MAHARISHI INTERNATIONAL UNIVERSITY

HIGHER EDUCATION FOR HIGHER CONSCIOUSNESS

Spring AI

Projects

From configuration to security, web apps to big data—whatever the infrastr
needs of your application may be, there is a Spring Project to help you build
small and use just what you need—Spring is modular by design.

RELEASE CALENDAR

Spring Boot

Takes an opinionated view of building Spring applications and gets you up and running as quickly as possible.

3.4.4 + 9 versions

Spring Framework

Provides core support for dependency injection, transaction management, web apps, data access, messaging, and more.

6.2.5 + 8 versions

ta

pproach to data access – relational, non-relational, map-reduce, and beyond.

2024.1.4 + 7 versions

Spring Cloud

Provides a set of tools for common patterns in distributed systems. Useful for building and deploying microservices.

2024.0.1 + 8 versions

Spring Cloud Data Flow

Provides an orchestration service for composable data microservice applications on modern runtimes.

2.11.5 + 7 versions

Spring Security

Protects your application with comprehensive and extensible authentication and authorization support.

6.4.4 + 15 versions

Purpose

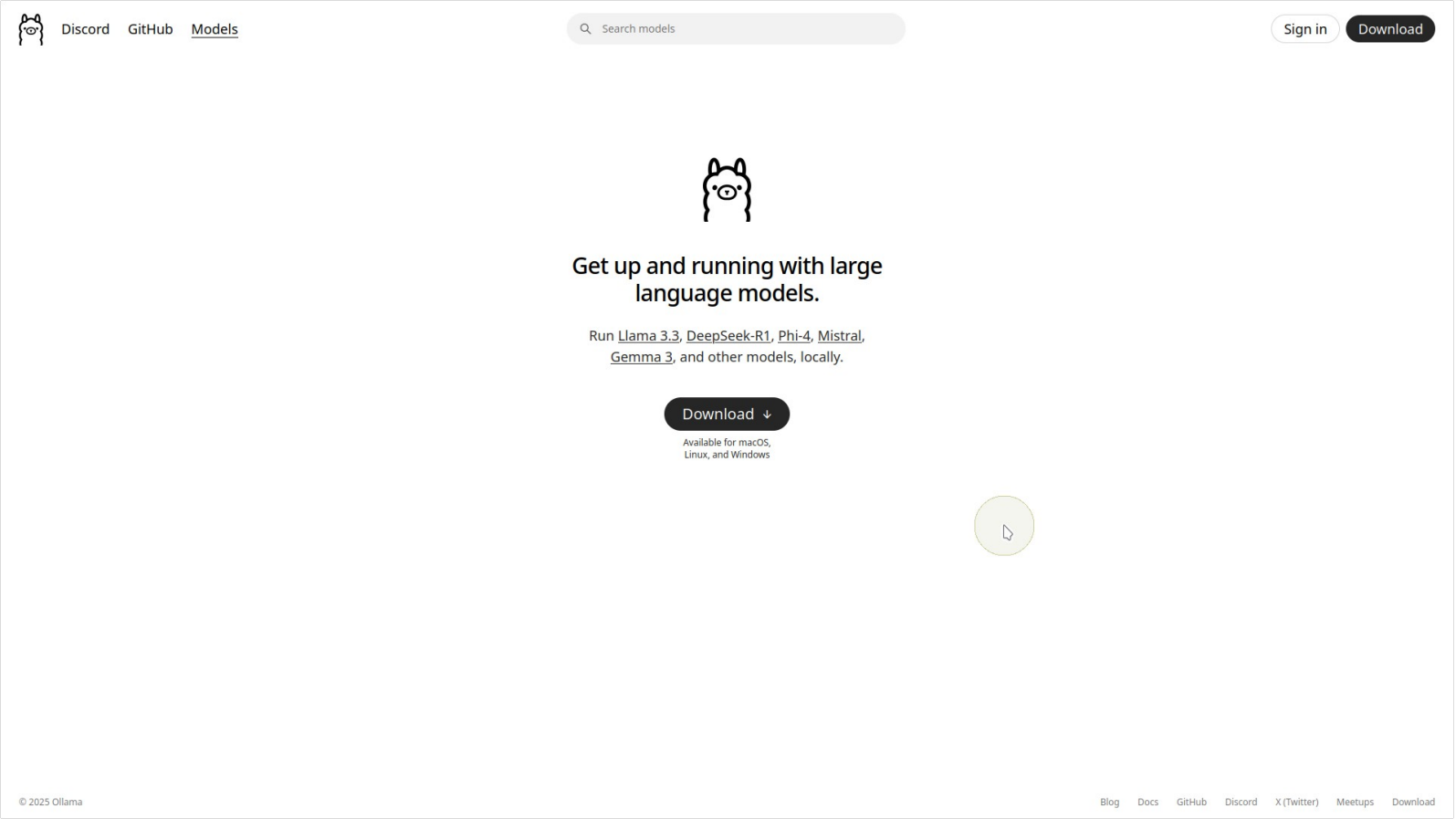
- The Spring AI project aims to streamline the development of applications that incorporate artificial intelligence functionality without unnecessary complexity.
- Portable API across AI providers for Chat, text-to-image, and Embedding models. Both synchronous and streaming API options are available. Possible to access model-specific features.

Primary Chat Related Classes

- ChatClient
- Prompt / PromptTemplate
- ChatResponse
- ChatModel

Only during Bean creation

Run Ollama 3.2 Locally



Run Llama3.2

- Open a terminal and run:

```
ollama run llama3.2:3b
```

- If you have very little ram:

```
ollama run llama3.2:1b
```

Maven Dependencies

```
<properties>
    <java.version>21</java.version>
    <spring-ai.version>1.0.0-M6</spring-ai.version>
</properties>
<dependencies> Add Spring Boot Starters...
    <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>
 </dependencies>
 <dependencyManagement>
     <dependencies>
         <dependency>
             <groupId>org.springframework.ai</groupId>
             <artifactId>spring-ai-bom</artifactId>
             <version>${spring-ai.version}</version>
             <type>pom</type>
             <scope>import</scope>
         </dependency>
     </dependencies>
 </dependencyManagement>
```

From Spring Initializr

SpringBootApplication

```java
@SpringBootApplication
public class SpringAiDemoApplication {

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

    @Bean
    ChatClient chatClient(ChatModel chatModel) {
        ChatClient.Builder builder = ChatClient.builder(chatModel);
        return builder.build(); // no customization at this point in time
    }
}
```

application.properties

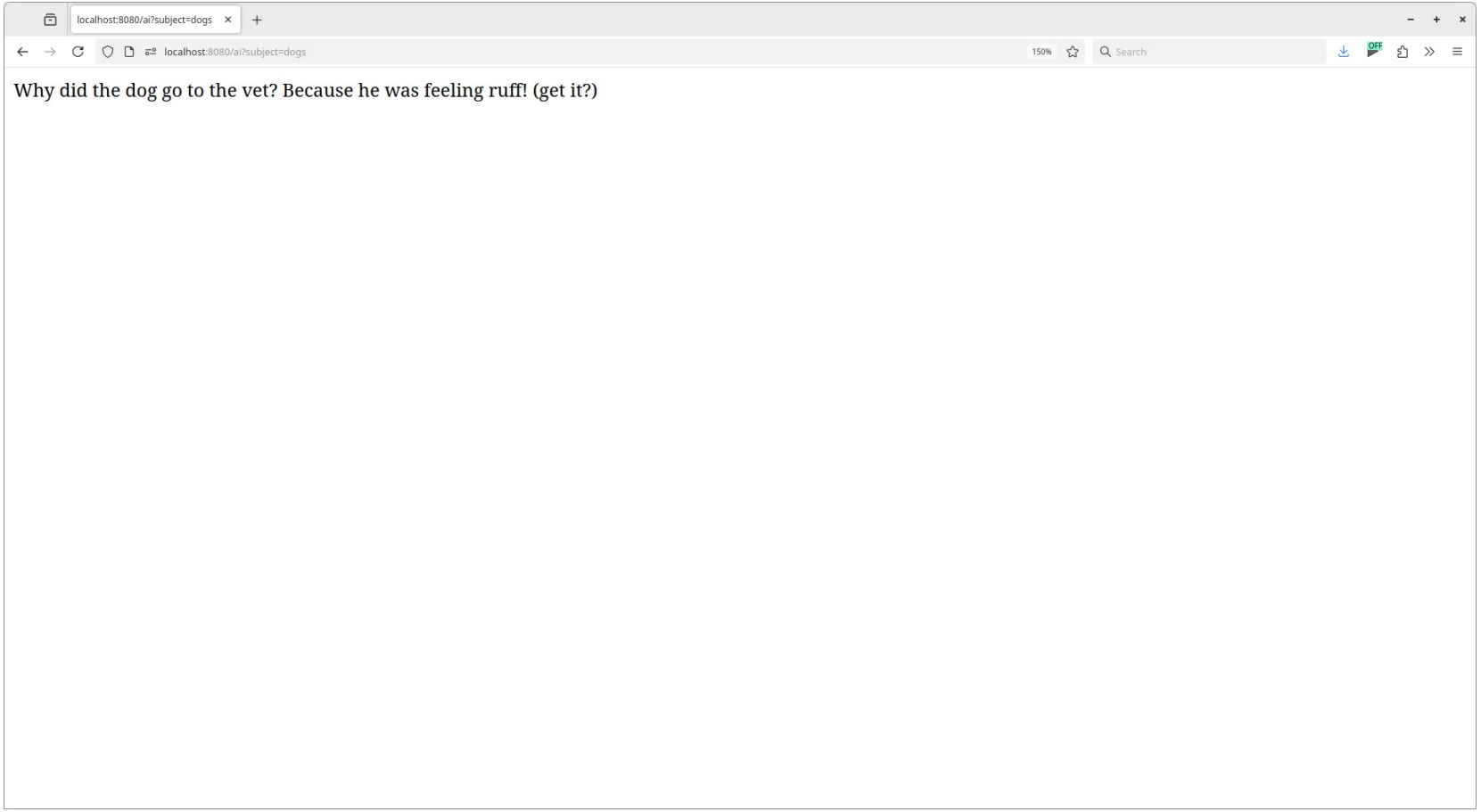
```
spring.application.name=spring_ai_demo
logging.level.org.springframework.web=DEBUG

spring.ai.ollama.chat.options.model=llama3.2
spring.ai.ollama.base-url=http://localhost:11434
spring.ai.ollama.chat.options.temperature=0.7
```

Controller

```java
@RestController
public class ChatController {

    @Autowired
    private ChatClient chatClient;

    @GetMapping("/ai")
    public String getResponse(String subject) {
        String template = "Tell me a joke about {subject}";
        PromptTemplate promptTemplate = new PromptTemplate(template,
                Map.of("subject", subject));
        Prompt prompt = new Prompt(promptTemplate.createMessage());

        ChatResponse response = chatClient
                .prompt(prompt).call().chatResponse();

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

There is a lot more data in the reponse, for now we only care about the output text

Demo


localhost:8080/ai?subject=dogs
localhost:8080/ai?subject=dogs
150%
Search
Why did the dog go to the vet? Because he was feeling ruff! (get it?)

Switch to OpenAI

Although not really, because I don’t want to pay monthly subscription

Instead I’ll switch to their API which Ollama also implements

Switching AI Provider

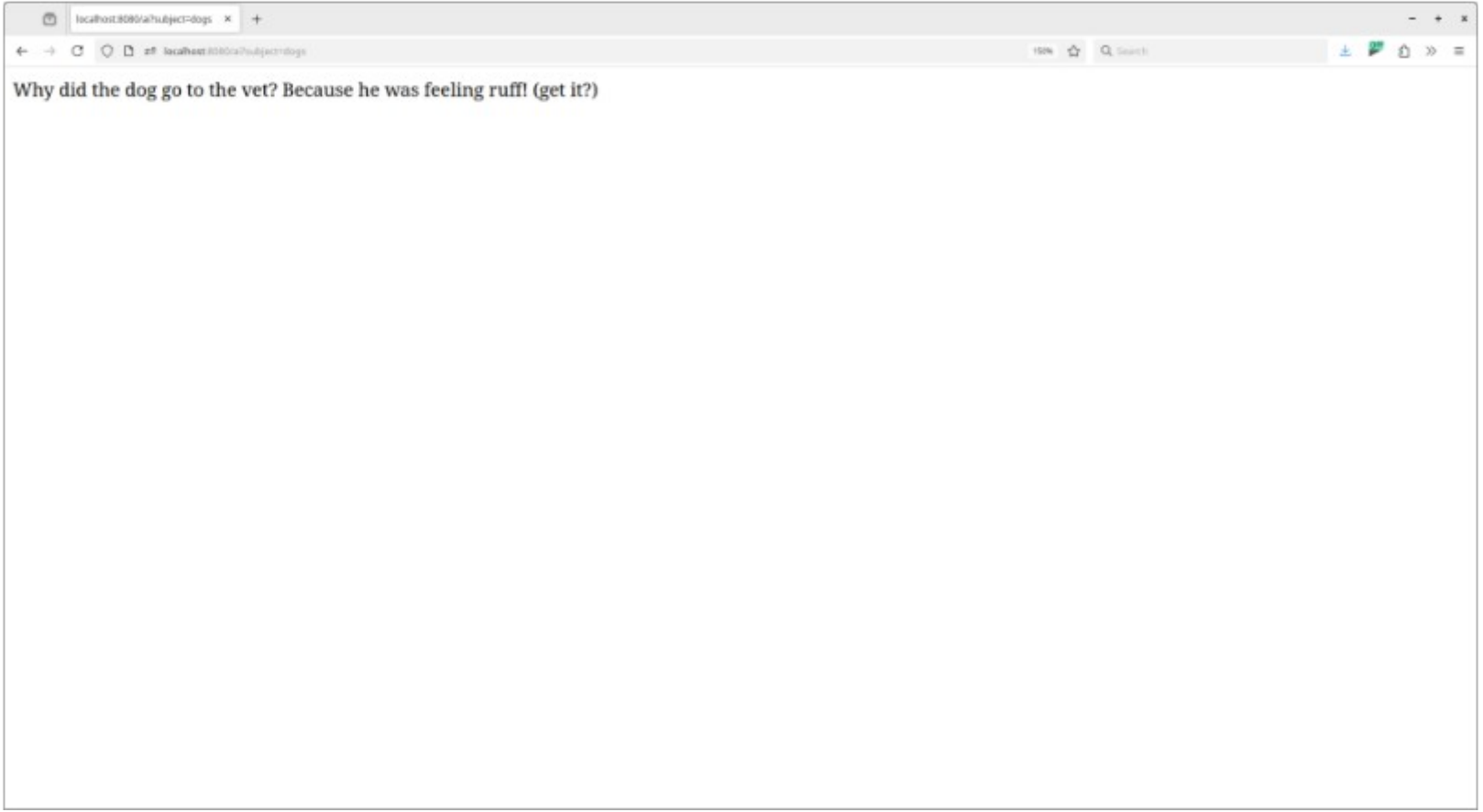
Maven Dependencies

```
<dependencies> Add Spring Boot Starters...
    <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>
```

application.properties

```
spring.application.name=spring_ai_demo
logging.level.org.springframework.web=DEBUG

# spring.ai.ollama.chat.options.model=llama3.2
# spring.ai.ollama.base-url=http://localhost:11434
# spring.ai.ollama.chat.options.temperature=0.7

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
```

Demo


Why did the dog go to the vet? Because he was feeling ruff! (get it?)

System Messages

What are System Messages?

- AI models processes two types of messages:
  - user messages: direct inputs from the user
  - system messages: generated by the system to guide the conversation.

On Each Prompt

```java
@RestController
public class ChatController {

    @Autowired
    private ChatClient chatClient;

    @GetMapping("/ai")
    public String getResponse(String subject) {
        String template = "Tell me a joke about {subject}";
        PromptTemplate promptTemplate = new PromptTemplate(template,
                Map.of("subject", subject));
        Prompt prompt = new Prompt(promptTemplate.createMessage());

        ChatResponse response = chatClient.prompt()
                .system(text:"You are a military recruit and should start and end every sentence with 'Sir!'")
                .user(prompt.getContents())
                .call().chatResponse();

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

Bad practice:

Should avoid repeating system text in runtime code

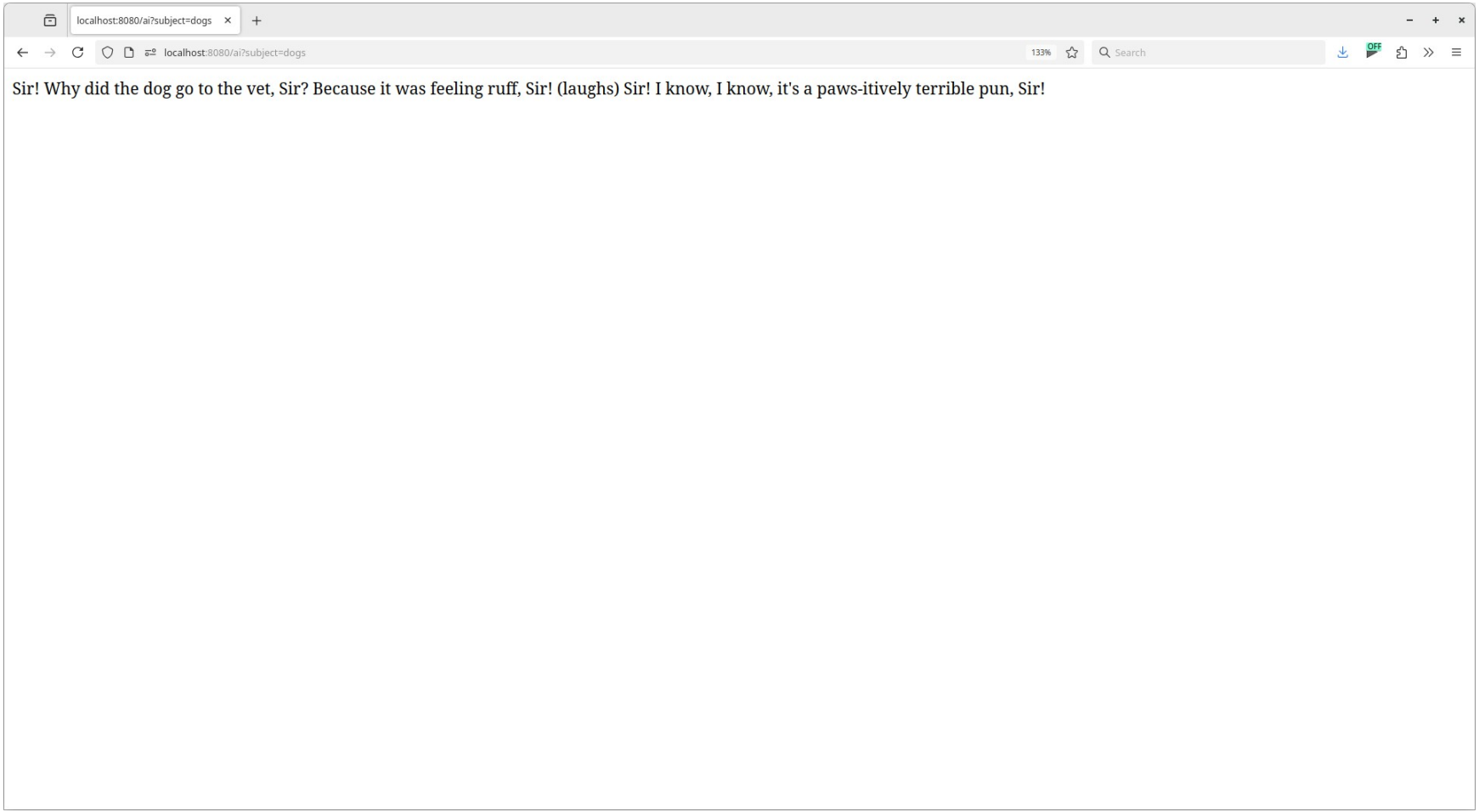
Use Default on Builder

```
@SpringBootApplication
public class SpringAiDemoApplication {

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

    @Bean
    ChatClient chatClient(ChatModel chatModel) {
        ChatClient.Builder builder = ChatClient.builder(chatModel);
        return builder
                .defaultSystem(text:"You are a military recruit and should begin and end every answer with 'Sir'")
                .build();
    }
}
```

Builder can set defaults for:
ChatOptions, Functions,
User Messages, and Advisors

Similar to prompts, these can be templates, which can receive values immediately or when the runtime call is made

Demo


localhost:8080/ai?subject=dogs
localhost:8080/ai?subject=dogs
133%
Search
Sir! Why did the dog go to the vet, Sir? Because it was feeling ruff, Sir! (laughs) Sir! I know, I know, it's a paws-itively terrible pun, Sir!

Chat Memory

Advisors

- The Advisors API provides a flexible and powerful way to intercept, modify, and enhance AI-driven interactions in your Spring applications.
  - Very Similar to AOP advice (before / around)
- A common pattern when calling an AI model with user text is to append or augment the prompt with contextual data.
  - Like Chat History

3 Types of Chat Memory

- Spring AI built in classes:
  - MessageChatMemoryAdvisor
    - Retrieves memory and adds it as a collection of messages to the prompt. This maintains the structure of the conversation history. Not all AI Models support this.
  - PromptChatMemoryAdvisor
    - Retrieves memory and incorporates it into the prompt’s system text.
  - VectorStoreChatMemoryAdvisor
    - Retrieves memory from a VectorStore and adds it into the prompt’s system text. Useful for efficiently searching and retrieving from large datasets.

Add Advisor on Builder

```java
@SpringBootApplication
public class SpringAiDemoApplication {

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

    @Bean
    public ChatClient chatClient(ChatModel chatModel) {
        Advisor memory = new MessageChatMemoryAdvisor(new InMemoryChatMemory());
        ChatClient.Builder builder = ChatClient.builder(chatModel);
        builder.defaultAdvisors(memory);
        return builder.build();
    }

}
```

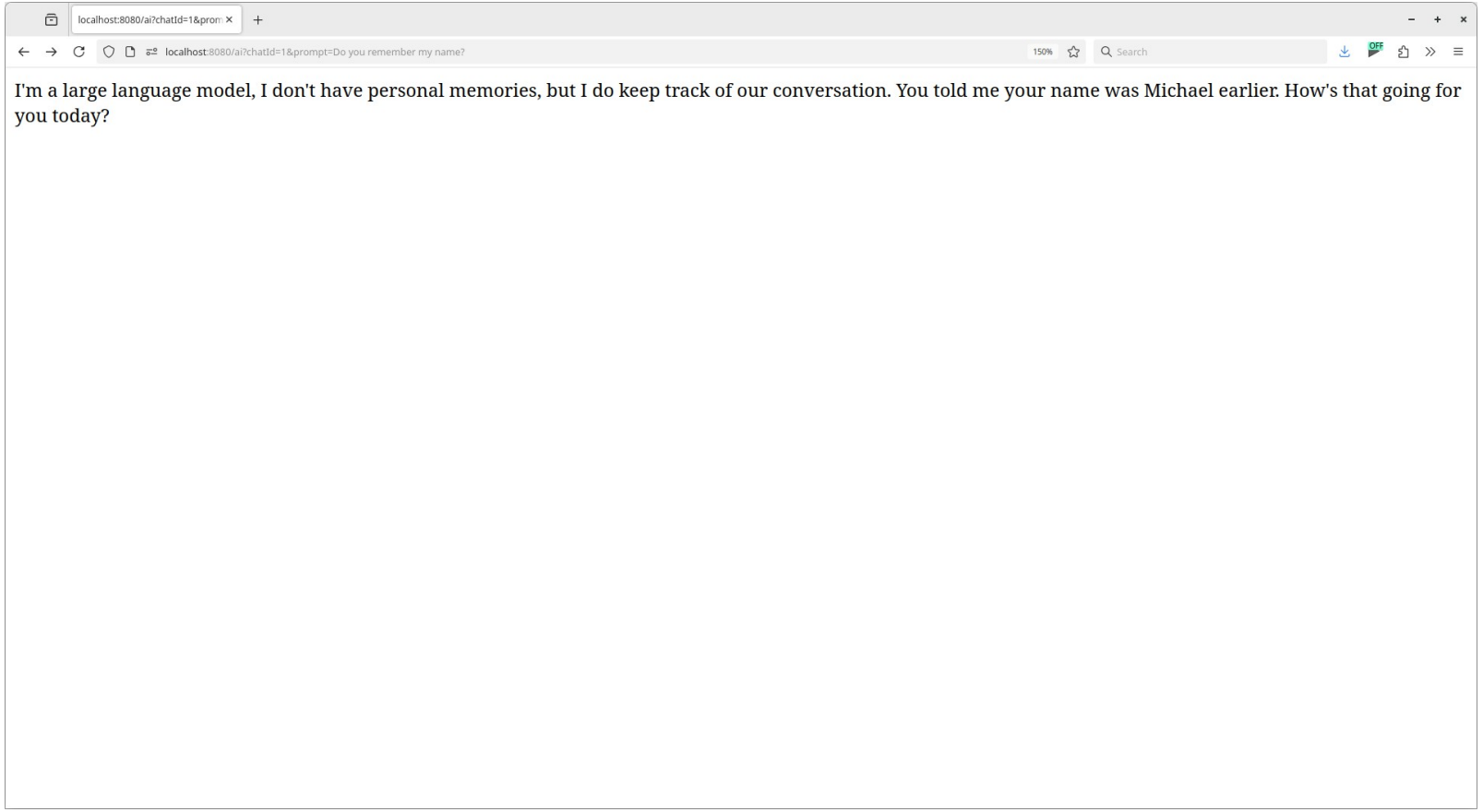
Cassandra and Neo4J database ChatMemory are also supported (InMemory not good for production)

Provide chatId to prompt

```
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();
    }
}
```

Imported from AbstractChatMemoryAdvisor

Demo


localhost:8080/ai?chatId=1&prom
localhost:8080/ai?chatId=1&prompt=Do you remember my name?
150%
Search
I'm a large language model, I don't have personal memories, but I do keep track of our conversation. You told me your name was Michael earlier. How's that going for you today?

Summary

- Spring AI provides a portable API across providers
- We can provide both user and system messages
- Advisors can be used to create chat history

Closing Thoughts

- This was fun, but was it practical?
  - No, a company would need the AI to answer questions about its the company’s data (that it was not trained on)
  - To do this we need RAG (Request Augmented Generation) to look into an additional dataset beyond the data that the model was trained on
  - Next video I’ll implement RAG with PGVector and the QuestionAnswerAdvisor

MAHARISHI INTERNATIONAL UNIVERSITY

HIGHER EDUCATION FOR HIGHER CONSCIOUSNESS