# Day20 - 自定義Advisor

延續昨天的 Advisor，RequestResponseAdvisor 的設計方式就如同 Filter Chain，每個 Chain 節點都會回傳 Request 與 Response

## 原始碼說明

```
public interface RequestResponseAdvisor {
    default AdvisedRequest adviseRequest(AdvisedRequest reque
        return request;
    }
    default ChatResponse adviseResponse(ChatResponse response
        return response;
    }
    default Flux<ChatResponse> adviseResponse(Flux<ChatRespon
        return fluxResponse;
    }
}
```

要實作 Advisor 的內容˙其實非常簡單，主要分成 Request 及 Response，Response 因為有 call 及 stream 流式呼叫，所以提供兩個方法

- AdvisedRequest request: 基本上要送給 AI 的所有內容都可在這找到，下面就是 AdvisedRequest 帶的參數

```
public record AdvisedRequest(ChatModel chatModel, String user
        List<Media> media, List<String> functionNames, List<F
        Map<String, Object> userParams, Map<String, Object> s
        Map<String, Object> advisorParams)
```

- ChatResponse response: AI 回應的訊息，包含 MetaData 以及 回應的內容

```
    private final ChatResponseMetadata chatResponseMetadata;
    private final List<Generation> generations;
```

- Map<String, Object> context: 貫穿所有 Advisor 的上下文參數，要讓不同 Advisor 可以共用 request / response 以外的參數，就能透過 context 傳遞

# 程式實作

## 目標: 寫一隻 Log Advisor，記錄傳送訊息以及使用 Token 數

首先實作 TokenUsageLogAdvistor 類別

```
@Slf4j
public class TokenUsageLogAdvistor implements RequestResponseA
    @Override
    public AdvisedRequest adviseRequest(AdvisedRequest reques
        log.info("Chat ID:{} User Message:{}",context.get("ch
        return RequestResponseAdvisor.super.adviseRequest(req
    }
    @Override
    public ChatResponse adviseResponse(ChatResponse response,
        log.info("Chat ID:{} Assistant Message:{}",context.ge
        log.info("PromptTokens:{}",response.getMetadata().getU
        log.info("GenerationTokens:{}",response.getMetadata()
        log.info("TotalTokens:{}",response.getMetadata().getU
        return RequestResponseAdvisor.super.adviseResponse(re
    }
}
```

在 ChatService 中加入增強器

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

    public String chat(String chatId, String userMessage) {
        return this.chatClient.prompt()
        .advisors( new MessageChatMemoryAdvisor(chatMemory, c
                  new TokenUsageLogAdvistor())
        .advisors(a -> {a.param("chatId", chatId);
                                        a.param("lastN", 30);
        .user(userMessage)
```

```
    .call().content();
    }
}
```

## 程式碼重點說明

- chatId、lastN 並不包含在 request 中，若要讓 log advisor 取得只能透過 context 傳送，傳送參數的程式碼如下

```
.advisors(a -> {a.param("chatId", chatId);
                            a.param("lastN", 30);
                        })
```

- context 是一個 map，只要用下面方式就能取得內容

```
context.get("chatId")
```

- ChatResponse 提供 `getMetadata()` 可取得資料回傳的額外資訊
- 一般 AI 的 Token 會區分請求的 Prompt Token 與生成的 Generation Token，兩者費用不太相同
- 若要進一步控制使用者的使用額度，可以將資料存於資料庫，並在調用前檢查

## 驗收成果

```
Chat ID:1 User Message:我是凱文大叔
Chat ID:1 Assistant Message:你好，凱文大叔！很高興認識你。請問有什麼我可以幫助你的嗎？
PromptTokens:13
GenerationTokens:28
TotalTokens:41
Chat ID:1 User Message:說一個笑話
Chat ID:1 Assistant Message:當然可以！這是一個簡單的笑話：

                            為什麼數學書總是感到憂鬱？

                            因為它有太多的「問題」！

                            希望這個笑話能讓你笑一笑！如果你想聽更多，隨時告訴我！

PromptTokens:54
GenerationTokens:67
TotalTokens:121
Chat ID:1 User Message:我是誰
Chat ID:1 Assistant Message:你是凱文大叔！如果你想分享更多關於自己的事情，或者有其他問題需要討論，隨時告訴我！
PromptTokens:131
GenerationTokens:34
TotalTokens:165
Chat ID:2 User Message:我是誰
Chat ID:2 Assistant Message:你是一位正在與我互動的用戶！如果你願意，可以告訴我更多關於你自己的事情，或者你有什麼想要討論的話題嗎？
PromptTokens:9
GenerationTokens:42
TotalTokens:51
```

可以看到 log 成功輸出，Chat ID 也能夠過 context 取得，另外多了 Token 數量輸出後可以很清楚知道每次發問使用的 Token 數量