User Can Sign In Via Facebook

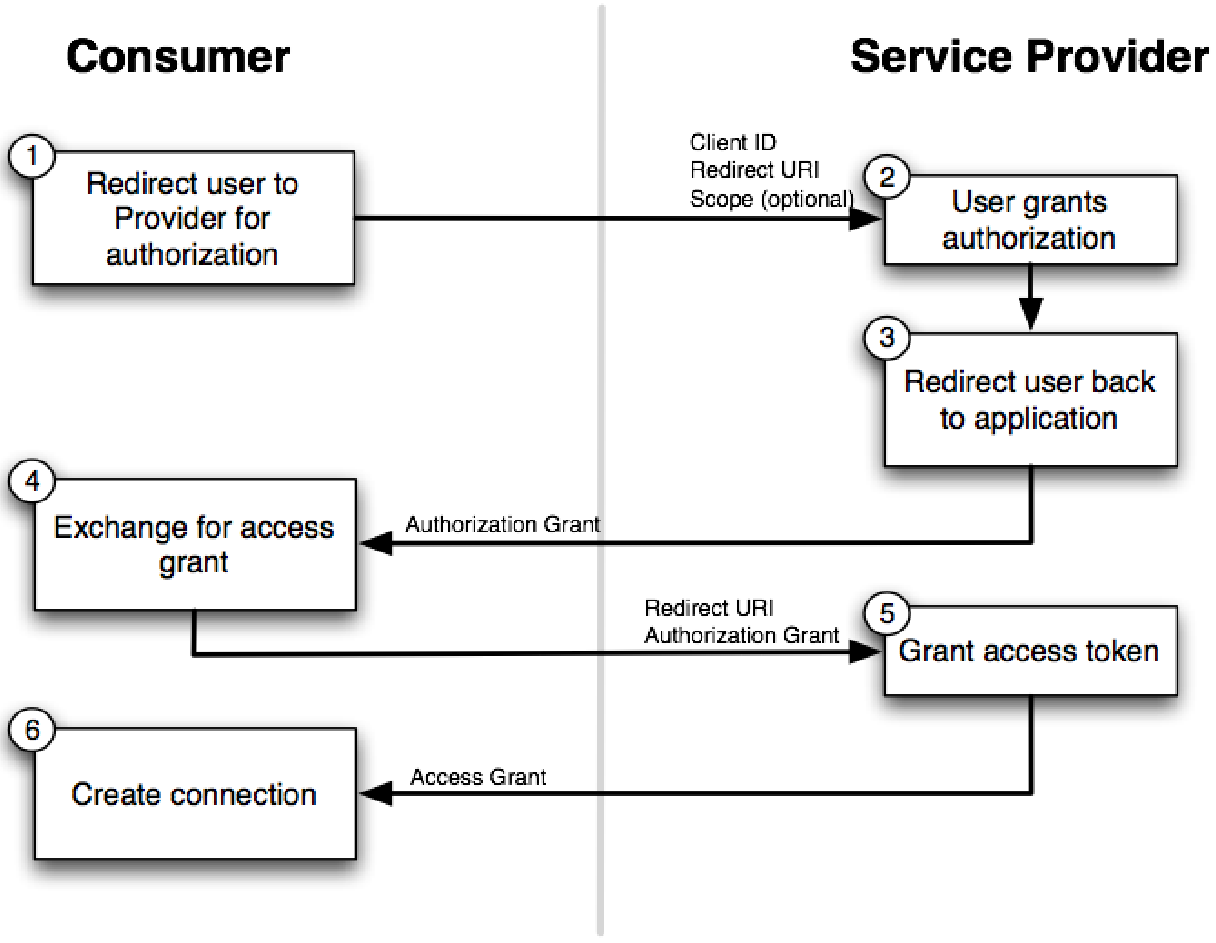
Let’s make your app even more convenient for your users!

Many apps offer the convenience of logging user in via other popular services such as Facebook, Twitter, Google, Github and more. There are many reasons for this. Some reasons include:

1. Ability for user to sign up for an account on the app without the need of typing any email or passwords.
2. Ability to have access to the other users’ applications such as reading their tweets, or making a tweet on their behalf.

OAuth is an open standard for web application security for apps to ‘talk’ to each other regarding **users**. Modern applications that applies the OAuth protocol are using the newer version of OAuth, namely OAuth 2.

You can look at the following diagram to understand the flow of an OAuth authentication process:



Difference between OAuth 1 and OAuth 2 is pretty well explained here: https://stackoverflow.com/questions/4113934/how-is-oauth-2-different-from-oauth-1

Objectives

Now, it's your turn to figure out how to actually implement this in your app!

Instead of handling the whole flow on your own like the diagrams above, an easier way to do this is to use a `gem`: omniauth-facebook, which uses the OAuth2 protocol.

In the previous exercise, you have built a user authentication with Clearance. You could also integrate Facebook authentication into your app with the help of the gem `'omniauth-facebook'`, while using the gem `'clearance'`. Let's get started!

Register your app at Facebook for Developers

Before you write code, let's **register our app** at Facebook for Developers. In this case, Facebook is the *authorization provider*.

To register your app, you can either log in using your personal Facebook account or signup for a new account for development purposes. After logging into your developer account, look at the right-hand corner to "Add new app". Fill in the necessary information required. Once you have registered your app with Facebook, you will get a *key pair* (i.e. an app key and app secret that is unique to your app). This helps the provider (i.e. Facebook) recognize your app when a user wants to log in to your app via their Facebook account. Now that you have the keys, keep them somewhere safe first!

Documentation Reading

When you are done, you can read this documentation for omniauth-facebook. When you have a better understanding of what the gem is about, let's implement it!

Implement Facebook Authentication

1. Add the gems `omniauth` and `omniauth-facebook` to your `Gemfile`.

PS: Don't forget to add the `figaro` gem if you haven't done so! Link

2. Run `bundle install`.

3. Paste your Facebook `app key` and `app secret` in `application.yml`. For example:

```
FACEBOOK_APP_KEY: '....'  # Note the use of the single quotes ' ' here.
FACEBOOK_APP_SECRET: '...'
```

4. According to the documentation, each OmniAuth strategy is a Rack middleware. We will now add the `OmniAuth::Strategies::Facebook` Rack middleware into our app. Create a file called `omniauth.rb` in the folder `config/initializers`. In the file, add the following code:

```
# In config/initializers/omniauth.rb
Rails.application.config.middleware.use OmniAuth::Builder do
  provider :facebook, ENV['FACEBOOK_APP_KEY'], ENV['FACEBOOK_APP_SECRET']
end
```

5. Before we move on, let’s understand the concept of an authentication hash which can be found here in this link.

An *authentication hash* is a hash that contains data provided from the provider about the user after authenticating with the provider. For Facebook, the list of information that can be obtained for the user is given here.

6. Now, in your User model (`app/models/user.rb`), write the following code. Note that your `auth_hash` might differ from the following code depending on the attributes you have for your User model. The documentation provides a list of possible configurations for the authentication hash. Remember you have to make the necessary changes to in your omniauth.rb file as well. Read the link above carefully to see what you need to do if you want other attributes beside name and email. (Hint: you will need some code in your `omniauth.rb` file, particularly `scopes`.)

```
# In app/models/user.rb
class User < ApplicationRecord
  has_many :authentications, dependent: :destroy


  def self.create_with_auth_and_hash(authentication, auth_hash)
    user = self.create!(
      name: auth_hash["name"],
      email: auth_hash["extra"]["raw_info"]["email"]
    )
    user.authentications << authentication
    return user
  end

  # grab fb_token to access Facebook for user data
  def fb_token
    x = self.authentications.find_by(provider: 'facebook')
    return x.token unless x.nil?
  end
end
```

The first method, create_with_auth_hash will create a User object based on the information given by the provider. What about the second method fb_token? What do you think it means?

7. Note that the above setup is meant for the user to key in his/her own Facebook password to log in to our app and such that we need not store this user's password. However, in our database schema, the encrypted_password column does not allow for a null column, due to a migration generated by the Clearance gem. Let's change it so that it can be a nullable column.

```
# In the terminal:
$ rails g migration change_password_required_for_users


# In the generated migration file:
```

```
change_column_null :users, :encrypted_password, true
```

Run `bundle exec rake db:migrate` to apply the change.

8. In the user model, you will have noticed that there is a `has_many` association with a model called **Authentication**. This would be useful if you have more than one authorization provider. (Eg. Twitter, Google, Github)

Next we will need to set up an authentication system for a user. Let's first generate a migration file with the following set up:

```
rails g model authentication uid:string token:string provider:string user:references
```

Remember to run `bundle exec rake db:migrate` in the to apply the change in the **new** migration file.

The attributes are described as follow:

- `uid`: this is a string generated by the provider that uniquely identifies a user
- `token`: this will take the form of a string and is an access token which takes the form of a string. It is generated by the provider after successful authentication by the provider of our app, and approved authorization by the user (note that we need not store this unless we want to perform various actions on the user's behalf - for example - fetch friends' information, post a status on our behalf, etc).
- `provider`: the authorization provider. Also a string.
- `user_id`: this user ID refers to the ID of the user stored in our app's database.

9. Next, in `app/models/authentication.rb`, set up the authentication class as follows:

```
class Authentication < ApplicationRecord
  belongs_to :user


  def self.create_with_omniauth(auth_hash)
```

```
    self.new(
      provider: auth_hash["provider"],
      uid: auth_hash["uid"],
      token: auth_hash["credentials"]["token"]
    )
  end


  def update_token(auth_hash)
    self.token = auth_hash["credentials"]["token"]
    self.save
  end
end
```

Make an educated guess of what these two methods mean. If you are thinking along the lines of creating an authentication object, **bingo**, you are on the right track!

10. Now that we have the authentication system set up, we will now need to set up a route in our app that matches to the callback URL, i.e. a URL that redirects the user from Facebook to your app. In your `routes.rb` file, add the following:

```
  get "/auth/:provider/callback" => "sessions#create_from_omniauth"
```

Note: remember that we have already defined our `:provider` as Facebook in the initializer files (i.e. omniauth.rb).

11. We will also need to have a sessions controller for the route defined above. Create a file called `sessions_controller.rb`and add the following:

```
class SessionsController < Clearance::SessionsController
  def create_from_omniauth
```

```
    auth_hash = request.env["omniauth.auth"]
    authentication = Authentication.find_by_provider_and_uid(auth_hash["provider"], auth_hash["uid"]) ||  Authentication.create_with_omniauth(auth_hash)

    # if: previously already logged in with OAuth
    if authentication.user
      user = authentication.user
      authentication.update_token(auth_hash)
      @next = root_url
      @notice = "Signed in!"
    # else: user logs in with OAuth for the first time
    else
      user = User.create_with_auth_and_hash(authentication, auth_hash)
      # you are expected to have a path that leads to a page for editing user details
      @next = edit_user_path(user)
      @notice = "User created. Please confirm or edit details"
    end

    sign_in(user)
    redirect_to @next, :notice => @notice
  end
end
```

Note that `request.env["omniauth.auth"]` returns data about the user in an authentication hash mentioned earlier. The documentation also provides an example of what the authentication hash contains. You can also see what the authentication hash contains by putting a byebug at the right place.

After the user information is obtained, the code above then checks if a user already exists for this particular authentication given the provider and uid, if yes, he/she will be signed in, else a new user object will be created. Now try to relate the methods given with the methods defined in the user model and authentication model. Which methods were defined earlier and which methods were not? If a method was defined earlier, where do you think it was defined? (Hint: we are using a gem.)

12. Finally, in your desired view file, for example, `welcome/index.html.erb`, you can provide a link for the user to sign in with Facebook, i.e.

```
<%= link_to "Sign in with Facebook", "/auth/facebook" %>
```

You can also play around with the several display formats.

Now that you have implemented the code, try to **match the methods and routes given with each step in the diagram above for the OAuth2 protocol**. For example, the path `/auth/facebook` corresponds to Step 1 in the diagram, the path `/auth/:provider/callback` corresponds to Step 3 in the diagram, and so on so forth. This will help reinforce your understanding of how the OAuth2 protocol works.

If you are more adventurous and want to truly understand things before taking the easier way out, try implementing Facebook authentication yourself without a pre-built library like `omniauth-facebook`.