2019-5-9 更新

**新進工程師訓練教材: 後端**

[使用說明](https://github.com/5xRuby/5xtraining/blob/master/README.md)

**概要**

**開發需求**

本教材的主要目的，是要開發一套「任務管理系統」。這個系統需要做到的事情有：

* 任務功能
  + 可新增自己的任務
  + 使用者登入後，只能看見自己建立的任務
  + 可設定任務的開始及結束時間
  + 可設定任務的優先順序（高、中、低）
  + 可設定任務目前的狀態（待處理、進行中、已完成）
  + 可依狀態篩選任務
  + 可以任務的標題、內容進行搜尋
  + 可為任務加上分類標籤
  + 任務列表，並可依優先順序、開始時間及結束時間等進行排序

滿足以上需求之後，還會需要如下的管理機制：

* 使用者的管理功能

**瀏覽器支援**

* 預設需要支援 macOS/Chrome 的最新版本

**開發工具**

請以下列程式語言、網站開發框架及資料庫系統的最新穩定版本進行開發：

* Ruby
* Ruby on Rails
* PostgreSQL

server 端請使用：

* Heroku

※ 本教材中對效能、資安沒有特別的要求，但仍需要有一定的品質。網站效能太差的話，會被要求改善。

**最終目標**

完成本教材後，我們會認為你已經具備以下能力：

* 可以實做基本的 Rails 網站以及做簡單的佈署
* 對於已經上線的 Rails 專案，能夠進行功能的追加和資料維護
* 知道如何在 GitHub 發 PR、merge 等協作流程，以及必須的 Git 指令：
  + 能將 commit 切成適度的大小
  + 能寫出適合的 PR 說明
  + 能針對 code review 進行修正
* 遇到問題時，能夠適時以口頭或線上工具向相關人員（在本例中為導師）求救

**參考資料**

* Git: <https://gitbook.tw/>
* Rails: <https://railsbook.tw/>

**必修課題**

**步驟1: 建立 Rails 的開發環境**

**1-1: 安裝 Ruby**

* 利用 [rbenv](https://github.com/rbenv/rbenv) 或 [RVM](https://rvm.io/) 安裝最新版本的 Ruby
* 以 ruby -v 指令來確認 Ruby 的版本

**1-2: 安裝 Rails**

* 以 gem 指令安裝 Rails
* 安裝最新版本的 Rails
* 以 rails -v 指令來確認 Rails 的版本

**1-3: 安裝資料庫（PostgreSQL）**

* 在你使用的 OS 下安裝 PostgreSQL
  + macOS 的話，請以 brew 等工具安裝

**步驟2: 在 GitHub 建立 repository**

* 在你的環境中安裝 Git
  + macOS 的話，請以 brew 等工具安裝
  + 以 git config 設定 user name 和 email
* 請考慮專案名稱（也等於 repo 名稱）
* 建立 repo
  + 如果沒有帳號的話，先申請帳號
  + 接著建立空白的 repo

**步驟3: 建立 Rails 專案**

* 以 rails new 指令，建立 Rails 應用程式最低限度的樣板和檔案
* 在 rails new 產生的專案目錄下，建立 docs 資料夾，並將本教程文件 commit 進去
  + 目的是為了方便之後開發時可以參考
* 將成品 push 到 GitHub 的 repo
* 將使用的 Ruby 版號寫進 Gemfile（也請確認 Rails 版號是否有標明）

**步驟4: 想像網站成品會是什麼樣子**

* 開始進行設計之前，先和導師一起討論對最終成品的預想。建議在紙上畫 prototype
* 請參照網站需求，開始想需要怎樣的資料結構
  + 需要哪些 model (或資料表)？
  + 資料表會需要哪些欄位？
* 有想法之後，將 model 的關係圖手繪出來
  + 完成後將關係圖拍照存檔，放進 repo 裡
  + 把 table schema 寫進 README.md（model 名稱、欄位名稱、資料形態）

※ 在這個階段，model 關係圖不需要是完全正確的。以現在所能預想的範圍來規劃就好（做到後面的步驟，發現需要修改時再來調整的概念）

**步驟5: 資料庫連接等週邊設定**

* 建立新的 topic 分支
  + 之後都在 topic 分支上開發並進行 commit
* 安裝 bundler
* 在 Gemfile 安裝 pg（PostgreSQL 的 adapter）
* 設定 database.yml
* 以 rails db:create 建立資料庫
* 以 rails db 確認有正確連接資料庫
* 在 GitHub 上建立 PR 並請人 review
  + 必要時，請在 PR 上標柱 WIP（Work In Progress）
  + 收到 Comment 後就做必要的處置。收到兩個 LGTM（Looks Good To Me） 後就可以 merge 回 master

**步驟6: 建立任務 model**

開始來做管理任務所需要的 CRUD。一開始先簡單做，只要能記錄名字和任務內容即可。

* 以 rails generate 指令建立 CRUD 所需的 model 類別
* 撰寫 migration 並以此建立資料表
  + 非常重要：migration 要確定能安全回到上一步的狀態！請養成以 redo 確認的習慣
* 以 rails c 指令，透過 model 確認有正確連接資料庫
  + 順便試著以 ActiveRecord 建立資料
* 在 GitHub 上發 PR 並請人 review

**步驟7: 新增、修改、刪除任務**

* 製作任務的列表、新增、檢視以及修改頁面
  + 以 rails generate 指令產生 controller
    - 請和導師討論要用哪一種 template engine（ERB / Slim / Haml..etc）
  + 實做 controller 和 view 必要的部分
  + 完成新增、修改、刪除之後需要在畫面上顯示的 Flash 訊息
* 修改 routes.rb，讓 http://localhost:3000/ 會顯示任務的列表頁面
* 在 GitHub 上發 PR 並請人 review
  + 之後的 PR，如果覺得過於龐大，就需要開始考慮分割成多個 PR

**步驟8: 寫 E2E 測試**

* 寫 spec 的事前準備
  + 準備 spec/spec\_helper.rb 、spec/rails\_helper.rb
* 針對任務的功能來寫 feature spec
* 導入 Travis CI 之類的 CI 工具，每次 Push 後自動跑 Spec
  + 太難的話可以請導師幫忙設定

**步驟9: 將網站中的中文部分共用化**

* 利用 Rails 的 i18n 功能，將 View / Controller / Model 中的語言部份共用化

※ i18n 化的好處是，之後的步驟中，各種訊息的處理會輕鬆很多

**步驟10: 設定 Rails 的時區**

* 將 Rails 的時區設為台灣（台北）

**步驟11: 任務列表以建立時間排序**

* 資料預設是以 id 進行排序，請試著讓它以建立時間排序
* 完成後，撰寫 feature spec

**步驟12: 資料驗證**

* 開始設定資料驗證
  + 請思考需要在哪個欄位上加入哪種驗證比較好
  + 與之配合的 DB 限制，請寫成 migration
  + 以 rails generate 指令產生 migration file
* 在頁面上加入驗證的錯誤訊息
* 撰寫對應的 model 測試
* 在 GitHub 上發 PR 並請人 review

**步驟13: 網站佈署**

目的：將 master 分支上的簡易任務管理系統推上線

* 試著將網站 deploy 到 Heroku 上
  + 沒有帳號的話，請建立帳號
* 看一下被推上 Heroku 的網站
  + 接下來就會在這裡建立任務並繼續開發
  + ※ 不過，推上 Heroku 後，就是在網路上公開了，請注意不要放敏感資料
    - 現階段，或許可以考慮加入 basic 認證
  + 今後，每個步驟完成後，就繼續將成品推上 Heroku
* 將佈署的方法寫進 README.md
  + 也將使用的 framework 版號等資料記下來

**步驟14: 加入結束時間**

* 任務可設定結束時間
* 列表頁可以結束時間排序
* 擴充 spec
* PR/review 後佈署

**步驟15: 加入狀態，並且能夠查詢**

* 在任務上加入狀態（待處理、進行中、完成）
  + 【選項】不是初學者的話，可以使用管理 state 的 gem
* 在列表頁面，要能夠以標題和狀態進行查詢
  + 【選項】不是初學者的話，可以使用 ransack 等 gem
* 在設定條件查詢時，請觀察 log 並確認 SQL 的變化
  + 之後的步驟也需要這麼做，請養成習慣
* 建立 search index
  + 準備一定程度的測試資料後，觀察 log/development.log 以確認加入 index 後對速度的改善
  + 【選項】使用 PostgreSQL 的 explain 等功能，檢視資料庫端的 index 使用狀況
* 針對查詢功能增加 model spec（feature spec 也要擴充）

**步驟16: 設定優先順序**

* 在任務上加入優先順序（高、中、低）
* 列表頁可依優先順序做排序
* 擴充 feature spec
* PR/review 後佈署

**步驟17: 增加分頁功能**

* 使用 kaminari gem 在列表頁面加入分頁功能

**步驟18: 加入設計**

* 使用 Bootstrap 4，為目前的作品套入設計
  + 【選項】自己寫 CSS 來設計

**步驟19: 支援多人使用**

* 建立使用者 model
* 以 seed 建立第一個使用者
* 建立使用者和任務的關聯
  + 建立關聯所需的 index
  + 要避免 N+1 問題
  + ※ 推上 Heroku 時，已經建立過的任務，要和使用者建立關係（資料維護）

**步驟20: 登入/登出功能**

* 這裡不使用任何 Gem，請自己實做
  + 不使用 devise 等便利的 Gem，是為了讓新人能更深入了解 Rails 中 HTTP cookie 和 session 的原理
  + 以及加強對一般認證機制的理解（例如密碼的處理）
* 實做登入的頁面
* 未登入時，不能進入任務管理頁面
* 請改成只能看到自己建立的任務
* 實做登出功能

**步驟21: 使用者管理頁面**

* 在頁面上新增管理選單
* 管理頁面的網址 /admin
  + 在修改 routes.rb 之前，請想一下 URL 以及 routing name（會變成 \*\_path 的部分）要怎麼設計
* 實做使用者的列表、新增、修改、刪除等功能
* 刪除使用者後，也一併刪除該使用者的任務
* 在使用者列表頁面，顯示使用者的任務數量
* 能夠看到使用者所建立的任務列表

**步驟22: 為使用者加入角色**

* 將使用者分為管理員和一般使用者
* 請改成只有管理員可以存取使用者管理頁面
  + 若一般使用者存取管理頁面時，需提示權限不足訊息無法存取，並轉向適當頁面
* 能在使用者管理頁面選擇角色
* 管理者只剩下一個人時，不能再被刪除
  + 利用 model 的 callback 實做
* ※ 可以自己決定是否要使用 Gem

**步驟23: 為任務加入標籤**

* 一個任務可以設定多個的標籤
* 能夠以標籤進行搜尋

**步驟24: 設定錯誤頁面**

* 客制化 Rails 的預設錯誤頁面
* 根據不同狀況，設定適合的錯誤頁面
  + 至少需要做 404 和 500 這兩頁

**後記**

辛苦了，恭喜你已經完成本次教材！

另外，雖然沒有包括在教程中，以下這些主題也是必要的技能，希望各位能漸漸掌握（大部分會在專案開發的過程中學習到）。

* 加深對網站應用程式的基本理解
  + HTTP 與 HTTPS
* Rails 稍微進階一點的用法
  + STI
  + logging
  + explicit transactions
  + 非同步處理
  + asset pipeline（佈署方面）
* JavaScript、CSS 等 frontend 技術
* 資料庫
  + SQL
  + query 的效能
  + index
* Server 環境
  + Linux OS (CentOS)
  + web server（nginx）的設定
  + application server（Passenger）的設定
  + PostgreSQL 的設定
* 佈署工具
  + Capistrano

**（番外篇）選修課題**

有別於前述的必修課題，任務管理系統還有以下選修課題。要做到什麼程度，請和導師一起討論決定。

**選修課題1: 任務接近或超過結束時間時，顯示提示訊息**

* 在登入時提醒有接近或超過結束時間的任務
* 如果能做出已讀、未讀狀態更好

**選修課題2: 讓使用者之間可以共享任務**

* 任務可讓多個的使用者檢視或進行修改
  + 例：在新人和導師之間共享
* 顯示任務作者

**選修課題3: 群組**

* 選修課題2的延續
* 新增群組設定的功能，並增加群組內分享的功能

**選修課題4: 附加檔案**

* 讓任務可以上傳附加檔案
* 使用 Heroku 的話，要能夠管理上傳到 S3 bucket 的檔案
* 請使用適合的 Gem

**選修課題5: 使用者大頭照**

* 讓使用者可以設定大頭照
* 由於上傳的檔案會被當成 icon 使用，請做成 thumbnail
* 請使用適合的 Gem

**選修課題6: 任務日曆**

* 為了將結束時間視覺化，在日曆上以結束時間來顯示任務
* 可以自己決定是否要使用其它套件

**選修課題7: 以拖曳方式排序**

* 在任務列表中，加入以拖曳方式排序的功能

**選修課題8: 標籤使用率的圖表**

* 來把統計資料視覺化吧
* 圖表的形式，要簡單易懂
* 可以自己決定是否要使用其它套件

**選修課題9: 任務到期通知信**

* 任務接近結束時間時，在背景以 email 進行通知
* 使用雲端服務來發信
  + Heroku 的話就是 SendGrid
  + VPS 的話就是 MailGun 或是公司的 Postal
* 以一天一次的頻率，批次發信
  + Heroku 的話用 Heroku Scheduler（add-on）
  + VPS 的話就設定 cron job

**選修課題10: 用 Let's Encrypt 在 VPS 上加上 SSL 憑證**

* 用 Certbot 申請 letsencrypt 憑證並設定在 Nginx 上