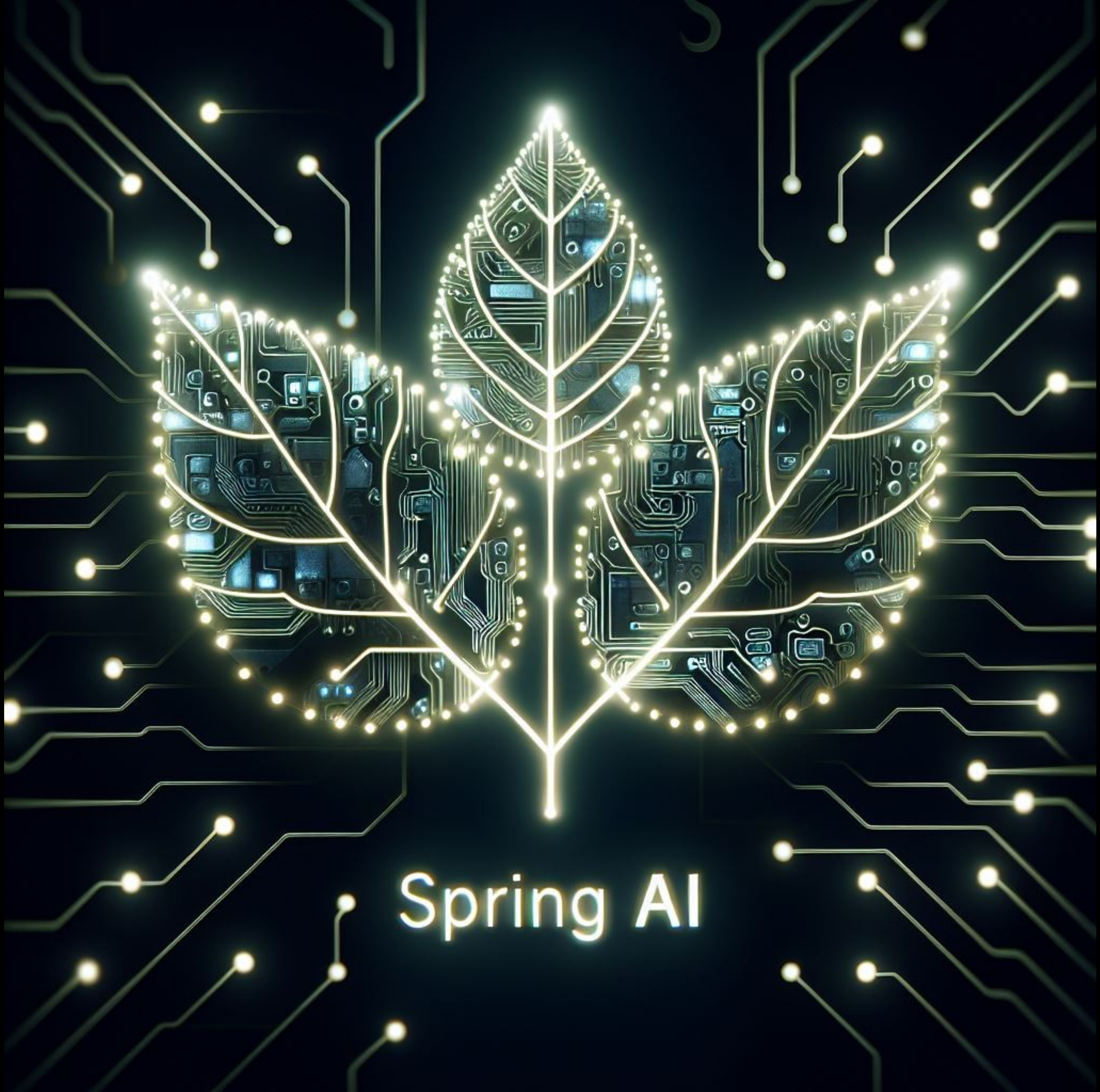
MAHARISHI INTERNATIONAL UNIVERSITY

HIGHER EDUCATION FOR HIGHER CONSCIOUSNESS


Spring AI

Overview

- Setting up PGVector
- Embedding Spring AI docs
- Retrieval Augmented Generation

Embedding and RAG

- To answer about data not trained on
  - Vectorize data and embed into a vectorstore
  - When query comes in, vectorstore is queried for related data (vectors used for closest matches)
  - Related data is added to the prompt along with instructions to base answer on it

Embedding stores data along with relations vector

Retrieval Augmented Generation

An embedding model is used for both embedding and retrieval of related data

About this code

- Spring AI is still very new, only just hit 1.0.0
  - During the process of building this I encountered a variety of bugs
  - I also saw features that are coming down the pipeline which will be very useful (not here yet)
  - I was able to make it work but things may change in the near future

My Setup

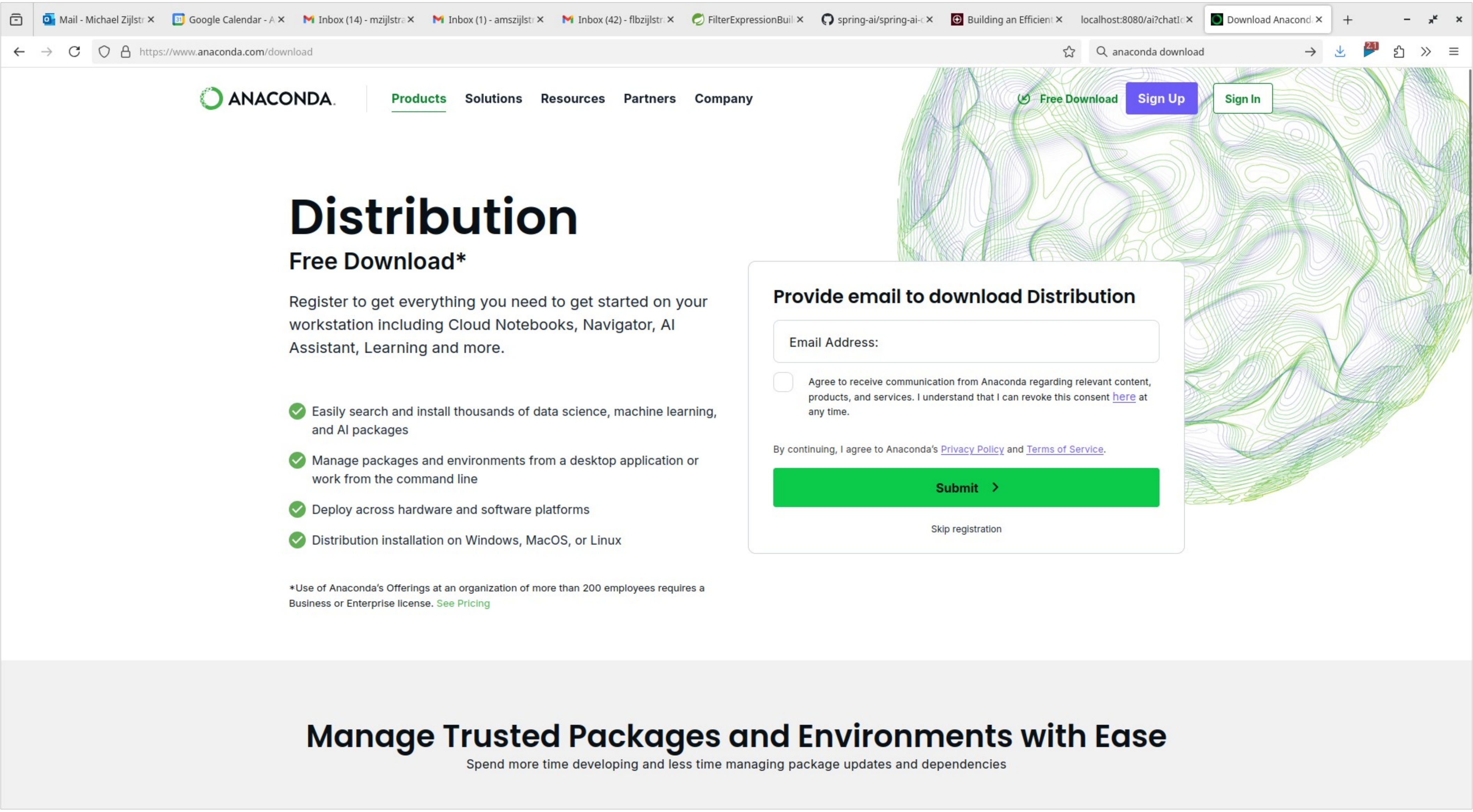
- I used Ollama running locally
  - Model for embedding: nomic-embed-text
  - Model for Chat: llama3.2

- Vector Store
  - PGVector: Postgresql with vector extension

Setting up PGVector

Anaconda

- The easiest way to get the PGVector extension under windows is through Anaconda
  - Anacoda is a cross-platform package manager (Windows/Mac OS/Linux) for AI related tools
  - Originally focused on Python AI tools, branched out to tools in any language
  - Free for students, requires a paid subscription for companies



Distribution

Free Download*

Register to get everything you need to get started on your workstation including Cloud Notebooks, Navigator, AI Assistant, Learning and more.

- Easily search and install thousands of data science, machine learning, and AI packages
- Manage packages and environments from a desktop application or work from the command line
- Deploy across hardware and software platforms
- Distribution installation on Windows, MacOS, or Linux

*Use of Anaconda's Offerings at an organization of more than 200 employees requires a Business or Enterprise license. See Pricing

Provide email to download Distribution

Email Address:

☐ Agree to receive communication from Anaconda regarding relevant content, products, and services. I understand that I can revoke this consent here at any time.

By continuing, I agree to Anaconda's Privacy Policy and Terms of Service.

Submit >

Skip registration

Manage Trusted Packages and Environments with Ease

Spend more time developing and less time managing package updates and dependencies

Install Anaconda

- For zsh users (Mac OS / Linux):
  - conda init zsh

Anaconda Environments

- To install you’ll first need an environment
  - Use the commandline conda utility
  - Easily create slightly different test environments
  - Only one environment can be active

```
conda create -n myenv
conda activate myenv
```

Install Postgresql and PGVector

```
conda install conda-forge::postgresql
conda install conda-forge::pgvector


initdb -D dbfiles
pg_ctl -D dbfiles -l logfile start
createdb embeddings
createuser --pwprompt postgres
psql -d embeddings
```

Inside the directory where you want to place the dbfiles directory

Make sure you don't already have a postgresql db running.

Will open the postgresql SQL commandline as superuser

Create vector_store table

```
CREATE EXTENSION IF NOT EXISTS vector;
CREATE EXTENSION IF NOT EXISTS hstore;
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

CREATE TABLE IF NOT EXISTS vector_store (
    id uuid DEFAULT uuid_generate_v4() PRIMARY KEY,
    content text,
    metadata json,
    embedding vector(768) -- 1536 is the default embedding dimension
);
CREATE INDEX ON vector_store USING HNSW (embedding vector_cosine_ops);
GRANT ALL ON vector_store TO postgres;
```

Nomic-embed-text embedding model wants a vector of size 768

Once you’re done you can exit with CTRL-D or \q

Embedding

Embed the Spring-AI docs

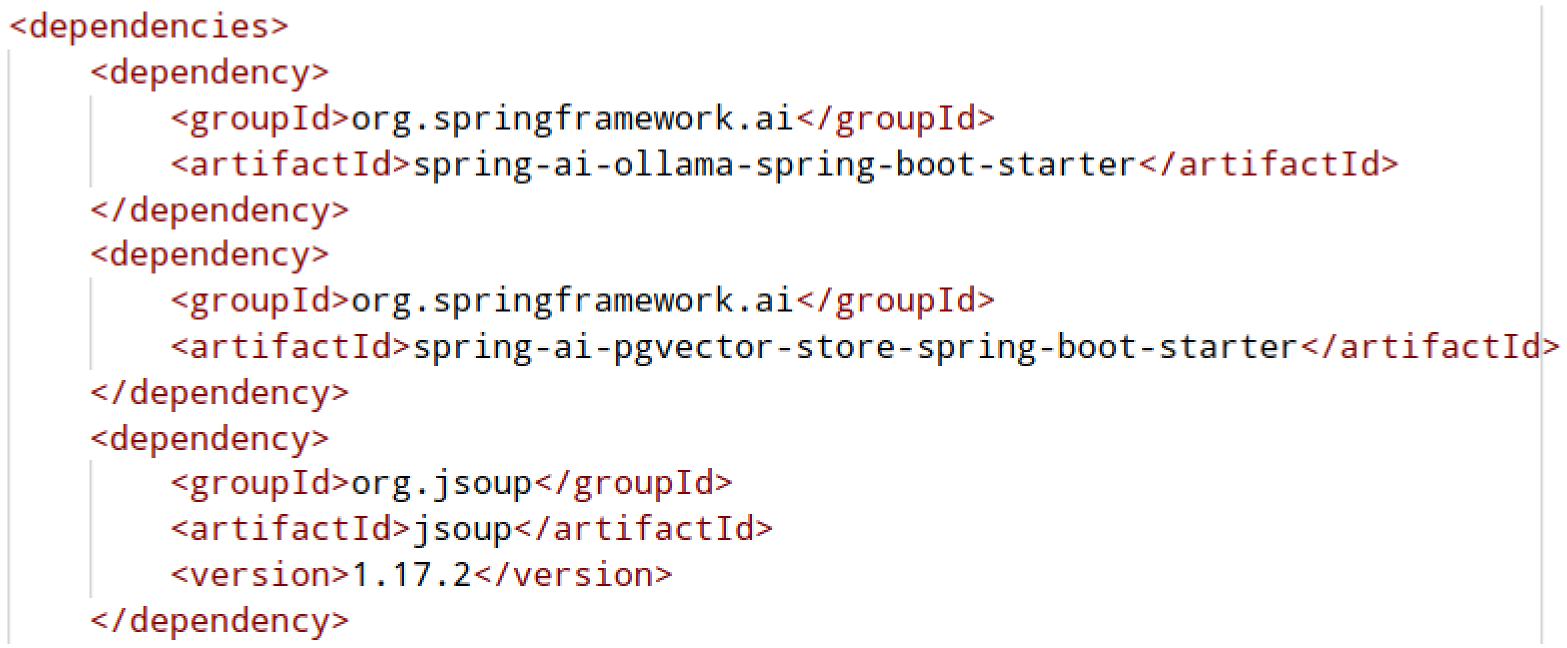
- Ollama 3.2 has no in-depth knowledge of it
- We’ll embed with nomic-embed-text
  - Currently most popular open embedding model

If you look at the Ollama site

```
ollama pull nomic-embed-text
```

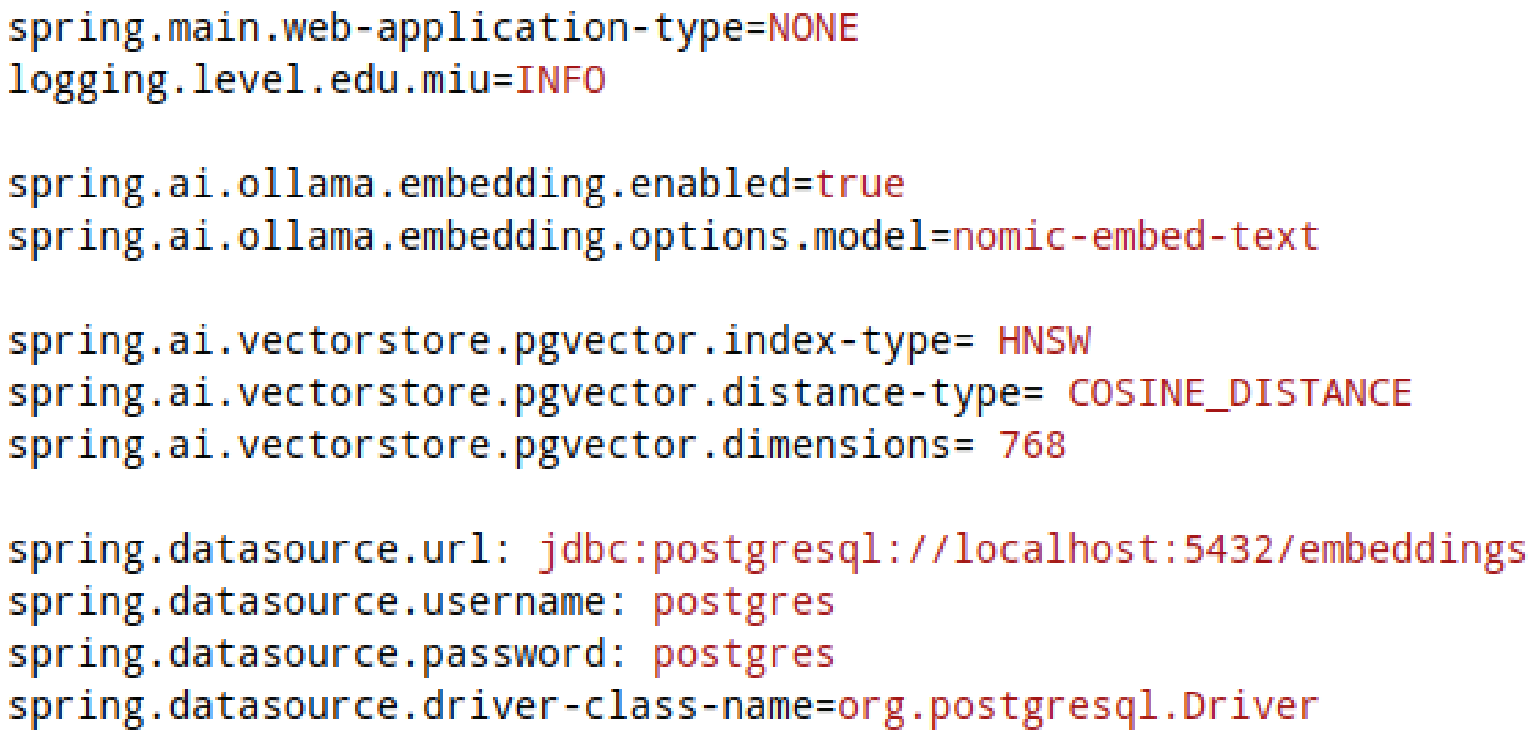
To install on local machine

Spring Boot Commandline Project

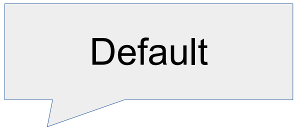


```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-ollama-spring-boot-starter</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-pgvector-store-spring-boot-starter</artifactId>
    </dependency>
    <dependency>
        <groupId>org.jsoup</groupId>
        <artifactId>jsoup</artifactId>
        <version>1.17.2</version>
    </dependency>
```

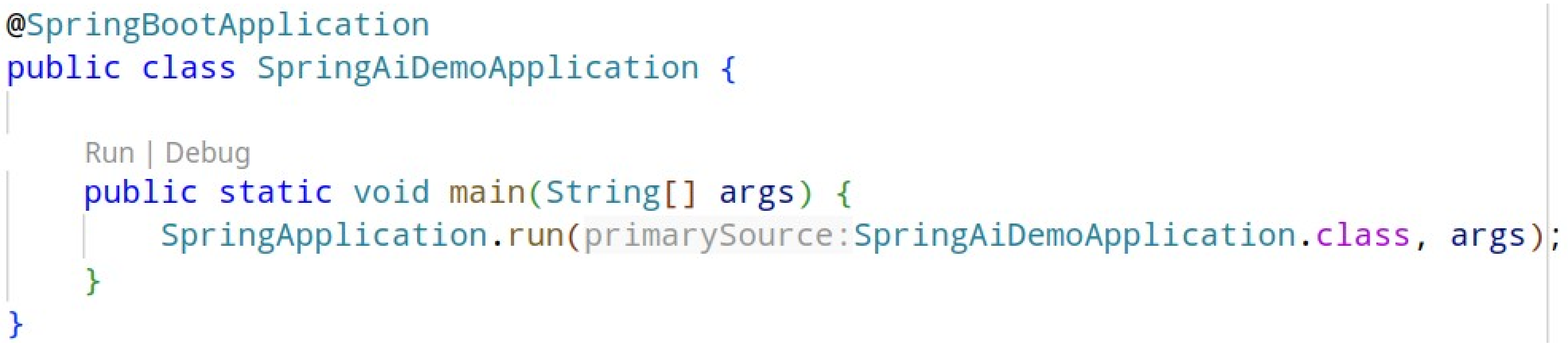
application.properties



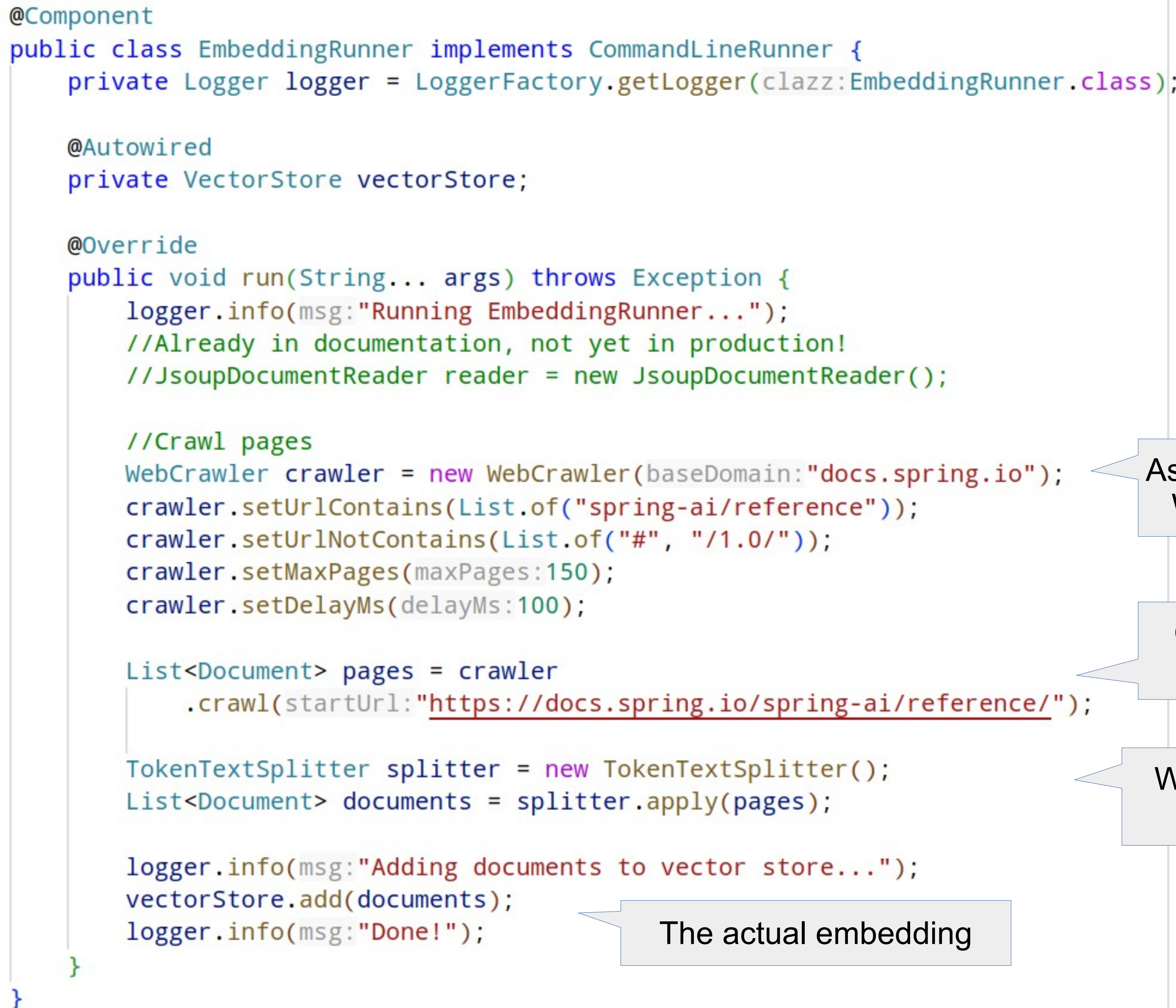
```
spring.main.web-application-type=NONE
logging.level.edu.miu=INFO

spring.ai.ollama.embedding.enabled=true
spring.ai.ollama.embedding.options.model=nomic-embed-text

spring.ai.vectorstore.pgvector.index-type= HNSW
spring.ai.vectorstore.pgvector.distance-type= COSINE_DISTANCE
spring.ai.vectorstore.pgvector.dimensions= 768

spring.datasource.url: jdbc:postgresql://localhost:5432/embeddings
spring.datasource.username: postgres
spring.datasource.password: postgres
spring.datasource.driver-class-name=org.postgresql.Driver
```



SpringBootApplication



```
@SpringBootApplication
public class SpringAiDemoApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource:SpringAiDemoApplication.class, args);
    }
}
```



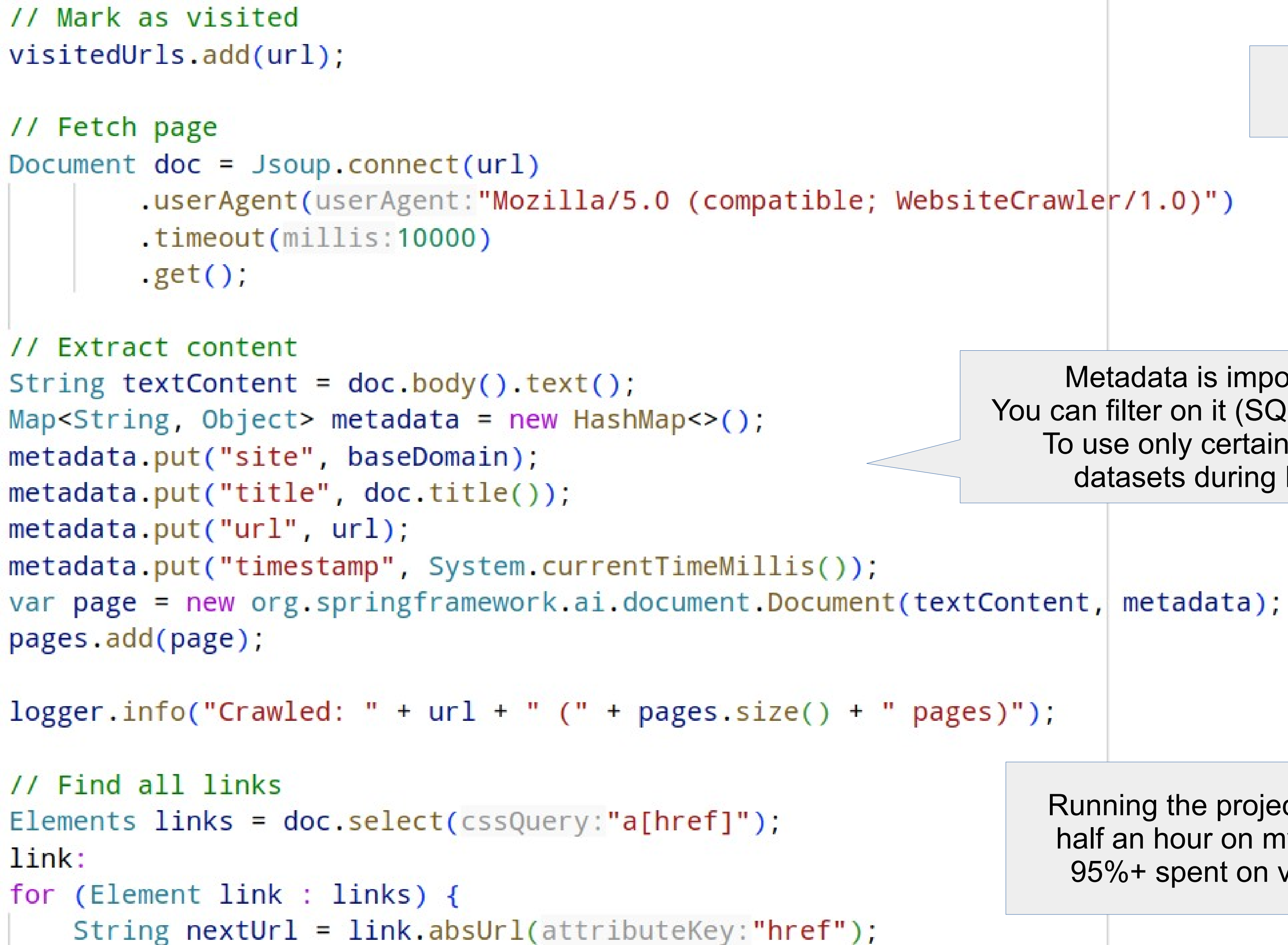
```java
@Component
public class EmbeddingRunner implements CommandLineRunner {
    private Logger logger = LoggerFactory.getLogger(clazz:EmbeddingRunner.class);

    @Autowired
    private VectorStore vectorStore;

    @Override
    public void run(String... args) throws Exception {
        logger.info(msg:"Running EmbeddingRunner...");
        //Already in documentation, not yet in production!
        //JsoupDocumentReader reader = new JsoupDocumentReader();

        //Crawl pages
        WebCrawler crawler = new WebCrawler(baseDomain:"docs.spring.io");
        crawler.setUrlContains(List.of("spring-ai/reference"));
        crawler.setUrlNotContains(List.of("#", "/1.0/"));
        crawler.setMaxPages(maxPages:150);
        crawler.setDelayMs(delayMs:100);

        List<Document> pages = crawler
            .crawl(startUrl:"https://docs.spring.io/spring-ai/reference/");

        TokenTextSplitter splitter = new TokenTextSplitter();
        List<Document> documents = splitter.apply(pages);

        logger.info(msg:"Adding documents to vector store...");
        vectorStore.add(documents);
        logger.info(msg:"Done!");
    }
}
```

Asked Grok to make me a WebCrawler with jsoup

Customized it to return Spring AI documents

We need to split before we can embed

The actual embedding



```java
// Mark as visited
visitedUrls.add(url);

// Fetch page
Document doc = Jsoup.connect(url)
        .userAgent(userAgent:"Mozilla/5.0 (compatible; WebsiteCrawler/1.0)")
        .timeout(millis:10000)
        .get();

// Extract content
String textContent = doc.body().text();
Map<String, Object> metadata = new HashMap<>();
metadata.put("site", baseDomain);
metadata.put("title", doc.title());
metadata.put("url", url);
metadata.put("timestamp", System.currentTimeMillis());
var page = new org.springframework.ai.document.Document(textContent, metadata);
pages.add(page);

logger.info("Crawled: " + url + " (" + pages.size() + " pages)");

// Find all links
Elements links = doc.select(cssQuery:"a[href]");
link:
for (Element link : links) {
    String nextUrl = link.absUrl(attributeKey:"href");
```

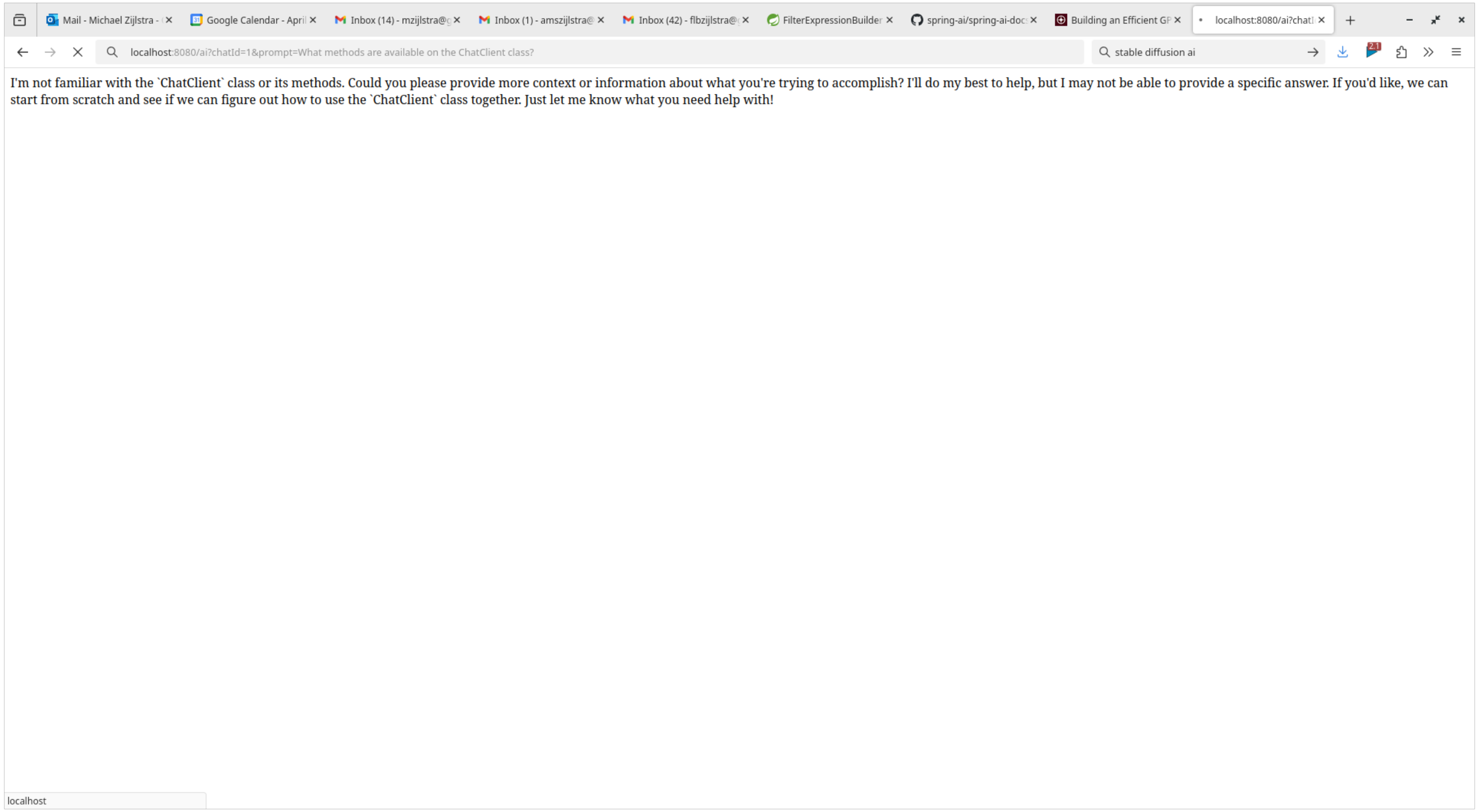
Core Crawler Code

Metadata is important!
You can filter on it (SQL WHERE)
To use only certain parts /
datasets during RAG

Running the project takes about
half an hour on my (old) laptop
95%+ spent on vectorization

Retrieval Augmented Generation

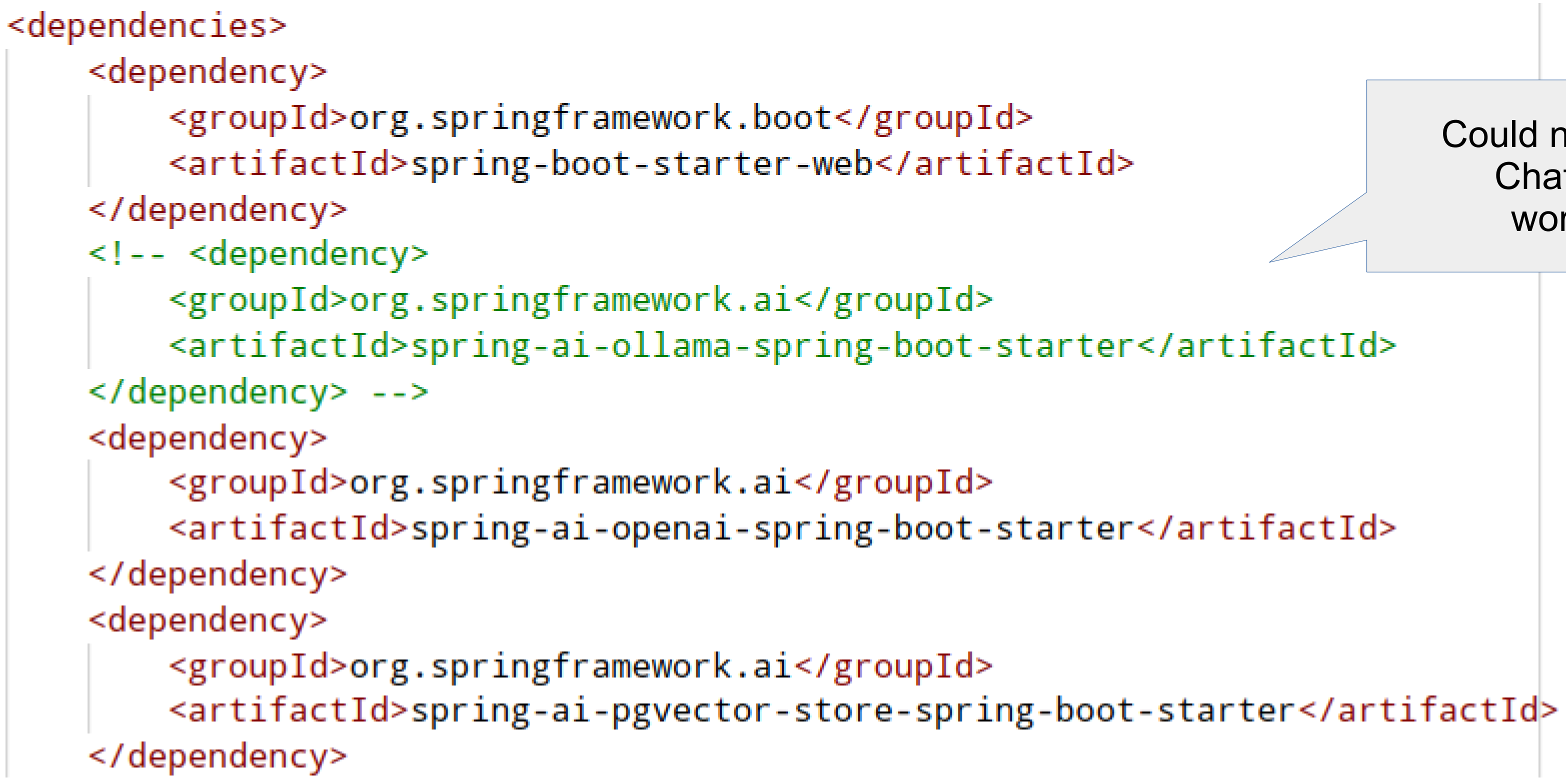
Start Demo



localhost:8080/ai?chatId=1&prompt=What methods are available on the ChatClient class?

I'm not familiar with the `ChatClient` class or its methods. Could you please provide more context or information about what you're trying to accomplish? I'll do my best to help, but I may not be able to provide a specific answer. If you'd like, we can start from scratch and see if we can figure out how to use the `ChatClient` class together. Just let me know what you need help with!

Add pgvector-store

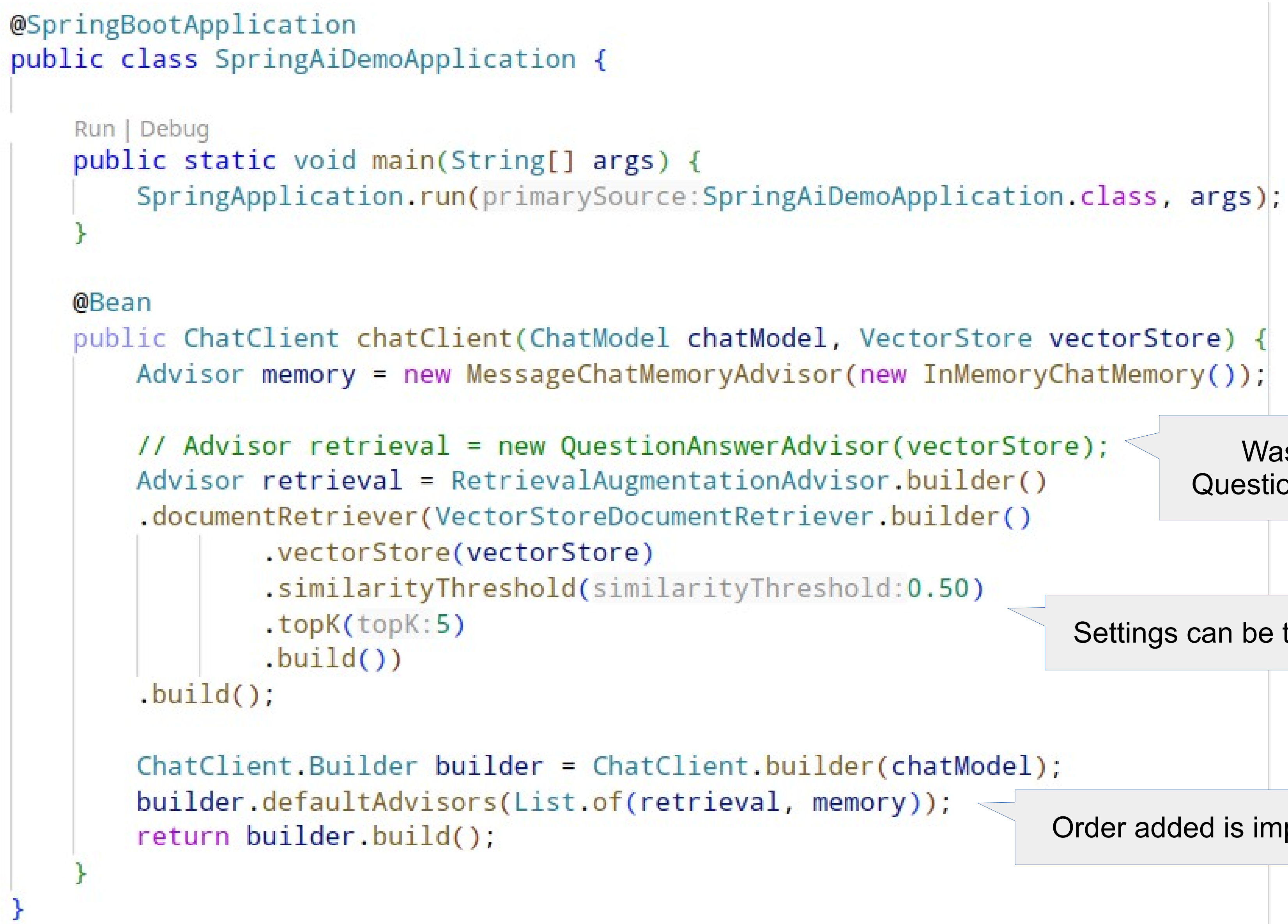


```xml
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <!-- <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-ollama-spring-boot-starter</artifactId>
    </dependency> -->
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-openai-spring-boot-starter</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.ai</groupId>
        <artifactId>spring-ai-pgvector-store-spring-boot-starter</artifactId>
    </dependency>
```

Could not make combination
ChatMemory and RAG
work with ollama API

application.properties

```
spring.ai.openai.chat.base-url=http://localhost:11434
spring.ai.openai.chat.options.model=llama3.2
spring.ai.openai.chat.options.temperature=0.7
spring.ai.openai.api-key=none

spring.ai.openai.embedding.enabled = true
spring.ai.openai.embedding.options.model=nomic-embed-text
spring.ai.openai.embedding.base-url=http://localhost:11434
spring.ai.openai.embedding.api-key= none

spring.ai.vectorstore.pgvector.index-type= HNSW
spring.ai.vectorstore.pgvector.distance-type= COSINE_DISTANCE
spring.ai.vectorstore.pgvector.dimensions= 768

spring.datasource.url: jdbc:postgresql://localhost:5432/pgvector
spring.datasource.username: postgres
spring.datasource.password: postgres
spring.datasource.driver-class-name=org.postgresql.Driver
```



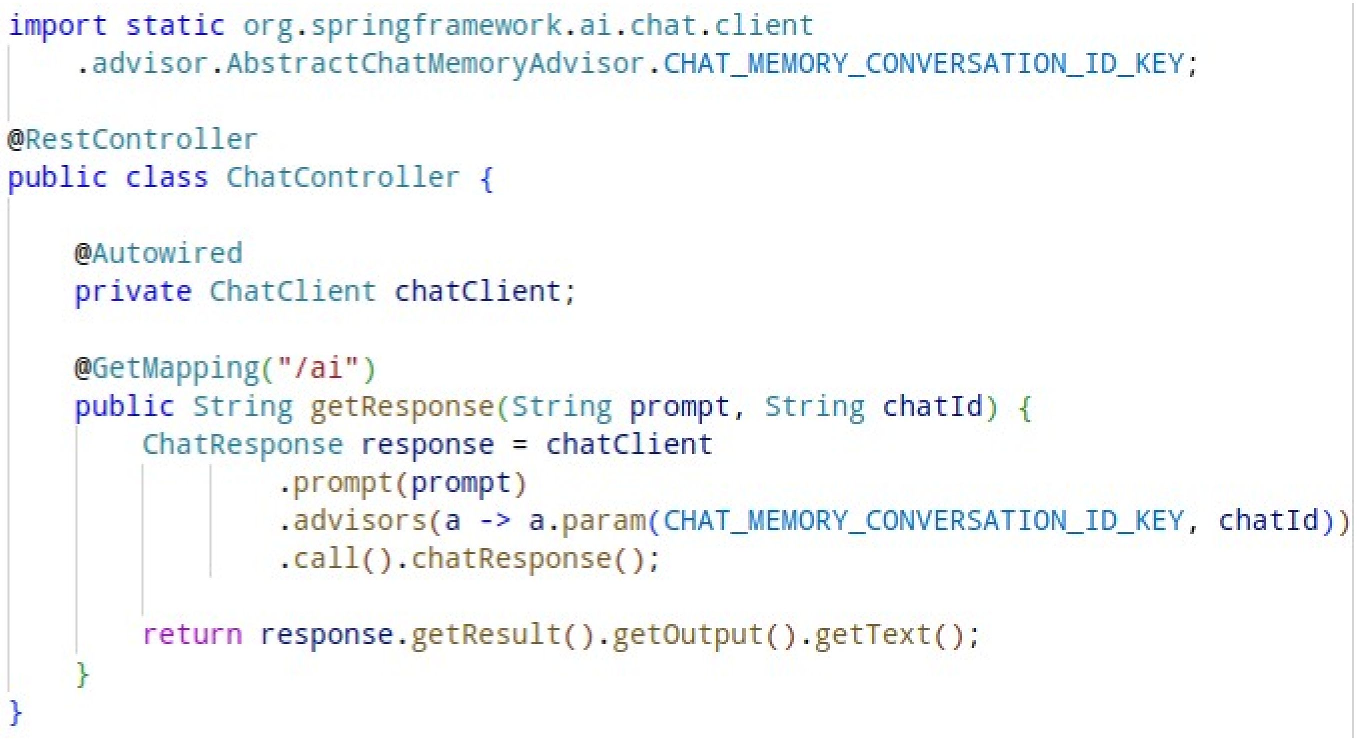
```java
@SpringBootApplication
public class SpringAiDemoApplication {

    Run | Debug
    public static void main(String[] args) {
        SpringApplication.run(primarySource:SpringAiDemoApplication.class, args);
    }

    @Bean
    public ChatClient chatClient(ChatModel chatModel, VectorStore vectorStore) {
        Advisor memory = new MessageChatMemoryAdvisor(new InMemoryChatMemory());

        // Advisor retrieval = new QuestionAnswerAdvisor(vectorStore);
        Advisor retrieval = RetrievalAugmentationAdvisor.builder()
        .documentRetriever(VectorStoreDocumentRetriever.builder()
                .vectorStore(vectorStore)
                .similarityThreshold(similarityThreshold:0.50)
                .topK(topK:5)
                .build())
        .build();

        ChatClient.Builder builder = ChatClient.builder(chatModel);
        builder.defaultAdvisors(List.of(retrieval, memory));
        return builder.build();
    }
}
```

Was not able to make QuestionAnswerAdvisor work

Settings can be tweaked

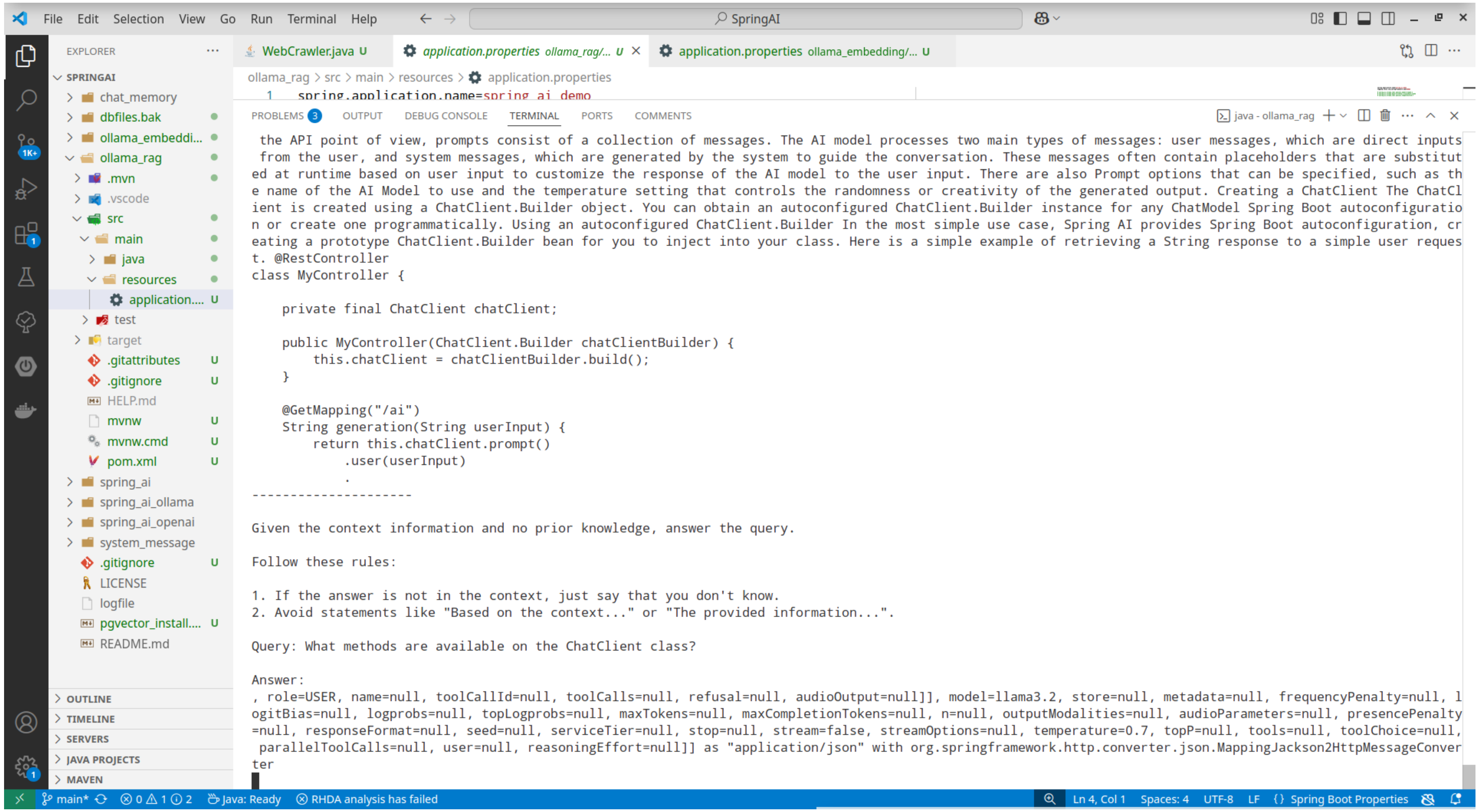
Order added is important!

No change to Controller



```java
import static org.springframework.ai.chat.client
    .advisor.AbstractChatMemoryAdvisor.CHAT_MEMORY_CONVERSATION_ID_KEY;

@RestController
public class ChatController {

    @Autowired
    private ChatClient chatClient;

    @GetMapping("/ai")
    public String getResponse(String prompt, String chatId) {
        ChatResponse response = chatClient
                .prompt(prompt)
                .advisors(a -> a.param(CHAT_MEMORY_CONVERSATION_ID_KEY, chatId))
                .call().chatResponse();

        return response.getResult().getOutput().getText();
    }
}
```

See the prompt being stuffed



```
spring.application.name=spring_ai_demo
```

```
 the API point of view, prompts consist of a collection of messages. The AI model processes two main types of messages: user messages, which are direct inputs
 from the user, and system messages, which are generated by the system to guide the conversation. These messages often contain placeholders that are substitut
ed at runtime based on user input to customize the response of the AI model to the user input. There are also Prompt options that can be specified, such as th
e name of the AI Model to use and the temperature setting that controls the randomness or creativity of the generated output. Creating a ChatClient The ChatCl
ient is created using a ChatClient.Builder object. You can obtain an autoconfigured ChatClient.Builder instance for any ChatModel Spring Boot autoconfiguratio
n or create one programmatically. Using an autoconfigured ChatClient.Builder In the most simple use case, Spring AI provides Spring Boot autoconfiguration, cr
eating a prototype ChatClient.Builder bean for you to inject into your class. Here is a simple example of retrieving a String response to a simple user reques
t. @RestController
class MyController {

    private final ChatClient chatClient;

    public MyController(ChatClient.Builder chatClientBuilder) {
        this.chatClient = chatClientBuilder.build();
    }

    @GetMapping("/ai")
    String generation(String userInput) {
        return this.chatClient.prompt()
            .user(userInput)
            .
---------------------

Given the context information and no prior knowledge, answer the query.

Follow these rules:

1. If the answer is not in the context, just say that you don't know.
2. Avoid statements like "Based on the context..." or "The provided information...".

Query: What methods are available on the ChatClient class?

Answer:
, role=USER, name=null, toolCallId=null, toolCalls=null, refusal=null, audioOutput=null]], model=llama3.2, store=null, metadata=null, frequencyPenalty=null, l
ogitBias=null, logprobs=null, topLogprobs=null, maxTokens=null, maxCompletionTokens=null, n=null, outputModalities=null, audioParameters=null, presencePenalty
=null, responseFormat=null, seed=null, serviceTier=null, stop=null, stream=false, streamOptions=null, temperature=0.7, topP=null, tools=null, toolChoice=null,
 parallelToolCalls=null, user=null, reasoningEffort=null]] as "application/json" with org.springframework.http.converter.json.MappingJackson2HttpMessageConver
ter
```

Summary

- Setting up PGVector
- Embedding Spring AI docs
- Retrieval Augmented Generation

Closing Thoughts

- We saw a practical example of embedding and Retrieval Augmented Generation
  - A lot of the advisors didn’t work that well
    - Ollama3.2 would just get confused
  - Maybe I should try writing my own?
- Next I want to integrate Ollama with VSCode
  - To have code suggestions generated locally
  - To not have company code sent to 3rd party

MAHARISHI INTERNATIONAL UNIVERSITY

HIGHER EDUCATION • FOR HIGHER CONSCIOUSNESS