

فراخوانی توابع با استفاده از مدل‌های هوش مصنوعی

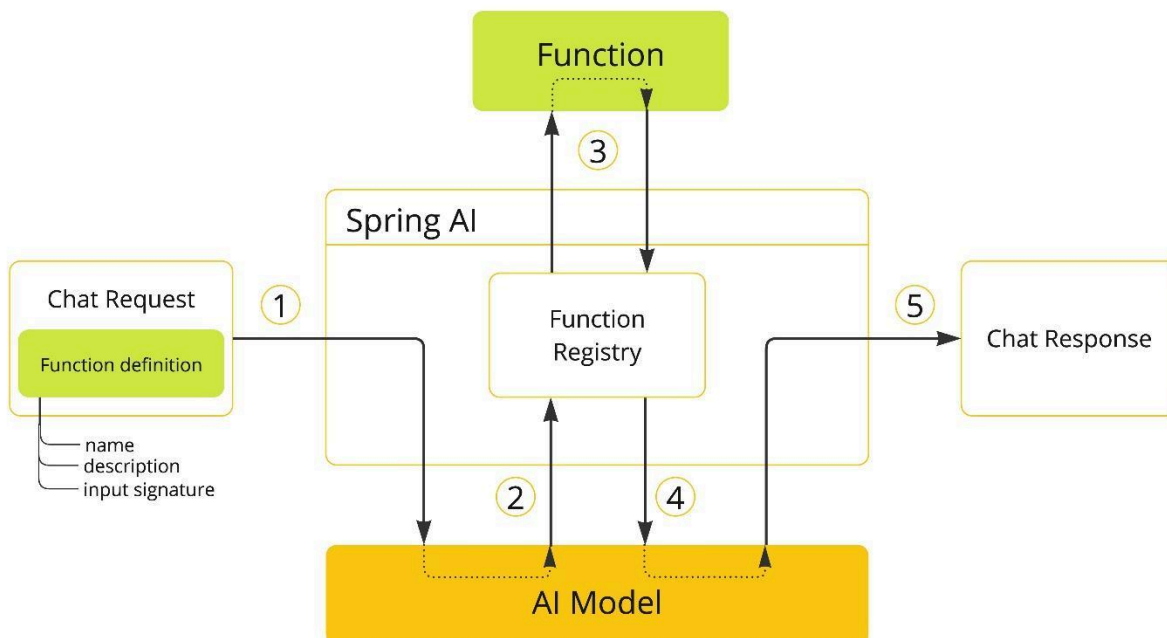
مقدمه ای بر Spring AI

سوگند صالحی، مهدی شفیعی، امیر محمد افتخار

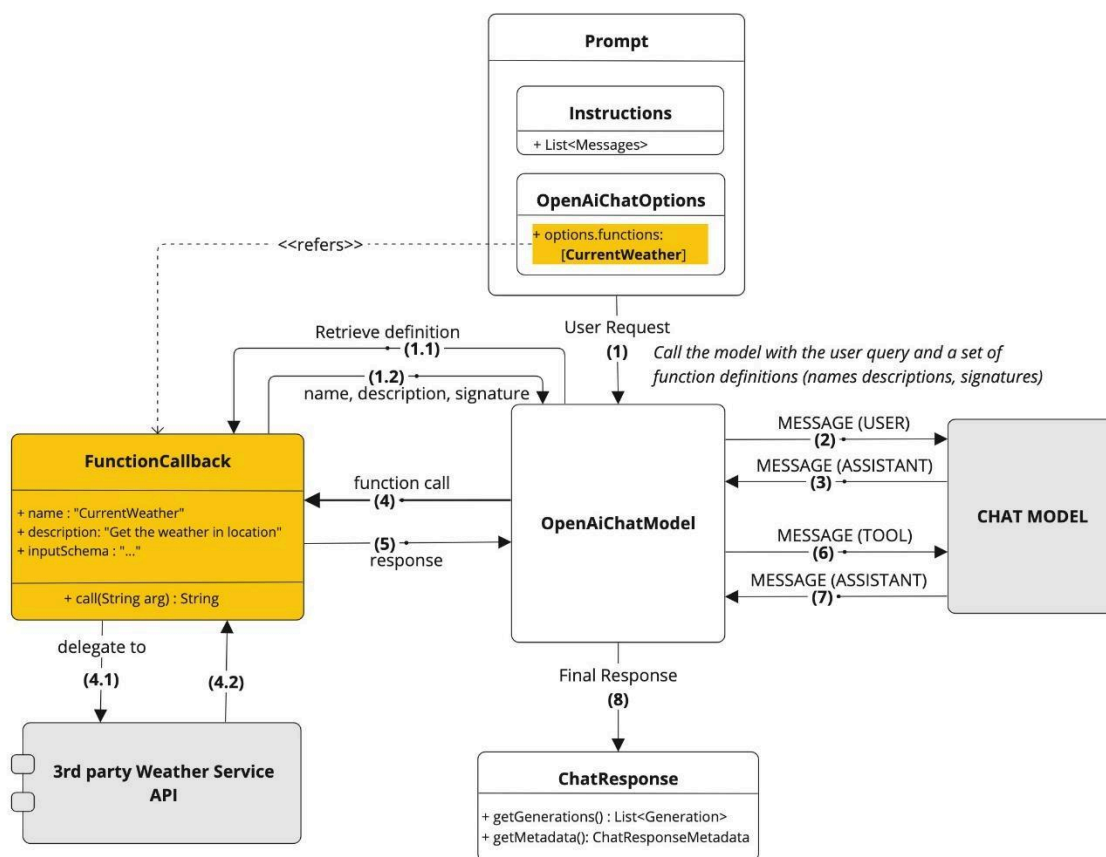
مسئله و ابزار

پشتیبانی از توابع در مدل‌های هوش مصنوعی، به مدل این اجازه را می‌دهد که ران کردن توابع سمت کلاینت را درخواست بدهد. در نتیجه آن، می‌تواند به اطلاعات لازم دسترسی داشته باشد و تسک‌هایش را به صورت داینامیک انجام دهد. در حال حاضر مدل‌های متنوعی توسط Spring AI ساپورت می‌شوند که Claude, OpenAI, Gemini, Groq, Mistral از معروفترین آن‌ها می‌باشند.

شمای کلی فراخوانی توابع با استفاده از مدل‌های هوش مصنوعی به صورت زیر می‌باشد:



و به صورت دقیق‌تر برای مثال openai داریم:



این قابلیت وجود دارد که توابع جاوای سفارشی را با **ChatClient** ثبت کنیم و مدل هوش مصنوعی به صورت هوشمند تصمیم می‌گیرد که یک آبجکت **JSON** شامل آرگومان‌های لازم برای فراخوانی یک یا چند تابع ثبت شده تولید کند. این امکان به ما اجازه می‌دهد که قابلیت‌های مدل زبانی را به ابزارها و **API**های خارجی متصل کنیم. مدل‌های هوش مصنوعی برای تشخیص زمان مناسب فراخوانی توابع آموزش دیده‌اند و به‌گونه‌ای پاسخ می‌دهند که **JSON** تولیدشده مطابق با امضای تعریف‌شده برای توابع باشد.

API به صورت مستقیم تابع را صدا نمی‌زند، بجای آن مدل یک **JSON** تولید می‌کند که می‌توان از آن استفاده کرد تا تابع در کد صدا زده شود و نتیجه‌اش به مدل برگردانده شود.

Spring AI روش‌های انعطاف‌پذیری را برای ثبت و فراخوانی توابع سفارشی ارائه می‌دهد. به طور کلی، توابع سفارشی باید شامل نام تابع، توضیحات، و امضای فراخوانی تابع (به صورت **JSON schema**) باشند تا مدل متوجه شود که تابع چه آرگومان‌هایی نیاز دارد. توضیحات به مدل کمک می‌کند زمان مناسب برای فراخوانی تابع را تشخیص دهد.

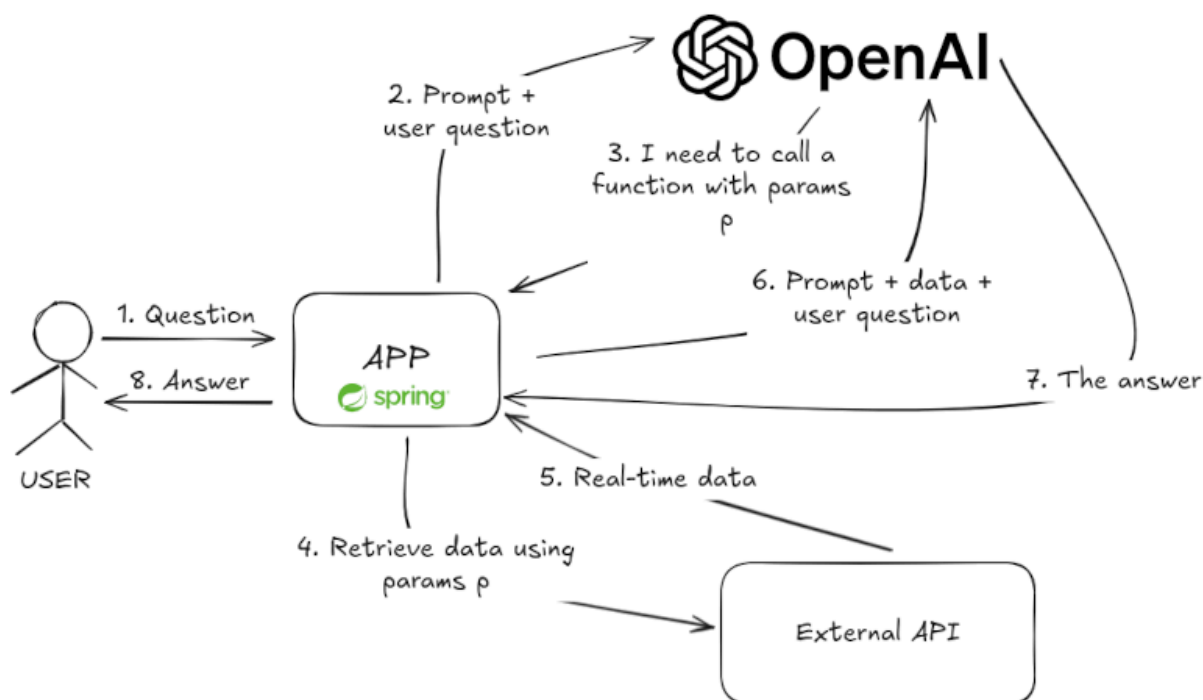
برای پاسخ به سوالاتی که مدل هوش مصنوعی اطلاعات لازم برای آن‌ها را ندارد (مثل دمای فعلی یک مکان)، می‌توان متادیتای مربوط به توابع را به مدل ارائه کرد. مدل هنگام تشخیص نیاز به اطلاعات بیشتر، یک درخواست خاص شامل نام تابع و آرگومان‌ها (به صورت JSON) ایجاد می‌کند. سپس کلاینت مسئول اجرای تابع مشخص‌شده و بازگرداندن پاسخ به مدل است. Spring AI این فرآیند را ساده می‌کند. می‌توانیم توابع خود را به‌عنوان یک Bean@ تعریف کنیم و نام آن‌ها را در گزینه‌های درخواست خود به مدل بدهیم. همچنین امکان استفاده از چندین تابع در یک درخواست وجود دارد.

برای مثال اگر می‌خواهیم یک چت بات بسازیم که با استفاده از فانکشن کال، وضعیت آب و هوا را ارائه دهد. مدل هنگام مواجهه با سوالی مثل "هوا در تهران چطور است؟"، مقدار مکان را به تابع ارسال می‌کند. این تابع می‌تواند از یک API آب و هوایی استفاده کند یا مثلاً، از یک MockWeatherService ساده با مقادیر از پیش تعیین‌شده برای مکان‌ها پاسخ دهد. نکات اضافه‌کشف شده از سرچ‌های پراکنده (: یکی از نکات کلیدی منطق تصمیم‌گیری مدل برای فراخوانی توابع است. مدل به‌طور هوشمندانه زمانی که نیاز به فراخوانی یک تابع داشته باشد، این کار را انجام می‌دهد و به جای پاسخ‌های پیش‌فرض از قبل هاردکد شده، بر اساس زمینه مکالمه، تابع مناسب را انتخاب می‌کند. همچنین، باید به مدیریت خطاها و پاسخگویی به شرایطی که داده‌ها یا خدمات خارجی در دسترس نیستند اشاره کرد، مانند زمانی که یک درخواست به API هواشناسی یا ارز دیجیتال با شکست مواجه می‌شود. در مورد مقیاس‌پذیری، اگر چندین تابع در یک درخواست فراخوانی شوند، باید به بررسی نحوه مدیریت هم‌زمانی و عملکرد پرداخته شود. نکته دیگری که اهمیت دارد، مسائل امنیتی و حریم خصوصی است؛ به‌ویژه زمانی که اطلاعات حساس به توابع ارسال می‌شود، باید از امنیت در انتقال داده‌ها اطمینان حاصل کرد اگر مدل‌ها به صورت پیش‌فرض ساپورتش نمی‌کنند. علاوه بر این، باید به محدودیت‌های توابع اشاره کرد، مانند محدودیت در نوع داده‌ها یا پیچیدگی امضای توابع، و همچنین امکان کاستومایز کردن بیشتر رفتار مدل برای تصمیم‌گیری درباره فراخوانی توابع. از دیگر نکات مهم در پیاده‌سازی، بررسی وثبت رویدادها است. باید روش‌هایی برای پیگیری و ثبت فراخوانی‌های توابع و تحلیل عملکرد آن‌ها در سیستم وجود داشته باشد (وابسته به سیستم و مدل فرق دارد). همچنین، در مورد نسخه‌بندی و به‌روزرسانی توابع باید توضیح داده شود، به‌ویژه زمانی که API‌ها یا سرویس‌های خارجی تغییر می‌کنند و بک‌اپ استاتیک ندارند. علاوه بر این، هنگام استفاده از فراخوانی‌های تابع با مدل‌های هوش مصنوعی، هیچ مفهومی از asynchronicity وجود ندارد. مدل می‌تواند پاسخی ناقص ارائه دهد و در نتیجه به کاربر این فرصت را می‌دهد تا در حالی که پردازش

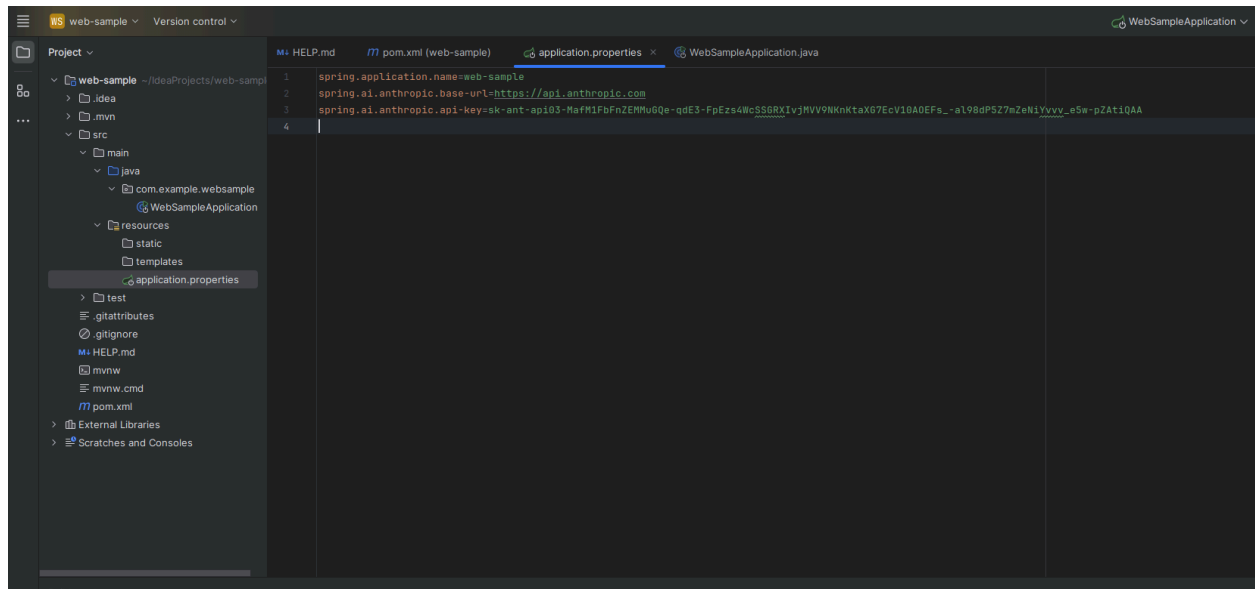
داده‌ها در پس زمینه انجام می‌شود، مکالمه را ادامه دهد. چت‌بات‌ها زمینه مکالمه را حفظ می‌کنند و دور از ذهن نیست که یک بات وسط یک مکالمه پیامی ارسال کند و مثلاً بگوید "راستی، این همان اطلاعاتی است که دو دقیقه پیش از من خواسته بودید."

مثال کاربردی

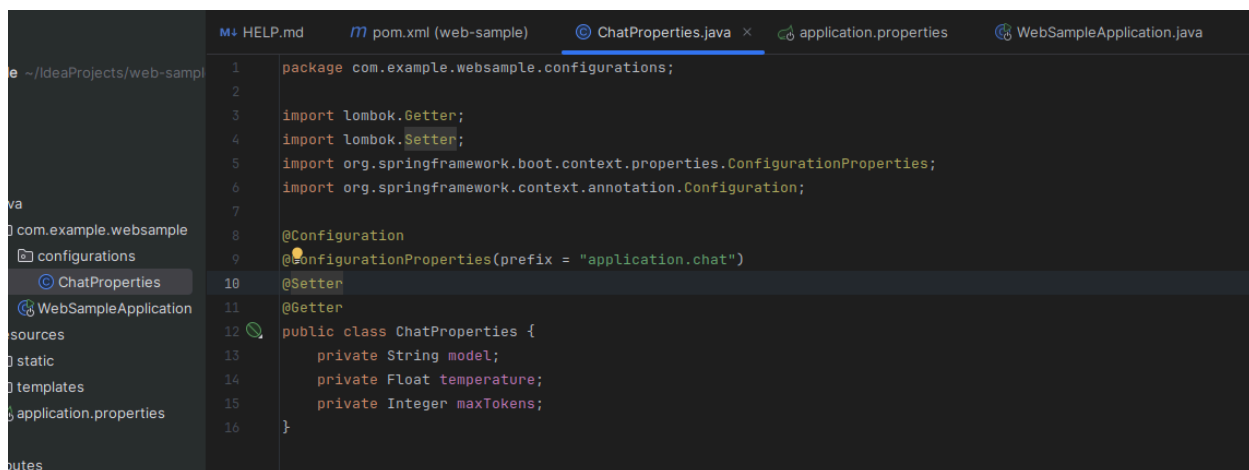
در این بخش، ما یک مثالی را بررسی می‌کنیم که در آن با استفاده از فانکشن کالینگ و API OpenAI، پاسخ کاربران را در ارتباط با cryptocurrency می‌دهیم. مدل زبانی بر اساس متن ورودی کاربر تصمیم می‌گیرد که تابع را کال کند یا نه. شمای کلی مثال به صورت زیر می‌شود:



در فایل `application.properties` به صورت زیر ویژگی‌های API را اضافه می‌کنیم:

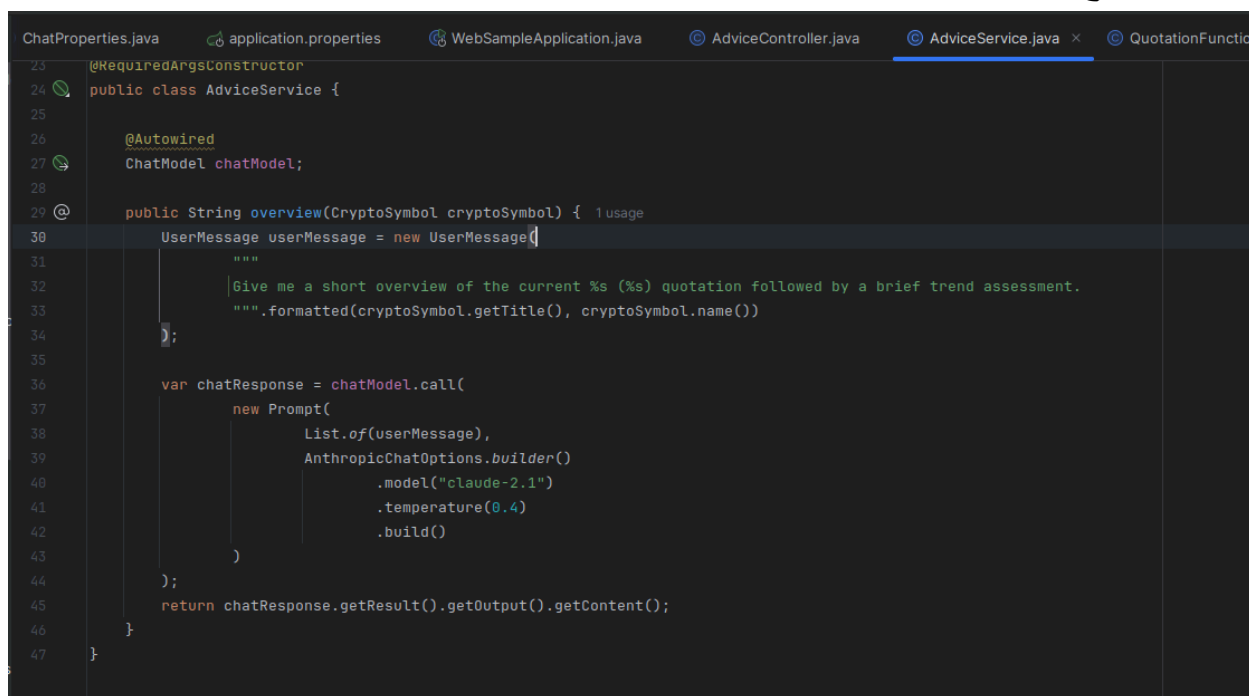


در گام بعد مطابق منبع یک Configuration درست می کنیم:



```
1 package com.example.websample.configurations;
2
3 import lombok.Getter;
4 import lombok.Setter;
5 import org.springframework.boot.context.properties.ConfigurationProperties;
6 import org.springframework.context.annotation.Configuration;
7
8 @Configuration
9 @ConfigurationProperties(prefix = "application.chat")
10 @Setter
11 @Getter
12 public class ChatProperties {
13     private String model;
14     private Float temperature;
15     private Integer maxTokens;
16 }
```

حال بخش اصلی Service است که Controller از آن استفاده می کند. Service وظیفه ی برقراری ارتباط با LLM ما را دارد. ولی در Service ما مشخص نکرده این که دقیقا چه سوالی مطرح بشود.



```
23 @RequiredArgsConstructor
24 public class AdviceService {
25
26     @Autowired
27     ChatModel chatModel;
28
29     @Service
30     public String overview(CryptoSymbol cryptoSymbol) {
31         UserMessage userMessage = new UserMessage(
32             """
33             Give me a short overview of the current %s (%s) quotation followed by a brief trend assessment.
34             """, cryptoSymbol.getTitle(), cryptoSymbol.name());
35
36         var chatResponse = chatModel.call(
37             new Prompt(
38                 List.of(userMessage),
39                 AnthropicChatOptions.builder()
40                     .model("claude-2.1")
41                     .temperature(0.4)
42                     .build()
43             ));
44         return chatResponse.getResult().getOutput().getContent();
45     }
46 }
47 }
```

برای این که بتوانیم از امکانات spring ai استفاده کنیم، از یک Bean دیگر استفاده می کنیم که به صورت یک function wrapper عمل می کند و به توجه به description و signature فانکشن سوال مربوطه را به LLM ارسال می کند. برای این که در مثال پیچیدگی الکی وجود نداشته باشد از یک Mock استفاده کرده و آن را هاردکد کردیم:

```

ChatProperties.java  application.properties  WebSampleApplication.java  AdviceController.java  AdviceService.java  QuotationFunctionMock.java x
1  package com.example.websample.app;
2
3  import java.util.function.Function;
4
5  public class QuotationFunctionMock implements Function<QuotationFunctionMock.Request, Quotation> { 3 usages
6
7      public record Request(String cryptoSymbol) {} 2 usages
8
9      @Override
10     @Override
11     public Quotation apply(Request r) {
12         return new Quotation(
13             r.cryptoSymbol(), name: "", priceUSD: 25000, priceYesterdayUSD: 26000, volumeYesterdayUSD: 1317774231.5192668
14         );
15     }
16

```

```

ChatProperties.java  application.properties  WebSampleApplication.java  AdviceController.java  AdviceService.java  QuotationFunctionMock.java  BeanProvider.java
8  import org.springframework.context.annotation.Configuration;
9
10 import java.util.stream.Collectors;
11 import java.util.stream.Stream;
12
13 @Configuration
14 public class BeanProvider {
15     @Bean
16     public FunctionCallback quotationFunction() {
17         return FunctionCallbackWrapper.builder(new QuotationFunctionMock())
18             .withName("CurrentQuotation")
19             .withDescription("""
20                 Get the current quotation of the cryptocurrency by its symbol.
21                 Available symbols are: %s
22             """).formatted(
23                 Stream.of(CryptoSymbol.values()).map(CryptoSymbol::name)
24                     .collect(Collectors.joining(delimiter: ", "))
25             ))
26         .build();
27     }
28

```

و در انتها کد را اجرا می کنیم. در نهایت یک درخواست به کمک curl به سرورمان می زنیم و مشاهده می کنیم که سوال hardcoded شده از LLM ما پرسیده شده است:

```

17s
➤ curl -X GET 'http://localhost:8080/advice/overview/BTC'
As of writing, Bitcoin is trading around $17,000 per coin. Over the past few months, Bitcoin has seen significant volatility but has remained in a range between roughly $15,500 and $17,500.

After reaching an all-time high of nearly $69,000 in November 2021, Bitcoin's price declined significantly in 2022 amid a broader crypto sell-off. However, more recently Bitcoin's price has stabilized. There was optimism the $17,000 level could act as support, but that remains to be seen.

Many analysts view Bitcoin as continuing to consolidate for now after the massive growth and subsequent sell-off in 2021-2022. There are still macroeconomic concerns like rising interest rates and recession fears that could weigh on more speculative crypto assets. However, there remains long-term optimism about Bitcoin's future as it continues seeing adoption, albeit at a slow pace. Overall the short-term trend is neutral to slightly positive, while the long-term trajectory depends greatly on if crypto gains mainstream traction.

```

لینک کد: <https://github.com/amirmohammedftekhar/web-spring-ai>

لینک ویدیو ضبط شده توسط گروه: <https://youtu.be/X7DMhaF1wBU>

- <https://johnnysn.hashnode.dev/function-calling-with-spring-ai>
- <https://www.tomaszezula.com/spring-ai-and-challenges-with-function-calling/>
- <https://www.datacamp.com/tutorial/open-ai-function-calling-tutorial>
- <https://www.geeksforgeeks.org/function-calling-and-java-integration-with-spring-ai-models/>
- <https://docs.spring.io/spring-ai/reference/api/chat/functions/openai-chat-functions.html>
- <https://docs.spring.io/spring-ai/reference/api/chat/functions/mistralai-chat-functions.html>
- <https://www.baeldung.com/spring-ai-mistral-api-function-calling>
- https://docs.spring.io/spring-ai/reference/api/chat/groq-chat.html#_function_calling
- <https://docs.spring.io/spring-ai/reference/api/chat/functions/vertexai-gemini-chat-functions.html>
- <https://docs.spring.io/spring-ai/reference/api/chat/functions/azure-open-ai-chat-functions.html>
- <https://docs.spring.io/spring-ai/reference/api/chat/functions/anthropic-chat-functions.html>
- <https://docs.spring.io/spring-ai/reference/api/functions.html>
- <https://www.baeldung.com/spring-ai-anthropics-claude-models>
- <https://docs.spring.io/spring-ai/reference/api/chat/anthropic-chat.html>