Integrating DeepSeek Locally (Free) with Spring Boot:

A Step-by-Step Guide

Pushkar Kumar
4 min read
·
Feb 2, 2025

Learn how to run the Open-Source model of DeepSeek locally and integrate it with Spring Boot to build powerful AI applications without relying on paid APIs

Springboot + Deepseek

AI is evolving at lightning speed, but running large language models (LLMs) often comes with cloud dependency, high costs, and API limitations. What if you could run a powerful AI model **entirely on your local machine** — for free?

In this guide, I'll walk you through integrating **DeepSeek LLM with Spring Boot**, enabling **real-time AI responses** without relying on expensive cloud-based APIs. This setup is not just cost-effective but also blazing fast, making it a **perfect playground for developers to experiment, build, and learn AI** hands-on. 🚀

Spring AI is here

Spring is also in the AI game with its new integration, **Spring AI**, making it incredibly easy to integrate AI models into Spring Boot applications. With built-in support for LLMs, vector stores, and AI-driven workflows, developers can now build intelligent applications effortlessly while maintaining scalability and flexibility.

Prerequisites

1. Java 17+
2. Spring Boot 3.2+
3. Maven/Gradle
4. IDE (IntelliJ/Eclipse)
5. DeepSeek API / DeepSeek on Local

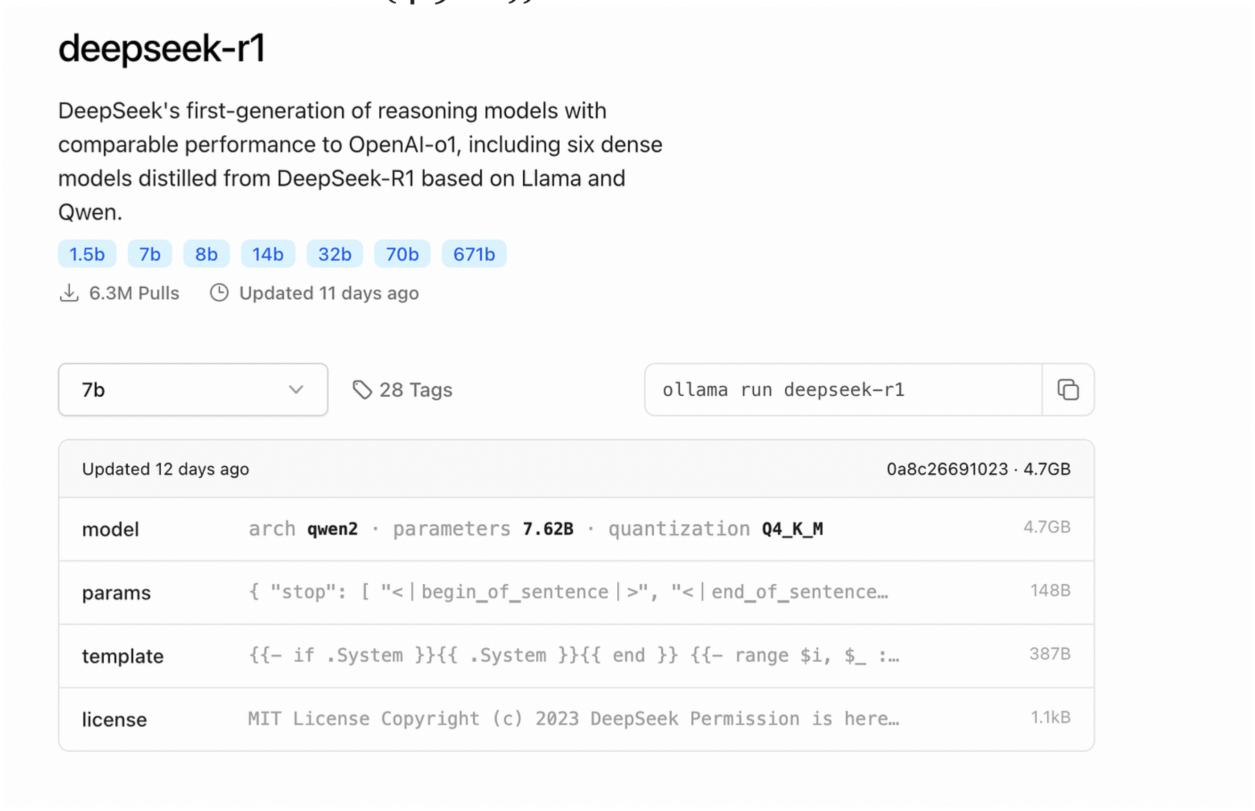
Setting up DeepSeek on Local

Step-1:- Installing Ollama

Download and install **Ollama** from the official website: https://ollama.com (You will receive an installer that you need to run to set up Ollama)

Step-2:- Installing DeepSeek LLM

- You can **download the DeepSeek LLM model** based on your choice from https://ollama.com/library/deepseek-r1
- run the command **ollama run deepseek-r1:8b** (here I have chosen 8b model (4.9GB))



deepseek-r1

DeepSeek's first-generation of reasoning models with comparable performance to OpenAI-o1, including six dense models distilled from DeepSeek-R1 based on Llama and Qwen.

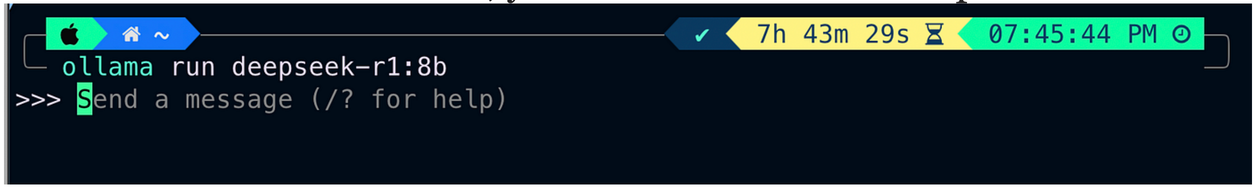
```
ollama run deepseek-r1
```

Updated 12 days ago — 0a8c26691023 · 4.7GB

model	arch **qwen2** · parameters **7.62B** · quantization **Q4_K_M**	4.7GB
params	{ "stop": ["<|begin_of_sentence|>", "<|end_of_sentence…	148B
template	{{- if .System }}{{ .System }}{{ end }} {{- range $i, $_ :…	387B
license	MIT License Copyright (c) 2023 DeepSeek Permission is here…	1.1kB

Ollama DeepSeek-r1 models

After successful installation, you will see the below output.



```
 ~                                     ✔  7h 43m 29s ⌛  07:45:44 PM ⏲
ollama run deepseek-r1:8b
>>> Send a message (/? for help)
```

Integrating DeepSeek with SpringBoot Application

Step-1:- Add Dependency in Pom.xml

```
<!-- pom.xml -->
<dependencies>

    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>

    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.13</version>
    </dependency>

    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-ollama-spring-boot-starter</artifactId>
    </dependency>

</dependencies>
```

Step-2:- Setup Application Yaml

```
<!-- For connecting with Deepseek API -->

deepseek.api.key = your_api_key_here
deepseek.api.url = https://api.deepseek.com/v1/chat/completions

<!-- ------------------------------ -->

<!-- For connecting with Deepseek Running on Local -->

spring.ai.ollama.chat.options.model=deepseek-r1:8b

<!-- ------------------------------ -->
```

Step-3:- Setup Rest Controller to receive the requests

```
@RestController
@RequestMapping("/ai-chat")
public class AIChatController {

    private final AIChatService aiChatService;

    public AIChatController(AIChatService aiChatService) {
        this.aiChatService = aiChatService;
    }


    @GetMapping
    public Flux<String> getAIChatResponse(@RequestParam String message) {
        return aiChatService.getAIResponse(message);
    }


}
```

Step-4:- Building Service Layer to integrate AI Chat

```
@Service
@Slf4j
public class AIChatService {

    private final ChatClient chatClient;

    AIChatService(ChatClient.Builder builder){
        this.chatClient = builder
                .defaultSystem("You are an advanced AI assistant integrated into a
Spring Boot application.Ensure clarity and accuracy in responses.")
                .build();
    }

    public Flux<String> getAIResponse(String message) {
        try {
            return chatClient.prompt(message)
                    .stream()
                    .content();
        } catch (Exception e) {
            throw new RuntimeException("Error occurred while fetching AI response, ex =
" + e.getMessage());
        }
    }
}
```

Step-5:- Testing it

→ Sample Request

```
curl --location 'http://localhost:8080/ai-chat?message=What%20is%20Spring%20AI'
```

→ Sample Response

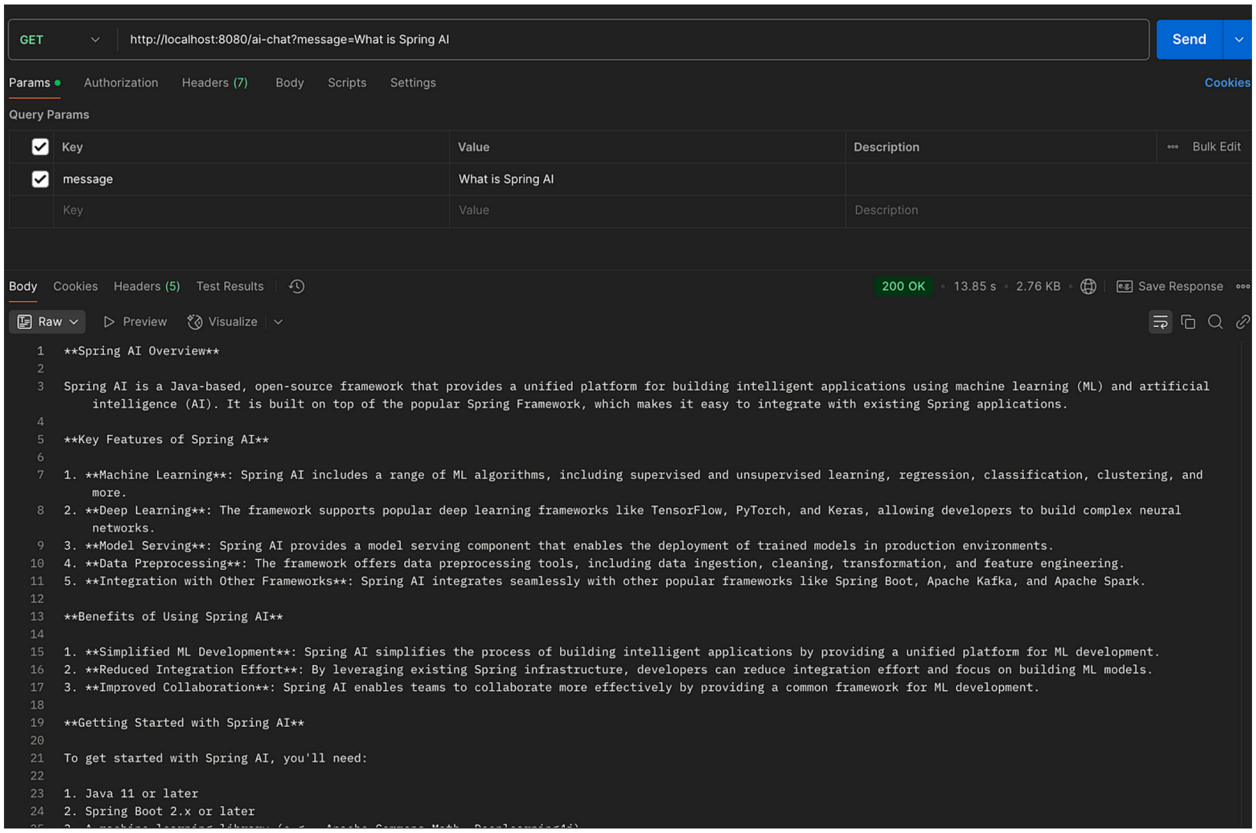
```
**Spring AI Overview**

Spring AI is a Java-based, open-source framework that provides a unified platform for
building intelligent applications using machine learning (ML) and artificial
intelligence (AI). It is built on top of the popular Spring Framework, which makes it
easy to integrate with existing Spring applications.

**Key Features of Spring AI**

1. **Machine Learning**: Spring AI includes a range of ML algorithms, including
supervised and unsupervised learning, regression, classification, clustering, and more.
2. **Deep Learning**: The framework supports popular deep learning frameworks like
TensorFlow, PyTorch, and Keras, allowing developers to build complex neural networks.
3. **Model Serving**: Spring AI provides a model serving component that enables the
deployment of trained models in production environments.
4. **Data Preprocessing**: The framework offers data preprocessing tools, including data
ingestion, cleaning, transformation, and feature engineering.
5. **Integration with Other Frameworks**: Spring AI integrates seamlessly with other
popular frameworks like Spring Boot, Apache Kafka, and Apache Spark.

**Benefits of Using Spring AI**
1. **Simplified ML Development**: Spring AI simplifies the process of building
intelligent applications by providing a unified platform for ML development.
2. **Reduced Integration Effort**: By leveraging existing Spring infrastructure,
developers can reduce integration effort and focus on building ML models.
3. **Improved Collaboration**: Spring AI enables teams to collaborate more effectively
by providing a common framework for ML development.
```



```
http://localhost:8080/ai-chat?message=What is Spring AI
```

Key	Value	Description
message	What is Spring AI	

```
**Spring AI Overview**

Spring AI is a Java-based, open-source framework that provides a unified platform for building intelligent applications using machine learning (ML) and artificial
    intelligence (AI). It is built on top of the popular Spring Framework, which makes it easy to integrate with existing Spring applications.

**Key Features of Spring AI**

1. **Machine Learning**: Spring AI includes a range of ML algorithms, including supervised and unsupervised learning, regression, classification, clustering, and
    more.
2. **Deep Learning**: The framework supports popular deep learning frameworks like TensorFlow, PyTorch, and Keras, allowing developers to build complex neural
    networks.
3. **Model Serving**: Spring AI provides a model serving component that enables the deployment of trained models in production environments.
4. **Data Preprocessing**: The framework offers data preprocessing tools, including data ingestion, cleaning, transformation, and feature engineering.
5. **Integration with Other Frameworks**: Spring AI integrates seamlessly with other popular frameworks like Spring Boot, Apache Kafka, and Apache Spark.

**Benefits of Using Spring AI**

1. **Simplified ML Development**: Spring AI simplifies the process of building intelligent applications by providing a unified platform for ML development.
2. **Reduced Integration Effort**: By leveraging existing Spring infrastructure, developers can reduce integration effort and focus on building ML models.
3. **Improved Collaboration**: Spring AI enables teams to collaborate more effectively by providing a common framework for ML development.

**Getting Started with Spring AI**

To get started with Spring AI, you'll need:

1. Java 11 or later
2. Spring Boot 2.x or later
3. A machine learning library (e.g., Apache Commons Math, Deeplearning4j)
```

GitHub Repo:- https://github.com/pushkar910/weatherAI

Conclusion

In conclusion, integrating Free DeepSeek locally with a Spring Boot application not only enhances your project's search capabilities but also provides an efficient and scalable solution for handling complex queries. By following the step-by-step guide, you can successfully implement a local AI chatbot that ensures faster, more relevant results, while maintaining full control over your data.

With Spring Boot's robust framework and the power of Free DeepSeek, you can unlock new possibilities for your application's performance and user experience. I hope this guide has helped you understand the integration process, and I encourage you to explore further customizations to suit your specific needs.