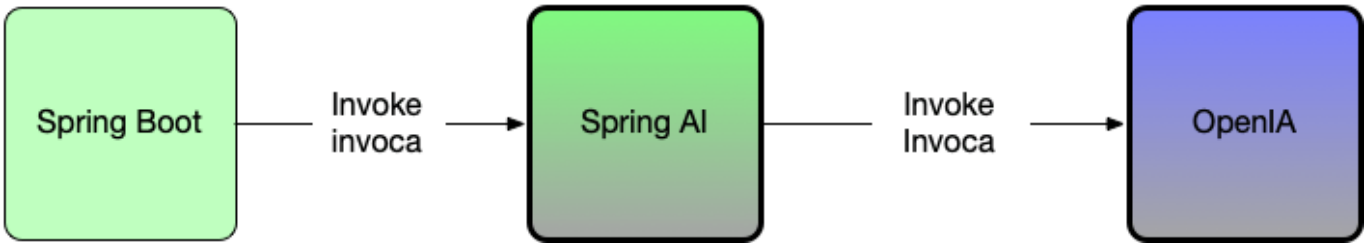
¿Que es Spring AI? . Poco a poco todos estamos empezando a oir a hablar de Spring AI como uno de los nuevos frameworks de Spring y indudablemente el más destacado. ¿Por qué? . Pues porque nos permitirá integrar herramientas y capacidades de IA dentro de nuestras aplicaciones . ¿Esto que quiere decir ? . Pues que el paradigma clásico de construcción de aplicaciones va a ser modificado y complementado por Spring AI . Este framework nos permite hacer llamadas a ChatGPT y soluciones similares e integrar sus respuestas dentro de nuestras soluciones.



Configuración y starters

¿Que necesitaremos para integrar Spring AI dentro de nuestra solución ? . Pues como siempre lo primero que necesitaremos es utilizar un Starter de AI en este caso el de OpenAI para configurar el acceso a las capacidades de IA

```
<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-
web</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework.ai</groupId>
			<artifactId>spring-ai-starter-model-
openai</artifactId>
		</dependency>

		<dependency>
```

```
                        <groupId>org.springframework.boot</groupId>
                        <artifactId>spring-boot-starter-
test</artifactId>
                        <scope>test</scope>
                </dependency>
        </dependencies>
```

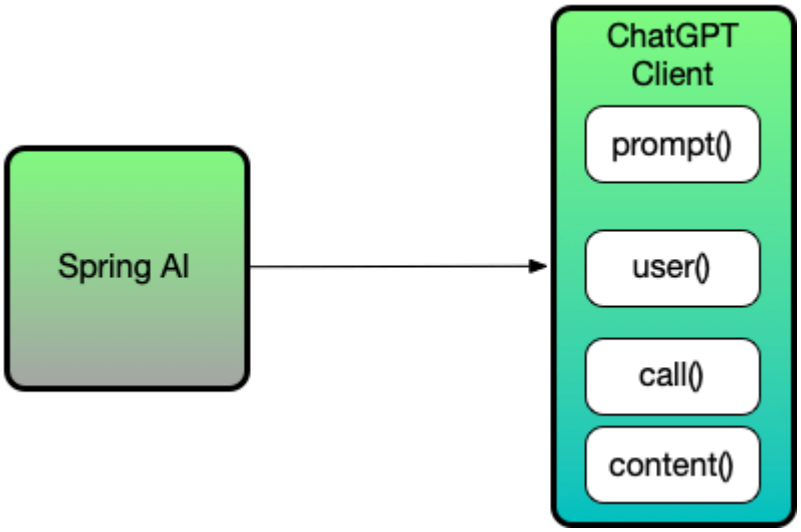
Como podemos ver tendremos que inicializar el starter de model-openai para poder tener acceso a las clases fundamentales de Spring IA que cargan las abstracciones de IA. Una vez hecho esto nos queda modificar el fichero de application.properties y añadir un pequeño grupo de propiedades entra las que destaca un api key que tienes que configurar en la web de open api.

```
spring.application.name=ia1
spring.ai.openai.api-key=sk-
proj-9fDQ59q94vV46aQ3IjzLrw4JuVrpU-3bHQqkncBScb5i_HbUnVcL9fcrz5UsieBed
rU47UJkdQT3BlbkFJJth84sZ0QIN_mRwWXTbKa-
KbGpqH61gSyKniWGUE_Dk8kUVvrcWUfjPeU7HNktyaJ-LZ
spring.ai.openai.model=gpt-4.1
spring.ai.openai.chat.options.max-tokens=1000
spring.ai.openai.chat.options.temperature=0.2
```

En este caso hemos usado un modelo clasico de chatgpt 4.1 asignando un maximo numero de tokens a procesar de 1000 y una temperatura de 0.2 que hace al modelo más estricto en cuanto a sus respuestas.

Spring AI y ChatClient

Es momento de configurar un servicio con ChatClient una clase que se encarga de abstraernos del modelo de IA exacto con el que trabajemos generando un conjunto de abstracciones que nos permiten comuninarnos de una forma efectiva.



Usamos inyección de dependencia a través de constructores para inicializarlo . Hecho esto luego dispondremos de varios métodos:

- prompt(): nos solicita la información relacionada con el prompt . Hay varias opciones en este caso usamos user().
- user(): Este método contiene como parametro el mensaje del usuario con la tarea que queremos realizar
- call(): Invoca al sistema de IA que tengamos detrás en este caso OpenIA
- content(): Recupera la respuesta del sistema

```
package com.arquitecturajava.ia1;

import org.springframework.ai.chat.client.ChatClient;

import org.springframework.ai.openai.OpenAiChatOptions;
import org.springframework.ai.openai.api.ResponseFormat;
import org.springframework.stereotype.Service;

@Service
public class ChatService {

    private  final ChatClient chatClient;
```

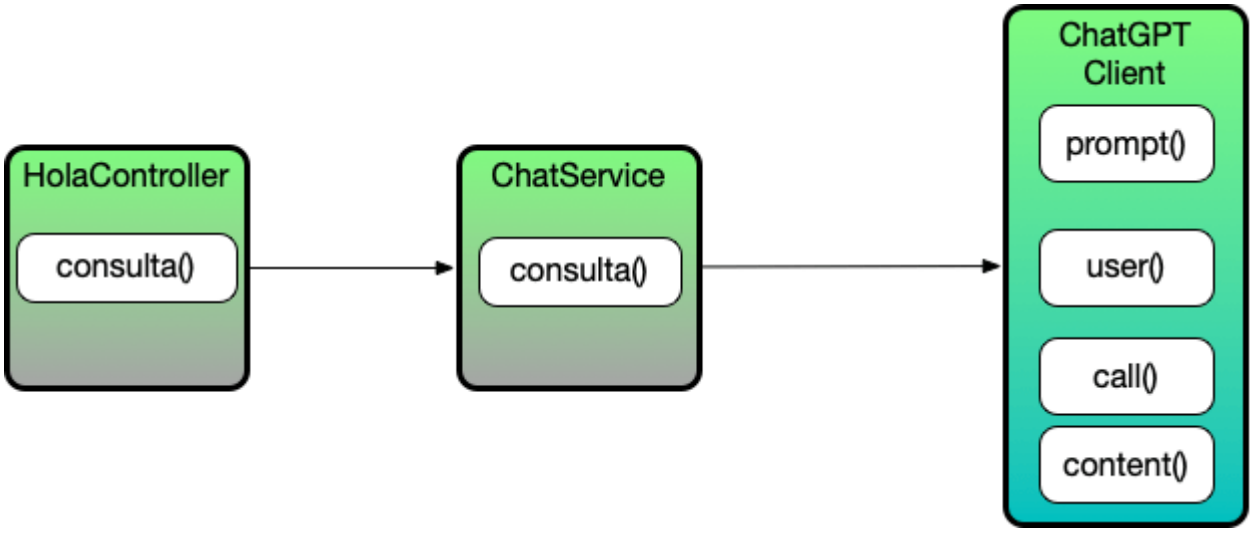
```
    public ChatService(ChatClient.Builder builder) {

    this.chatClient = builder.build();
    }

    public String consulta(String mensaje) {

        return chatClient.prompt().user(mensaje).call().content();
    }


}
```

Nos queda definir un servicio REST a través del cual podamos invocar de forma cómoda al ChatClient:

```
package com.arquitecturajava.ia1;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/chat")
public class HolaChatGPTController {


    private final ChatService chatService;
    public HolaChatGPTController(ChatService chatService) {

        this.chatService = chatService;
    }
```

```
    @PostMapping("/consulta")
    public String consultaPost(@RequestBody UserInput input) {

        return chatService.consulta(input.consulta());

    }
}
```

Usamos un sencillo postMapping y hacemos que el servicio REST a través de la url /chat/consulta delege en el servicio de Spring y en el ChatClient



Esto nos permitirá acceder desde cualquier sitio a las capacidades de ChatGPT . Recordemos que tendremos que tener creditos de tokens disponibles aparte de nuestra habitual cuenta profesional.

Webinar Spring AI Gratuito

Apuntate a mi Webinar Gratuito de Spring IA y pierde el miedo a usar esta tecnología aprender a construir los primeros ejemplos de Spring AI desde cero :).

Apuntate al Webinar Spring AI Gratuito !!!

Otros artículos relacionados

- Spring Boot Starter
- Spring Boot Actuator
- ¿Spring Boot Que es?