Lecture 11: Spring AI Memory Advisor

Problem with Default ChatClient

- When you ask the model a question, it replies correctly.
- If you then say *"more"*, the model doesn't understand—it forgets the earlier context.
- Reason: **LLMs are memoryless by default.** (stateless)
- Tools like ChatGPT appear to remember, but that's their own wrapper adding memory.

Introducing Advisors in Spring AI

- **Advisors** allow you to intercept or modify requests/responses to the LLM.
- You can use them for:
    - Adding memory to your chat.
    - Safeguarding or censoring respo
    - Monitoring or altering input/output.

Memory with Advisors

- To give your ChatClient memory, use:
    - MessageChatMemoryAdvisor → adds memory capability.
    - InMemoryChatMemory stores conversation in memory.
- This lets the model understand follow-up messages like "more" or "explain in one line".

Implementation:

```java
@RestController
public class OpenAIController {
    private ChatClient chatClient;

    // Create in-memory chat memory
    ChatMemory chatMemory = MessageWindowChatMemory.builder().build();

    // Build ChatClient with memory advisor
    public OpenAIController(ChatClient.Builder builder) {
        this.chatClient = builder
                .defaultAdvisors(MessageChatMemoryAdvisor.builder(chatMemory).build())
                .build();
    }

    @GetMapping("/api/{message}")
    public ResponseEntity<String> getAnswer(@PathVariable String message) {
        ChatResponse chatResponse = chatClient.prompt(message)
                .call()
                .chatResponse();

        String response = chatResponse.getResult().getOutput().getText();
        return ResponseEntity.ok(response);
    }
}
```

Key Takeaways

- **LLMs don’t remember past chats** unless you add memory.
- Spring AI provides **Advisors** to extend functionality.
- Using MessageChatMemoryAdvisor with InMemoryChatMemory allows the app to remember conversations.
- Now follow-up queries like “more” work as expected.