

Angular 16 進階開發練功坊

實作課程

多奇數位創意有限公司

全端工程師 黃升煌 (Mike)

<https://fullstackladder.dev>





課程目標

課程目標

- 完成一個簡易的內容管理系統，功能包含
 - 登入
 - 顯示全部文章
 - 顯示單筆文章
 - 新增文章（需登入）
- 透過路由設定切換各頁面
- 透過表單驗證來確定輸入資料內容正確
- 其他補充技巧



前置準備

下載前端起始專案

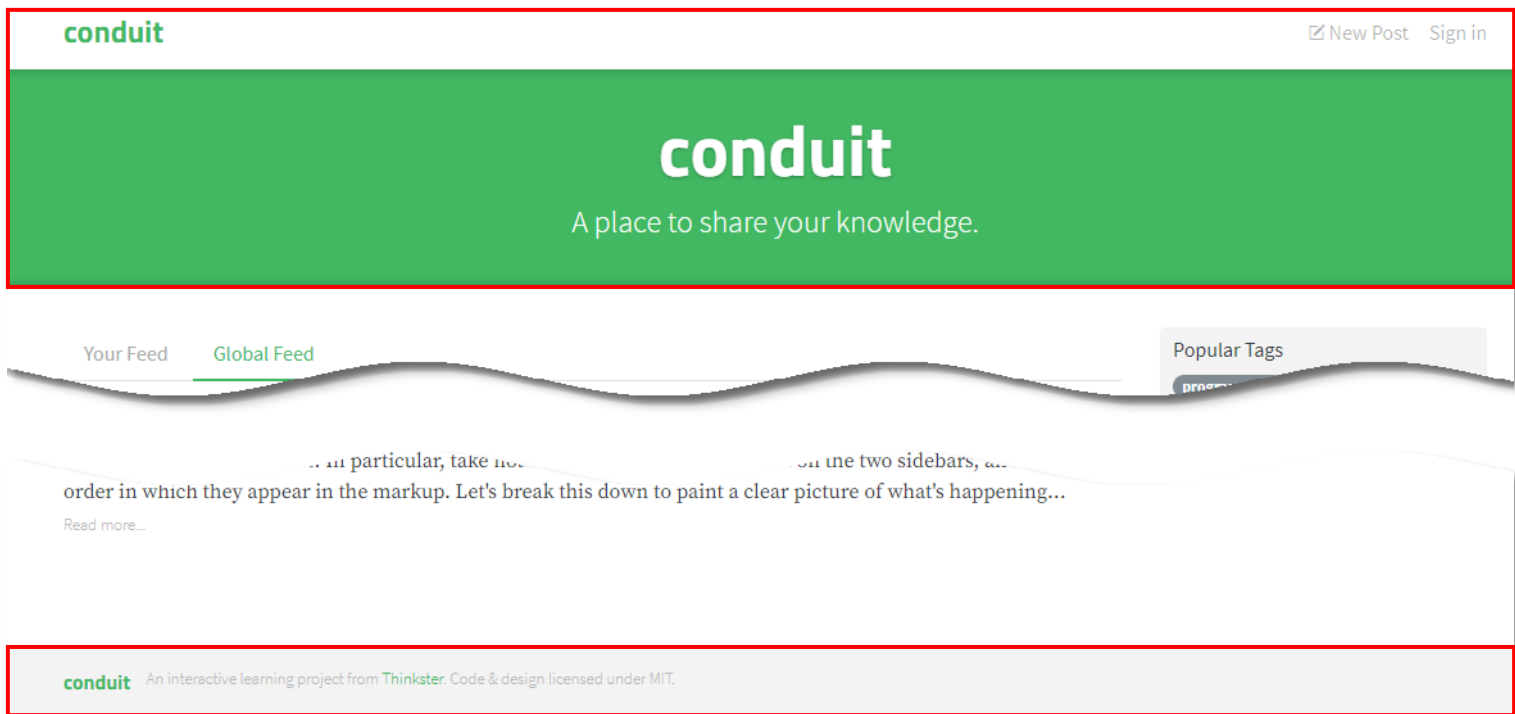
- [前端起始專案位置](#)
- 使用命令提示字元進入目錄，安裝套件並啟動
 - `npm install`
 - `npm start`
- [完成版的前端範例程式](#)

起始專案架構說明

- **templates**：包含基本的樣板 HTML 原始檔
- **app\interfaces**：包含串接 API 所需的 interface
- **app\login.service.ts**：串接登入 API 的 service
- **app\post.service.ts**：串接文章相關 API 的 service

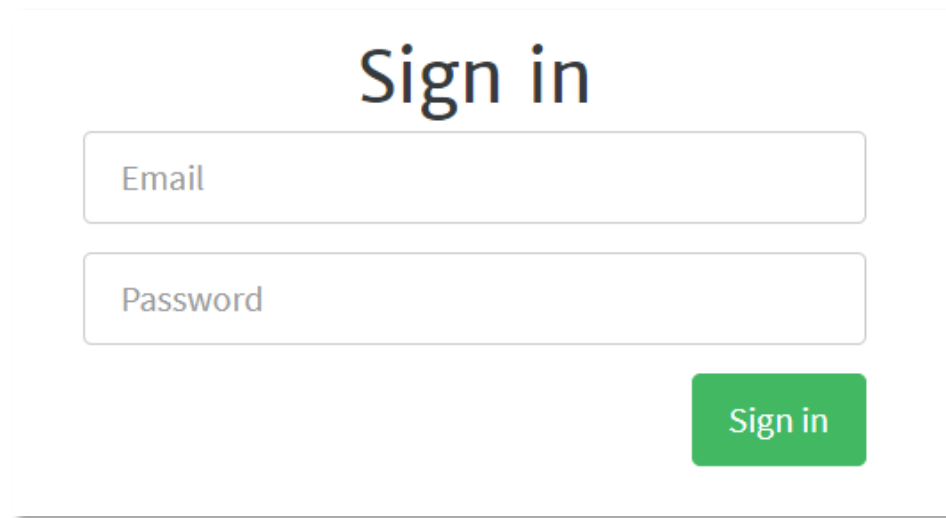
樣板 - Layout

- templates/layout.html
 - 上面的 NavBar 和下面的 Footer 為共用的 layout



樣板 - 登入頁面

- templates/login.html



Sign in

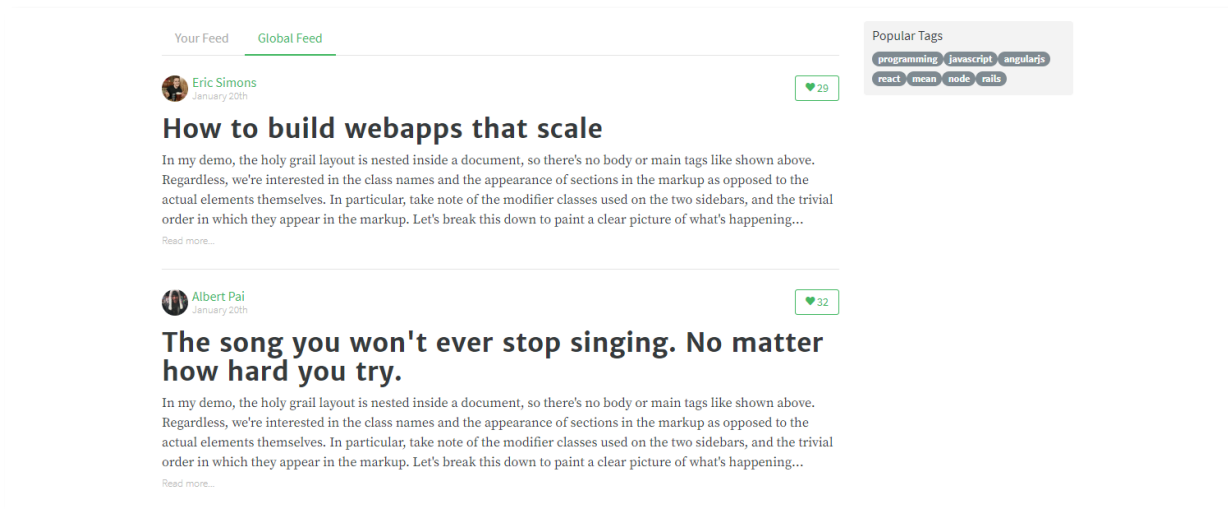
Email

Password

Sign in

樣板 - 顯示全部文章


- templates/posts.html



樣板 - 顯示單篇文章

- templates/post.html

How to build webapps that scale

Eric Simons
January 20th

[+ Follow Eric Simons \(10\)](#)[♥ Favorite Post \(29\)](#)

Web development technologies have evolved at an incredible clip over the past few years. We've gone from rudimentary DOM manipulation with libraries like jQuery to supercharged web applications organized & powered by elegant MV* based frameworks like AngularJS. Pair this with significant increases in browser rendering speeds, and it is now easier than ever before to build production quality applications on top of Javascript, HTML5, and CSS3.

While these advances have been incredible, they are only just starting to affect the clear platform of the future: mobile. For years, mobile rendering speeds were atrocious, and the MVC frameworks & UI libraries provided by iOS and Android were far superior to writing mobile apps using web technologies. There were also some very public failures -- Facebook famously wrote their first iOS app in 2011 using HTML5 but ended up scrapping it due to terrible performance.

For years now, hybrid apps have been mocked and jeered by native app developers for being clunky and ugly, having subpar performance, and having no advantages over native apps. While these may have been valid reasons in 2011, they are now virtually baseless, thanks to a collection of new technologies that have emerged over the past two years. With these technologies, you can design, build, and deploy robust mobile apps faster than you could with native technologies, all while incurring little to no app performance penalties. This is thanks in large part to super fast mobile browser rendering speeds and better JavaScript performance. This course is designed to teach you how to effectively use these new technologies to build insanely great mobile apps.

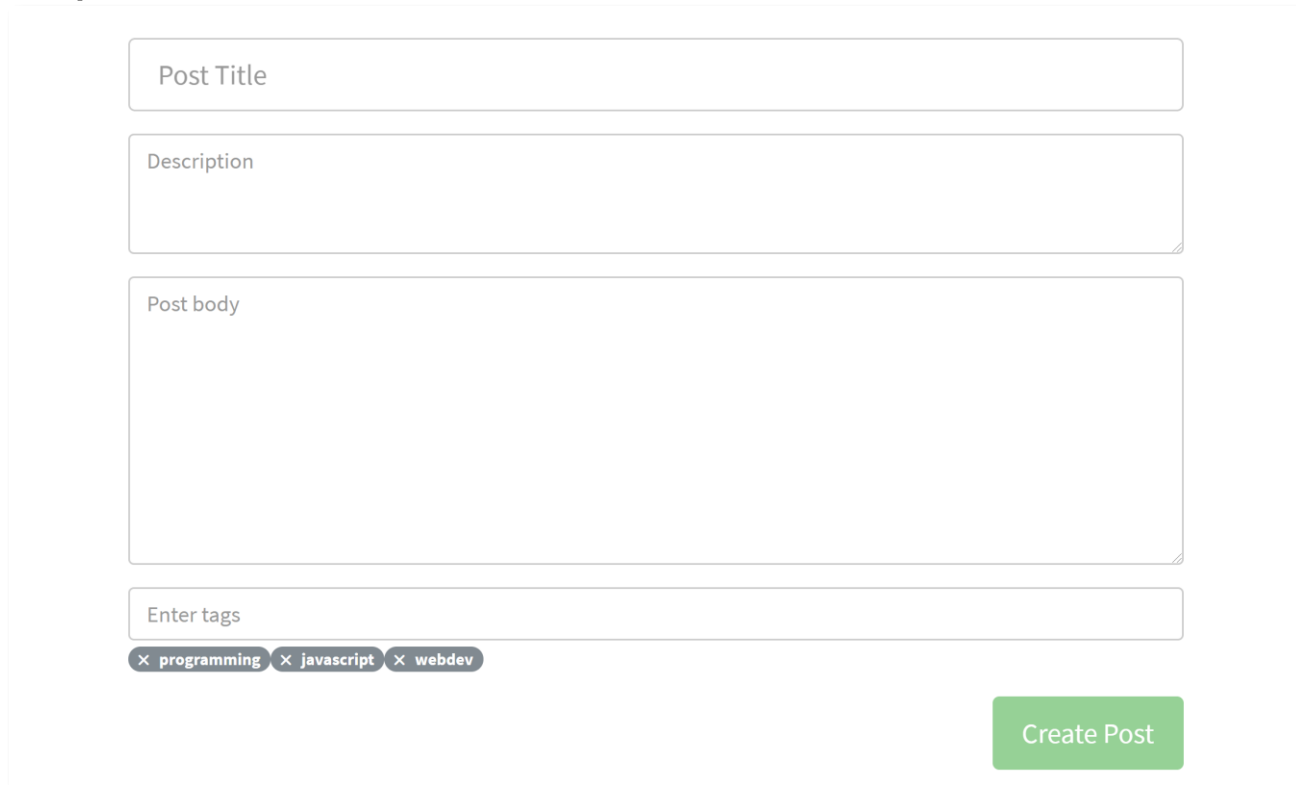
Without further ado, we'd like to welcome you to the future of mobile app development, freed from the shackles of native languages & frameworks. Let's learn what the new mobile stack consists of and how it works.

Introducing Ionic.

Before, building hybrid apps was a chore -- not because it was hard to build web pages, but because it was hard to build full-

樣板 - 建立文章

- templates/create.html



A screenshot of a web form for creating a post. The form is contained within a light gray box with a subtle shadow. It features four main input areas: a text box for 'Post Title', a text box for 'Description', a large text area for 'Post body', and a text box for 'Enter tags'. Below the 'Enter tags' box, there are three tags displayed: 'programming', 'javascript', and 'webdev', each preceded by a small 'x' icon. A green button labeled 'Create Post' is positioned at the bottom right of the form.

Post Title

Description

Post body

Enter tags

x programming x javascript x webdev

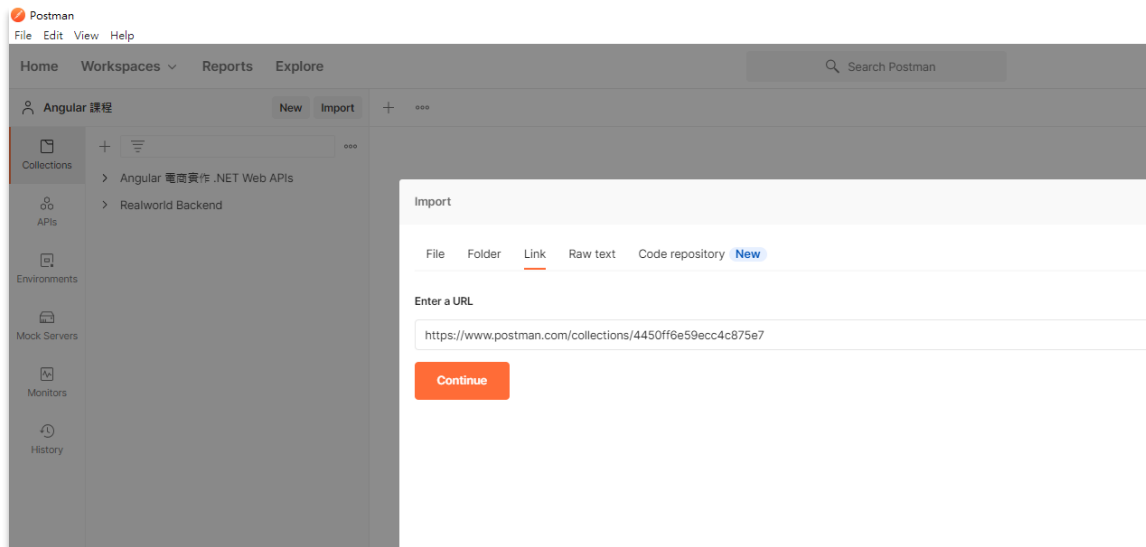
Create Post

下載後端範例

- 後端範例位置
 - <https://github.com/coolrare/realworld-backend>
 - 使用命令提示字元進入目錄，安裝套件並啟動
 - `npm install`
 - `npm start`
- Postman 匯入連結
 - <https://www.getpostman.com/collections/4450ff6e59ecc4c875e7>

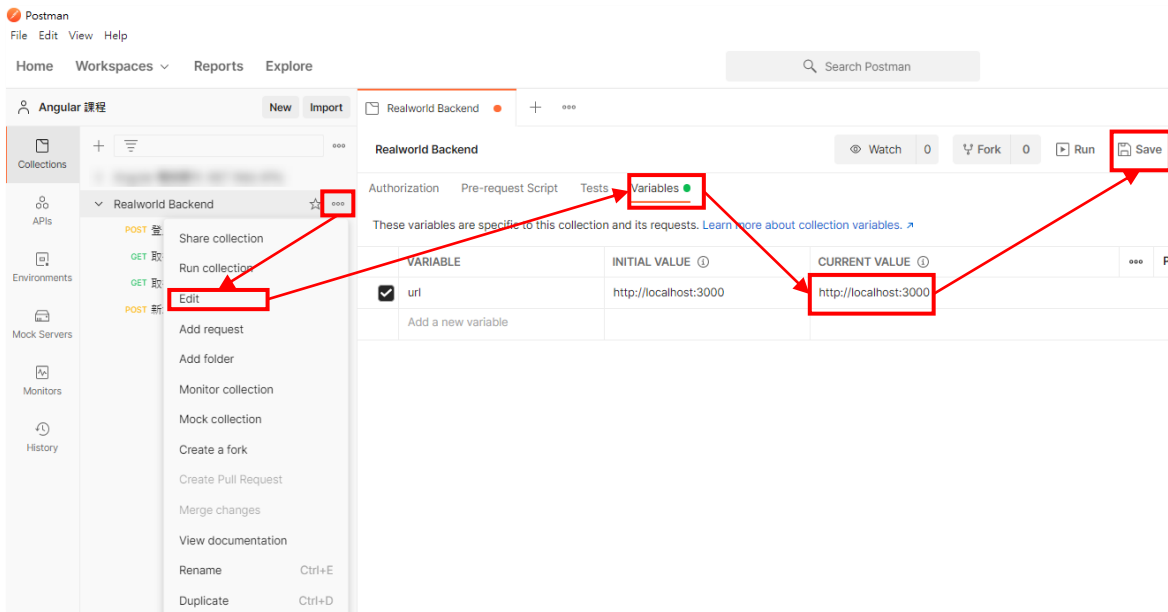
匯入 Postman 的方式

- 安裝 [postman](#)
- 點擊上方 import 按鈕
- 切換到 link 頁籤
- 輸入 Postman 匯入連結
- Continue



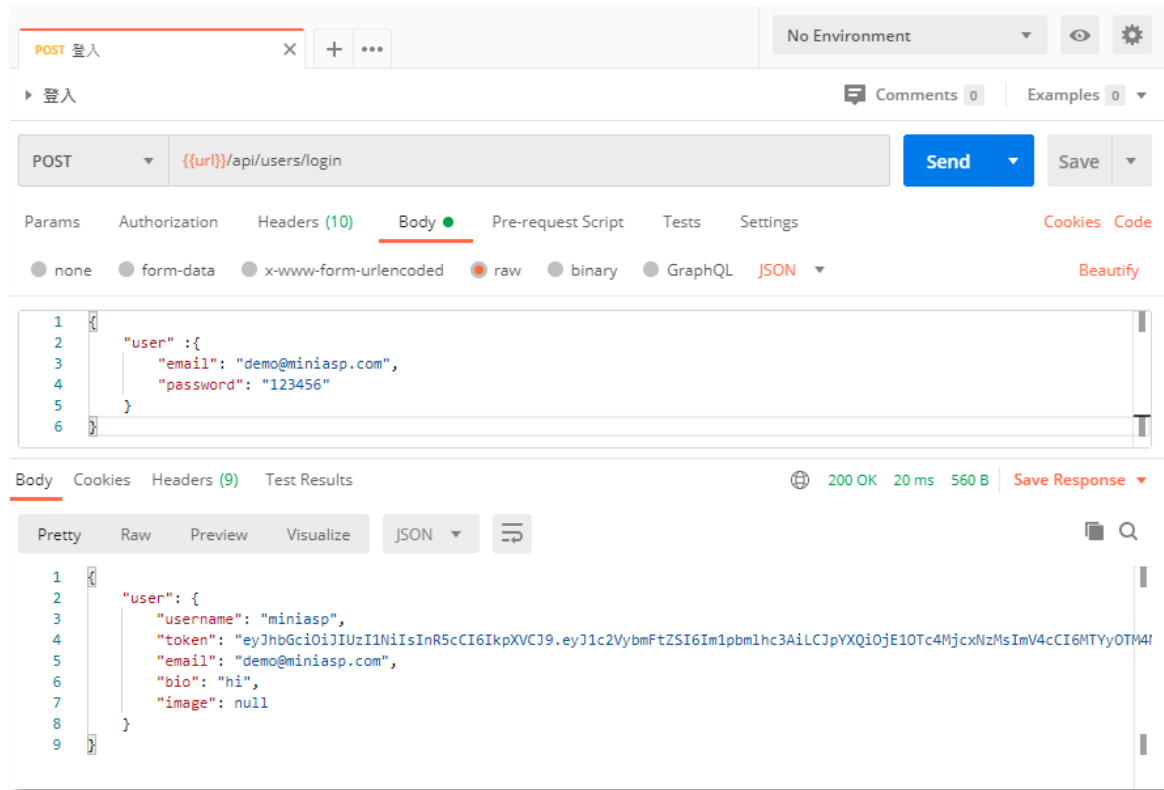
設定 Postman Collection

- 點擊 Realworld Backend 旁邊的 ... 按鈕
- 選擇 Edit
- 切到 Variables 頁籤
- url 的 CURRENT VALUE 確認
 - <http://localhost:3000>
 - 如果沒有請自行輸入
 - 按下 Save 儲存



後端 API 說明：登入 API

- POST `{{url}}/api/users/login`



後端 API 說明：文章清單 API

- GET `{{url}}/api/articles`

The screenshot displays a REST client interface with the following details:

- Request:** Method `GET`, URL `{{url}}/api/articles`. The `Send` button is highlighted.
- Params:** The `Query Params` tab is active, showing a table with columns `KEY`, `VALUE`, and `DESCRIPTION`. A single row is present with `Key` and `Value`.
- Response:** The `Body` tab is active, showing a `200 OK` status with a response time of `14 ms` and a size of `992 B`. The response is formatted as `JSON` and contains the following data:

```
1 {
2   "articles": [
3     {
4       "id": "1",
5       "title": "Title (1)",
6       "description": "This is description (1)",
7       "body": "This is article body (1)",
8       "tagList": [
9         "angular"
10      ],
11       "createdAt": "2019-04-25T12:45:37.095Z",
12       "updatedAt": "2019-04-25T12:45:37.095Z",
13       "author": "miniasp"
14     },
15     {
16       "id": "2",
17       "title": "Title (2)",
18       "description": "This is description (2)",
```


後端 API 說明：單篇文章 API

- GET `{{url}}/api/articles/{{id}}`

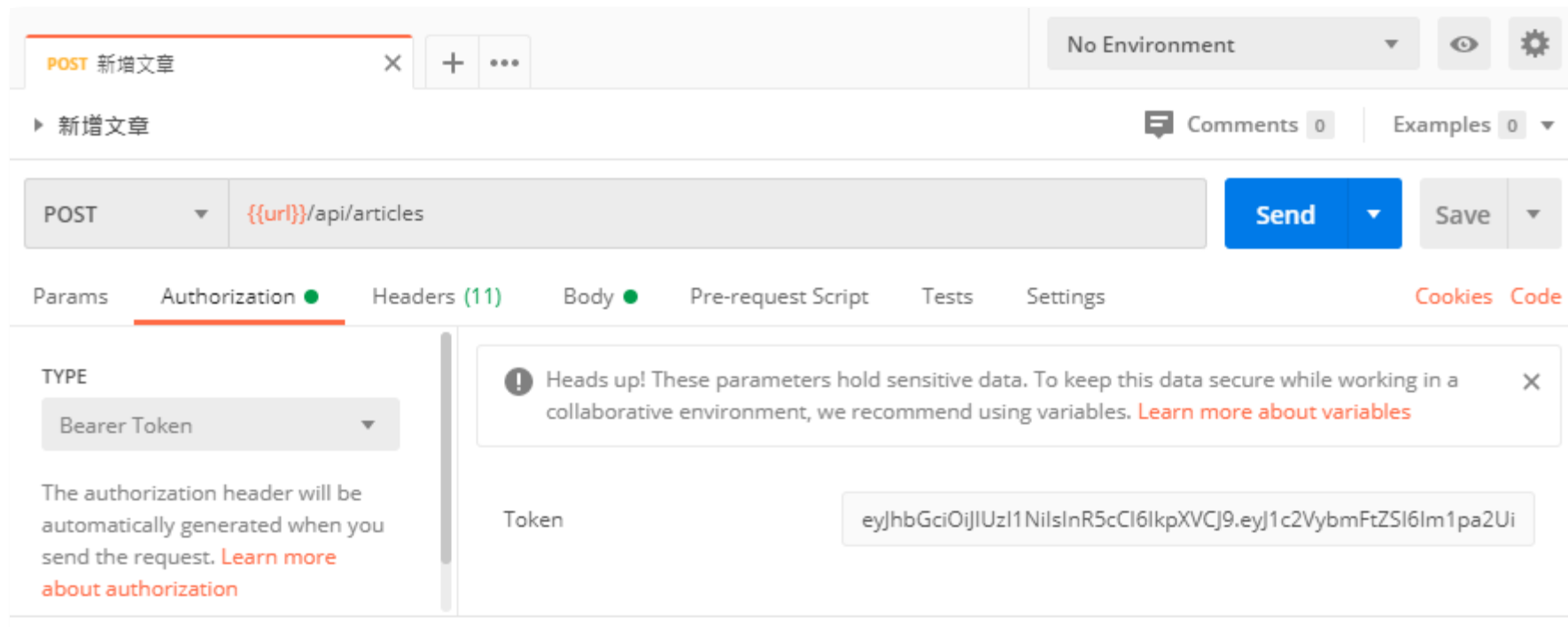
The screenshot displays a REST client interface with the following details:

- Request Method:** GET
- Request URL:** `{{url}}/api/articles/{{id}}`
- Response Status:** 200 OK, 12 ms, 546 B
- Response Body (JSON):**

```
{  "article": {    "id": "1",    "title": "Title (1)",    "description": "This is description (1)",    "body": "This is article body (1)",    "tagList": [      "angular"    ],    "createdAt": "2019-04-25T12:45:37.095Z",    "updatedAt": "2019-04-25T12:45:37.095Z",    "author": "miniasp"  }
```

後端 API 說明：新增文章 API

- POST `{{url}}/api/articles`
 - 需額外帶入 JWT Token 才可從登入 API 取得資料





實戰練習

任務 01：建立並設定路由機制

- 請依照下列表格建立對應元件
- 並依照路徑設定好**延遲載入**元件

路由	子路由	元件	樣板
/		LayoutComponent	templates/layout.html
	/	直接跳轉到 /posts 頁面	
	posts	PostsComponent	templates/posts.html
	post/:id	PostComponent	templates/post.html
	create	CreateComponent	templates/create.html
/login		LoginComponent	templates/login.html

任務 02：使用 RouterLink 與 RouterLinkActive

conduit

☑ New Post Sign in

1. 點擊 conduit 後轉到首頁

2. 點擊 New Post 後轉到 `/create` 頁面

3. 轉到 `/create` 頁面後，New Post 的連結加入 `active` 樣式

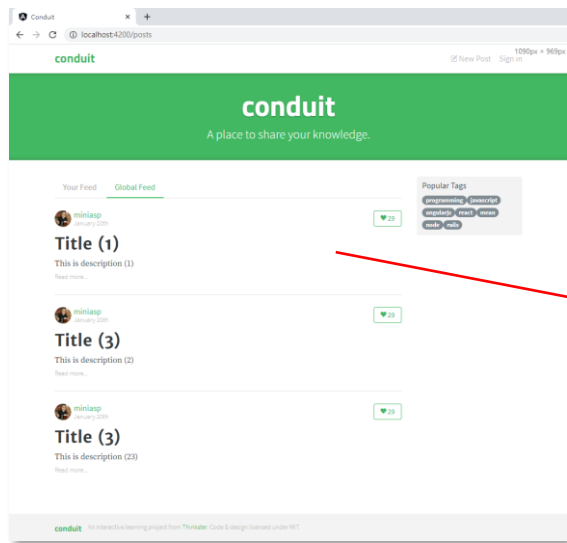
4. 點擊 Sign in 後轉到 `/login` 頁面

```
.nav-item.active {  
  background: yellow;  
}
```

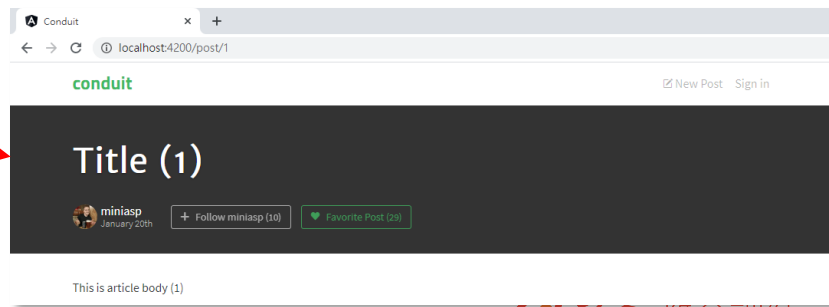
目前 style.css 中的 active 樣式參考

任務 03：在程式中取得路由參數

- 在 `/posts` 頁面，串接**文章列表 API** 顯示全部文章列表
 - 文章列表 API：呼叫 `PostService` 的 `getArticles()` 方法
- 單篇文章連結至 `/post` 頁面，並將**文章 id** 附在網址後面 (`/post/:id`)
- 在 `/post/:id` 頁面，串接**單篇文章API** 顯示單篇文章內容
 - 單篇文章 API：呼叫 `PostService` 的 `getArticle(id)` 方法

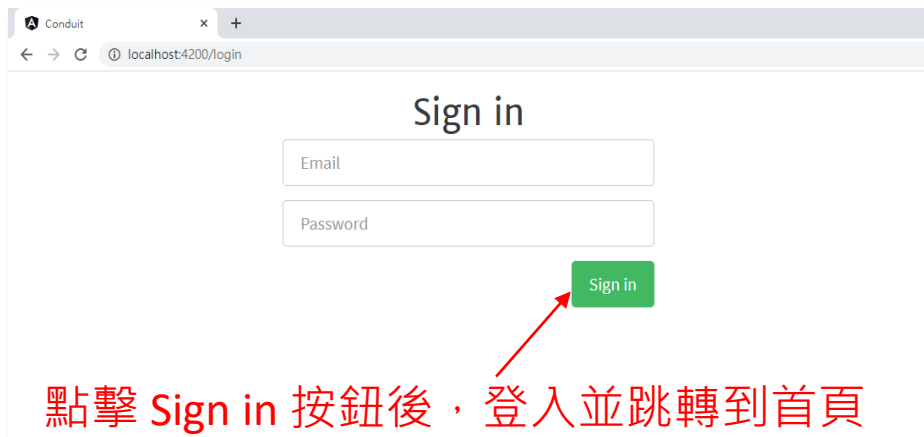


`/posts` 頁面顯示全部文章
點擊單篇文章後跳轉到 `/post/:id` 並顯示該篇文章內容



任務 04：在程式中動態切換路由

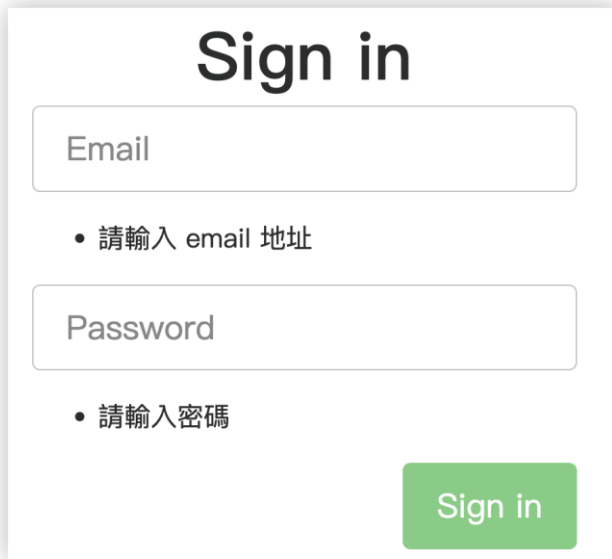
- 在 Sign in 頁面，按下按鈕
 - 使用 Router 的 **navigate()** 方法，跳轉到**首頁**
 - 串接登入 API，取得 JWT Token 並暫存在 **localStorage**
 - 登入 API：呼叫 **LoginService** 的 **login()** 方法
 - 測試 API 帳號密碼：demo@miniasp.com / 123456



點擊 Sign in 按鈕後，登入並跳轉到首頁

任務 05：Template-driven Form 驗證表單資料

- 目標：登入頁面驗證 Email 與 密碼 欄位格式，錯誤時提示訊息
 - Email：必填、必須為 Email 格式
 - 密碼：必填、長度 ≥ 4
- 驗證錯誤時，Sign in 按鈕為 disabled



A mockup of a 'Sign in' form. The title 'Sign in' is centered at the top. Below it are two input fields: 'Email' and 'Password'. Each field has a bullet point below it indicating a requirement: '• 請輸入 email 地址' for the email field and '• 請輸入密碼' for the password field. At the bottom right is a green 'Sign in' button.

Sign in

Email

- 請輸入 email 地址

Password

- 請輸入密碼

Sign in

任務 06：Reactive Forms 繫結表單資料

- 使用 Reactive Forms 操作 CreateComponent 元件內的表單
 - 使用 FormBuilder 物件
 - 建立 **表單模型**
 - 使用 formGroup 與 formControlName 和 formArrayName 指令
 - 顯示 **表單資料**
 - 在程式中操作表單模型

任務 07：Reactive Forms 驗證表單資料

- 目標：新增文章頁面驗證 標題 與 內文 欄位格式，錯誤時提示訊息
 - 標題 (title)：必填
 - 內文 (body)：必填、長度 ≥ 10
- 驗證錯誤時，Create Post 按鈕為 disabled

任務 08：使用路由守門員 (Route Guard)

- 目標：進入新增文章頁面前，使用檢查是否登入
 - 使用路由守門員 (Route Guard) 限制進入
 - 必須在 **localStorage** 內有設定 token 才允許進入
 - 未登入則先導向登入頁面 (**/login**), 並在網址參數中帶入新增文章頁面網址 (**?redirect=create**)

任務 09：練習使用 `HttpInterceptor`

- 目標：串接新增文章 API，在 API 發送前加入 `localStorage` 內的 JWT Token
 - 新增文章 API：呼叫 `PostService` 的 `createArticle()` 方法
 - 透過 `HttpInterceptor` 替所有請求加入驗證 header
 - `Authorization: Bearer {JWT_TOKEN}`
- 文章新增完成後，跳轉到首頁



多奇·教育訓練

THANK YOU!

Q&A