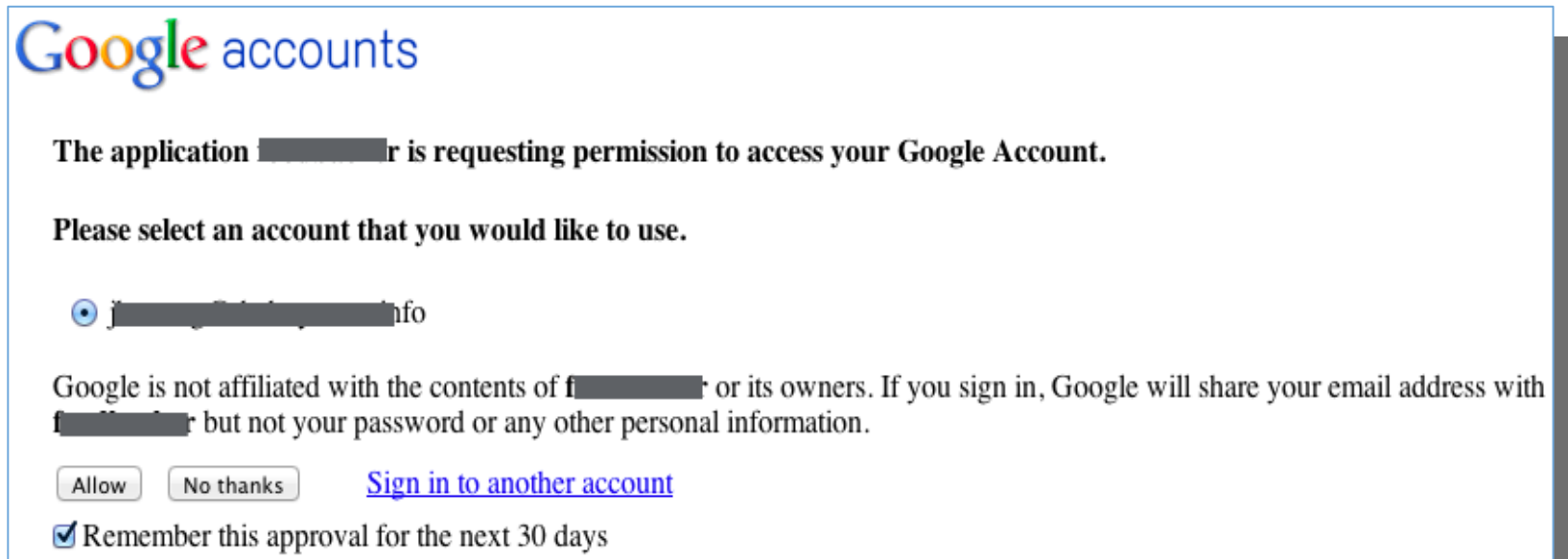
python

```
user = users.get_current_user()
if user:
    greeting = ("Welcome, %s! % (user.nickname()))
else:
    greeting = ("<a href=\"%s\">Sign in or register</a>." % users.
create_login_url("/"))

if user:
    if users.is_current_user_admin():
        print "<a href=\"/admin/\">Go to admin area</a>"
```

Grant Permission for User APIs

Open to All Google Accounts: After login to Google, prompts user upon initial access to authorize application to view email.



The screenshot shows a Google Accounts authorization dialog. At the top is the 'Google accounts' logo. Below it, a message states: 'The application [redacted] is requesting permission to access your Google Account.' This is followed by the instruction: 'Please select an account that you would like to use.' There is a single account listed with a radio button and the email address '[redacted]@hfo'. Below the account list, a disclaimer reads: 'Google is not affiliated with the contents of f[redacted] or its owners. If you sign in, Google will share your email address with f[redacted] but not your password or any other personal information.' At the bottom, there are two buttons: 'Allow' and 'No thanks', followed by a link 'Sign in to another account'. A checkbox labeled 'Remember this approval for the next 30 days' is checked.

Google accounts

The application [redacted] is requesting permission to access your Google Account.

Please select an account that you would like to use.

☒ [redacted]@hfo

Google is not affiliated with the contents of f[redacted] or its owners. If you sign in, Google will share your email address with f[redacted] but not your password or any other personal information.

[Sign in to another account](#)

☒ Remember this approval for the next 30 days

Restricted to single Google Apps domains - Does not prompt user for authorization

Custom Authentication

How?

- Setup application as "Open to All Google Account users"
- Do not restrict any access in your configuration files
- Implement custom authentication mechanism

Custom authentication mechanism within application

- Enterprise SSO systems (only if uses 80/443 ports, try Socket APIs)
- Encrypt Username/password in datastore



Authorization

Google App Engine



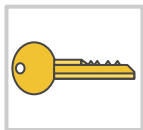
Google Cloud Platform

Authorization in App Engine Application



Internal access control within your application

- Use Users API to check the role
- Use Google Apps groups to check group membership
- Custom



Delegation to external systems with OAuth

- Access external protected resources (Consumer)



Expose application externally using Google Cloud Endpoints

- Expose content to external resources (Service Provider)

Internal Access Control

- **Users API role**

- Only support Admin role (project viewers, developers or owners)
- Check current user's role

- **Google Apps group**

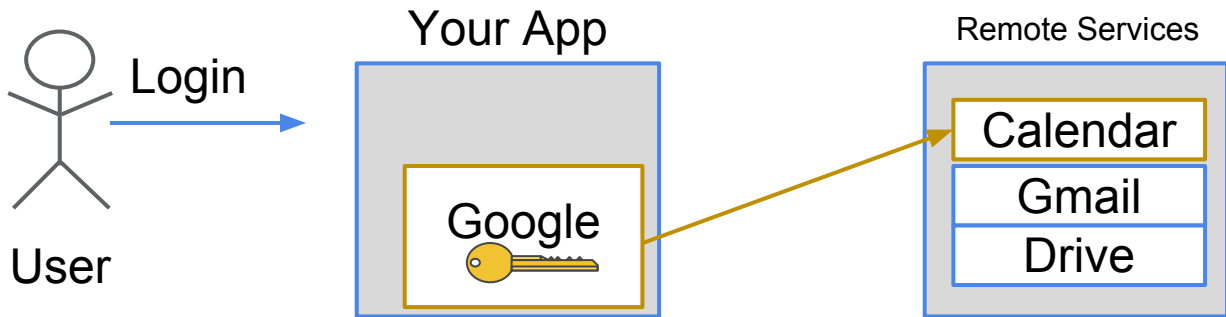
- Enterprise Google Apps customers sync Active Directory
- Requires Google Apps Admin SDK (Directory API)
- Caching the result for performance and avoid stale data

- **Custom**

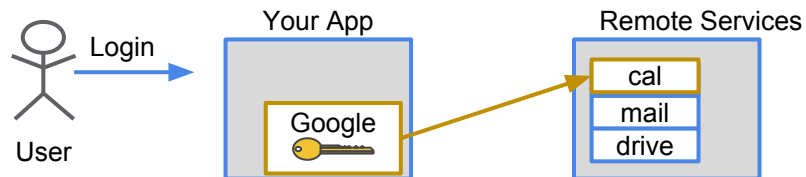
- Maintain list of roles attached to users (i.e., user ID)
- Needs to implement user administration functionalities

Delegation with OAuth

Access external protected resources

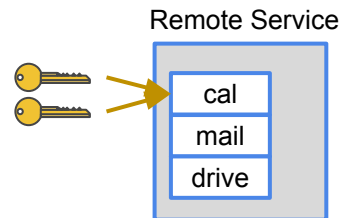


Access to External Resources



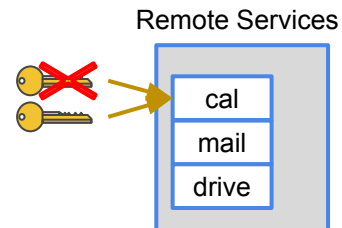
Scoped access to resources

Key	User	Scope
	manager@company.com	calendar
	user2@company.com	calendar
	manager@company.com	drive



Can be revoked by user or services at any time


Key	User	Scope
	manager@company.com	calendar
	user2@company.com	calendar
	manager@company.com	drive





Revoke Permissions

Users can revoke permission at accounts.google.com

Account Permissions ?


**Conference Central**
Has access to Google Calendar, Google Cloud Platform


**Google APIs Explorer**
Has access to Google Calendar

**Conference Central**

Revoke access

Conference Central has access to:

**Google Calendar**
Manage your calendars

**Google Cloud Platform**
View and manage your data in Google BigQuery










Authorized
June 26, 2013

OAuth Playground

OAuth 2.0 Playground ✕

▼ **Step 1** Select & authorize APIs

Select the **scope** for the APIs you would like to access or input your own OAuth scopes below. Then click the **"Authorize APIs"** button.

- ▶  Ad Exchange Buyer API v1.3
- ▶  Ad Exchange Seller API v1.1
- ▶  AdSense Host API v4.1
- ▶  AdSense Management API v1.4
- ▶  Admin Reports API reports_v1
- ▶  BigQuery API v2
- ▶  Blogger API v3
- ▶  Books API v1
- ▼  Calendar API v3
 - `https://www.googleapis.com/auth/calendar`
 - ✓ `https://www.googleapis.com/auth/calendar.readonly`

Authorize APIs

Use the OAuth Playground to experiment with OAuth:

<https://developers.google.com/oauthplayground/>

OAuth Best Practices

- Use SSL for all communications
- Request the smallest scope needed
- Request a refresh token only if needed
- Store the refresh token securely
- Use the same access token for as long as it is valid

Trick: store in memcache for validity
(just less than expiration)

- Be prepared to handle auth exception if access token expires
- Request new token before access token expires
- If processing OAuth on mobile device, open in full browser, not embedded browser display - *Allows users to verify URL and verify security*

Quiz

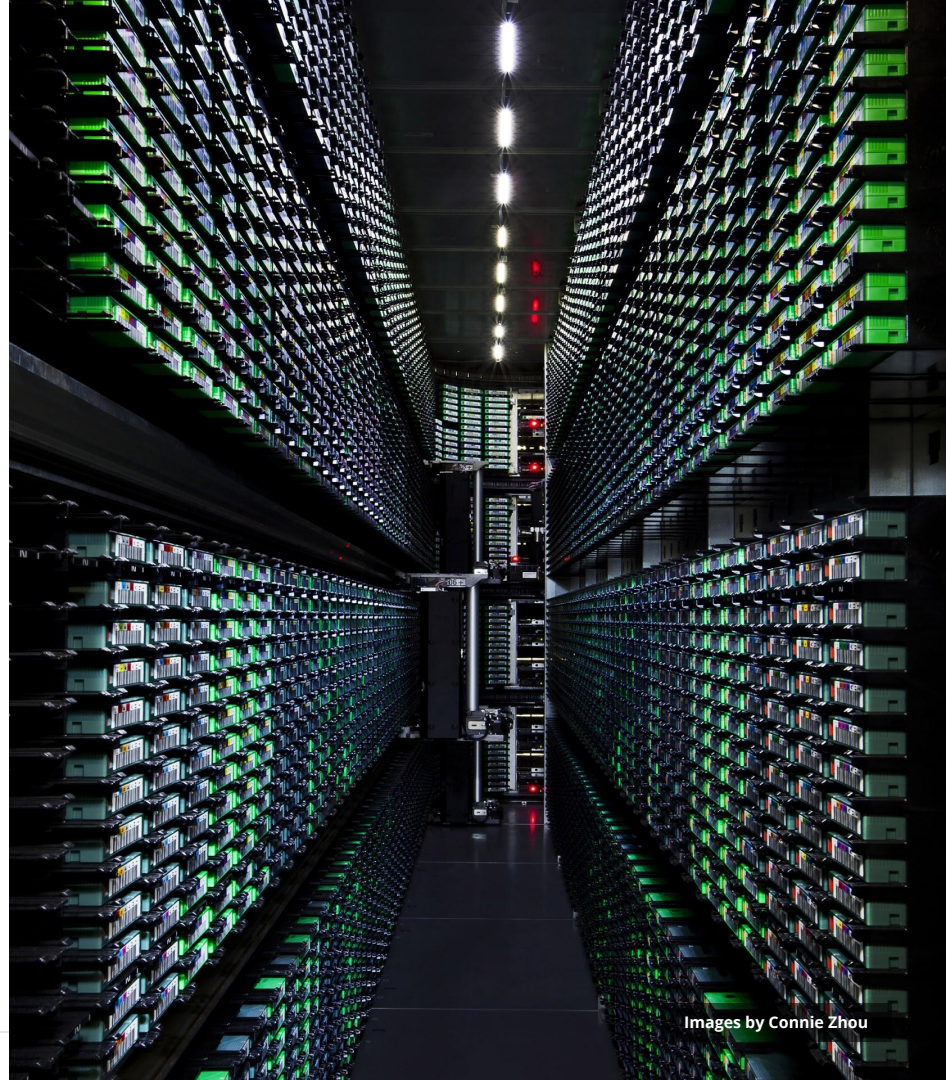
Which of the following **project permission** levels maps to the role of **admin** in the Users API?

*(select **3** of the available options)*

- ☐ Viewers
- ☐ Developers
- ☐ Administrators
- ☐ Owners
- ☐ All of the above
- ☐ None of the above

Codelab

- Implementing the Users API in Conference Central



Images by Connie Zhou

Resources

- Users Java API Overview
<https://cloud.google.com/appengine/docs/java/users/>
- Users Python API Overview
<https://cloud.google.com/appengine/docs/python/users/>
- Configuring Your App's Authentication
<https://cloud.google.com/appengine/articles/auth>



cloud.google.com