Day15 - ChatModel vs ChatClient

前言：既生瑜，何生亮

記得我們 Day3 提到要自動綁定 ChatClient 卻失敗吧，要建立 ChatClient 只能使用 Builder 建立

```
@RestController
class MyController {
    private final ChatClient chatClient;
    public MyController(ChatClient.Builder chatClientBuilder)
        this.chatClient = chatClientBuilder.build();
    }
    @GetMapping("/ai")
    String generation(String userInput) {
        return this.chatClient.prompt()
            .user(userInput)
            .call()
            .content();
```

```
    }
}
```

若有多個 Component 會使用到則可以在 Config 類別中使用 Bean 的建立方式

```
@Configuration
class Config {
    @Bean
    ChatClient chatClient(ChatClient.Builder builder) {
        return builder.build();
    }
}
```

> 其實凱文大叔一直覺得之後的改版一定會讓 ChatClient 可以自動綁定

前面我們 ChatModel 也用得好好的，為何在講記憶之前要先說 ChatClient？

在 Spring 框架中，結尾冠上 Client 的類別，就是更高級的封裝，例如：RestClient、WebClient、JdbcClient等，而且這一系列的類別都提供 fluent 風格的 API，雖然 ChatModel 也提供一部分的 fluent API，不過許多設定還是得在其他物件設定好後才能引入，例如 Prompt

ChatClient 最大的差別就是將這些外部類別設定的資料也一起整合進來，下面就來比較一下 ChatModel 與 ChatClient 寫法的差異吧

1. Prompt

ChatModel:

```
ChatResponse response = chatModel
                .call(new Prompt(
                    new UserMessage("Tell me a joke"),
                ));
```

ChatClient:

```
ChatResponse chatResponse = chatClient.prompt()
    .user("Tell me a joke")
    .call()
    .chatResponse();
```

這個範例看起來雖然程式碼差不多，不過可以看到 ChatClient 在設定 Prompt 時是直接使用 .prompt() 這也是 Fluent API 的精髓

2. PromptTemplate

ChatModel:

```
@GetMapping("/template")
    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(te
        Prompt prompt = promptTemplate.create(Map.of("llm", l
        ChatResponse response = chatModel.call(prompt);
        return response.getResult().getOutput().getContent();
    }
```

ChatClient:

```
    @GetMapping("/template")
    public String template(@RequestParam String llm) {
        String template = "請問{llm}目前有哪些模型，各有甚麼特殊能力
        ChatResponse response = chatClient.prompt()
                .user(u -> u.text(template)
                        .param("llm", llm)
                        )
                .call()
                .chatResponse();
        return response.getResult().getOutput().getContent();
    }
```

這個例子可以看出 ChatClient 在設定 Message 時可直接使用 Lambda 建立 PromptTemplate 並使用 param 帶入參數，一氣呵成

3. **Structured Output Converter**

ChatModel:

```
record ActorsFilms(String actor, List<String> movies) {};
@GetMapping("/films")
public ActorsFilms films(String actor) {
    String template = """
            列出演員{actor}最有名的五部電影，需用繁體中文回答
            {format}
            """;
    BeanOutputConverter<ActorsFilms> beanOutputConverter =
            new BeanOutputConverter<>(ActorsFilms.class);
    String format = beanOutputConverter.getFormat();
    Generation generation = chatModel.call(
            new Prompt(new PromptTemplate(template, Map.of("a
    ActorsFilms actorsFilms = beanOutputConverter.convert(gen
    return actorsFilms;
}
```

ChatClient:

```
record ActorsFilms(String actor, List<String> movies) {};
@GetMapping("/films")
public ActorsFilms films(String actor) {
String template = """
            列出演員{actor}最有名的五部電影，需用繁體中文回答
            """;
    ActorFilms actorFilms = chatClient.prompt()
    .user(u -> u.text(template).param("actor", actor))
    .call()
    .entity(ActorFilms.class);
  return actorFilms;
}
```

可以感受出差異嗎？其實跟 PromptTemplate 一樣，ChatClient 就是將 BeanOutputConverter 寫在 Fluent API 裡

4. Using Defaults

在 ChatClient 中能設定一些 Default 參數，只要執行時若沒帶入參數就會使用預設的內容，舉個實際例子

```
@Configuration
class Config {
    @Bean
    ChatClient chatClient(ChatClient.Builder builder) {
        return builder.defaultSystem("你是個友善的聊天機器人,不管
                .build();
    }
}
```

之後執行 `chatClient` 時若沒使用 `.system()` 覆蓋 `SystemMessage`，Spring AI 就會使用 Default 內容來執行呼叫，以下預設的方法以及正式的方法

Default	Standard
defaultSystem	system
defaultUser	user
defaultFunction	function
defaultFunctions	functions
defaultOptions	options
defaultAdvisors	advisors

有注意到 `advisors` 嗎？這是前面沒提過的內容，也會是後面記憶跟 RAG 最重要的方法，就留在後面慢慢說明囉