

MPS 第23回ミーティング (2015/3/14) 資料

Python で OAuth を使ってみよう！

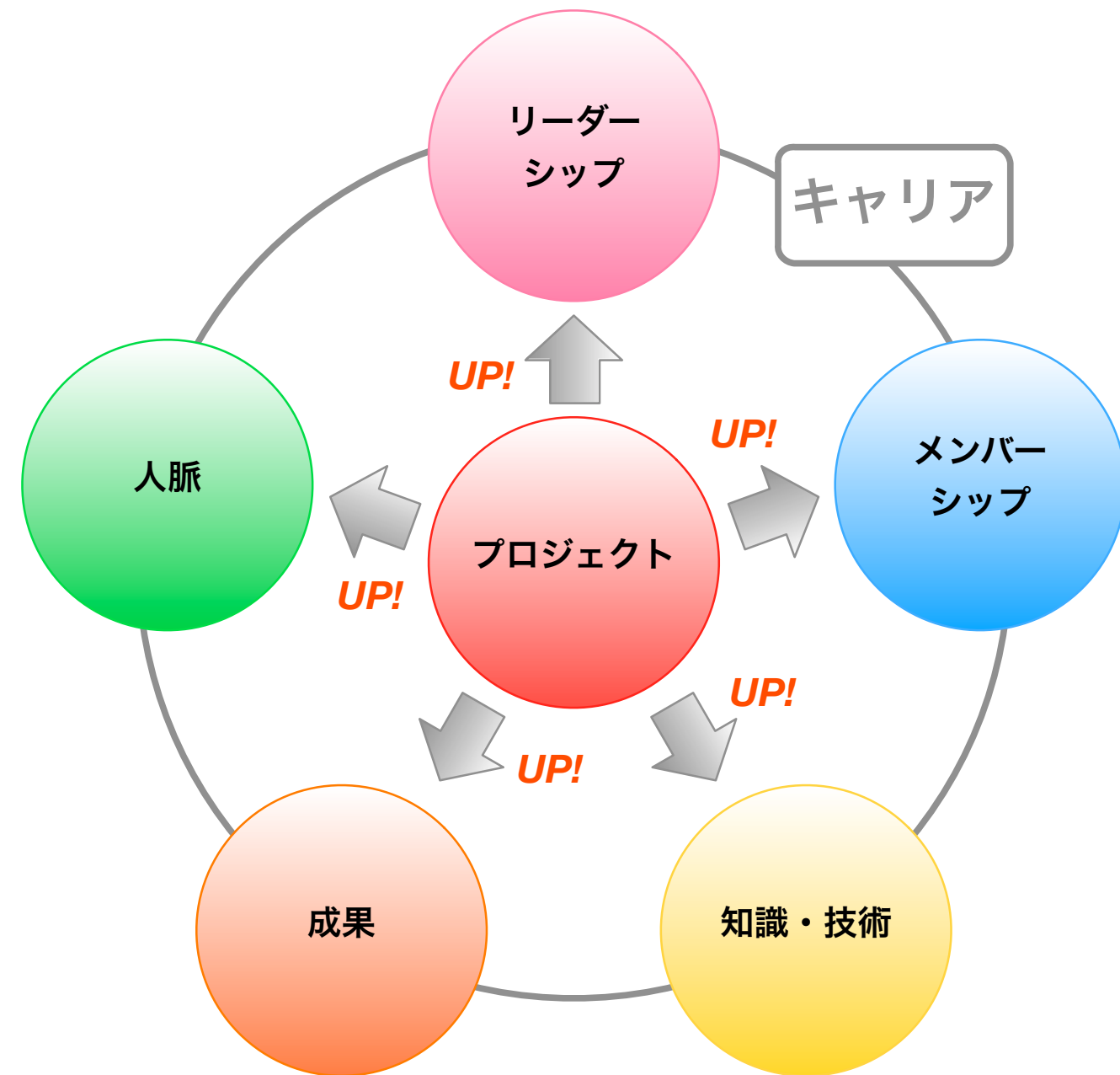
Morning Project Samurai 代表

金子純也

目次

- **MPS とは**
- 今回作成するアプリ
- セキュリティの基礎知識 (認証と承認)
- OAuth とは
- 認証コード認可 (Authorization Code Grant)
- アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)
- アプリ作成

Morning Project Samurai (MPS)



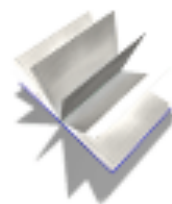
- Morning
 - 土曜の朝を有意義に
- Project
 - プロジェクト指向
- Samurai
 - 謙虚に学習
 - プロジェクトをバッサバッサと斬りまくる

これまでに行った活動

- 勉強会 (プレゼン)
 - Webアプリの安全性について (XSS実習)
 - コンピュータが動くメカニズム (論理回路基礎)
 - プログラムテストについて
 - JavaScript 入門 (実習)
 - Python を用いた Youtube 動画リストの作成
(プログラム基礎、オブジェクト指向、サーバーからのデータ取得、ドキュメントの検索と読み方、UML基礎)
 - Python から WebAPI を使ってみよう！
 - Python で簡易キャッシュを実装してみよう！
- プロジェクト
 - MPS HP
 - ぶらさぼり (東京メトロオープンデータ活用コンテスト)

MPS

Morning Project Samurai



MPS (Morning Project Samurai) は、
メンバー一人一人が「主体的」に「世の中の役に立つソフトウェア開発プロジェクト」を
提案し、
「リーダーとなる機会を持つことのできる環境」を作ります。

次回予告

組織概要

活動ガイド

メンバー紹介

ぶらサボり

(東京メトロオープンデータコンテスト出展作品)



Python + Django で開発

我々の今後の主な活動の一部

- ぶらサボりのコードの理解
- ぶらサボりのアップデート & リファクタリング
- ぶらサボりの海外展開 (Burasabori Abroad)
- ダイワハウスのスマートハウスコンテスト
- 手軽な情報交換デバイス
- サーバー環境の整備

目次

- MPS とは
- **今回作成するアプリ**
- セキュリティの基礎知識 (認証と承認)
- OAuth とは
- 認証コード認可 (Authorization Code Grant)
- アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)
- アプリ作成

今回作成するアプリ

OAuth を通して Google から

**特定ユーザーの情報にアクセスする承認をもらい、
承認を受けた情報を表示する**

アプリ



Virtualenv, Django, Heroku を用いて開発
するケロ！

目次

- MPS とは
- 今回作成するアプリ
- セキュリティの基礎知識 (認証と承認)
- OAuth とは
- 認証コード認可 (Authorization Code Grant)
- アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)
- アプリ作成

セキュリティってなんだろう

- 何詞？

セキュリティってなんだろう

- 何詞？
 - 名詞
- 何を指す名詞？

セキュリティってなんだろう

- 何詞？
 - 名詞
- 何を指す名詞？
 - 安全な状態
- コンピュータにおける安全な状態ってどんな状態？

セキュリティ

A system condition in which system resources are free from unauthorized access and from unauthorized or accidental change, destruction, or loss.

(RFC 4949)

リソースが承認されていないアクセス、変更、破壊、
消失に脅かされることのないシステムの状態

セキュリティを確保する方法
の例を考えてみよう！

セキュリティを確保する方法例

- ファイアウォール
 - Linux の iptables
 - ルーターのパケットフィルタリング機能
- ログイン機能
 - Windows や Mac OSX のログイン機能
 - GMail や Hotmail へのログイン機能
- パーミッション機能
 - Windows や Mac OSX のファイルパーミッション機能
- SSL (Secure Socket Layer)
 - https で始まるウェブサイト (MPS のウェブサイトなど)

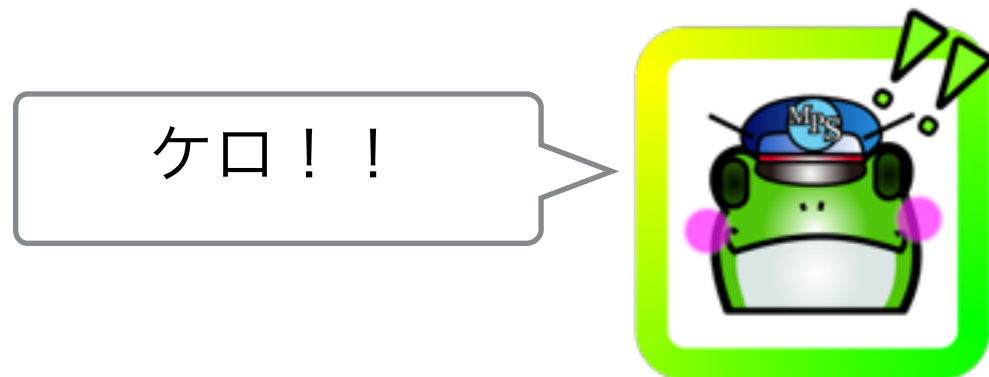
認証 (Authentication)

The process of verifying a claim that a system entity or system resource has a certain attribute value.

(RFC 4949)

システムの要素やリソースがある特定の属性値を持っているか検証するプロセス

例: パスワード認証、公開鍵認証、ケルベロス認証



承認 (Authorization)

A process for granting approval to a system entity to access a system resource.

(RFC 4949)

システムの要素がシステムリソースにアクセスする権利を承認するプロセス

例: パーミッションによるファイルへのアクセス制御



セキュリティ確保のための2ステップ

- 認証 (Authentication)

システムにアクセスしてきた者 (物) が誰か (何か) 特定する

- 承認 (Authorization)

システムにアクセスしてきた者 (物) のシステム上の権利を特定する

目次

- MPS とは
- 今回作成するアプリ
- セキュリティの基礎知識 (認証と承認)
- **OAuth とは**
- 認証コード認可 (Authorization Code Grant)
- アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)
- アプリ作成

OAuth とは

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, [...]

(出典: RFC 6749)

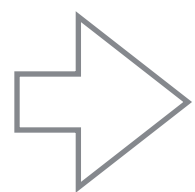
サードパーティー製のアプリケーションの
HTTP サービスへの制約付きアクセスを可能とする、
承認フレームワーク

OAuth とは

The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, [...]

(出典: RFC 6749)

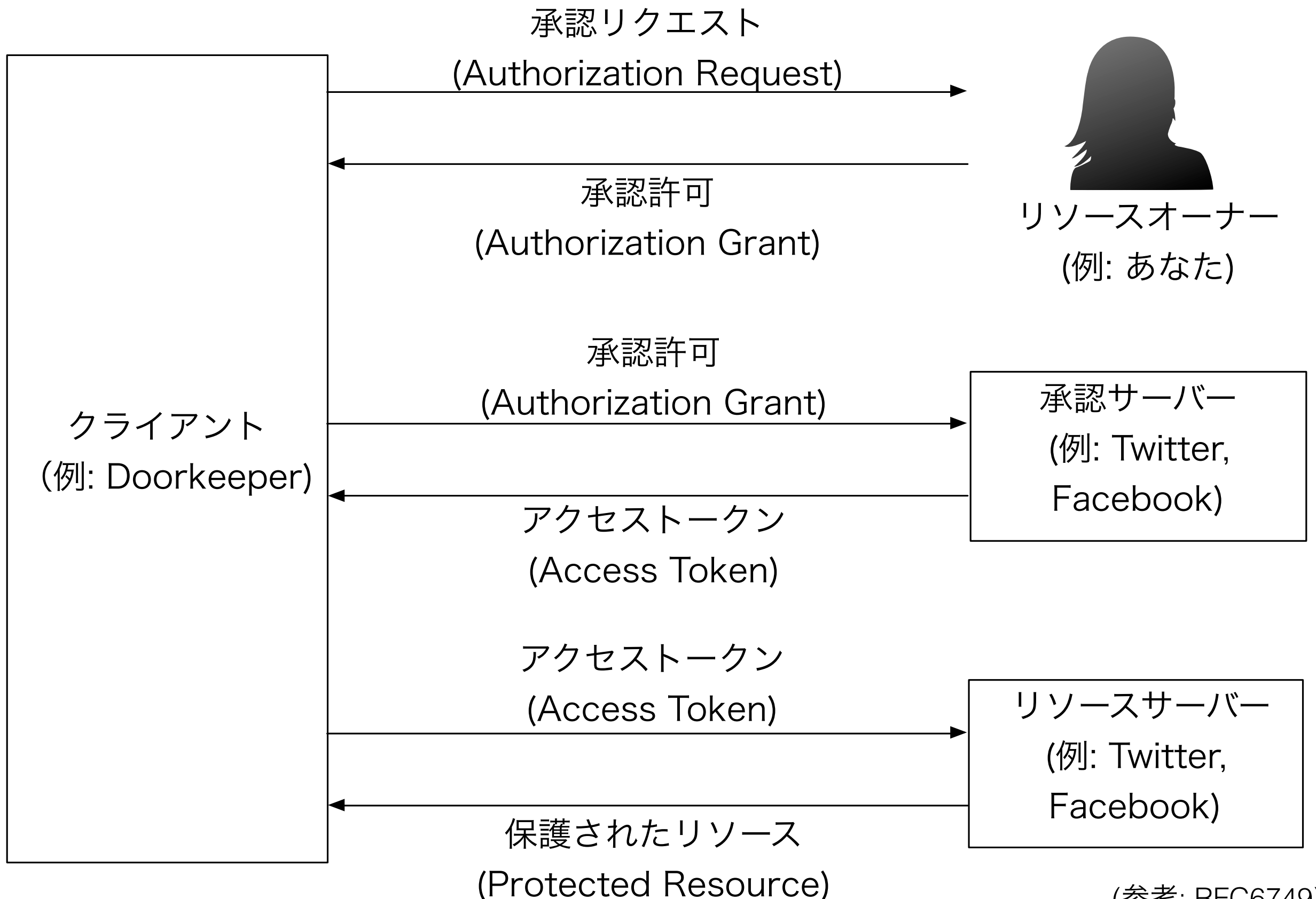
サードパーティー製のアプリケーションの
HTTP サービスへの制約付きアクセスを可能とする、
承認フレームワーク



~~「誰がアプリケーションにアクセスしているか」~~

**「サードパーティー製のアプリケーションが
どのリソースにアクセス可能か」**

OAuth による承認とリソースの取得



(参考: RFC6749)

気になるところ (問題編)

- 情報を本人 (リソースオーナー) でなく、
第三者 (リソースサーバー) から得るメリットは？
- 情報を第三者 (リソースサーバー) から得る際に
OAuth を使うメリットは？

気になるところ (回答編)

- 情報を本人 (リソースオーナー) でなく、
第三者 (リソースサーバー) から得るメリットは？
 - ユーザーエクスペリエンスの向上
 - 自分のサーバーで管理する個人情報の削減
- 情報を第三者 (リソースサーバー) から得る際に
OAuth を使うメリットは？
 - 標準化が進んでおり開発が容易
 - ユーザー認証情報の漏洩防止

OAuth の嬉しいところ

- リソースオーナー
 - 自分の認証用パスワードの公開が不要
 - 同じ情報をサービス毎に入力する手間の削減
- クライアント
 - 他のサービスのリソースを容易に利用可能
 - 自分のサーバーで管理する個人情報の削減
 - ユーザーエクスペリエンスの向上

4つの承認方法

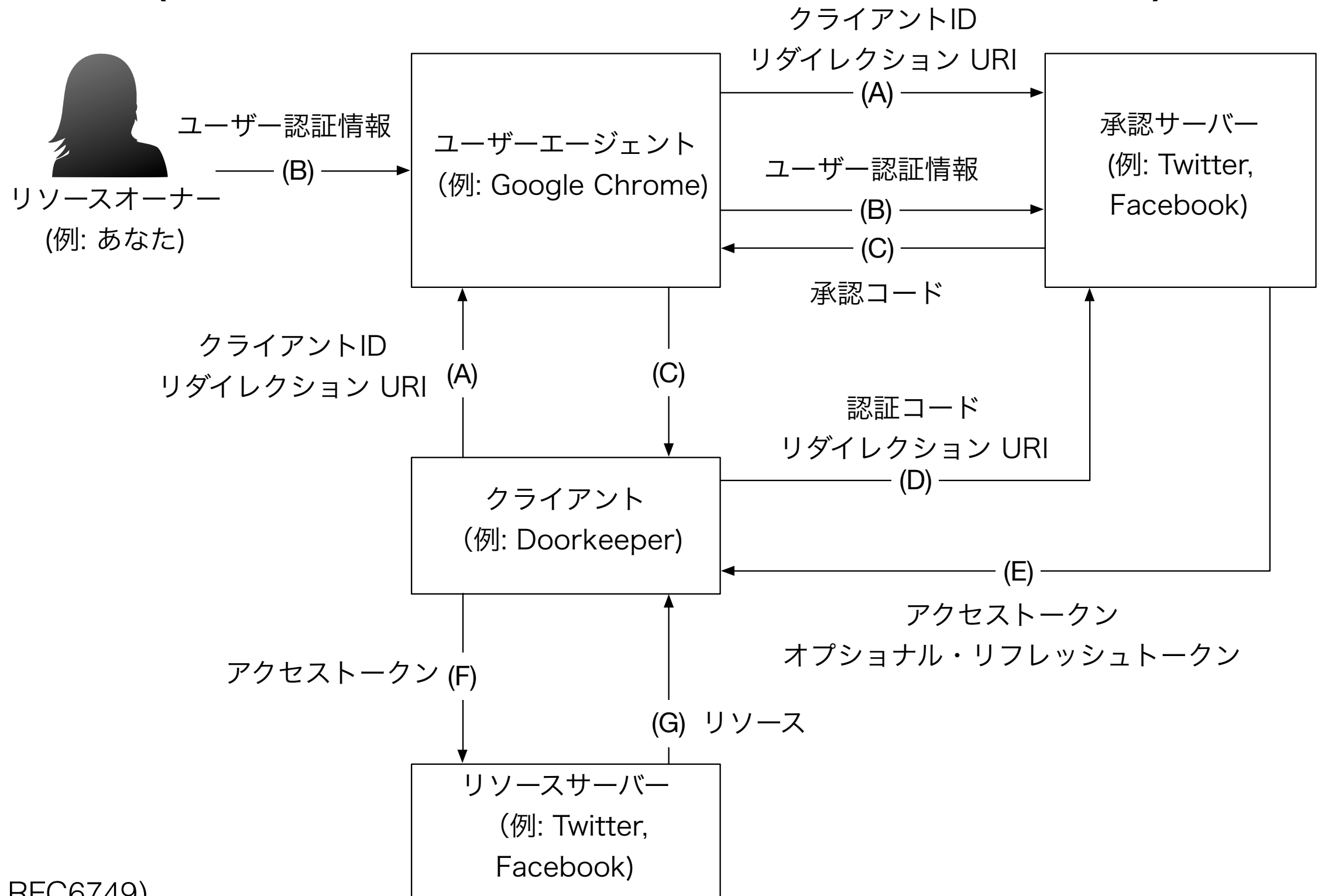
- 承認コード認可 (Authorization Code Grant)
- 暗黙的認可 (Implicit Grant)
- リソースオーナーパスワード認証情報認可
(Resource Owner Password Credentials Grant)
- クライアント認証情報認可 (Client Credentials Grant)

目次

- MPS とは
- 今回作成するアプリ
- セキュリティの基礎知識 (認証と承認)
- OAuth とは
- **認証コード認可 (Authorization Code Grant)**
- アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)
- アプリ作成


承認コード認可

(Authorization Code Grant)



(参考: RFC6749)

OAuth を用いた承認スタート！



The screenshot shows the Doorkeeper login interface. At the top, the Doorkeeper logo is on the left, and links for '新規登録' (New Registration), 'ログイン' (Login), and 'English' are on the right. The main content area is titled 'Doorkeeper ログイン'. It features four social login buttons: 'Facebookでログイン', 'Twitterでログイン', 'GitHubでログイン' (which is highlighted with a red dashed border), and 'LinkedInでログイン'. Below these is a section for email login, starting with 'またはメールでログイン'. It includes input fields for 'メールアドレス' (Email Address) and 'パスワード' (Password), both marked with a red asterisk. There is a checkbox for '次回から自動的にログイン' (Log in automatically next time) and a link for 'パスワードを忘れた方はこちら' (Click here if you forgot your password). A blue 'ログイン' (Login) button is at the bottom.

Doorkeeper ログイン

f Facebookでログイン

Twitterでログイン

GitHubでログイン

in LinkedInでログイン

またはメールでログイン

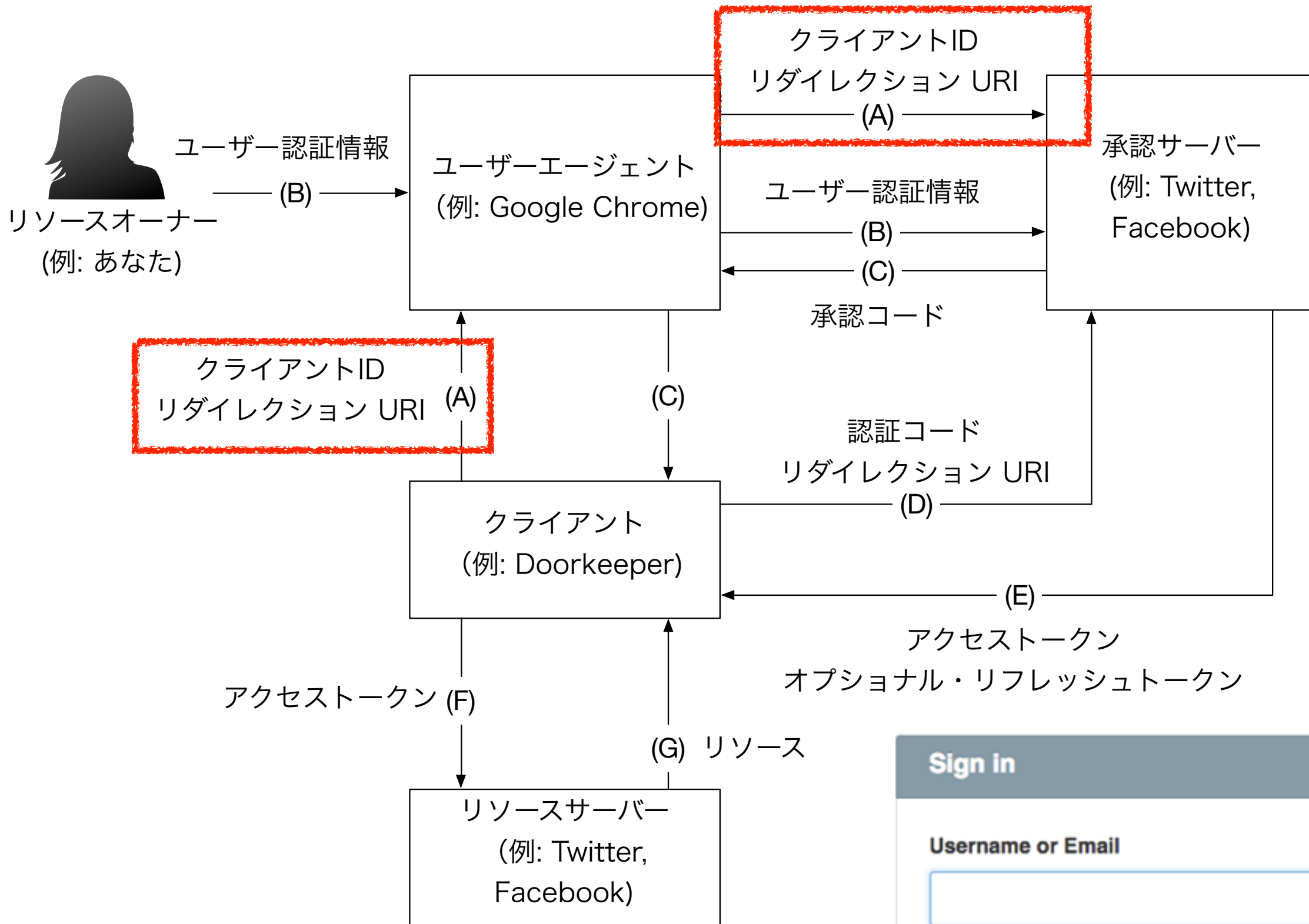
メールアドレス *

パスワード *

☒ 次回から自動的にログイン

パスワードを忘れた方はこちら

ログイン



Sign in

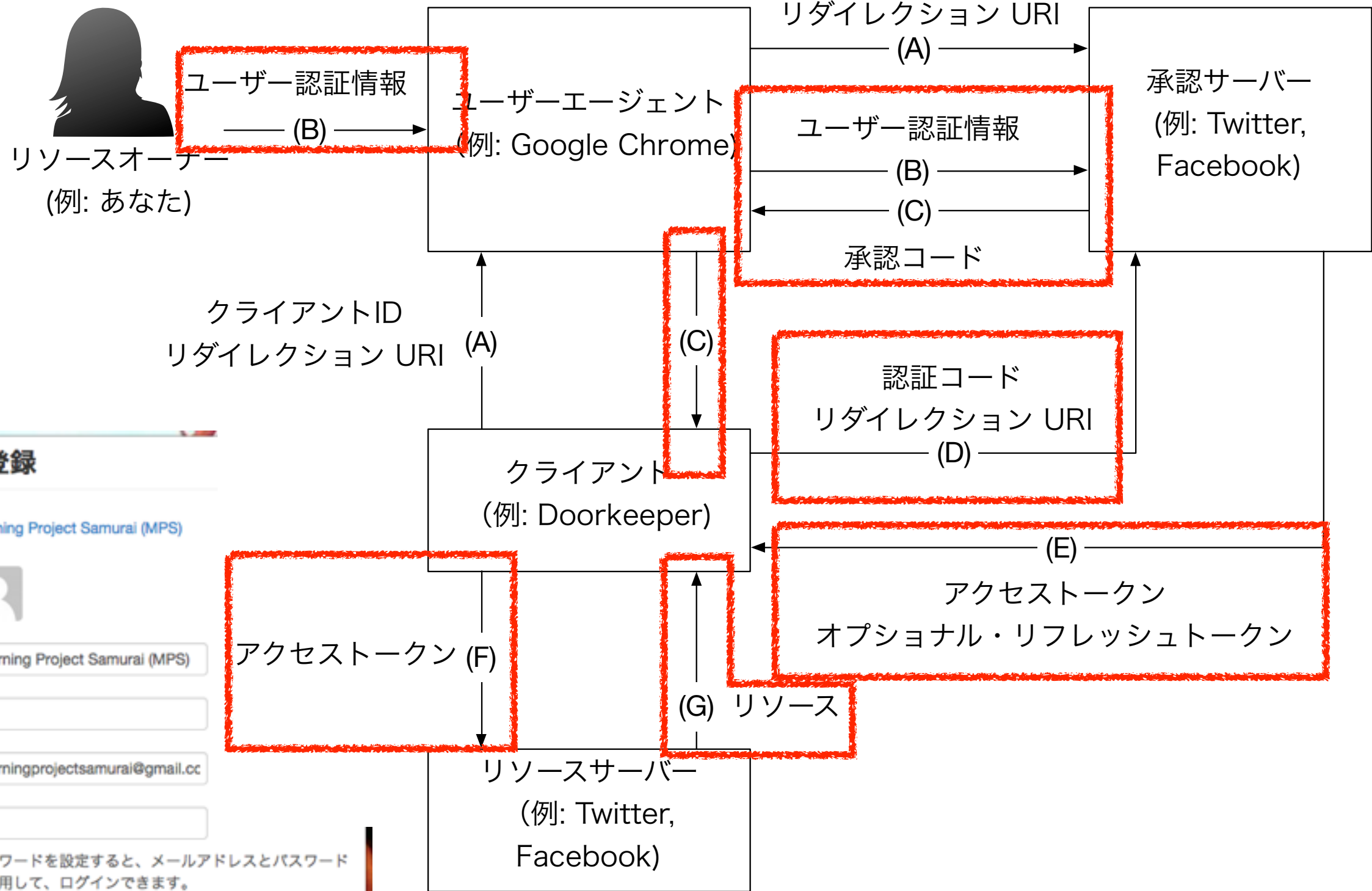
Username or Email

Password (forgot password)

Sign in

GitHub に Doorkeeper が渡す情報

`https://github.com/login?
return_to=/login/oauth/authorize?
client_id=515d17e1c4ffbc7816b7&
redirect_uri=
https://manage.doorkeeper.jp/user/auth/github/callback&
response_type=code&
scope=user email&
state=718f3af0d28620437fcb9b0ed0a72868665cccb59aabc148`



Doorkeeper 新規登録

GitHub アカウント Morning Project Samurai (MPS)

プロフィール写真

名前 * Morning Project Samurai (MPS)

名前 (カタカナ)

メールアドレス * morningprojectsamurai@gmail.cc

パスワード

パスワードを設定すると、メールアドレスとパスワードを使用して、ログインできます。

☒ 近日開催予定のイベントに関する週刊ニュースレターを受け取る

アカウントを作成すると、[利用規約](#)に同意したことになります。

(参考: RFC6749)

目次

- MPS とは
- 今回作成するアプリ
- セキュリティの基礎知識 (認証と承認)
- OAuth とは
- 認証コード認可 (Authorization Code Grant)
- **アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)**
- アプリ作成

Virtualenv のインストール

仮想の Python 実行環境を構築するアプリ

1. virtualenv 12.0.7 をダウンロード
2. コマンドラインで下記を実行

```
$ tar xvfz virtualenv-12.0.7.tar.gz
```

```
$ cd virtualenv-12.0.7
```

```
$ sudo python3 setup.py install
```

Virtualenv を使ってみよう

1. 仮想環境の作成

```
$ mkdir mps_env  
$ virtualenv mps_env
```

2. 環境の切り替え

```
$ mv mps_env  
$ source bin/activate
```

3. 環境の確認

```
(mps_venv)$ which python  
mps_env/bin/python にパスが通っていれば OK
```

Django のインストール

Python 上で動く

ウェブアプリケーションフレームワーク

1. インストール

```
(mps_env)$ pip install django
```

3. インストールできているか確認

```
(mps_env)$ which django-admin.py  
mps_venv/bin/django-admin.py にパスが
```

通っていれば OK

Heroku への登録と設定

1. www.heroku.com にアクセスしてユーザー登録
2. Documentation -> Get Started -> Python -> Setup
と進み、Heroku Toolbelt をダウンロードして
インストール
3. (mps_env) \$ heroku login
コマンドラインから Heroku にログインできれば OK
4. 今回のアプリの動作に必要なモジュールのインストール
(mps_env) \$ pip install django-toolbelt
(mps_env) \$ pip install xmltodict

Heroku へのアプリケーション登録

1. アプリケーションの鋳型のダウンロード

```
(mps_env)$ git clone https://github.com/morningprojectsamurai/20150314MPS.git
```

2. アプリケーションの Heroku への登録

```
(i) (mps_env) $ cd 20150314MPS
```

```
(ii) (mps_env) $ heroku create
```

3. Heroku 上のアプリケーションの更新

```
(i) (mps_env) $ git add —all
```

```
(ii) (mps_env) $ git commit -m “any comment”
```

```
(iii) (mps_env) $ git push heroic master
```

Google の設定

1. Google Developer Console にアクセス

<https://console.developers.google.com/project>

2. プロジェクト作成

Create Project をクリックして必要事項の入力

3. クライアント ID の作成

(i) 作成したプロジェクト名 -> APIs & Auth -> Credentials

-> Create new Client ID をクリック

(ii) Web Application を選択し、AUTHORIZED REDIRECT URI に

下記 URL を記入してCreate Client ID をクリック

URL: 作成した Heroku 上のウェブアプリのURL/callback/

4. プロダクト名の設定

Consent Screen をクリックし、PRODUCT NAME を適当に

埋めて save をクリック

目次

- MPS とは
- 今回作成するアプリ
- セキュリティの基礎知識 (認証と承認)
- OAuth とは
- 認証コード認可 (Authorization Code Grant)
- アプリ作成の下準備
(Virtualenv, Django, Heroku, Google)
- **アプリ作成**

今回のアプリの目的

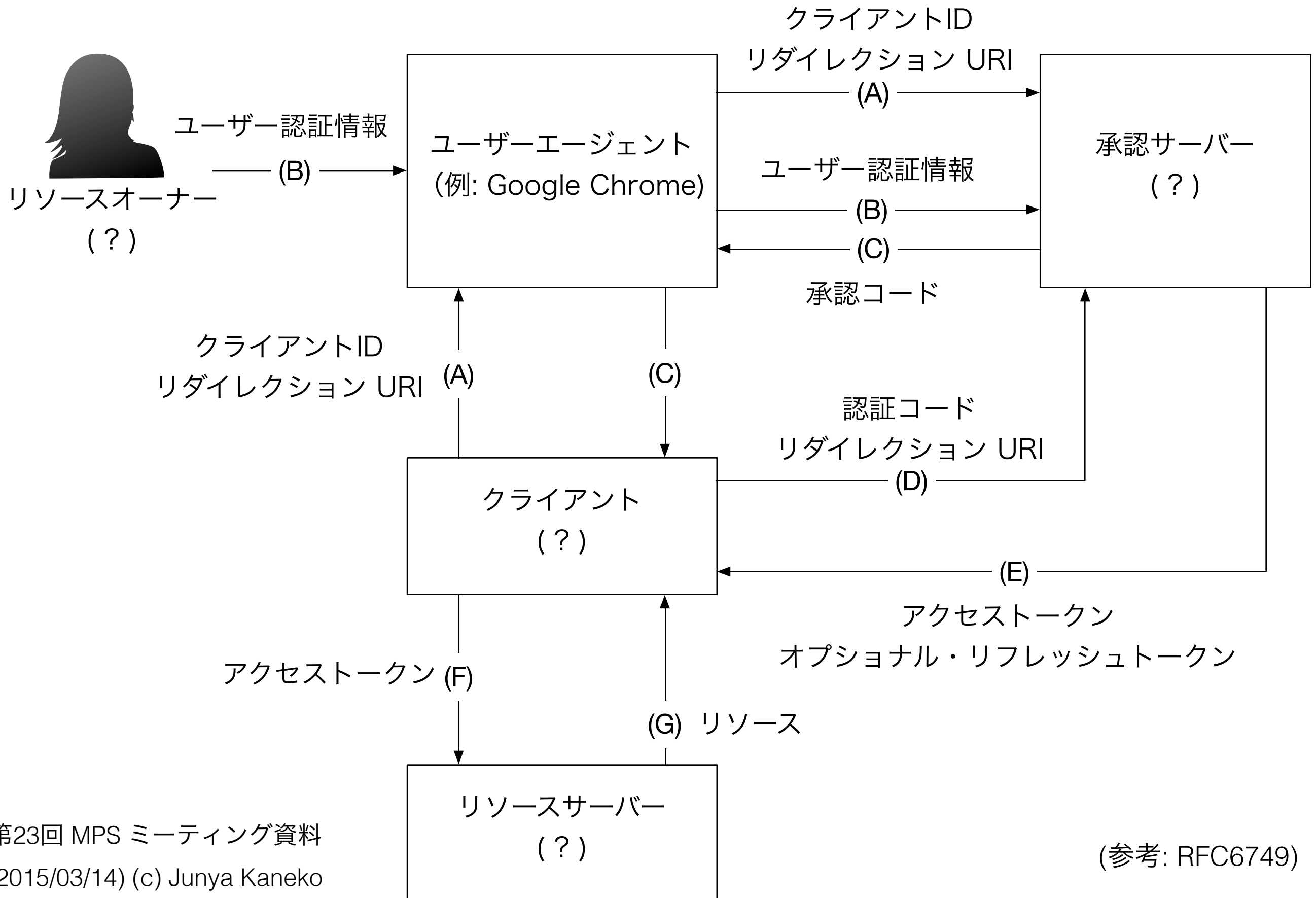
OAuth を通して Google から

**特定ユーザーの情報にアクセスする承認をもらい、
承認を受けた情報を表示する**

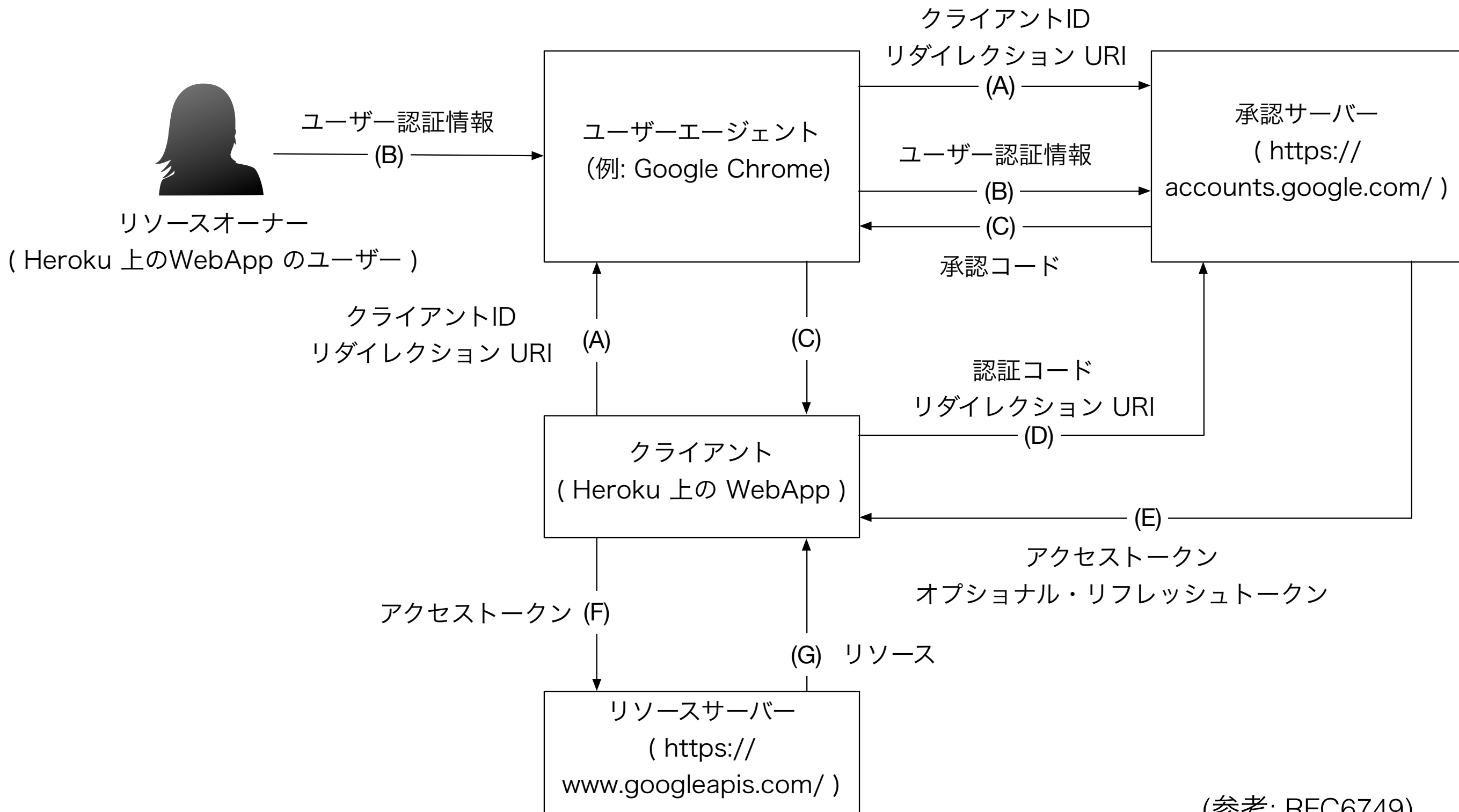


Virtualenv, Django, Heroku を用いて開発
するケロ！

Q. (?) をみんなで埋めてみよう！



A. 回答例

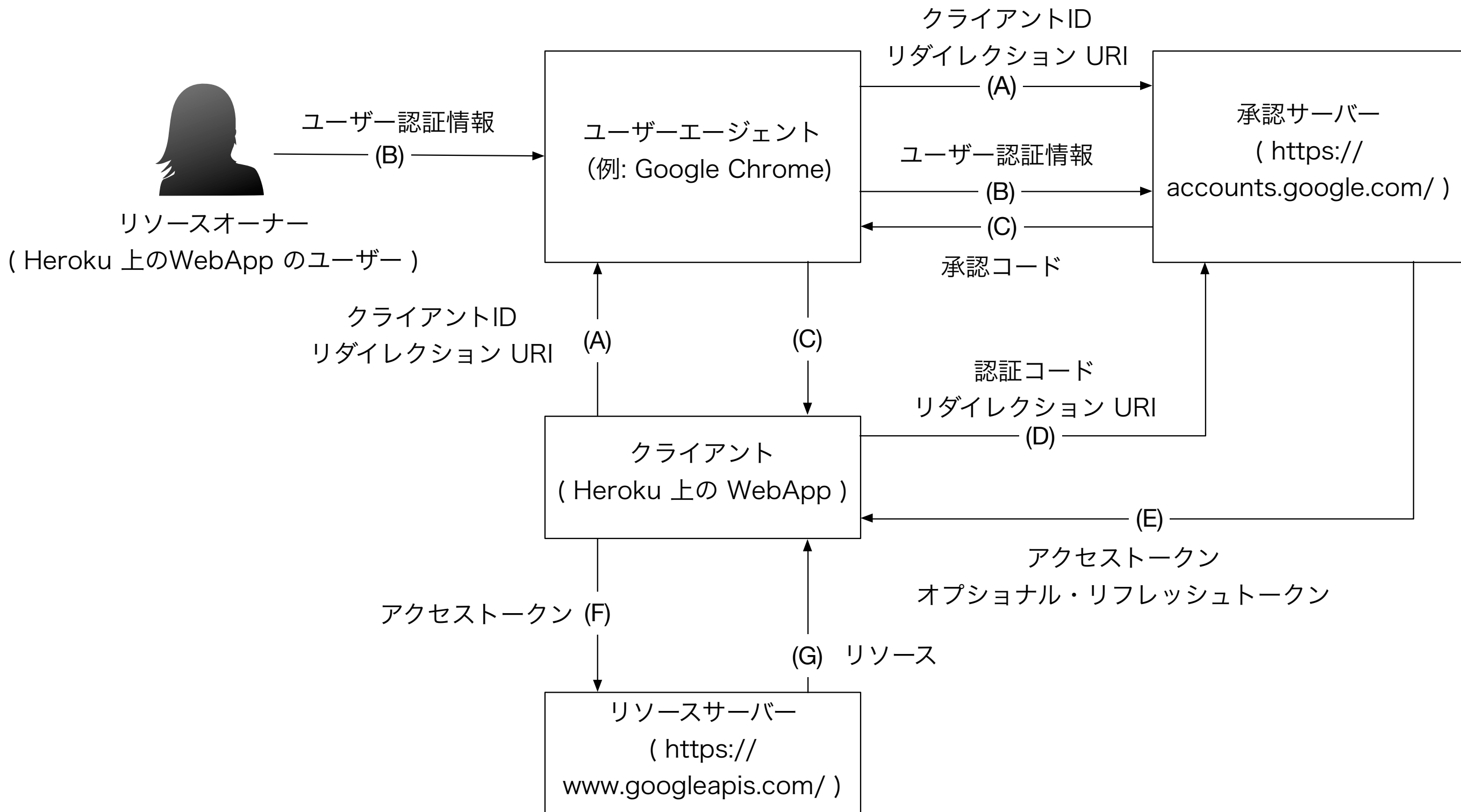


(参考: RFC6749)

各サーバーのエンドポイント

- 承認サーバー
 - <https://accounts.google.com/o/oauth2/auth>
 - <https://accounts.google.com/o/oauth2/token>
- リソースサーバー
 - <https://www.googleapis.com/oauth2/v2/userinfo>

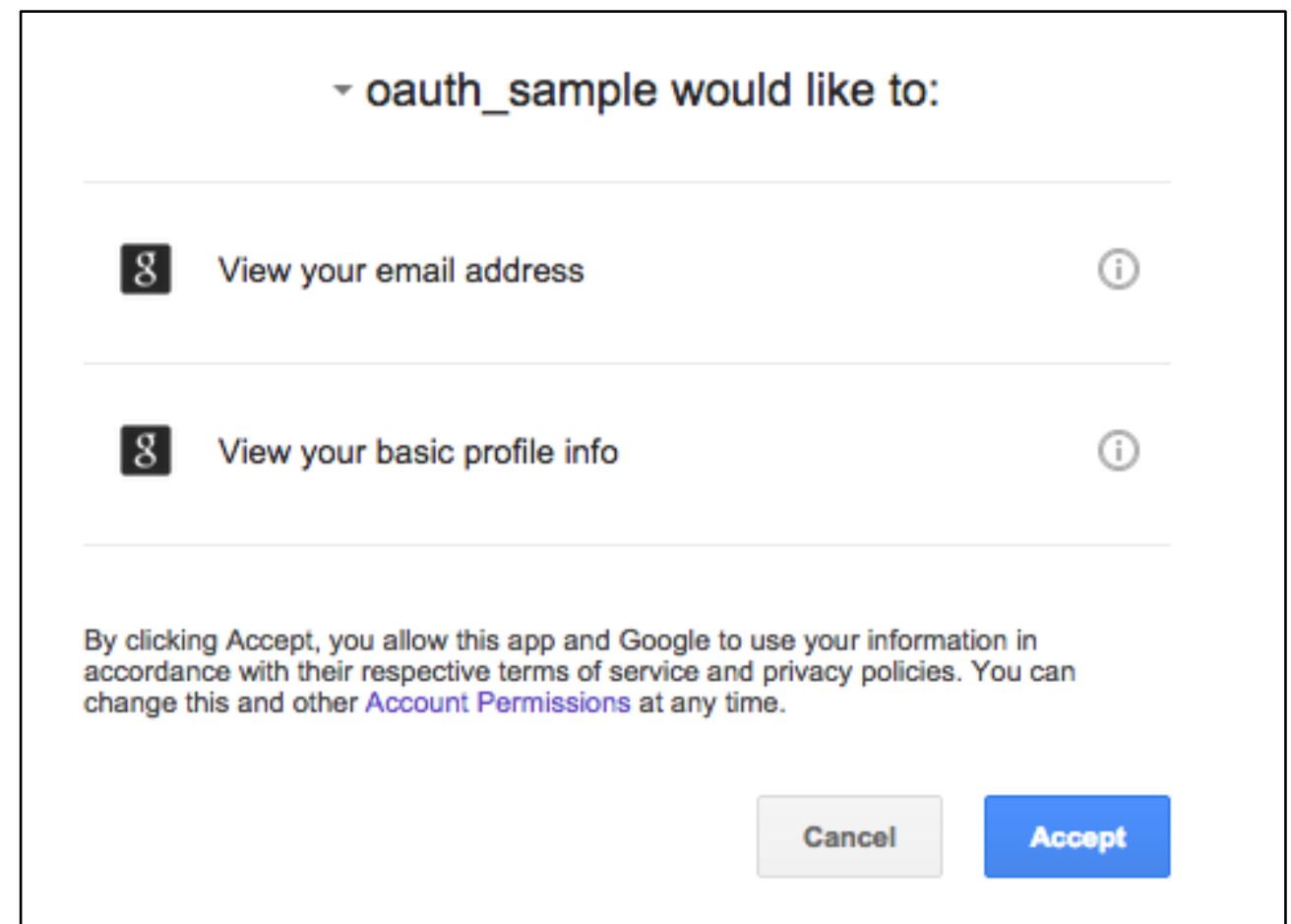
必要な画面と遷移を考えてみよう！



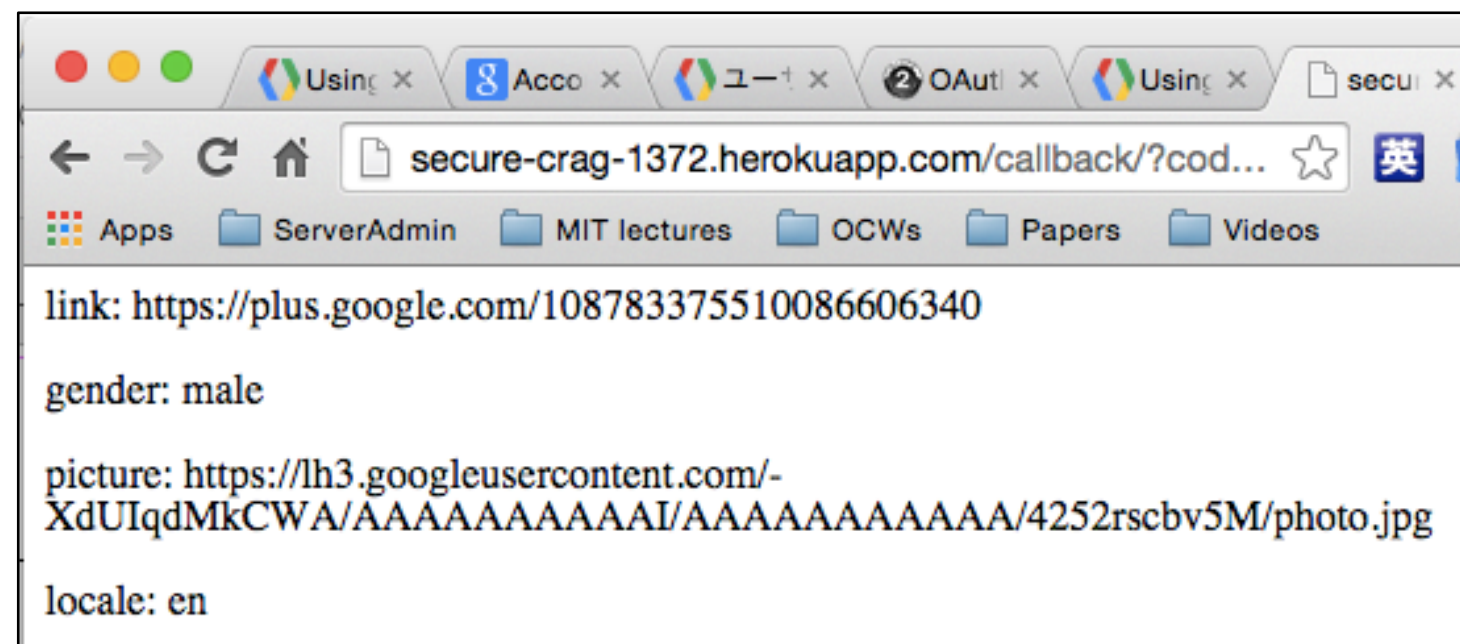
初期画面



承認画面



プロフィール画面

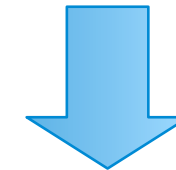
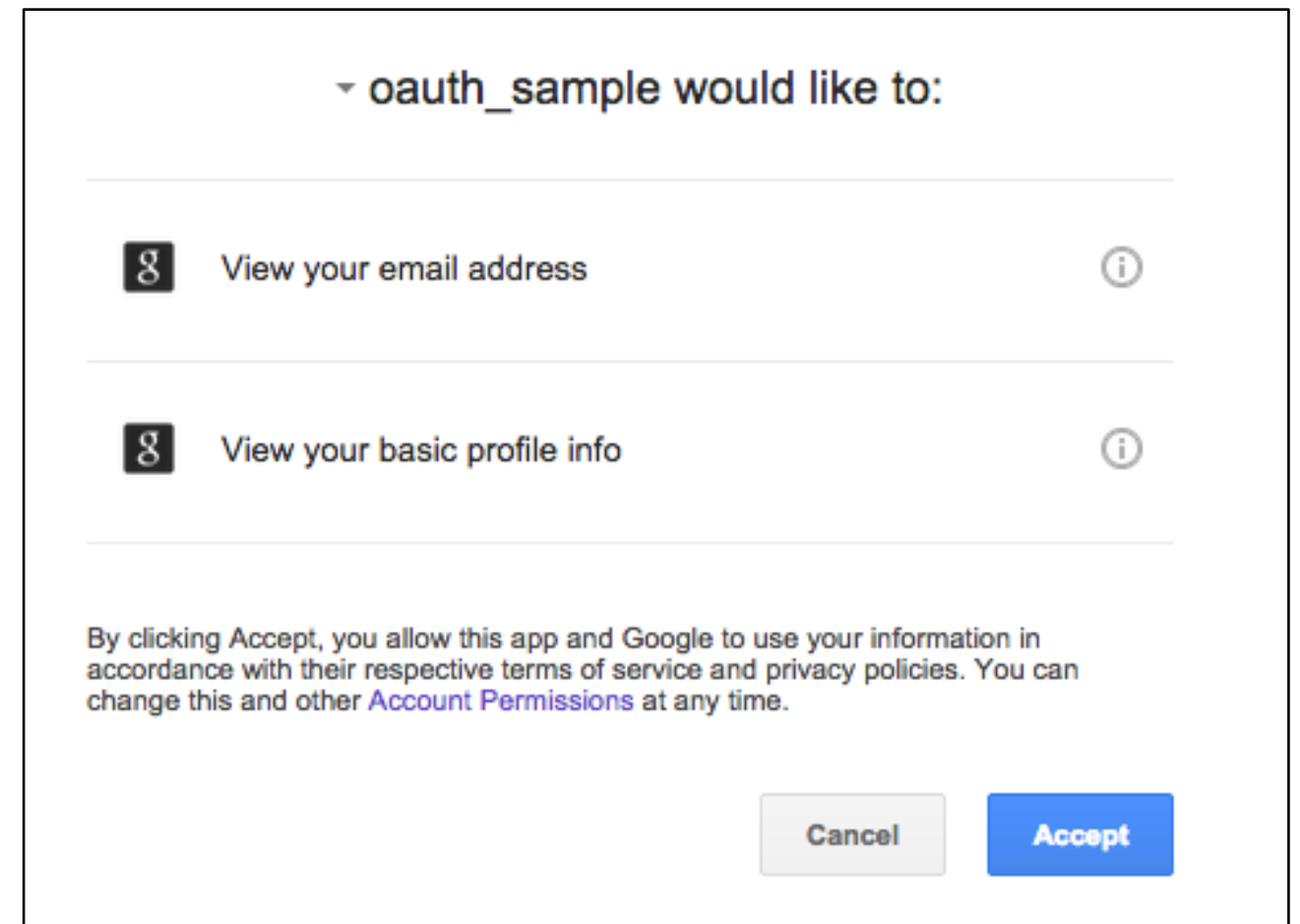
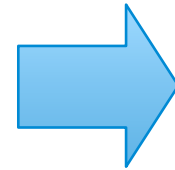


初期画面

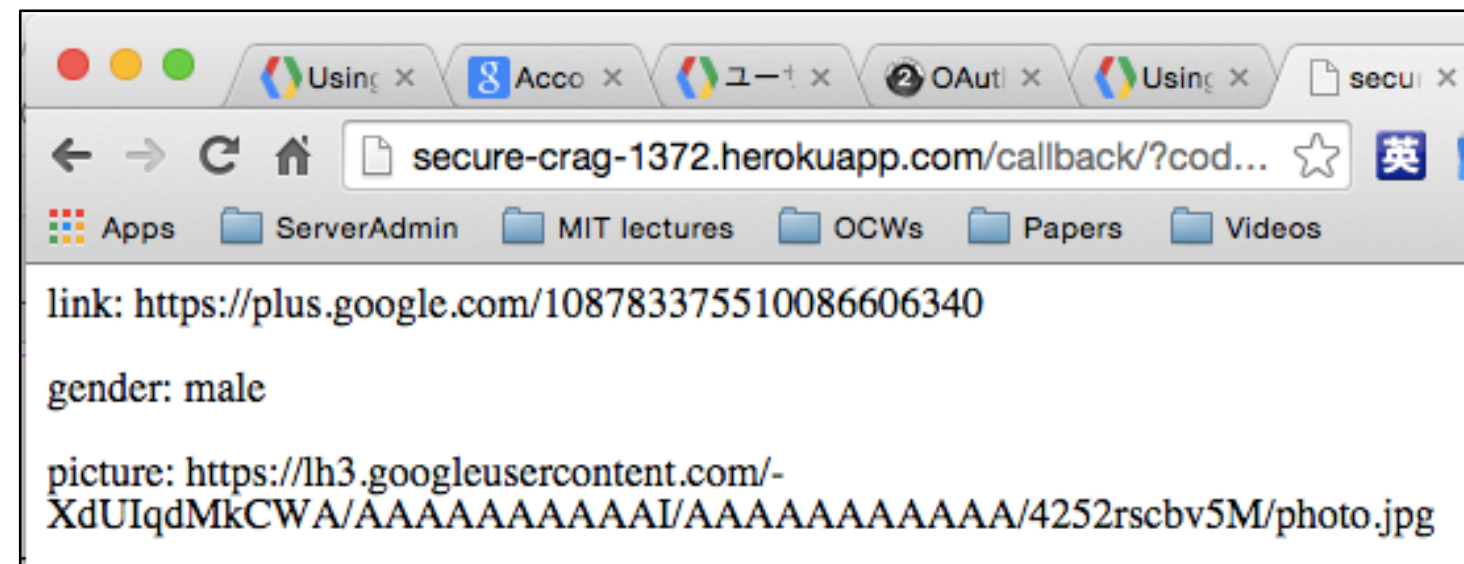


secure-crag-1372.herokuapp.com

承認画面

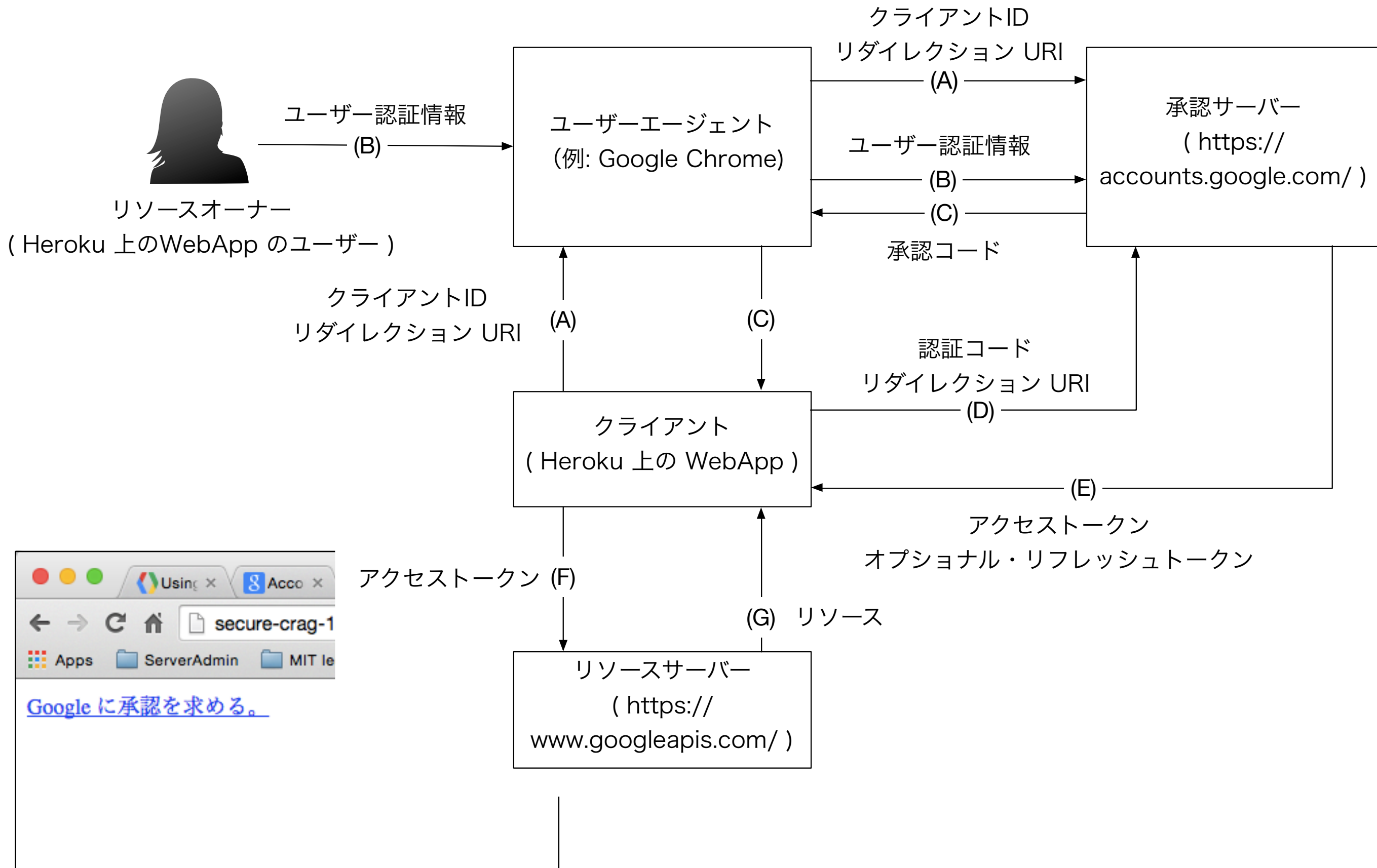


プロフィール画面

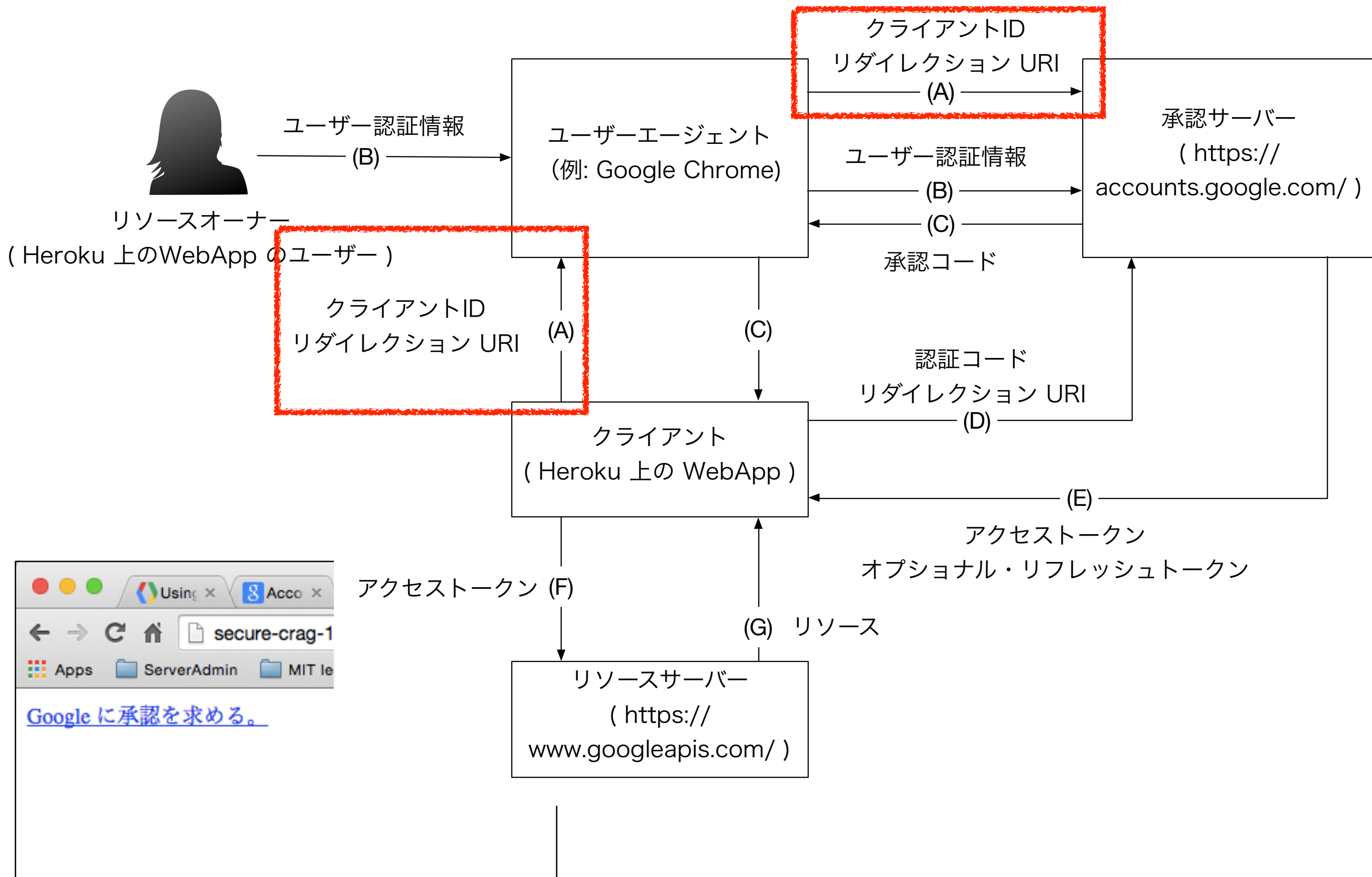


secure-crag-1372.herokuapp.com/callback/

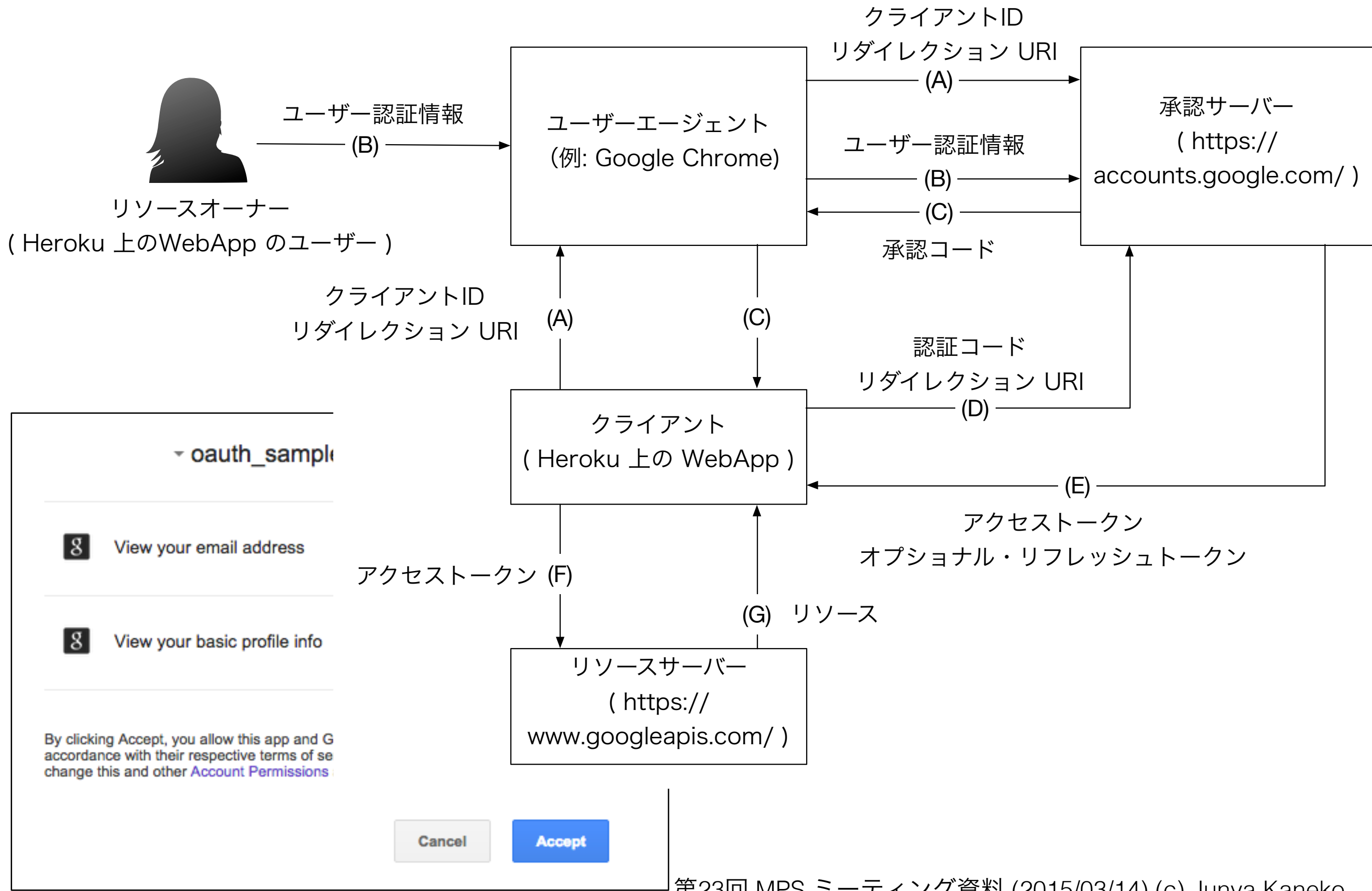
画面と処理の対応関係 - 初期画面 -



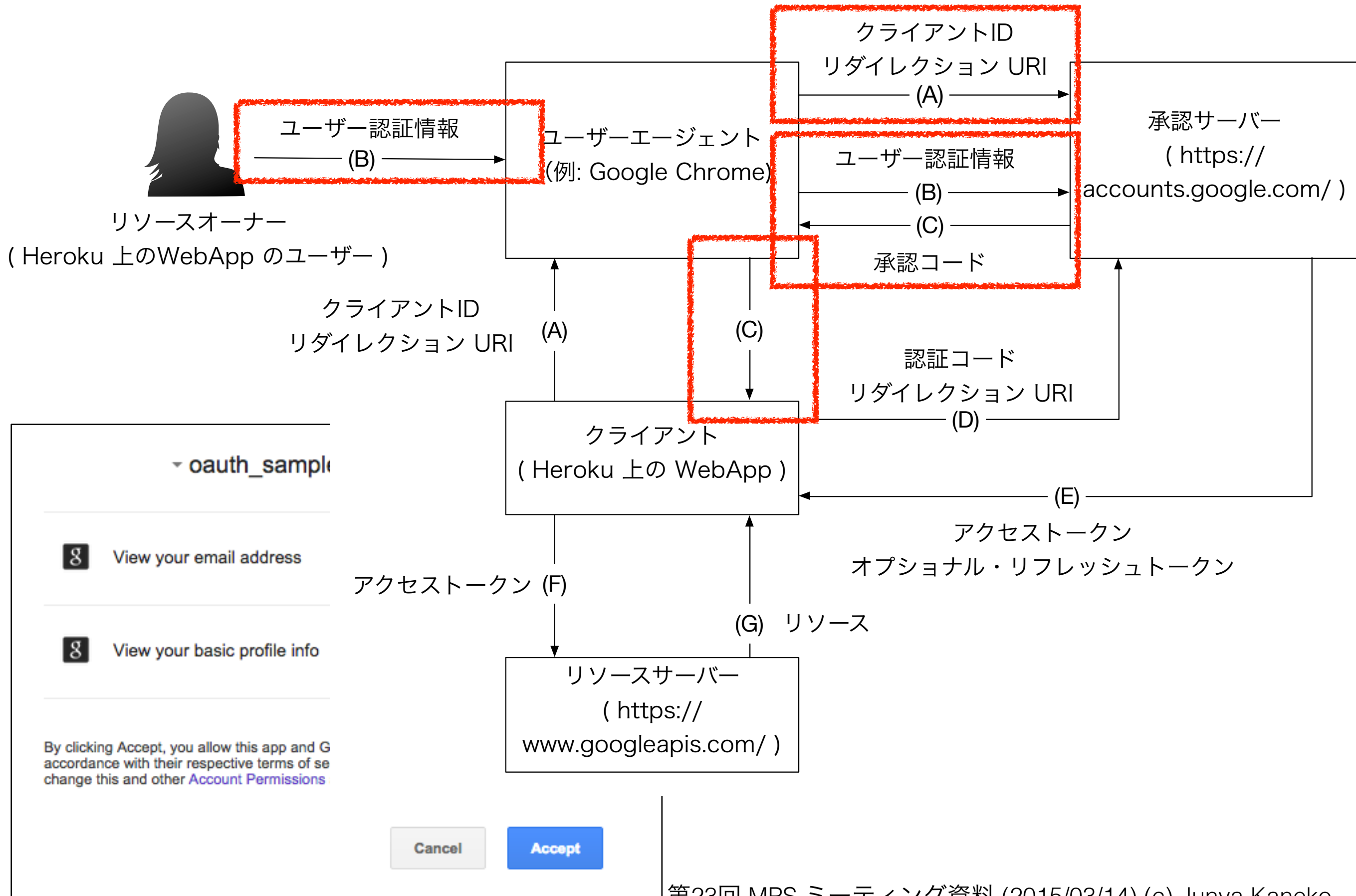
画面と処理の対応関係 - 初期画面 -



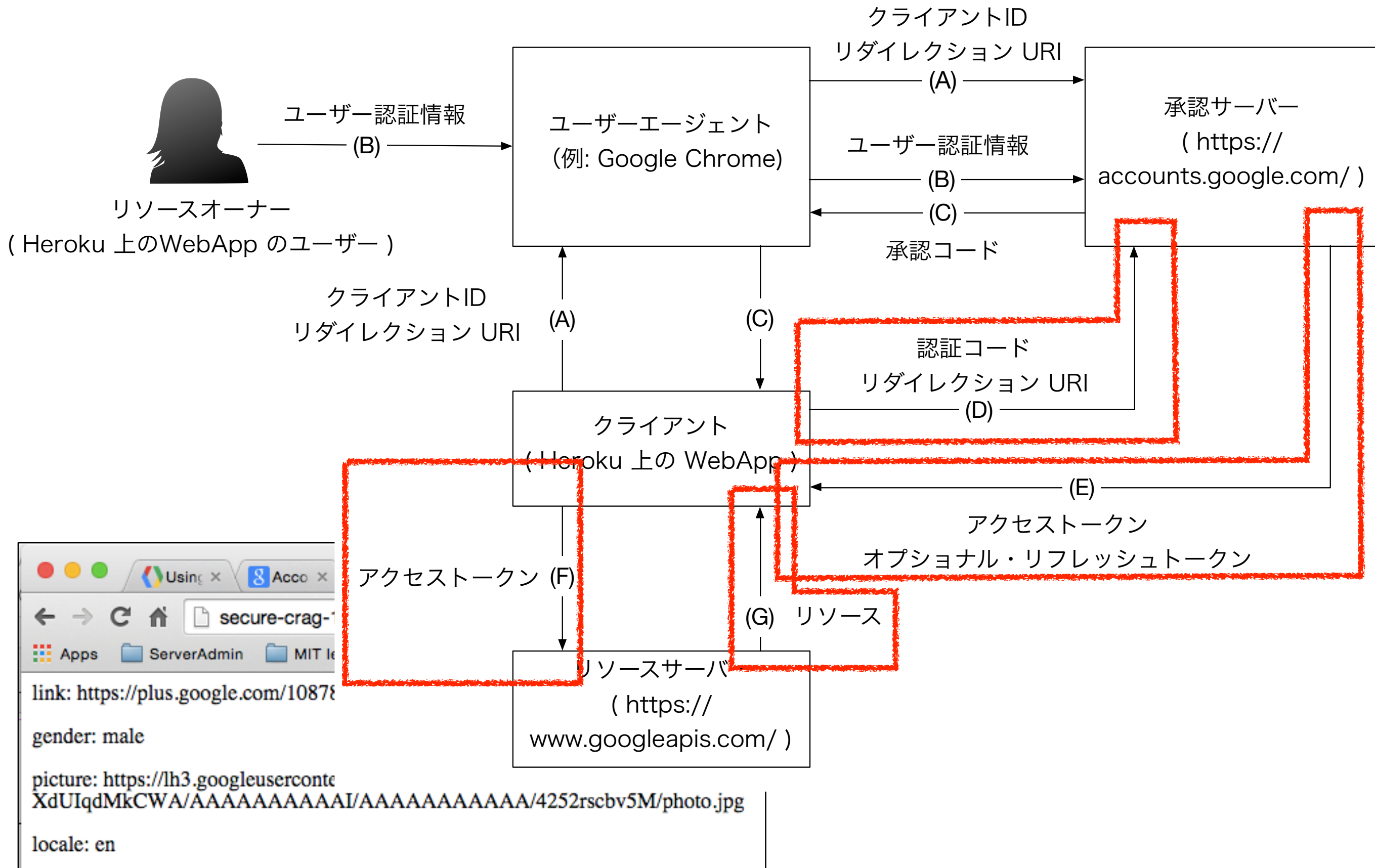
画面と処理の対応関係 - 承認画面 -



画面と処理の対応関係 - 承認画面 -



画面と処理の対応関係 - プロフィール画面 -



Django における画面の設計

- views.py に画面に関わる処理を書く
- 1つの画面に対して1つのクラス

例: google/views.py

```
class WelcomeView(View):  
    def get(self, request):  
        # GET メソッドでリクエストを受けた時の処理  
  
    def post(self, request):  
        # POST メソッドでリクエストを受けた時の処理
```

画面、クラス、 URL の対応付け

- 初期画面

クラス: WelcomeView

URL: Heroku 上のアプリのアドレス/

- プロフィール画面

クラス: CallbackView

URL: Heroku 上のアプリのアドレス/callback/



この対応付けは、urls.py を使って行われるケロ。
今回は、すでに設定済みだケロ。

まずは自分で作ってみよう！

- 20150314MPS/google/views.py を編集
(できたら Heroku 上のアプリを更新)
- 主に必要な知識は urllib と WebAPI の使い方
(前回までの企画を思い出そう!)
- Google OAuth のドキュメント
<https://developers.google.com/accounts/docs/OAuth2WebServer>



まずは、ドキュメントを読みながら試行錯誤するケロ！