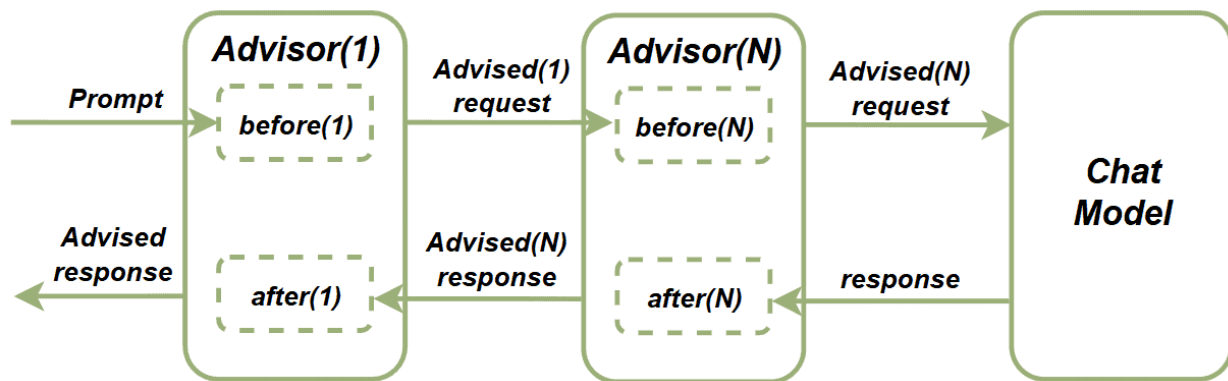


Mastering Advisors - The Interceptors of Spring AI

1. What are Advisors?

- Advisors are the middleware of Spring AI. Just like Filters in Spring Web or Aspects in Spring AOP, they sit between your code and the AI model.
- They intercept the Request (to modify the prompt) and the Response (to handle the output) without changing your core business logic.



2. Giving the AI Context

LLMs are stateless. Advisors solve this by injecting history into every prompt.

- **MessageChatMemoryAdvisor** (Short-Term Memory)
 - **How it works:** Fetches the last **N** messages from a store (e.g., in-memory or JDBC) and appends them to the prompt.
 - **Best for:** Standard chatbots where recent context matters most.
 - **Config:** Use **MessageWindowChatMemory** to limit the sliding window (e.g., last 10 messages) to control costs.

```
MessageChatMemoryAdvisor.builder(chatMemory)
    .conversationId(userId)
    .build()
```

- **VectorStoreChatMemoryAdvisor (Long-Term Memory)**
 - **How it works:** Instead of the "last 10" messages, it searches your history for the "most relevant" messages using embeddings.
 - **Best for:** "Recall" features (e.g., "What did we agree on 3 months ago?").

```
VectorStoreChatMemoryAdvisor.builder(vectorStore)
    .conversationId(userId)
    .chatHistoryWindowSize(10) // Max results to retrieve
    .build()
```

3. The RAG Family (Giving the AI Knowledge)

- **QuestionAnswerAdvisor**
 - **How it works:** Takes the user's question, searches a Vector Store for matching documents, and injects them into the system prompt as "Context".
 - **Best for:** "Chat with your Data" applications.
 - **Key Config:** Use `.filterExpression()` to ensure users only search their own documents.

```
QuestionAnswerAdvisor.builder(vectorStore)
    .searchRequest(SearchRequest.builder()
        .similarityThreshold(0.8) // Only high confidence
        .filterExpression(b.eq("owner", userId)).build())
```

```
.build())  
.build()
```

4. The Safety & Governance Family (Keeping the AI in Check)

- **SafeGuardAdvisor**
 - **How it works:** Checks the input prompt (and optionally the response) against a list of forbidden words. If a match is found, it throws an exception or blocks the request.
 - **Best for:** Preventing basic jailbreaks or usage of sensitive internal terms.

```
new SafeGuardAdvisor(List.of("password", "secret_key", "ignore all in  
structions"))
```

5. The Observability & Logic Family (Monitoring & reasoning)

- **SimpleLoggerAdvisor**
 - **How it works:** Logs the full Request and Response to your console.
 - **Best for:** Debugging. Seeing exactly what the RAG advisor injected into the prompt.
 - **Code:** `new SimpleLoggerAdvisor()`
- **TokenUsageAdvisor (Custom Implementation)**
 - **How it works:** A custom **CallAdvisor** that calculates the cost of the request by extracting usage metadata from the response.
 - **Best for:** Production monitoring, billing users, and cost analysis.

Summary Table: When to use what?

Scenario	Advisor
----------	---------

I want a normal chatbot.	MessageChatMemoryAdvisor
I want to search PDFs.	QuestionAnswerAdvisor
I want to remember chats from last year.	VectorStoreChatMemoryAdvisor
I need to stop users from asking about politics.	SafeGuardAdvisor
I need to debug what is being sent to OpenAI.	SimpleLoggerAdvisor
I need to track how much money I'm spending.	TokenUsageAdvisor (Custom)