專案實作

在開始專案之前

User Story

- 使用者可以註冊、登入、登出系統
- 使用者可以建立新的團體，而且成為團體的管理員。
- 管理者可以刪除團體。
- 團體管理者可以加入使用者到團體中。
- 使用者可以CRUD標籤
- 使用者可以建立帳單

Issus Tracking

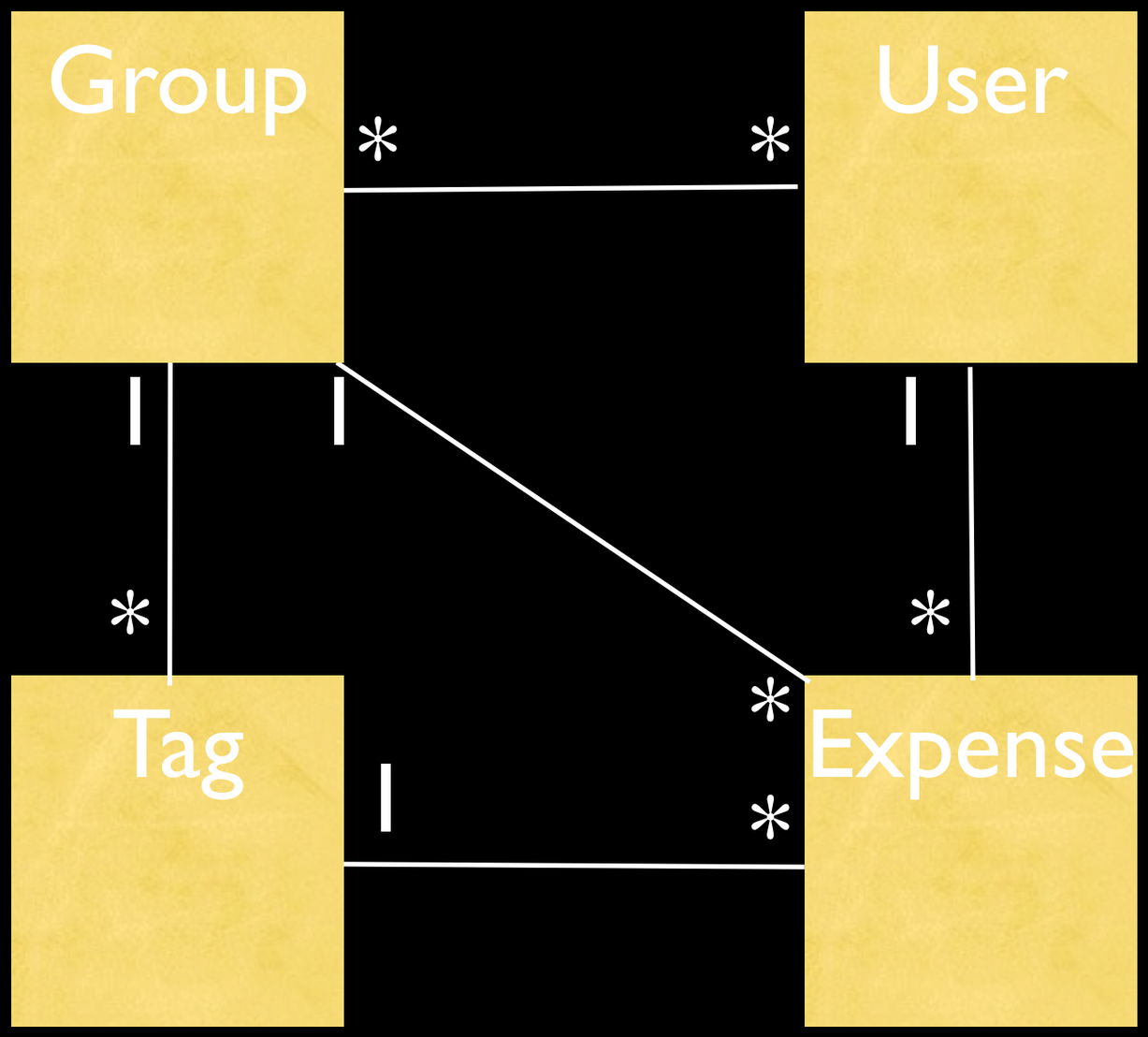
- 利用Issus Tracking來追蹤進度
- http://agilezen.com/project/40126/board

Version Control

- Github
- https://github.com/

Prototype

DB relationship



建立專案

新增專案

- 請在cmd中適當的目錄中執行:
  - rails new gurupu
  - cd gurupu

建立測試環境

- 在Gemfile檔案中加入

```
group :development, :test do
  gem 'rspec-rails'
end
```

- 在cmd中執行：
  - bundle install
  - rails generate rspec:install

建立首頁 (I)

- 在cmd中執行：
  - rails generate controller Welcome index
- 在config/route.rb中加入

```
root :to => 'welcome#index'
```

建立首頁 (II)

- 刪除 public/index.html
- 增加你要的文案到views/welcome/index.html.erb
- 在cmd中執行：
  - rails server
- 打開瀏覽器，輸入網址http://localhost:3000

Devise

一個讓使用者註冊、登入、登出的系統

Module

- Database Authenticatable
- Confirmable
- Registerable
- Rememberable
- Trackable
- Timeoutable
- Validatable
- Lockable

Extensions

- https://github.com/plataformatec/devise/wiki/Extensions

把Devise 加入Rails

- 請把下一行，加入到Gemfile檔案中
  - gem “devise”
- 在cmd中執行：
  - bundle install
  - rails generate devise:install

把Devise 加入Rails

- 在config/environments/development.rb中加入：

```
config.action_mailer.default_url_options =
{ :host => 'localhost:3000' }
```

Create User Model

- 請在cmd中執行：
  - rails generate devise User
- 會建立三個檔案：
  - db/migrate/xxxx_devise_create_users.rb
  - app/models/user.rb
  - spec/models/user_spec.rb

User model

- 請開啟app/models/user.rb
- devise方法的參數，就是你要使用哪些module。

attr_accessible

- 正向列表哪些可以被mass-assign
- User.new(params[:user])

DB Migration

- 開啟db/migrate/xxxx_devise_create_user.rb
- 請在cmd中執行：
  - rake db:migrate
- 插撥migration簡介

Route (I)

- 在config/route.rb中已經新增了：

  ```
  devise_for :users
  ```

- 請在cmd中執行：
  - rake routes

Route (II)

```
       new_user_session GET    /users/sign_in(.:format)       devise/sessions#new
           user_session POST   /users/sign_in(.:format)       devise/sessions#create
   destroy_user_session DELETE /users/sign_out(.:format)      devise/sessions#destroy
          user_password POST   /users/password(.:format)      devise/passwords#create
      new_user_password GET    /users/password/new(.:format)  devise/passwords#new
     edit_user_password GET    /users/password/edit(.:format) devise/passwords#edit
                        PUT    /users/password(.:format)      devise/passwords#update
cancel_user_registration GET    /users/cancel(.:format)        devise/registrations#cancel
       user_registration POST   /users(.:format)               devise/registrations#create
   new_user_registration GET    /users/sign_up(.:format)       devise/registrations#new
  edit_user_registration GET    /users/edit(.:format)          devise/registrations#edit
                        PUT    /users(.:format)               devise/registrations#update
                        DELETE /users(.:format)               devise/registrations#destroy
```

Filter & Helper

- before_filter :authenticate_user!
- user_signed_in?
- current_user
- user_session

Simple Layout

建立layout (I)

- 一開始的Layout可以先不要考慮CSS。
- 依據Prototype，我們可以把每一個頁面都分成三個部分：
  - Header
  - Content
  - Footer

建立layout (II)

- 開啟views/layout/application.html.erb

```
<body>
    <div id='header'>
        <h1><%= link_to "GURUPU", root_path %></h1>
    </div>
    <div id="content">
        <%= yield %>
    </div>
    <div id="footer">
        Copyright © 2012
        <%=link_to "Handlino", "http://handlino.com" %>
    </div>
</body>
```

建立menu

- 一般的慣例，有兩個menu
  - Main Menu：針對系統功能
  - User Menu：針對使用者的操作
- 習慣用pratial template來放。

Main Menu

- 建立檔案：views/menus/_main.html.erb

```
<div class="main-nav">
    <ul>
        <% if user_signed_in? %>
            <li><%=link_to "Home", root_path %></li>
        <% else -%>
        <% end -%>
    </ul>
</div>
```

User Menu

- 建立檔案：views/menus/_user.html.erb

```
<div class="user-nav">
    <ul>
        <% if user_signed_in? %>
            <li><%=link_to "Sign out",
                           destroy_user_session_path,
                           :method => :delete %>
                    </li>
        <% else -%>
            <li><%=link_to "Sign up",
                           new_user_registration_path %>
                    </li>
            <li><%=link_to "Sign in",
                          new_user_session_path %> </li>
        <% end -%>
    </ul>
</div>
```

Add menu to layout

- 打開app/views/layout/application.html.erb
- 加入：

```
<div id='header'>
   <h1><%= link_to "GURUPU", root_path %></h1>
    <%= render "menus/main" %>
    <%= render "menus/user" %>
</div>
```

Add Group

需求

- 使用者可以建立Group
- 建立Group的使用者為管理員
- 管理員可以編輯Group資訊
- 管理員可以刪除Group

Add Group model

- 請在cmd中執行
  - rails generate model Group name:string\ secret_str:string owner_id:integer
  - rake db:migrate

Group Owner (I)

- Group的建立者，就是管理員，所以在這裡管理員與Group是一種one-to-many的關係。

Group Owner (II)

- 請在app/models/group.rb中加入

```
class Group < ActiveRecord::Base
  #...
  belongs_to :owner,
             :class_name => "User",
             :foreign_key => "owner_id"
end
```

Group Owner (II)

- 請在app/models/users.rb中加入

```
class User < ActiveRecord::Base
  #...
  has_many :owned_groups,
           :class_name => "Group",
            :foreign_key => "owner_id"
end
```

在console中試看看

- 請在cmd中執行：
  - rails console

請在console中執行下列指令：

- u = User.first
- u.owned_groups.create(:name => “test”)
- p u.owned_groups

插撥：Model Association

設定secret_str (I)

- 打開app/models/group.rb
- 加入：

```
class Group < ActiveRecord::Base
  #...
  before_create :set_secret_str
  protected
  def set_secret_str
    self.secret_str = SecureRandom.hex(10)
  end
end
```

插撥：Model Callback

設定secret_str (II)

- 利用Rails console來驗證
  - u = User.first
  - g = u.owned_groups.new(:name => “IT”)
  - g.save
  - p g

Validation

- name不能是空值。
- name, secret_str不可以重複。

```
class Group < ActiveRecord::Base
  #...
  validates :name,
            :presence => true,
            :uniqueness => true
  validates :secret_str, :uniqueness => true
  #...
end
```

插撥：Model Validation

小練習

- group name的長度需介於5-20之間。
- 利用Rails console故意讓group的建立不符合validation，看看會發生什麼事。