

Day6 - 提示詞範本

有些人就是不知如何問 AI，得到的解答自然也不具參考價值，這時提示詞範本就派上用場了

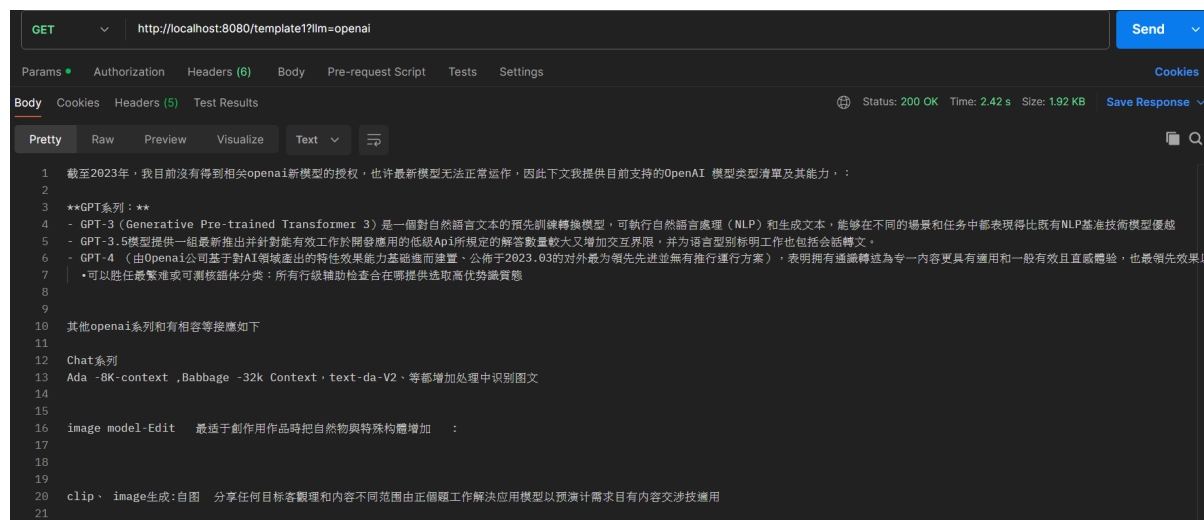
我們可以預設一些提示詞，只需將關鍵字換掉即可，下面是一個簡單的範例

```
@GetMapping(value = "/template1")
public String template1(@RequestParam String llm) {
    String template = "請問{llm}目前有哪些模型，各有甚麼特殊能力"
    PromptTemplate promptTemplate = new PromptTemplate(template)
    Prompt prompt = promptTemplate.create(Map.of("llm", llm))
    ChatResponse response = chatModel.call(prompt);
    return response.getResult().getOutput().getContent();
}
```

程式重點說明

- template 中包含一個 `{llm}`，可以在建立 Prompt 時替換
- PromptTemplate 就是提示詞範本類別，若 template 中有可替換的字串(前一點的 `{llm}`)，可在執行 create 時傳入Map替換，若有多個變數需替換就傳入多組 Map
- PromptTemplate 可以使用 create 得到 Prompt 物件，也可以使用 createMessage 得到 Message 物件，這部分取決於後面要如何應用

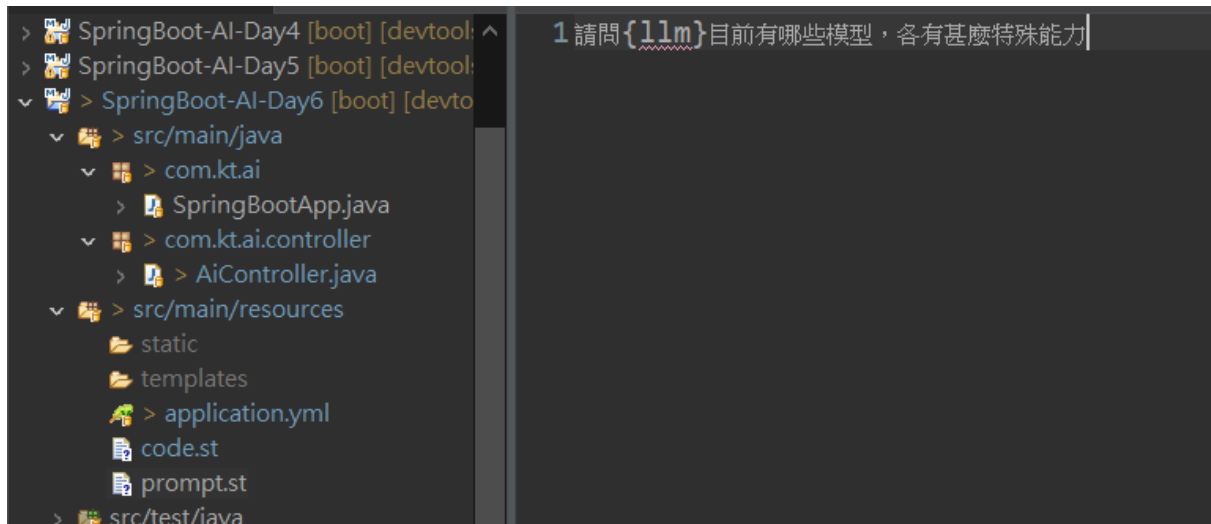
看看 Postman 測試得到的結果



可以看到我們只需輸入 openai，AI 就能依據我們定義的模板來詢問，甚至能依特定格式產出結果

有時模板太長打在程式裡不美觀也不容易維護，PromptTemplate 也支援使用 Resource 讀取文字檔

首先在 resources 目錄下建立一個文字檔: `prompt.st` 內容就是上個範例的模板字串



程式碼改成如下

```
@Value("classpath:prompt.st")
private Resource templateResource;

@GetMapping(value = "/template2")
public String template2(@RequestParam String llm) {
    PromptTemplate promptTemplate = new PromptTemplate(templateResource);
    Prompt prompt = promptTemplate.create(Map.of("llm", llm));
    ChatResponse response = chatModel.call(prompt);
    return response.getResult().getOutput().getContent();
}
```

這樣看起來是不是簡潔許多？

最後我們來寫一個演算法產生器，查詢時只需輸入程式語言以及需要的演算法，AI 就能給我們詳細的程式以及說明了

1. 先在 resources 增加一個 code.st，內容放入

```
language {language}  
method {methodName}
```

請提供 language 語言的 method 範例，並提供詳細的中文說明

2. 因為上面有兩個需替換的字串，程式碼需要有兩個 *@RequestParam*

```
@GetMapping(value = "/template3")  
public String template3(@RequestParam String language, @R
```

3. 由於有兩個參數，建立 Prompt 也需要傳入兩個 Map

```
Prompt prompt = promptTemplate.create(Map.of("language", lang
```

最終程式碼如下

```
@Value("classpath:code.st")  
private Resource templateResource2;  
  
@GetMapping(value = "/template3")  
public String template3(@RequestParam String language, @R  
    PromptTemplate promptTemplate = new PromptTemplate(te  
    Prompt prompt = promptTemplate.create(Map.of("languag  
    ChatResponse response = chatModel.call(prompt);  
    return response.getResult().getOutput().getContent();  
}
```

來看看執行結果(中間程式碼太長，將其省略，完整內容可自行測試)

```
GET http://localhost:8080/template3?language=java&methodName=快速排序

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

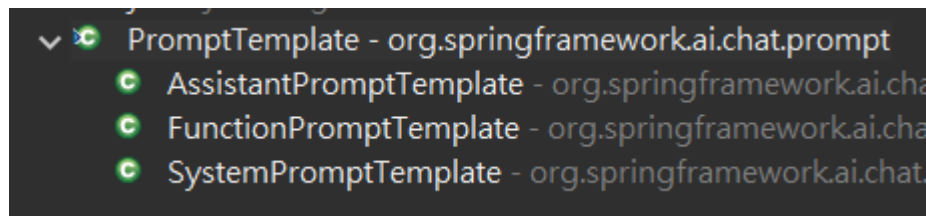
Body Cookies Headers (5) Test Results Status: 200 OK Time: 2.99 s Size: 2.36 KB Save Response

Pretty Raw Preview Visualize Text

1  **快速排序 (Quicksort) **
2
3  快速排序是一種高效的排序算法，適用於大型數據集。它的平均時間複雜度為  $O(n \log n)$ ，最壞情況下為  $O(n^2)$ 。快速排序的基本思想是選擇一個ivot元素，將數據分成兩部分，一部分小於ivot，一部分大於ivot。
4
5  **Java 實現**
6
7  以下是快速排序的 Java 實現：
8  ```java
9  public class QuickSort {
10     public static void sort(int[] arr) {
11         quickSort(arr, 0, arr.length - 1);
12     }
13
14     private static void quickSort(int[] arr, int low, int high) {
15         if (low < high) {
16             int pivotIndex = partition(arr, low, high);
17             quickSort(arr, low, pivotIndex - 1);
18             quickSort(arr, pivotIndex + 1, high);
19         }
20     }
21 }
```

```
51  **解說**
52
53  1. 'sort' 方法是快速排序的入口，調用 'quickSort' 方法進行排序。
54  2. 'quickSort' 方法是遞歸的核心，接受三個參數：'arr'（數組）、'low'（低位索引）和 'high'（高位索引）。
55  3. 如果 'low' 小於 'high'，則選擇一個ivot元素，將數據分成兩部分，並遞歸地對兩部分進行排序。
56  4. 'partition' 方法選擇ivot元素，並將數據分成兩部分。它返回ivot元素的索引。
57  5. 'swap' 方法交換兩個元素的值。
58
59  **例子**
60
61  ```java
62  int[] arr = {5, 2, 8, 3, 1, 6, 4};
63  QuickSort.sort(arr);
64  System.out.println(Arrays.toString(arr)); // [1, 2, 3, 4, 5, 6, 8]
65  ```
66  快速排序是一種高效的排序算法，適用於大型數據集。但是，它的最壞情況下時間複雜度為  $O(n^2)$ ，因此需要注意數據的分布情況。
```

還記得昨天提到 Message 有四種嗎？PromptTemplate 產出的 Prompt 預設就是使用 UserMessage 建立，Spring AI 也同時準備了其他三個 Message 的 PromptTemplate



回顧一下今天學到甚麼

- 使用 PromptTemplate 將模板字串的關鍵字換為輸入的內容再去 AI 查詢結果
- PromptTemplate 使用 Resource 讀取存於檔案中的模板字串

今日的原始碼:

<https://github.com/kevintsai1202/SpringBoot-AI-Day6.git>