

# Rails 簡介

# 什麼是Rails (I)

- 開放原始碼 (採用MIT授權) 的 Web 框架，主要用於開發database-backed 的網站應用程式
- MVC (Model-View-Control )模式

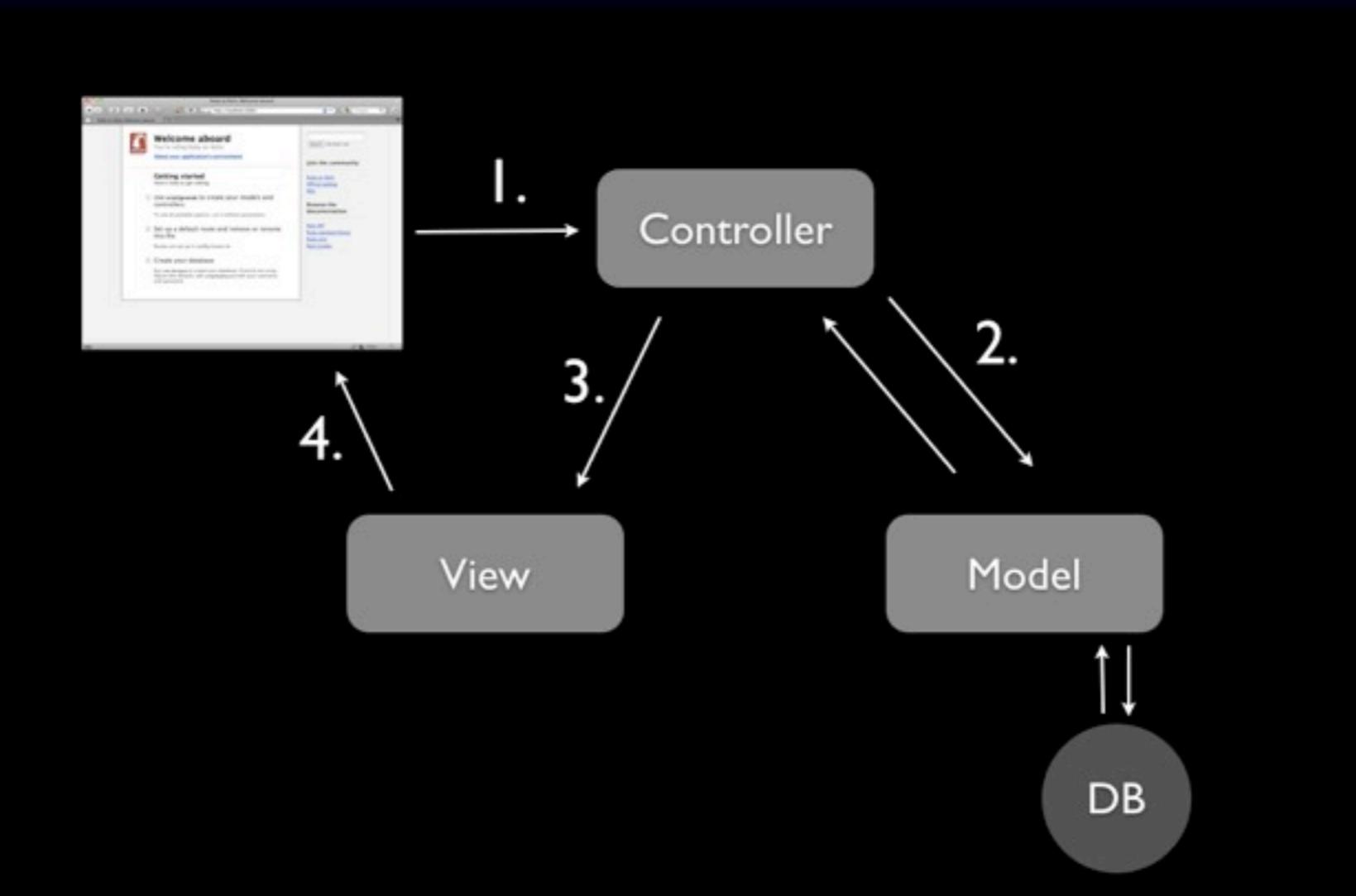
# 什麼是Rails(II)

- (幾乎可以)完全地 Ruby 開發環境，包括支援 Ajax、定義資料庫結構、ORM (object-relational-mapping) 機制操作資料庫等等。
- 2004 年由 David Heinemeier Hanson(DHH) 從 37signals 產品中獨立出來。

# 淺談MVC (I)

- 一種軟體架構的設計模式，將軟體分成三個部分：Model、Controller、View
  - Model 物件包裝了資料與商業邏輯
  - View 表示使用者介面，顯示及編輯表單
  - Controller 負責將資料送進送出

# 淺談MVC (II)



# 淺談MVC (III)

- 分離商業邏輯和使用者介面，讓前端與後端開發者可以獨立作業
- 容易保持程式的不重複性
- (DRY: Don't repeat yourself)
- 讓程式碼有著一致性的結構，位置清楚、容易維護。

# ORM

- Object-relational mapping
- 使用物件導向語法來操作關聯式資料庫

```
select * from orders, users  
where orders.user_id = users.id and  
orders.status = "Paid"  
limit 5 order by order.id;
```



```
Order.where(:status => "paid").includes(:user).limit(5).order("id")
```

# ORM的好處

- ORM容易使用、寫程式比較有效率。
- 將商務邏輯與資料庫操作邏輯分開。
- 容易閱讀，增加程式碼的維護性。
- 我不用學SQL了。

# 你的第一個Rails網站

# 安裝Rails 4

- `gem install rails --version 4.0.0.rc1`

# 建立Rails專案

- 在console中執行：

```
rails new together  
cd together  
rails server
```

- 在Browser中連結<http://localhost:3000>

# 目錄結構(I)

```
▼ app/
  ▼ assets/
    ▶ javascripts/
    ▶ stylesheets/
  ▼ controllers/
    ▶ concerns/
      application_controller.rb
  ▼ helpers/
    application_helper.rb
  ▶ mailers/
  ▼ models/
    ▼ concerns/
  ▼ views/
    ▼ layouts/
      application.html.erb
```

# 目錄結構(II)

```
▶ bin/
▼ config/
  ▶ environments/
  ▶ initializers/
  ▶ locales/
    application.rb
    boot.rb
    database.yml
    environment.rb
    routes.rb
▼ db/
  development.sqlite3
  seeds.rb
▶ lib/
▶ log/
```

# 目錄結構(III)

```
> public/
> test/
> tmp/
> vendor/
config.ru
Gemfile
Gemfile.lock
Rakefile
README.rdoc
```

# Rails 指令

- rails -h
- rails new
- rails console (c)
- rails generate (g)
- rails server (s)

# Gemfile

- Gemfile 是 bundler 的設定檔, 用來紀錄專案中需要使用的 rubygem 與其來源, 檔案中格式為 ruby 語言。
- Gemfile.lock是在gems安裝後的紀錄檔。

# Gemfile 語法(I)

- source
  - 放置在 Gemfile 的最開頭, 紀錄 Rubygems repository 的位置。
  - 範例:

```
source 'https://rubygems.org'
```

# Gemfile 語法(II)

- gem: 紀錄需要使用的 rubygem 的相關資料
- 範例:

```
gem 'sqlite3'  
gem 'rails', '4.0.0.rc1'  
gem 'uglifier', '>= 1.3.0'  
gem 'therubyracer', platforms: :ruby  
gem 'debugger', group: [:development, :test]
```

# Bundle常用指令 (I)

- bundle install
  - 依照Gemfile 與 Gemfile.lock 安裝 rubygems
- bundle update
  - 更新 rubygems 到最新版本

# Bundle常用指令 (I)

- bundle package
  - 將目前有用到的 gems 打包到 vendor/cache 中
- bundle exec
  - 在目前的 bundle 設定環境下執行指令

# Bundle常用指令 (I)

- bundle config
  - 顯示目前的 bundle 設定值
- bundle check
  - 檢查 Gemfile 中指定的 rubygems 是否都安裝完畢

# Zeus

- 預載(preload)你Rails環境，讓執行Rails指令快速很多。
- 官網：<https://github.com/burke/zeus>
- 安裝：gem install zeus
- 不需要放到Gemfile中。

# 建立首頁(I)

- rails generate controller Welcome index

```
mbpr:together weijen$ rails generate controller Welcome index
  create  app/controllers/welcome_controller.rb
  route   get "welcome/index"
  invoke  erb
  create   app/views/welcome
  create   app/views/welcome/index.html.erb
  invoke  test_unit
  create   test/controllers/welcome_controller_test.rb
  invoke  helper
  create   app/helpers/welcome_helper.rb
  invoke  test_unit
  create   test/helpers/welcome_helper_test.rb
  invoke  assets
  invoke  coffee
  create   app/assets/javascripts/welcome.js.coffee
  invoke  scss
  create   app/assets/stylesheets/welcome.css.scss
```

# 建立首頁(II)

- app/controllers/welcome\_controller.rb

```
class WelcomeController < ApplicationController
  def index
  end
end
```

# 建立首頁(III)

- app/views/welcome/index.html.erb

```
<h1>Welcome#index</h1>
<p>Find me in
app/views/welcome/index.html.erb
</p>
```

你可以用Brower連結  
<http://localhost:3000/welcome/index>

# 建立首頁(IV)

- 開啟 config\route.rb 添加上  
`root :to => 'welcome#index'`
- 用瀏覽器開啟 `http://localhost:3000`

# 揪專網 - 活動頁面

# User Story

- 使用者可以建立活動
  - 活動名稱、描述、開始時間
- 使用者可以看到活動列表
- 使用者可以看到單一活動頁面
- 使用者可以編輯活動
- 使用者可以刪除活動

# Scaffold

- rails generate scaffold Event name:string \  
description:text start\_at:datetime
- rake db:migrate
- rails server
- 瀏覽器打開：<http://localhost:3000/friends>

# Scaffold

- rails generate scaffold Event name:string \  
description:text start\_at:datetime
- rake db:migrate
- rails server
- 瀏覽器打開：<http://localhost:3000/friends>

是的，你做完了。

# Database

- 設定檔：config/database.yml
- 預設是使用sqlite3。

# DB改用MySQL (I)

- Gemfile中加入  
gem "mysql2"

# DB改用MySQL (I)

- in config/database.yml

```
development:  
  adapter: mysql2  
  encoding: utf8  
  database: together_development  
  host: localhost  
  username: root  
  password: ""
```

# DB改用MySQL (I)

- 在console中輸入

```
bundle install  
rake db:create  
rake db:migrate
```

# 操作資料庫時常用的 rake指令

- rake db:create
- rake db:drop
- rake db:migrate
- rake db:rollback
- rake db:seed

# Rails environment

- Rails預設有三種不同的環境：
  - development
  - test
  - production
- 你可以在config/environments/目錄下面看到三種不同的環境，會有不同的參數設定。

# DB Migration

- DB Migration (資料庫遷移) 可以讓你用 Ruby 程式來修改資料庫結構。
- 檔案：
  - db/migrate/xxxxxxxxxxxx\_create\_events.rb

# Model

- 操作資料，與商業邏輯的檔案。
- 檔案：
  - app/models/event.rb

# rails console (I)

- rails console是類似irb的環境，但是已經載入了目前專案的內容。包含rails、相關的gem、資料庫連線、你寫的程式。
- 在console中執行：rails console

# rails console (II)

```
Event.create(:name => "KTV",
             :description => "Singing together",
             :start_at => (Time.now + 2.day))
Event.all
Event.where(:name => "KTV")
Event.where(:name => "KTV").order("name")
```

# rails console (II)

```
event = Event.where(:name => "KTV").first  
e.update_attribute(:name, "event-ktv-1")  
e.update(name: "KTV")  
e.destroy
```

# Validate

```
validates :name, presence: true,  
          length: { minimum: 3 }
```

# Routes

- 在config/routes.rb中新增了：

```
resources :events
```

- 這代表了：

events	GET	/events(.:format)	events#index
	POST	/events(.:format)	events#create
new_event	GET	/events/new(.:format)	events#new
edit_event	GET	/events/:id/edit(.:format)	events#edit
event	GET	/events/:id(.:format)	events#show
	PATCH	/events/:id(.:format)	events#update
	PUT	/events/:id(.:format)	events#update
	DELETE	/events/:id(.:format)	events#destroy

# Routes

- 在config/routes.rb中新增了：

```
resources :events
```

- 這代表了：

events	GET	/events(.:format)	events#index
	POST	/events(.:format)	events#create
new_event	GET	/events/new(.:format)	events#new
edit_event	GET	/events/:id/edit(.:format)	events#edit
event	GET	/events/:id(.:format)	events#show
	PATCH	/events/:id(.:format)	events#update
	PUT	/events/:id(.:format)	events#update
	DELETE	/events/:id(.:format)	events#destroy

# RESTful簡介(I)

- RESTful 是一種標準化的 controller, action, url 命名方式。
- 他大幅的簡化了我們對於 route 的設定,
- 減少對於團隊維護程式碼命名的負擔。

# PUT vs PATCH

- PATCH : partial modifications to a resource.
- PUT : overwrite a resource with a complete new body.

# Controller (I)

- 負責收集使用者所需要的資料，然後轉交給view呈現給使用者。
- 檔案:
  - `app\controllers\events_controller.rb`

# Controller (II)

- `before_action`
  - 進入action前需要執行的程式。
  - 例如使用者認證。
- `after_action`
  - action結束後要執行的程式。
  - 例如Log。

# Controller (III)

- respond\_to
- 不同的格式需求，回應不同的格式。
- 不用重複action。

# Controller (IV)

- render:
  - 產生view。
- redirect\_to:
  - 轉到其他的action。

# Controller (V)

- Strong Parameter
- 可被接受的參數，在Controller中個別處理。

`params.require(:event).permit(...)`

1. **require**: 在params中必須存在: event
2. **permit**: 在params[:event]中只允許哪些 key/value pair。
3. 回傳在params[:event]中的key/value pair。

# View

- 資料呈現給使用者的樣子。
- Rails預設是使用ERb(Embedded Ruby)。

# Layout

- Layout可以視為”網頁共用的外框”。
- 檔案：app/views/application.html.erb。
- 其中的**yield**方法，是我們插入每個action所產生的頁面的地方。

# Assets

- 靜態檔案集中管理的地方。
- 例如js, css, images, sound等檔案。
- 目錄在：app/assets/。

# Execute instructions in erb

- <%= ... %>
- 將執行結果轉成html safe字串。
- <% ... %>
- 不會呈現執行結果。

# Partial Template

- 將Templaet中重複的程式碼抽出來。
- Partial Template的命名慣例是底線開頭，但是呼叫時不需加上底線。
- 可以參考：
  - app/views/events/new.html.erb
  - app/views/events/edit.html.erb
  - app/views/events/\_form.html.erb

# form\_for ()

- 協助你產生form的view helper
- 分為兩類：
  - 有對應到Model物件的form\_for
  - 沒有對應到Model物件的form\_tag

# form\_for (II)

- `text_field`
- `text_area`
- `radio_button`
- `check_box`
- `select`
- `select_date`, `select_datetime`
- `submit`

# form\_for (III)

- `form_for` 會判斷傳入的 `object`，並依照 RESTful 的慣例，產生適當的 `url` 與 `method`。
- `object` 未存入 `db`, 會產生
  - `:url => events_path,`
  - `:method => :post`
- `object` 已經存入 `db`, 會產生
  - `:url => event_path(@event)`
  - `:method => :put`

# form\_for (IV)

- options:

- :url
- :method
- :html

```
<%= form_for(@event,
  :url => events_path,
  :method => :post,
  :html => { :class => "new_event"}) do |f| %>
  ...
<% end %>
```

# 有點醜！？

- Twitter Bootstrap: <http://twitter.github.io/bootstrap/>
- 常用的gem:
  - <https://github.com/anjlab/bootstrap-rails>
  - [https://github.com/plataformatec/simple\\_form](https://github.com/plataformatec/simple_form)
  - [https://github.com/nickpad/will\\_paginate-bootstrap](https://github.com/nickpad/will_paginate-bootstrap)

# 揪專網 - 報名人

# User Story

- 可以瀏覽活動的所有報名人
- 使用者可以報名某個活動
  - 姓名、Email、電話、備註
- 使用者可以編輯報名資訊
- 使用者可以刪除報名資訊

# Model

- 建立Attendee model
- rails generate model Attendee  
**event:references** name:string email:string  
phone:string comment:text

# 建立關聯

- 在app/models/event.rb中加入：

```
has_many :attendees
```

- 在app/models/attendee.rb中：

```
belongs_to :event
```

要注意單複數。

# Rails Console

```
e = Event.first
e.attendees.create(
  name: "Wei Jen",
  email: "weijenlu@gmail.com",
  phone: "0912345678")
e.attendees
e.attendees.where(name: "Wei Jen")
e.attendees.first
```

# Migration

- `references` 方法會建立foreign key。  
`t.references :event, index: true`
- 上面的例子會建立欄位`event_id`，以及名為`event_id`的index。

# Routes

- 針對某個event的報名人進行操作：

```
resources :events do
  resources :attendees, except: [:show]
end
```

Prefix	Verb	URI Pattern	Controller#Action
event_attendees	GET	/events/:event_id/attendees(.:format)	attendees#index
	POST	/events/:event_id/attendees(.:format)	attendees#create
new_event_attendee	GET	/events/:event_id/attendees/new(.:format)	attendees#new
edit_event_attendee	GET	/events/:event_id/attendees/:id/edit(.:format)	attendees#edit
event_attendee	PATCH	/events/:event_id/attendees/:id(.:format)	attendees#update
	PUT	/events/:event_id/attendees/:id(.:format)	attendees#update
	DELETE	/events/:event_id/attendees/:id(.:format)	attendees#destroy

# Controller & Views

- console中執行：
  - rails generate controller attendees index new edit

# 取得event

```
before_action :set_event
```

```
private  
def set_event  
  @event = Event.find(params[:event_id])  
end
```

# Index

```
def index  
  @attendees = @event.attendees  
end
```

view: app/views/attendees/index.html.erb

接下來就靠各位了

# 練習

- 把 Attendee#new 、 edit 、 create 、 update 、 destroy 完成。只有 new 跟 edit 有頁面（當然也需要 \_form ）。 create 、 update 、 destroy 成功後都轉回 Attendee#index 。
- I18n

# 練習（進階）

- 使用Twitter-bootstrap套版。
- 使用者可以登入。
- ref: <https://github.com/plataformatec/devise>