Action Dispatch & Action Controller

RESTful

REST

- Representational State Transfer
- REST是一種軟體架構。
- 核心精神：
  - 使用Resource來當做識別。
  - 同一個Resource則可以有不同的Representations格式變化。

RESTful

- 使用了REST概念來建立一整組的命名路由(named routes)。
- 用一種比較標準化的方式來命名跟組織Controllers和Actions。

預設路由 (I)

- 當我們在config/route.rb中加入
  resources :groups 遇設會產生下列對應：

```
    groups GET    /groups(.:format)          groups#index
           POST   /groups(.:format)          groups#create
 new_group GET    /groups/new(.:format)      groups#new
edit_group GET    /groups/:id/edit(.:format) groups#edit
     group GET    /groups/:id(.:format)      groups#show
           PATCH  /groups/:id(.:format)      groups#update
           PUT    /groups/:id(.:format)      groups#update
           DELETE /groups/:id(.:format)      groups#destroy
```

預設路由 (II)

- 由四個網址搭配上五個HTTP動詞，對應到7個Action。
  - /groups + GET => groups#index
  - /groups + POST => groups#create
  - /groups/new + GET => groups#new
  - /groups/:id/edit + GET => groups#edit

預設路由 (II)

- /groups/:id + GET => groups#show
- /groups/:id + PUT => groups#update
- /groups/:id + PATCH => groups#update
- /groups/:id + DELETE => groups#destroy

Helper

Helper	URL
groups_path	/groups
new_group_path	/groups/new
edit_group_path(@group)	/groups/:id/edit
group_path (@group)	/groups/:id

需注意兩件事：

1. 單複數
2. 是否需要參數

Helper的使用

```
<%= link_to "列表", groups_path %>
<%= link_to "新增", new_group_path %>
<%= link_to "修改", edit_group_path(@group) %>
<%= link_to "删除",
            group_path(@group),
            :method=>:delete %>
```

respond_to

- respond_to可以讓我們在同一個Action中，支援不同的資料格式。

```
def destroy
  @group.destroy
  respond_to do |format|
    format.html { redirect_to groups_url }
    format.json { head :no_content }
  end
end
```

Route

ActionDispatch

Routing做了什麼事？

- 將網址對應到適當的Controller#Action。
- 處理網址內的參數字串
  - /groups/1 => params[:id] = 1
- 辨識link_to和redirect_to的參數產生URL字串
  - groups_path => /groups
  - {:controller => ‘users’, :action => “approve”} =>

    /users/approve

一般路徑

- get "groups/list"
- get "groups/list" => "groups#index"
- get "groups/:id/users" => "groups#users"

Named Routes

- get "/list" => "groups#index", :as => "list"
- 會有helper methods:
  - list_path => 相對路徑
  - list_url => 絕對路徑

設定首頁

- root :to => 'welcome#index'

限制動詞

```
match "groups/list" => "groups#index", :via => :get
get   "groups/list" => "groups#index"
```

```
match "groups/:id/approve" => "groups#approve",
      :via => [:post, :put]
post  "groups/:id/approve" => "groups#approve"
put   "groups/:id/approve" => "groups#approve"
```

限制條件

```
constraints(:id => /(^127.0.0.1$)|(^192.168.[0-9]
{1,3}.[0-9]{1,3}$)/) do
    match "groups/:id/approve" => "groups#approve"
end
```

RESTful Route

- 複數Resource
  - resources :groups
- 單數Resource
  - resource :email_config

```
     email_config POST   /email_config(.:format)       email_configs#create
 new_email_config GET    /email_config/new(.:format)   email_configs#new
edit_email_config GET    /email_config/edit(.:format)  email_configs#edit
                  GET    /email_config(.:format)       email_configs#show
                  PATCH  /email_config(.:format)       email_configs#update
                  PUT    /email_config(.:format)       email_configs#update
                  DELETE /email_config(.:format)       email_configs#destroy
```

Nested Resources (I)

```ruby
resources :groups do
  resources :users
end
```

```
    group_users GET    /groups/:group_id/users(.:format)          users#index
                POST   /groups/:group_id/users(.:format)          users#create
 new_group_user GET    /groups/:group_id/users/new(.:format)      users#new
edit_group_user GET    /groups/:group_id/users/:id/edit(.:format) users#edit
     group_user GET    /groups/:group_id/users/:id(.:format)      users#show
                PATCH  /groups/:group_id/users/:id(.:format)      users#update
                PUT    /groups/:group_id/users/:id(.:format)      users#update
                DELETE /groups/:group_id/users/:id(.:format)      users#destroy
```

Nested Resources (II)

- Helper的參數
  - group_users_path(@group)
  - new_group_user_path(@group)
  - edit_group_user_path(@group, @user)
  - group_user_path(@group, @user)
- 建議不要超過兩層。

自定Action (I)

- 針對多筆Resources

```
resources :groups do
  collection do
    get :alive
    get :closed
  end
end
```

```
 alive_groups GET    /groups/alive(.:format)      groups#alive
closed_groups GET    /groups/closed(.:format)     groups#closed
```

自定Action (II)

- 針對單筆Resource

```
resources :groups do
  get :close, :on => :member
end
```

```
close_group GET    /groups/:id/close(.:format)    groups#close
```

NameSpace

```
namespace :admin do
  resources :groups
end
```

```
    admin_groups GET    /admin/groups(.:format)          admin/groups#index
                 POST   /admin/groups(.:format)          admin/groups#create
 new_admin_group GET    /admin/groups/new(.:format)      admin/groups#new
edit_admin_group GET    /admin/groups/:id/edit(.:format) admin/groups#edit
     admin_group GET    /admin/groups/:id(.:format)      admin/groups#show
                 PATCH  /admin/groups/:id(.:format)      admin/groups#update
                 PUT    /admin/groups/:id(.:format)      admin/groups#update
                 DELETE /admin/groups/:id(.:format)      admin/groups#destroy
```

Routing Concerns

- 把重複的route設定抽取出來。

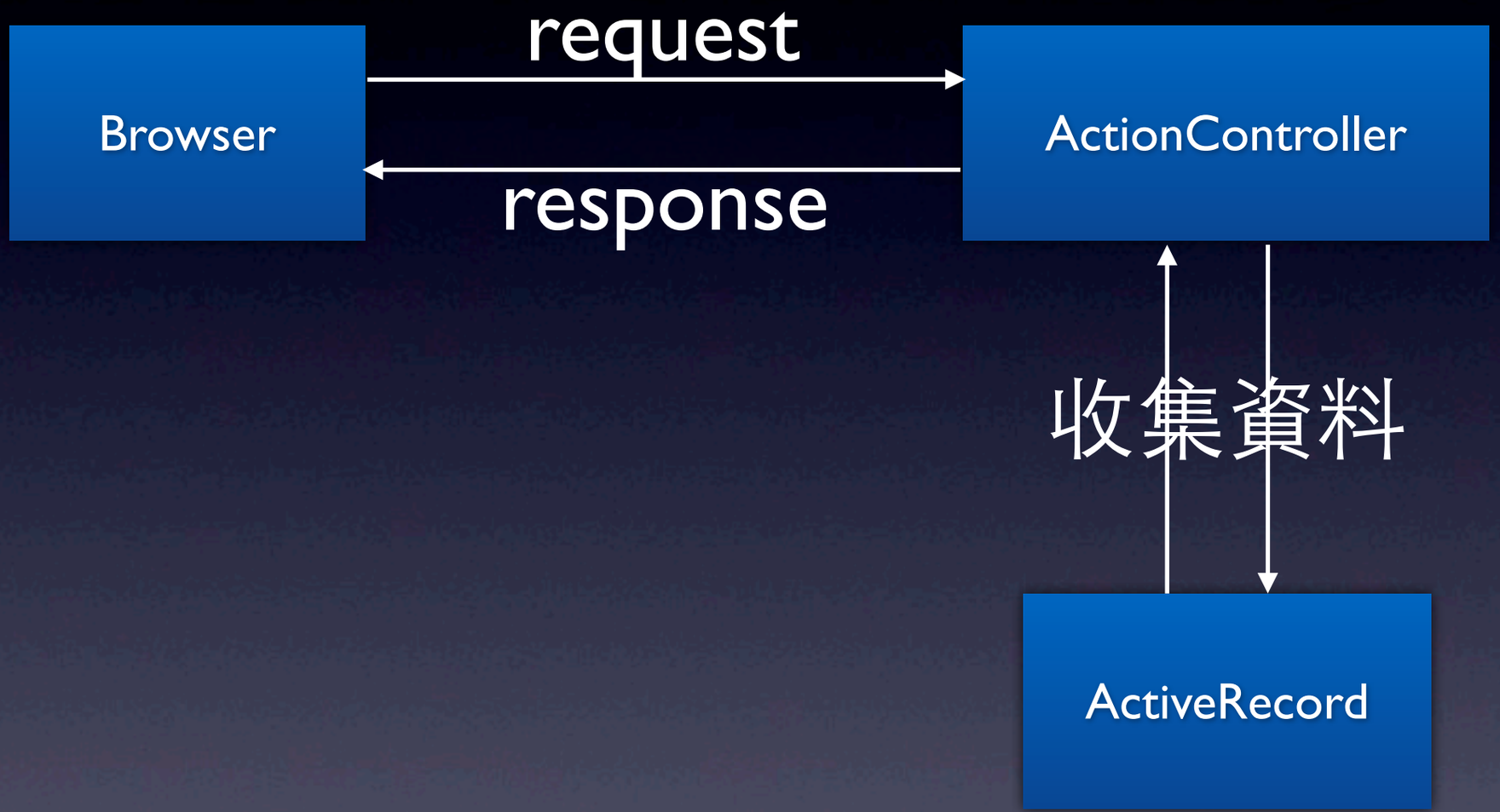
```
Gurupu::Application.routes.draw do
  concern :commentable do
    resources :comments
  end
  resources :groups,
            concerns: [:commentable]
  resources :expenses,
            concerns: [:commentable]
end
```

提醒

- config/route.rb中的設定，如果有多組相同的，則會越前面的越優先。

ActionController

HTTP Request流程



ApplicationController

- app/controllers/application_controller.rb
- 預設所有的Controller都會繼承ApplicationController。
- 你可以在這邊定義一些其他controller都會共用的methods。
- 共用的methods要放在protected或private內，否則就變成actions了。

protect_from_forgery

- 所有不是GET的Request，都需要帶一個Token才能進到action中執行。
- 在Layout中有<%= csrf_meta_tags %>
- 取消：

```
skip_before_filter :verify_authenticity_token
```

Controller & Action

- app/controllers/groups_controller.rb
- 在Action中需要處理：
  - 收集request資訊
  - 存取model
  - 回傳response

收集Request資訊

- action_name
- cookies
- headers
- params
- session
- request

params (I)

- 例如：http://localhost:3000?a=b&c=d
- 在params中會有：
  - params[:a] #=> b
  - params[:c] #=> d
- 還有從route所產生的如：
  - params[:id]
  - params[:action]
  - params[:controller]

params (II)

- params是ActionController::Parameters物件。
- params[:action] == params[“action”]

Render (I)

- 如果你沒有action，Rails還是會執行 render。
- 慣例是使用：
  - app/views/{controller_name}/{action_name}

Render (II)

- 直接回傳
  - render text: "Hi, Rails"
  - render xml: @groups.to_xml
  - render json: @groups.to_json
  - render nothing: true

Render (III)

- 使用其他template
  - `render template: "groups/index"`
  - `render action: "index"`

Render (IV)

- 其他參數
  - render status: 500
  - render layout: false
  - render layout: "admin"

redirect_to

- 在原本的action中跳轉到別的action去。
  - 例如groups#create後跳轉到groups#index
    - redirect_to groups_path

send_data

- 回傳的資料讓使用者可以下載下來，成為一個檔案。
- send_data data, [options]

```
def send_text
  send_data "大家好", :filename => "hi.txt"
end
```

send_file

- 回傳檔案
- send_file file_location, [options]

```
def send_pdf
  send_file "public/pdf/vim.pdf",
            filename: "vim.pdf"
end
```

session (I)

- 讓app可以在多次的requests，仍能記得某些資料。例如購物車。
- session 是 Hash。
  - session[:group] = @group

session (II)

- Rails預設把session的資料加密後存在瀏覽器的Cookie。但是只有4k大小。
- 也支援用ActiveRecord或Memcached存放。

flash (I)

- 用於在redirect時，可以將訊息帶到下一個action。

```
def create
  #...
  if group.save
    flash[:notice] = "Created successful"
    else
    flash[:error] = "Created fail"
  end

  redirect_to groups_path
end
```

flash (II)

- flash是一個Hash。
- 慣用的key：(for Twitter-Bootstrap)
  - success
  - error
  - info

flash (III)

- 在template中呈現出來需要：

```
<% flash.each do |key, value| %>
    <div class="alert alert-<%=key%>">
        <%= value %>
    </div>
<% end %>
```

Cookie

- 設定 session cookie.
  - cookies[:user_name] = "weijen"
- 設定 cookie 過期時間
  - cookies[:login] = { value: "XJ-122", expires: 1.hour.from_now }
- 刪除 cookie 內容
  - cookies.delete :user_name

Filters

- before_action
- after_action
- prepend_before_action
- prepend_after_action
- 都可以有:only, :except參數。

Skip Filters

- skip_before_action
- skip_after_action