# Spring AI
# A Inteligência Artificial ao Alcance do Desenvolvedor Spring

Spring AI é um projeto guarda-chuva da Spring que visa simplificar a integração de funcionalidades de Inteligência Artificial (IA) em aplicações Java, especialmente aquelas construídas com o ecossistema Spring.

Ele oferece uma abstração sobre diversas plataformas e modelos de IA, permitindo que os desenvolvedores interajam com essas tecnologias de forma consistente e idiomática, sem a necessidade de se aprofundar nos detalhes específicos de cada implementação.
História do Surgimento e Ideologia:

O surgimento do Spring AI está intrinsecamente ligado à crescente democratização da Inteligência Artificial e à proliferação de modelos de linguagem grandes (LLMs) e outros serviços de IA acessíveis via APIs.

A equipe do Spring percebeu a oportunidade de aplicar sua filosofia central de simplificação e abstração para facilitar a adoção dessas tecnologias pelos desenvolvedores Java.

A ideologia por trás do Spring AI pode ser resumida em alguns pontos chave:

- **Simplicidade**: Oferecer uma API intuitiva e consistente para interagir com diferentes provedores de IA, ocultando a complexidade das integrações individuais.
- **Abstração**: Permitir que os desenvolvedores troquem entre diferentes modelos e provedores de IA com o mínimo de alterações de código, promovendo a flexibilidade e a portabilidade.
- **Integração com o Ecossistema Spring:** Aproveitar os conceitos e as ferramentas familiares do Spring, como injeção de dependência, configuração e abstrações como RestTemplate e WebClient.
- **Foco no Desenvolvedor**: Tornar o processo de integração de IA o mais direto e produtivo possível para os desenvolvedores Spring.

Em essência, o Spring AI busca ser para a IA o que o Spring Data é para o acesso a dados: uma camada de abstração poderosa que simplifica uma tarefa complexa e a torna acessível a um público mais amplo de desenvolvedores.

## Boas Práticas:

Ao trabalhar com Spring AI, algumas boas práticas podem otimizar o desenvolvimento e a manutenção das aplicações:

- **Abstração de Provedores**: Evite acoplamentos fortes a provedores de IA específicos. Utilize as abstrações do Spring AI para facilitar a troca de provedores no futuro.
- **Configuração Centralizada:** Utilize os mecanismos de configuração do Spring (como arquivos application.properties ou application.yml) para gerenciar as chaves de API, URLs e outros parâmetros específicos dos provedores de IA.
- **Tratamento de Erros**: Implemente mecanismos robustos de tratamento de erros para lidar com falhas de comunicação com as APIs de IA, limites de taxa e outros problemas potenciais.
- **Testes**: Desenvolva testes unitários e de integração para garantir o correto funcionamento das interações com os serviços de IA. Mockar as chamadas às APIs externas pode ser útil em testes unitários.
- **Monitoramento**: Implemente mecanismos de monitoramento para acompanhar o uso das APIs de IA, identificar gargalos e diagnosticar problemas.

- **Segurança:** Gerencie as chaves de API e outras informações confidenciais de forma segura, evitando armazená-las diretamente no código. Utilize variáveis de ambiente ou cofres de segredos.
- **Design para Escalabilidade:** Considere o potencial de escalabilidade da sua aplicação ao interagir com serviços de IA, especialmente em cenários de alto volume de requisições.

## Principais Dependências:

O Spring AI é um projeto modular, e as dependências específicas que você precisará incluir em seu projeto dependerão dos provedores de IA que você deseja utilizar.

Algumas das principais dependências incluem:

- **spring-ai-core**: Contém as abstrações e interfaces principais do Spring AI.

Dependências específicas do provedor de LLM:

- **spring-ai-openai:** Para integração com a API da OpenAI (GPT-3, GPT-4, etc.).
- spring-ai-azure-openai: Para integração com a API da Azure OpenAI Service.
- **spring-ai-huggingfac**e: Para integração com modelos hospedados na Hugging Face Inference API.

## Dependências para outras funcionalidades de IA:

- **spring-ai-vectorstore-c**hroma: Para utilizar o Chroma como um banco de dados vetorial.
- **spring-ai-vectorstore-pinecone:** Para utilizar o Pinecone como um banco de dados vetorial.

Outras dependências para outros bancos de dados vetoriais e funcionalidades específicas.

- **Spring Web**: Frequentemente utilizado para construir APIs que interagem com serviços de IA.

Para incluir essas dependências em um projeto Maven, você adicionaria as seguintes entradas no arquivo pom.xml (exemplo para OpenAI):

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

Para Gradle, as dependências seriam adicionadas no arquivo build.gradle:

```
implementation 'org.springframework.ai:spring-ai-core'
implementation 'org.springframework.ai:spring-ai-openai'
implementation 'org.springframework.boot:spring-boot-starter-web'
```

# Exemplos Bem Simples:

Vamos considerar um exemplo simples de interação com um modelo de linguagem da OpenAI para gerar uma resposta a uma pergunta.

**1. Configuração:**
Primeiro, você precisará configurar a chave da API da OpenAI no seu arquivo application.properties ou application.yml:

```
spring.ai.openai.api-key=SUA_CHAVE_API_OPENAI
```

**2. Serviço Spring:**
Crie um serviço Spring para interagir com o modelo de linguagem:

```
import org.springframework.ai.client.AiClient;
import org.springframework.ai.client.prompt.PromptTemplate;
import org.springframework.stereotype.Service;

import java.util.Map;

@Service
public class ChatService {

    private final AiClient aiClient;

    public ChatService(AiClient aiClient) {
        this.aiClient = aiClient;
    }

    public String askQuestion(String question) {
        PromptTemplate promptTemplate = new PromptTemplate("Responda à seguinte pergunta: {question}");
        return aiClient.generate(promptTemplate.create(Map.of("question", question))).getGeneration().getText();
    }
}
```

**3. Controller Spring:**
Crie um controller Spring para expor um endpoint que utiliza o serviço de chat:

```
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class ChatController {

    private final ChatService chatService;

    public ChatController(ChatService chatService) {
        this.chatService = chatService;
    }

    @GetMapping("/ask")
    public String ask(@RequestParam String question) {
        return chatService.askQuestion(question);
    }
}
```

Neste exemplo, ao acessar o endpoint /ask com um parâmetro question (por exemplo, /ask?question=Qual a capital do Brasil?), a aplicação Spring AI utilizará a API da OpenAI para gerar uma resposta e retorná-la.

**Onde Aplicar Spring AI:**
Spring AI pode ser aplicado em uma vasta gama de cenários onde a integração de inteligência artificial pode agregar valor, incluindo:

- **Chatbots e Assistentes Virtuais**: Construir interfaces de conversação inteligentes para suporte ao cliente, vendas ou informações.
- **Geração de Conteúdo:** Automatizar a criação de textos, e-mails, posts de redes sociais, resumos e outros tipos de conteúdo.
- **Análise de Sentimento:** Analisar textos para determinar a polaridade (positiva, negativa, neutra) de opiniões e comentários.
- **Classificação e Categorização de Texto**: Automatizar a organização e o agrupamento de documentos e informações.
- **Extração de Informação**: Identificar e extrair dados relevantes de textos não estruturados.
- **Sistemas de Recomendaçã**o: Utilizar embeddings e bancos de dados vetoriais para construir sistemas de recomendação personalizados.
- **Aumento de Produtividad**e: Integrar funcionalidades de IA em fluxos de trabalho existentes para automatizar tarefas e fornecer insights.
- **Aplicações Multimodais**: Em combinação com outras bibliotecas, pode ser usado para processar e gerar conteúdo envolvendo texto, imagens e áudio.

## Como Aplicar Spring AI:

A aplicação do Spring AI geralmente envolve os seguintes passos:

- **Adicionar as Dependências**: Inclua as dependências necessárias no seu projeto, dependendo dos provedores e funcionalidades de IA que você pretende usar.
- **Configurar as Credenciais**: Forneça as chaves de API e outras configurações necessárias para autenticar e acessar os serviços de IA no seu arquivo de configuração do Spring.

**Utilizar as Abstrações do Spring AI:**

- **AiClient**: A principal interface para interagir com modelos de linguagem. Permite enviar prompts e receber gerações de texto.
- **PromptTemplate**: Facilita a criação de prompts dinâmicos com base em templates e dados.
- **ChatClient**: Uma interface especializada para interações de conversação com histórico de mensagens.
- **EmbeddingClient**: Permite gerar representações vetoriais (embeddings) de texto para tarefas como similaridade semântica e busca vetorial.
- **VectorStore**: Abstração para interagir com bancos de dados vetoriais para armazenar e pesquisar embeddings.

**Criar Serviços e Controllers**: Desenvolva serviços Spring que utilizem os clientes do Spring AI para interagir com os modelos de IA e exponha esses serviços através de controllers para serem consumidos pela sua aplicação.

**Tratar as Respostas**: Implemente a lógica para processar as respostas dos modelos de IA e integrá-las ao fluxo da sua aplicação.

Em resumo, Spring AI oferece uma maneira poderosa e flexível para desenvolvedores Spring incorporarem a inteligência artificial em suas aplicações, aproveitando a familiaridade do ecossistema Spring para simplificar a integração com uma variedade de serviços e modelos de IA.

Ao seguir as boas práticas e entender as principais abstrações, os desenvolvedores podem construir soluções inovadoras e inteligentes de forma eficiente.

***EducaCiência FastCode para a comunidade***