



This image shows a single sheet of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There are approximately 20 lines visible. The paper has a slight shadow on the right side, suggesting it's resting on a surface.

SPRING AI

GENERATIVE ARTIFICIAL INTELLIGENCE CON JAVA

Simone Scannapieco

Corso avanzato per Venis S.p.A, Venezia, Italia

Novembre 2025

Note

[illegible]

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

-  Simone Scannapieco

Note

This image shows a single sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

 Simone Scannapieco
 Spring AI - Corso avanzato
 Venis S.p.A, Venezia, IT
 4 / 13

[illegible]

-
- The diagram illustrates the Spring AI Tool Dispatching Flow, showing the interaction between a Chat Request, Spring AI, an AI Model, and a Tool.
- Components:**
- Chat Request:** Contains a **Tool Definition** (name, description, input schema).
 - Spring AI:** Contains the **Dispatch Tool Call Requests** component.
 - AI Model:** The model that generates the response.
 - Tool:** The external tool being dispatched.
- Flow Steps:**
- 1:** The **Chat Request** (containing the **Tool Definition**) is sent to the **Dispatch Tool Call Requests** component within **Spring AI**.
 - 2:** The **Dispatch Tool Call Requests** component sends the request to the **AI Model**.
 - 3:** The **AI Model** sends the response back to the **Dispatch Tool Call Requests** component.
 - 4:** The **Dispatch Tool Call Requests** component sends the request to the **Tool**.
 - 5:** The **Tool** sends the response back to the **Dispatch Tool Call Requests** component.
 - 6:** The **Dispatch Tool Call Requests** component sends the final **Chat Response**.

 Simone Scannapieco

Spring AI - Corso avanzato

 Venis S.p.A, Venezia, IT

5 / 13

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

-
- The diagram illustrates the Spring AI Tool Dispatching Flow, showing the interaction between a Chat Request, Spring AI, an AI Model, and a Tool.
- Components:**
- Chat Request:** Contains a **Tool Definition** (name, description, input schema).
 - Spring AI:** Contains the **Dispatch Tool Call Requests** component.
 - AI Model:** The model that processes the request.
 - Tool:** The external tool that performs the action.
- Flow Steps:**
- 1:** The **Chat Request** (containing the **Tool Definition**) is sent to the **Dispatch Tool Call Requests** component within **Spring AI**.
 - 2:** The **Dispatch Tool Call Requests** component sends the request to the **AI Model**.
 - 3:** The **AI Model** sends the request back to the **Dispatch Tool Call Requests** component.
 - 4:** The **Dispatch Tool Call Requests** component sends the request to the **Tool**.
 - 5:** The **Tool** sends the response back to the **Dispatch Tool Call Requests** component.
 - 6:** The **Dispatch Tool Call Requests** component sends the final **Chat Response**.

 Simone Scannapieco

Note

[illegible]

-
- The diagram illustrates the Spring AI Tool Dispatching Flow, showing the interaction between a Chat Request, Spring AI, an AI Model, and a Tool.
- Components:**
- Chat Request:** Contains a **Tool Definition** (name, description, input schema).
 - Spring AI:** Contains the **Dispatch Tool Call Requests** component.
 - AI Model:** The model that processes the request.
 - Tool:** The external tool that performs the action.
- Flow Steps:**
- 1:** The **Chat Request** (containing the **Tool Definition**) is sent to the **Dispatch Tool Call Requests** component within **Spring AI**.
 - 2:** The **Dispatch Tool Call Requests** component sends the request to the **AI Model**.
 - 3:** The **AI Model** sends the result back to the **Dispatch Tool Call Requests** component.
 - 4:** The **Dispatch Tool Call Requests** component sends the request to the **Tool**.
 - 5:** The **Tool** sends the result back to the **Dispatch Tool Call Requests** component.
 - 6:** The **Dispatch Tool Call Requests** component sends the final **Chat Response**.

 Simone Scannapieco

Spring AI - Corso avanzato

 Venis S.p.A, Venezia, IT

5 / 13

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

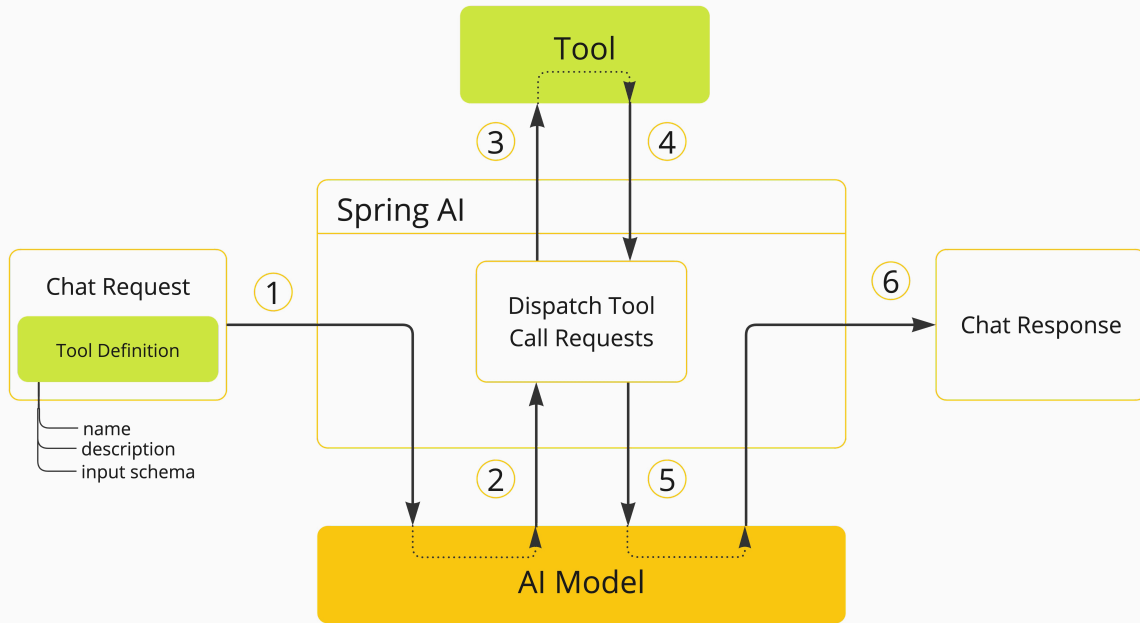
-
- The diagram illustrates the Spring AI Tool Call Flow, showing the interaction between a Chat Request, Spring AI, an AI Model, and a Tool.
- Components:**
- Chat Request:** Contains a **Tool Definition** (green box) with attributes: `name`, `description`, and `input schema`.
 - Spring AI:** Contains the **Dispatch Tool Call Requests** component (white box).
 - AI Model:** (Yellow box) Receives the request and returns a response.
 - Tool:** (Green box) Receives the request and returns a response.
- Flow Steps:**
- 1:** The **Chat Request** is received by **Spring AI**.
 - 2:** **Spring AI** dispatches the request to the **AI Model**.
 - 3:** The **AI Model** returns a response to **Spring AI**.
 - 4:** **Spring AI** dispatches the request to the **Tool**.
 - 5:** The **Tool** returns a response to **Spring AI**.
 - 6:** **Spring AI** returns the final **Chat Response**.

 Simone Scannapieco

Note

[illegible]

- 5 Il sistema invoca nuovamente il LLM con la richiesta utente precedente ma ulteriormente contestualizzata dall'*output* del *tool*



©Spring AI

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

-
- The diagram illustrates the Spring AI Tool Dispatching Architecture, showing the flow of data and control between various components:
- Chat Request** (Left): Contains a **Tool Definition** (Green box). It lists attributes: `name`, `description`, and `input schema`.
 - Spring AI** (Center): A large container box housing the **Dispatch Tool Call Requests** component (White box with yellow border).
 - Tool** (Top): A green box representing the tool interface.
 - AI Model** (Bottom): A yellow box representing the AI model.
 - Chat Response** (Right): The final output of the process.
- The flow is numbered 1 through 6:
- 1**: Arrow from **Tool Definition** to **Dispatch Tool Call Requests**.
 - 2**: Arrow from **Dispatch Tool Call Requests** to **AI Model**.
 - 3**: Arrow from **AI Model** to **Tool**.
 - 4**: Arrow from **Tool** to **Dispatch Tool Call Requests**.
 - 5**: Arrow from **Dispatch Tool Call Requests** to **AI Model**.
 - 6**: Arrow from **Dispatch Tool Call Requests** to **Chat Response**.
- Dashed arrows indicate return paths from the **AI Model** back to the **Tool** (via path 3) and from the **Tool** back to the **AI Model** (via path 5).

 Simone Scannapieco

Spring AI - Corso avanzato

 Venis S.p.A, Venezia, IT

5 / 13

Note

[illegible]

-
- The diagram illustrates the flow of a tool call request in Spring AI. It shows the interaction between a Chat Request, Spring AI, an AI Model, and a Tool.
- Chat Request:** Contains a **Tool Definition** (green box) and a **returnDirect=true** flag (blue box).
 - Spring AI:** Contains a **Dispatch Tool Call Requests** component.
 - AI Model:** A yellow box at the bottom.
 - Tool:** A green box at the top.
- The flow is numbered 1 through 5:
- 1:** The Chat Request is sent to the Spring AI Dispatch Tool Call Requests component.
 - 2:** The Spring AI Dispatch Tool Call Requests component sends the request to the AI Model.
 - 3:** The AI Model sends the response back to the Spring AI Dispatch Tool Call Requests component.
 - 4:** The Spring AI Dispatch Tool Call Requests component sends the response to the Tool.
 - 5:** The Tool sends the final Chat Response back to the Spring AI Dispatch Tool Call Requests component.

 Simone Scannapieco

Spring AI - Corso avanzato

 Venis S.p.A, Venezia, IT

6 / 13

Note

[illegible]

- 7/13

[illegible]

TOOL CALLING

METHOD AS TOOL

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

```
@Component
public class TimeTools {

    @Tool(name="getCurrentLocalTime",
        description="Ottieni l'ora corrente nel fuso orario dell'utente.")
    String getCurrentLocalTime() {
        ...
    }

    @Tool(name="getCurrentTime",
        description="Ottieni l'ora corrente nel fuso orario specificato.",
        returnDirect=true)
    public String getCurrentTime(@ToolParam(description = "Valore che rappresenta il fuso orario.") String timeZone) {
        ...
    }
}
```

- Simone Scannapieco
 Spring AI - Corso avanzato
 Venis S.p.A, Venezia, IT
 8 / 13

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

```
@Component
public class TimeTools {

    @Tool(name="getCurrentLocalTime",
        description="Ottieni l'ora corrente nel fuso orario dell'utente.")
    String getCurrentLocalTime() {
        ...
    }

    @Tool(name="getCurrentTime",
        description="Ottieni l'ora corrente nel fuso orario specificato.",
        returnDirect=true)
    public String getCurrentTime(@ToolParam(description = "Valore che rappresenta il fuso orario.") String timeZone) {
        ...
    }
}
```

Processo di *method as tool*

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

Tool-aware ChatClient - approccio statico

```
@Bean
public ChatClient ollamaTimeToolsChatClient(OllamaChatModel ollamaChatModel, TimeTools timeTools) {

    ChatClient.Builder chatClientBuilder = ChatClient.builder(ollamaChatModel);

    return chatClientBuilder
        .defaultTools(timeTools)
        ...
        .build();

}
```

Tool-aware ChatClient - approccio dinamico

```
@Override
public Answer getOllamaToolLocalTimeAnswer(QuestionRequest request) {

    return new Answer(this.ollamaToolChatClient
        .prompt()
        .tools(timeTools)
        ...
        .call()
        .content());
}
```

Note

[illegible]

```
public class TimeTools {  
    String getCurrentLocalTime() {  
        ...  
    }  
}
```

```
Method method = MethodToolCallbackReflectionUtils.findMethod(TimeTools.class, "getCurrentLocalTime");
ToolCallback toolCallback = MethodToolCallback.builder()
    .toolDefinition(ToolDefinitions.builder(method)
        .description("Ottieni l'ora corrente nel fuso orario dell'utente.")
        .build())
    .toolMethod(method)
    .toolObject(new TimeTools())
    .build();
```

- 10 / 13

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

-  Simone Scannapieco

This image shows a single sheet of white paper with horizontal blue ruling lines. The lines are evenly spaced and run across the width of the page. There are no margins, text, or other markings on the paper.

TOOL CALLING

FUNCTION AS TOOL

Note

This image shows a blank sheet of white paper with horizontal ruling lines. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings on the paper.

Definizione di *tool* come @Bean

```
@Configuration(proxyBeanMethods = false)
class WeatherTools {

    @Bean
    @Description("Get the weather in location")
    Function<WeatherRequest, WeatherResponse> currentWeather() {
        return weatherService;
    }

    @Bean
    @Description("Get the current temperature in a specific city")
    Function<TemperatureRequest, TemperatureResponse> cityTemperature() {
        return temperatureService;
    }
}
```

- ➔ Alternative all'approccio con annotazioni di metodo (`@Tool`)
- ➔ Il nome del `@Bean` diventa l'identificativo del `tool`
- ➔ `@Description` fornisce la descrizione del `tool`
 - ❗ Aiutare il modello a comprendere lo scopo del `tool`
- ➔ **Tipi funzionali supportati:** `Function<I,O>`, `Supplier<O>`, `Consumer<I>`, `BiFunction<I1,I2,O>`

Note

[illegible]

```
@Configuration(proxyBeanMethods = false)
class WeatherTools {

    @Bean
    @Description("Get the weather in location")
    Function<WeatherRequest, WeatherResponse> currentWeather() {
        return weatherService;
    }

    @Bean
    @Description("Get the current temperature in a specific city")
    Function<TemperatureRequest, TemperatureResponse> cityTemperature() {
        return temperatureService;
    }
}
```

- ✗ Tipi primitivi
- ✗ Optional
- ✗ Collezioni (List, Map, Array, Set)
- ✗ Tipi asincroni e reattivi

Note

[illegible]

```
// Utilizzo dei tool definiti come @Bean
ChatClient.create(chatModel)
    .prompt("What's the weather like in Copenhagen?")
    .toolNames("currentWeather") // riferimento al nome del @Bean
    .call()
    .content();

// Oppure con tool di default condivisi
ChatClient.Builder builder = ChatClient.builder(chatModel)
    .defaultToolNames("currentWeather", "cityTemperature");

ChatClient client = builder.build();
client.prompt("What's the weather in Rome?").call().content();
```

- ➔ Riferimento al *tool* tramite *nome del bean* con `toolNames()`
- ➔ Possibilità di definire *tool di default* condivisi tra più richieste con `defaultToolNames()`
 - ⚠ Attenzione alle *implicazioni di sicurezza* con *tool di default*
- ➔ **Generazione automatica dello schema** degli input
- ➔ `@ToolParam` permette personalizzazione per descrizioni e stato *required* anche per tipi nested

Note

[illegible]