Lecture 10: ChatClient Builder

Why ChatClient Builder?

- we created ChatClient **programmatically** using:

```
ChatClient.create(chatModel)
```

- But Spring AI also provides a **ChatClient.Builder**.
- Two options:
  - **Programmatic** (ChatClient.create(model)) → good when you want **multiple models**.
  - **Builder** (ChatClient.Builder) works best when you are using **only one model**.

How Builder Works:

- ChatClient.Builder is **auto-configured** by Spring Boot.
- It automatically detects which model (OpenAI, Anthropic, Llama, etc.) is present in your classpath.
- You don't need to pass the ChatModel explicitly.
- Simply inject the builder and call .build().

Implementation:

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

    // Using Builder instead of manual ChatModel
    public OpenAIController(ChatClient.Builder builder) {
        this.chatClient = builder.build();
    }

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

        // Print model being used
        System.out.println(chatResponse.getMetadata().getModel());

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

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

When to Use What:

- **ChatClient.create(model)**
  - Needed when you have multiple models in the same project.
  - You can control which model to use explicitly.
- **ChatClient.Builder**
  - Works when you have only one model.
  - Cleaner and simpler because Spring auto-configures it.

Key Takeaways:

- Builder makes life easier if you're sticking with one AI provider.
- For projects with multiple models → prefer the **programmatic approach**.
- Both ways are valid; choose based on your project setup.
- Use **Builder** for simplicity **when only one model** is in use.