Ruby (https://cloud.google.com/ruby/) Guides

Authenticating users with Ruby

This part of the Bookshelf app tutorial shows how to create a sign-in workflow for users and how to use profile information to provide users with personalized functionality.

By using Google Identity Platform (https://developers.google.com/identity/), you can easily access information about your users while ensuring their sign-in credentials are safely managed by Google. Use OAuth 2.0 (https://developers.google.com/identity/protocols/OpenIDConnect) to provide a sign-in workflow for all users of your app. It provides your app with access to basic profile information about authenticated users.

This page is part of a multipage tutorial. To start from the beginning and read the setup instructions, go to Ruby Bookshelf app (https://cloud.google.com/ruby/getting-started/tutorial-app).

Creating a web app client ID

A web app client ID lets your app authorize users and access Google APIs.

1. In the Google Cloud Console, go to the **Credentials** page.

   GO TO THE CREDENTIALS PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/CREDENTIALS)
2. Click **OAuth consent screen**.
3. In the **Application name** field, enter `Ruby Bookshelf App`.
4. In the **Authorized domains** field, enter your App Engine app name as `[YOUR_PROJECT_ID].appspot.com`. Replace `[YOUR_PROJECT_ID]` with the name of your Google Cloud project ID.
5. Fill in any other relevant, optional fields, and then click **Save**.

6. Click **Create credentials** > **OAuth client ID**.
7. Under **Application type**, select the **Web application** option.
8. In the **Name** field, enter `Ruby Bookshelf Client`.
9. In the **Authorized redirect URIs** field, enter the following URLs, one at a time:

```
http://localhost:3000/auth/google_oauth2/callback
```

```
http://[YOUR_PROJECT_ID].appspot.com/auth/google_oauth2/callback
```

```
https://[YOUR_PROJECT_ID].appspot.com/auth/google_oauth2/callback
```

10. Click **Create**.
11. Copy the client ID and client secret for later use.

Installing dependencies

Go to the `getting-started-ruby/4-auth` directory, and enter this command:

```
bundle install
```

Configuring settings

1. Copy the example settings file.

```
cp config/settings.example.yml config/settings.yml
```

2. Edit the `settings.yml` file. Replace `[YOUR_CLIENT_ID]` with your web app client ID and `[YOUR_CLIENT_SECRET]` with your web app secret.

```
default: &default
  oauth2:
    client_id: [YOUR_CLIENT_ID]
    client_secret: [YOUR_CLIENT_SECRET]
```

3. In the `settings.yml` file, replace the values for `project_id` and `gcs_bucket` with the values you used in the Using Cloud Storage (https://cloud.google.com/ruby/getting-started/using-cloud-storage) section.

   For example, suppose your web app client ID is `XYZCLIENTID`, your client secret is `XYZCLIENTSECRET`, your project name is `my-project`, and your Cloud Storage bucket name is `my-bucket`. Then the default section of your `settings.yml` file would look like this:

```
default: &default
  project_id: my-project
  gcs_bucket: my-bucket
  oauth2:
    client_id: XYZCLIENTID
    client_secret: XYZCLIENTSECRET
```

4. Copy the `database.example.yml` file.

```
cp config/database.example.yml config/database.yml
```

5. Configure the sample app to use the same database that you set up during the Using structured data (https://cloud.google.com/ruby/getting-started/using-structured-data) portion of this tutorial:

   CLOUD SQL | POSTGRESQL | DATASTORE

   - Edit `database.yml`. Uncomment the lines in the Cloud SQL portion of the file.

```
mysql_settings: &mysql_settings
  adapter: mysql2
  encoding: utf8
  pool: 5
  timeout: 5000
  username: [MYSQL_USER]
  password: [MYSQL_PASS]
  database: [MYSQL_DATABASE]
  socket: /cloudsql/[YOUR_INSTANCE_CONNECTION_NAME]
```

     - Replace `[MYSQL_USER]` and `[MYSQL_PASS]` with your Cloud SQL instance username and password that you created previously.
     - Replace `[MYSQL_DATABASE]` with the name of the database that you created previously.

- Replace `[YOUR_INSTANCE_CONNECTION_NAME]` with the `Instance Connection Name` of your Cloud SQL instance.

Note: You can retrieve the Cloud SQL instance connection name by running `gcloud beta sql instances describe [YOUR_INSTANCE_NAME]`.

- Run migrations.

```
bundle exec rake db:migrate
```

Running the app on your local machine

1. Start a local web server.

```
bundle exec rails server
```

2. In your web browser, enter the following address:

   http://localhost:3000 (http://localhost:3000)

Now you can browse the app's web pages, sign in using your Google account, add books, and see the books you've added using the **My Books** link in the top navigation bar.

To exit the local web server, press `Control+C`.

Deploying the app to the App Engine flexible environment

1. Compile the JavaScript assets for production.

```
RAILS_ENV=production bundle exec rake assets:precompile
```

2. Deploy the sample app.

```
gcloud app deploy
```

3. In your web browser, enter the following address.

```
https://[YOUR_PROJECT_ID].appspot.com
```

If you update your app, you can deploy the updated version by entering the same command you used to deploy the app the first time. The new deployment creates a new version (https://console.cloud.google.com/appengine/versions) of your app and promotes it to the default version. The older versions of your app remain, as do their associated VM instances. Be aware that all of these app versions and VM instances are billable resources.

You can reduce costs by deleting the non-default versions of your app.

To delete an app version:

1. In the Cloud Console, go to the **Versions** page for App Engine.

   GO TO THE VERSIONS PAGE (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APPENGINE/VERSIONS)

2. Select the checkbox for the non-default app version you want to delete.

   Note: The only way you can delete the default version of your App Engine app is by deleting your project. However, you can stop the default version in the Cloud Console (https://console.cloud.google.com/appengine/versions). This action shuts down all instances associated with the version. You can restart these instances later if needed.

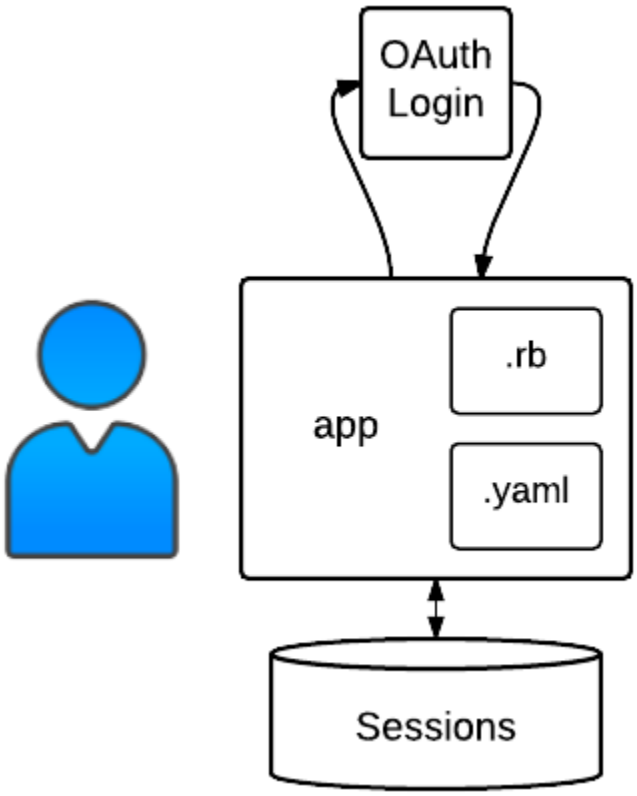
   In the App Engine standard environment, you can stop the default version only if your app has manual or basic scaling.

3. Click **Delete** to delete the app version.

For complete information about cleaning up billable resources, see the Cleaning up (https://cloud.google.com/ruby/getting-started/using-pub-sub#clean-up) section in the final step of this tutorial.

App structure

The following diagram shows the app's components and how they connect to each other.



Understanding the code

This section walks you through the sample code and explains how it works.

User sign-in

The form for the app's home page has a new link so users can sign in.

4-auth/app/views/layouts/application.html.erb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/views/layouts/application.html.erb)

TFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/VIEWS/LAYOUTS/APPLICATION.HTML.ERB)

```
<%= link_to "Login", login_path %>
```

The `/login` route is configured in the `config/routes.rb` file.

4-auth/config/routes.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/config/routes.rb)

```
3.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/CONFIG/ROUTES.RB)
```

```
get "/login", to: redirect("/auth/google_oauth2")
```

When you click **Log in**, you are redirected to the Google OAuth 2.0 user consent screen. The OmniAuth (https://github.com/intridea/omniauth) and Google OAuth2 (https://github.com/zquestz/omniauth-google-oauth2) Ruby gems provide support for the sign-in workflow.

4-auth/Gemfile (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/Gemfile)

```
PS://GITHUB.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/GEMFILE)
```

```
gem "omniauth"
gem "omniauth-google-oauth2"
```

The `omniauth.rb` initializer file configures OmniAuth to use Google OAuth 2.0 for user sign-in. The code in the `omniauth.rb` file reads configuration settings from the `oauth2` section of `config/settings.yml`.

4-auth/config/initializers/omniauth.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/config/initializers/omniauth.rb)

```
LOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/CONFIG/INITIALIZERS/OMNIAUTH.RB)
```

```
Rails.application.config.middleware.use OmniAuth::Builder do
  config = Rails.application.config.x.settings["oauth2"]

  provider :google_oauth2, config["client_id"],
                           config["client_secret"],
                           image_size: 150
end
```

When you click **Allow** on the Google OAuth 2.0 user consent screen, the authorization service issues an `HTTP GET` request for the sample app's `/auth/google_oauth2/callback` route, which is configured in the `routes.rb` file.

4-auth/config/routes.rb
(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/config/routes.rb)

3.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/CONFIG/ROUTES.RB)

```
get "/auth/google_oauth2/callback", to: "sessions#create"

resource :session, only: [:create, :destroy]
```

The handler for the callback route is the `create` method of the sample app's `SessionController` class. The `omniauth.auth` request variable provides your profile information, which includes your identifier, display name, and profile image. The `create` method reads the profile information and stores it in a `User` object that is serialized into the session.

4-auth/app/controllers/sessions_controller.rb
(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/controllers/sessions_controller.rb)

ORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/CONTROLLERS/SESSIONS_CONTROLLER.RB)

```
class SessionsController < ApplicationController

  # Handle Google OAuth 2.0 login callback.
  #
  # GET /auth/google_oauth2/callback
  def create
    user_info = request.env["omniauth.auth"]

    user           = User.new
    user.id        = user_info["uid"]
    user.name      = user_info["info"]["name"]
    user.image_url = user_info["info"]["image"]

    session[:user] = Marshal.dump user

    redirect_to root_path
  end
```

The app's home page has new elements to display information about the signed-in user.

4-auth/app/views/layouts/application.html.erb
(https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/views/layouts/application.html.erb)

TFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/VIEWS/LAYOUTS/APPLICATION.HTML.ERB)

```
<% if logged_in? %>
  <% if current_user.image_url %>
    <%= image_tag current_user.image_url, class: "img-circle", width: 24 %>
  <% end %>
  <span>
    <%= current_user.name %>  
    <%= link_to "(logout)", logout_path %>
  </span>
```

The `logged_in?` and `current_user` helper methods read the `User` object in the user's session.

4-auth/app/controllers/application_controller.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/controllers/application_controller.rb)

M/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/CONTROLLERS/APPLICATION_CONTROLLER.RB)

```
class ApplicationController < ActionController::Base
  helper_method :logged_in?, :current_user

  def logged_in?
    session.has_key? :user
  end

  def current_user
    Marshal.load session[:user] if logged_in?
  end
```

List a user's books

When a signed-in user adds a new book, their user ID is associated with the book:

4-auth/app/controllers/books_controller.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/controllers/books_controller.rb)

ATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/CONTROLLERS/BOOKS_CONTROLLER.RB)

```
def create
  @book = Book.new book_params
```

```
  @book.creator_id = current_user.id if logged_in?

  if @book.save
    flash[:success] = "Added Book"
    redirect_to book_path(@book)
  else
    render :new
  end
end
```

The app's home page has a new `Mine` link, so signed-in users can view only the books they have added.

4-auth/app/views/layouts/application.html.erb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/views/layouts/application.html.erb)

TFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/VIEWS/LAYOUTS/APPLICATION.HTML.ERB)

```
<% if logged_in? %>
  <li><%= link_to "Mine", user_books_path %></li>
<% end %>
```

The `index` action of the `UserBooksController` queries for books that were added by the signed-in user.

CLOUD SQL / POSTGR... DATASTORE

4-auth/structured_data/active_record/app/controllers/user_books_controller.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/structured_data/active_record/app/controllers/user_books_controller.rb)

'S/4-AUTH/STRUCTURED_DATA/ACTIVE_RECORD/APP/CONTROLLERS/USER_BOOKS_CONTROLLER.RB)

```
@books = Book.where(creator_id: current_user.id).
             limit(PER_PAGE).offset(PER_PAGE * page)
```

User sign-out

The form for the app's home page has a new `logout` link so that users can sign out.

4-auth/app/views/layouts/application.html.erb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/views/layouts/application.html.erb)

TFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/VIEWS/LAYOUTS/APPLICATION.HTML.ERB)

```
<%= link_to "(logout)", logout_path %>
```

The `logout_path` is configured in the `config/routes.rb` file.

4-auth/config/routes.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/config/routes.rb)

3.COM/GOOGLECLOUDPLATFORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/CONFIG/ROUTES.RB)

```
get "/logout", to: "sessions#destroy"
```

The `destroy` action of the `SessionsController` deletes the `user` from the session.

4-auth/app/controllers/sessions_controller.rb (https://github.com/GoogleCloudPlatform/getting-started-ruby/blob/steps/4-auth/app/controllers/sessions_controller.rb)

ORM/GETTING-STARTED-RUBY/BLOB/STEPS/4-AUTH/APP/CONTROLLERS/SESSIONS_CONTROLLER.RB)

```
def destroy
  session.delete :user

  redirect_to root_path
end
```

< PREV (HTTPS://CLOUD.GOOGLE.COM/RUBY/GETTING-STARTED/USING-CLOUD-STORAGE)

NEXT > (HTTPS://CLOUD.GOOGLE.COM/RUBY/GETTING-STARTED/LOGGING-APPLICATION-EVENTS)

Last updated December 4, 2019.