Day18 - Spring AI官方記憶體

前言:動手寫程式的機會越來越少了

使用 Spring Boot 開發程式真的很快，現在連記憶儲存的類別都不讓我們動手，在不努力點就要被 AI 取代了XD

InMemoryChatMemory 是 **ChatMemory** 的唯一實作類別 (本來 Snapshot版本還有另一個)，下面是Spring AI 的程式碼，除了記錄歷史訊息外，透過 Chat ID 還能識別不同對話

InMemoryChatMemory 使用 `ConcurrentHashMap<String, List<Message>>` 儲存資料，每個 Chat ID 就能有獨立的記憶，而 **ConcurrentHashMap** 有**線程安全**跟**無阻塞存取**的特性，非常適合多人同時使用的

另外取得資料時也加入 lastN 的參數，透過 Lambda 語法快速篩選出最近 N 筆資料

InMemoryChatMemory 原始碼:

```
public class InMemoryChatMemory implements ChatMemory {
    Map<String, List<Message>> conversationHistory = new Conc
    @Override
    public void add(String conversationId, List<Message> mess
        this.conversationHistory.putIfAbsent(conversationId,
        this.conversationHistory.get(conversationId).addAll(m
    }
    @Override
    public List<Message> get(String conversationId, int lastN
        List<Message> all = this.conversationHistory.get(conv
        return all != null ? all.stream().skip(Math.max(0, al
    }
    @Override
    public void clear(String conversationId) {
        this.conversationHistory.remove(conversationId);
    }
}
```

將昨天的程式改用 InMemoryChatMemory 存取對話，另外因為系統開始複雜了，我們將複雜的邏輯獨立到 Service 元件處理

```
@RequiredArgsConstructor
@Service
public class ChatService {
    private final ChatClient chatClient;
    private ChatMemory chatMemory = new InMemoryChatMemory();

    public String chat(String chatId, String userMessage) {
        chatMemory.add(chatId, new UserMessage(userMessage));
        return this.chatClient.prompt()
                .messages(chatMemory.get(chatId, 30)) //取對話
                .call()
                .content();
    }
}
```

原本的 Controller 專心處理轉派

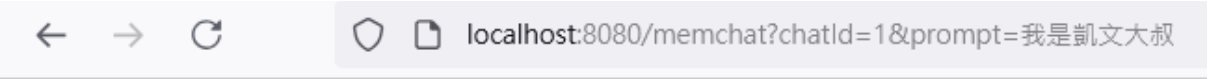
```
@RestController
@RequiredArgsConstructor
public class AiClientController {
    private final ChatService chatService;

    @GetMapping("/memchat")
    public String chat(@RequestParam String chatId, @RequestP
        return chatService.chat(chatId, prompt);
    }
}
```

驗收成果

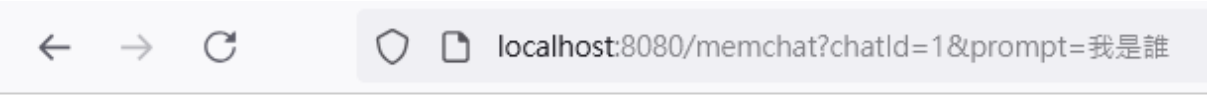
測試1:

Chat ID = 1

提問1: 我是凱文大叔

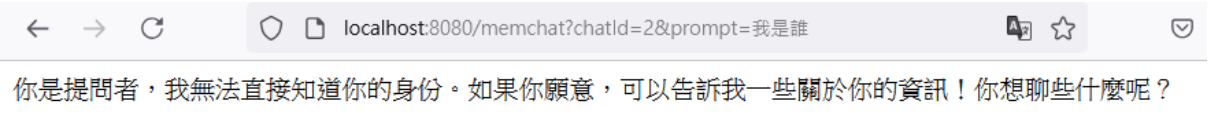


提問2: 我是誰



Chat ID = 2

提問3: 我是誰

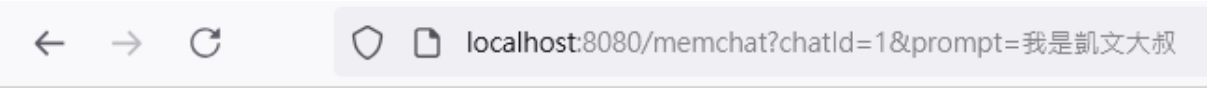


可以看出在 Chat ID = 1 時，能記住我是凱文大叔，一但切到 Chat ID = 2 又是另一段記憶

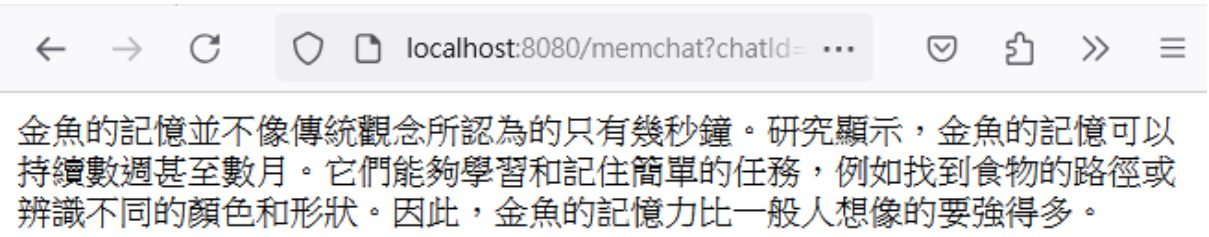
測試2:

這次將記憶數量改成2 `.messages(chatMemory.get(chatId, 2))` ，確認問第三次時能否有之前的記憶

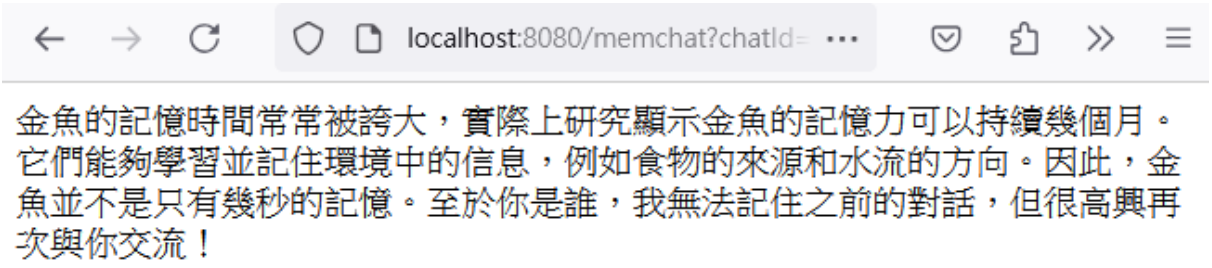
提問1: 我是凱文大叔



提問2: 金魚的記憶有幾秒?



提問3: 還記得我是誰嗎?



看起來記憶上限的參數也有作用

通常歷史訊息的數量可設為 20~30 左右，超過的話除了對 AI 的回答沒幫助外，也容易超過每次傳訊的 Token 限制，當然越多 Token 花的錢也越多，有效控制歷史訊息數量就非常重要了

回顧:

今日學到甚麼?

1. Spring AI 提供的 InMemoryChatMemory 如何實作
2. InMemoryChatMemory 的紀錄與取用方式
3. 測試不同 Chat ID 與超過上限的對話結果

程式碼下載: https://github.com/kevintsai1202/SpringBoot-AI-Day18.git