Spring AI: Beginner to Guru

Generative AI with Spring

Function Calling with OpenAI
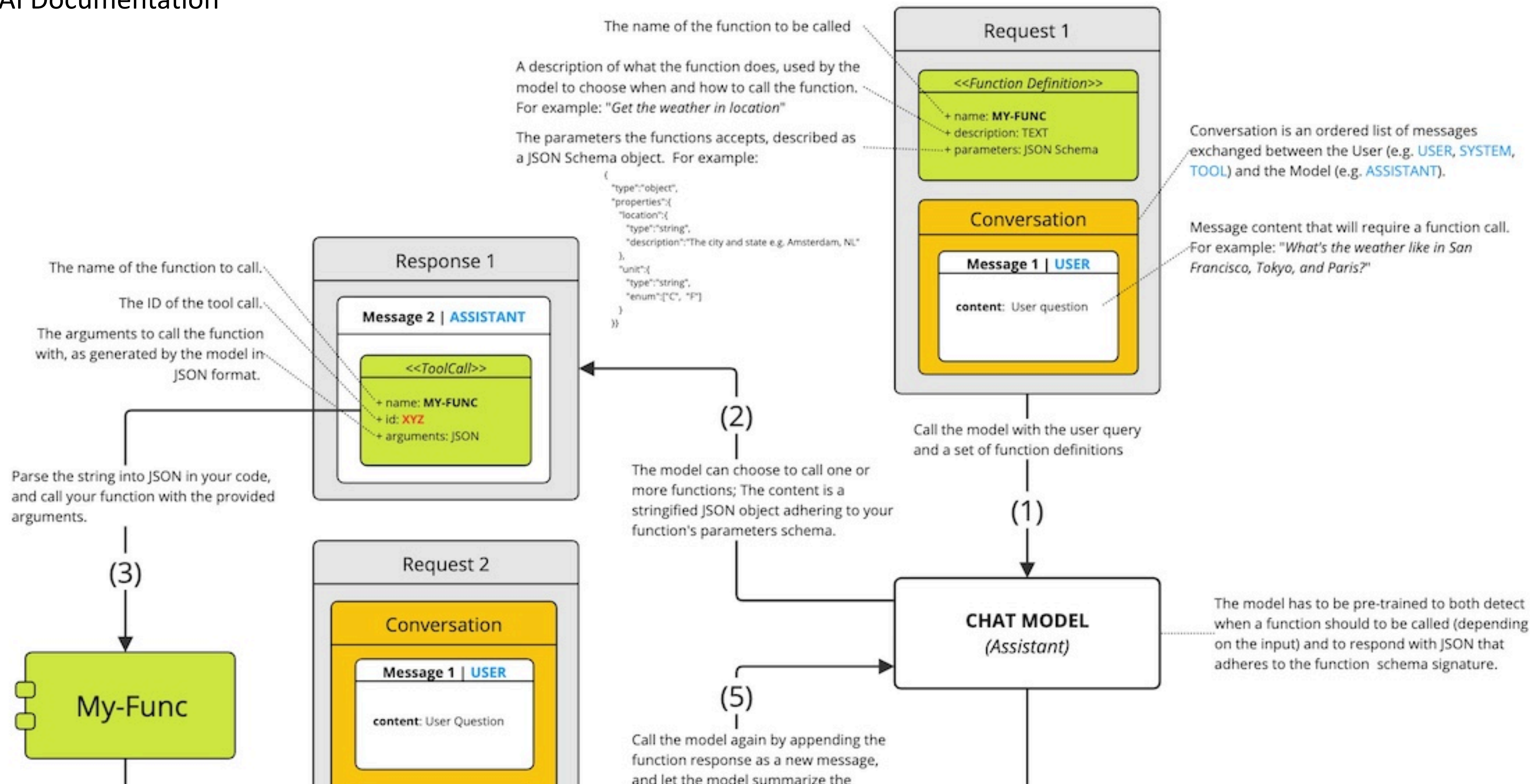
Function Calling with OpenAI

- OpenAI has the ability to invoke functions to complete tasks presented to it

- Functions typically are used to all the model to ask for additional data

- The model only 'knows' about what it's training data contains

- The model does not know things at the current time

- Depending on the model, its data can be out of date by months or even years

- Using functions, we can provide the model with the capability to get the current weather, stock prices, exchange rates, etc

The name of the function to be called

A description of what the function does, used by the model to choose when and how to call the function. For example: "Get the weather in location"

The parameters the functions accepts, described as a JSON Schema object. For example:

```
{
  "type":"object",
  "properties":{
    "location":{
      "type":"string",
      "description":"The city and state e.g. Amsterdam, NL"
    },
    "unit":{
      "type":"string",
      "enum":["C", "F"]
    }
  }
}
```

Request 1

<<Function Definition>>

+ name: **MY-FUNC**
+ description: TEXT
+ parameters: JSON Schema

Conversation is an ordered list of messages exchanged between the User (e.g. USER, SYSTEM, TOOL) and the Model (e.g. ASSISTANT).

Conversation

Message 1 | USER

content: User question

Message content that will require a function call. For example: "What's the weather like in San Francisco, Tokyo, and Paris?"

The name of the function to call.

The ID of the tool call.

The arguments to call the function with, as generated by the model in JSON format.

Response 1

Message 2 | ASSISTANT

<<ToolCall>>

+ name: **MY-FUNC**
+ id: **XYZ**
+ arguments: JSON

Parse the string into JSON in your code, and call your function with the provided arguments.

(2)

The model can choose to call one or more functions; The content is a stringified JSON object adhering to your function's parameters schema.

Call the model with the user query and a set of function definitions

(1)

(3)

My-Func

Request 2

Conversation

Message 1 | USER

content: User Question

(5)

Call the model again by appending the function response as a new message, and let the model summarize the

CHAT MODEL
(Assistant)

The model has to be pre-trained to both detect when a function should to be called (depending on the input) and to respond with JSON that adheres to the function schema signature.

```
"type":"object",
"properties":{
  "location":{
    "type":"string",
    "description":"The city and state e.g. Amsterdam, NL"
  },
  "unit":{
    "type":"string",
    "enum":["C", "F"]
  }
}
}}
```

Conversation

Message 1 | USER

content: User question

Message content that will require a function call. For example: "*What's the weather like in San Francisco, Tokyo, and Paris?*"

Call the model with the user query and a set of function definitions

(1)

Response 1

Message 2 | ASSISTANT

<<ToolCall>>

+ name: **MY-FUNC**
+ id: **XYZ**
+ arguments: JSON

The name of the function to call.

The ID of the tool call.

The arguments to call the function with, as generated by the model in JSON format.

(2)

The model can choose to call one or more functions; The content is a stringified JSON object adhering to your function's parameters schema.

Parse the string into JSON in your code, and call your function with the provided arguments.

(3)

CHAT MODEL
(Assistant)

The model has to be pre-trained to both detect when a function should to be called (depending on the input) and to respond with JSON that adheres to the function schema signature.

Request 2

Conversation

Message 1 | USER

content: User Question

My-Func

Add a new (role=Toll) message to the conversation, containing the result of one function call, with a toolCall ID referencing the id from the <<tool call>>.

(4)

Message2 | ASSISTANT

<<ToolCall>>

+ name: **MY-FUNC**
+ id: **XYZ**
+ arguments: JSON

(5)

Call the model again by appending the function response as a new message, and let the model summarize the results back to the user.

(6)

The conversation **must** contain the complete message history, including the Assistant responses.

Message 3 | TOOL

+ **content**: Function response.
+ **toolCallId** : **XYZ**

The tool Message containing the function response for the corresponding tool-call.

Response 2

Message 4 | ASSISTANT

content: Answer including the function response.

Tool call that this message is responding to. Should match the tool-call ID in Message2.

SPRING FRAMEWORK GURU