Lecture 09: ChatResponse and Metadata

What We Did Earlier:

- With ChatClient, we used .prompt(message).call().content() to get a **string response**.
- That works fine, but what if we need **more than just text**?

About chatResponse:

- Instead of .content(), we can use .chatResponse().
- This returns a ChatResponse object, not just plain text.
- Why? Because it contains:
    - The AI response (text).
    - Extra information (metadata) like model name, usage, rate limits, etc.

Fetching Response Text:

```
ChatResponse chatResponse = chatClient.prompt(message)
        .call()
        .chatResponse();

String response = chatResponse
        .getResult()
        .getOutput()
        .getText();
```

- Here, getResult() → gives the generation object.
- From that → getOutput().getText() → gives the plain text answer.

Fetching Metadata:

```
System.out.println(chatResponse.getMetadata().getModel());
```

- getMetadata() returns details such as:
    - Here, the above statement prints 'gpt-4o-mini' in the console.
    - Which **model** is running (e.g., GPT-4o-mini)?
    - **Usage info** (tokens).
    - **Rate limits** and prompt details.
- Useful for **debugging, logging, and monitoring**.

Overall Code:

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

    public OpenAIController(OpenAiChatModel chatModel) {
        this.chatClient = ChatClient.create(chatModel);
    }

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

        // Print model info on console
        System.out.println(chatResponse.getMetadata().getModel());

        // Extract response text
        String response = chatResponse
                .getResult()
                .getOutput()
                .getText();

        return ResponseEntity.ok(response);
    }
}
```

Key Takeaways:

- .content() → Quick way to get only text.
- .chatResponse() → Full control: response + metadata.
- Metadata helps track **model, usage, rate limits, etc.**
- Good practice: Use ResponseEntity for proper HTTP response.
- With ChatResponse, you can now handle **both the AI's text** and **extra details** that help in monitoring and debugging.