INNOQ Technology Night / Offenbach / 23.10.2024

LLMs mit Spring AI integrieren

INNOQ

MICHAEL VITZ
SENIOR CONSULTANT

MICHAEL VITZ

Java Champion
Senior Consultant at INNOQ

Preamble

This talk is purely about technical integration

This talk ignores many topics like prompt engineering, fine tuning, model selection, ...

This talk ignores the important topics like sense/nonsense, sustainability, social impact, …

Ollama

Get up and running with large language models.

Run Llama 3.1, Phi 3, Mistral, Gemma 2, and other models. Customize and create your own.

Download ↓

Available for macOS, Linux, and Windows (preview)

https://ollama.com/

Models

Filter by name...

Featured

llama3.1

Llama 3.1 is a new state-of-the-art model from Meta available in 8B, 70B and 405B parameter sizes.

Tools 8B 70B 405B

5.2M Pulls 94 Tags Updated 2 days ago

gemma2

Google Gemma 2 is a high-performing and efficient model available in three sizes: 2B. 9B. and 27B.

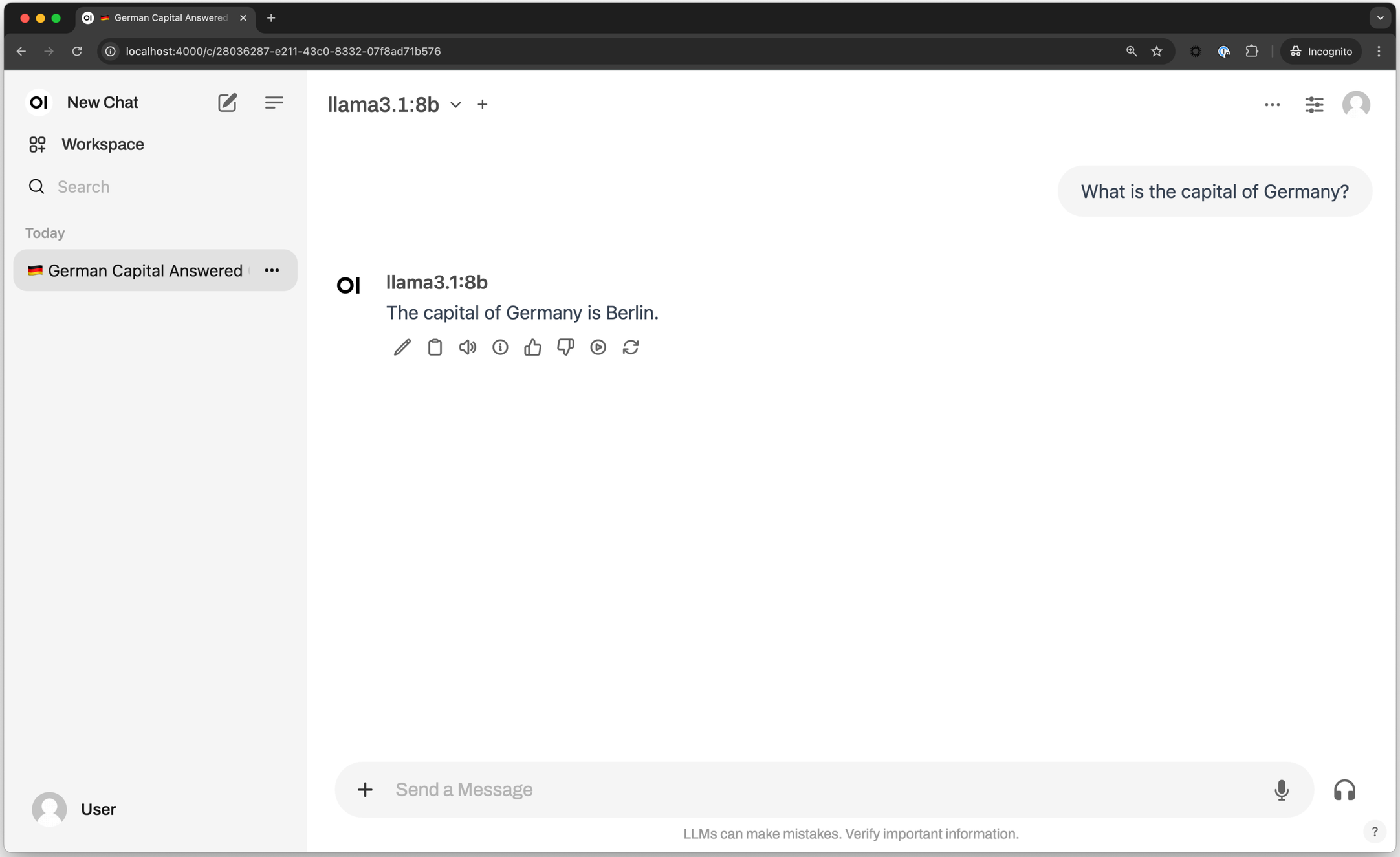
https://ollama.com/library

```
$ ollama pull llama3.1:8b
...

$ ollama run llama3.1:8b
>>> What is the capital of Germany?
The capital of Germany is Berlin.

>>> And of France?
The capital of France is Paris.

>>> /bye
```



https://openwebui.com/

```
$ curl http://localhost:11434/api/generate \
    -d '
{
  "model": "llama3.1:8b",
  "stream": false,
  "prompt": "What is the capital of Germany?"
}'
```

```
{
  "model": "llama3.1:8b",
  "created_at": "...",
  "response": "The capital of Germany is Berlin.",
  "done": true,
  "done_reason": "stop",
  "context": [
    ...
  ],
  "total_duration": 1352687287,
  "load_duration": 24182739,
  "prompt_eval_count": 17,
  "prompt_eval_duration": 172446000,
  "eval_count": 8,
  "eval_duration": 1157563000
}
```

```
$ curl http://localhost:11434/api/chat \
    -d '
{
  "model": "llama3.1:8b",
  "stream": false,
  "messages": [
    {
      "role": "user",
      "content": "What is the capital of Germany?"
    },
    {
        "role": "assistant",
        "content": "The capital of Germany is Berlin."
    },
    {
      "role": "user",
      "content": "And of France?"
    }
  ]
}'
```

```
{
  "model": "llama3.1:8b",
  "created_at": “...”,
  "message": {
    "role": "assistant",
    "content": "Paris is the capital of France."
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 1672865740,
  "load_duration": 23987996,
  "prompt_eval_count": 38,
  "prompt_eval_duration": 580858000,
  "eval_count": 8,
  "eval_duration": 1064965000
}
```


Spring AI

```xml
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.springframework.ai</groupId>
      <artifactId>spring-ai-bom</artifactId>
      <version>1.0.0-M2</version>
      <type>pom</type>
      <scope>import</scope>
    </dependency>
  </dependencies>
</dependencyManagement>
```

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

```
spring.ai.ollama.chat.options.model=llama3.1:8b
```

```
private String askQuestion(ChatClient.Builder clientBuilder) {
    var client = clientBuilder.build();

    var prompt = new Prompt(
            new UserMessage("What is the capital of Germany?"));

    return client
            .prompt(prompt)
            .call()
            .content();
}
```

```
private String askQuestion(ChatClient.Builder clientBuilder) {
    var client = clientBuilder.build();

    return client
            .prompt()
            .user("What is the capital of Germany?")
            .call()
            .content();
}
```

```java
private void chat(ChatClient.Builder clientBuilder) {
    var client = clientBuilder
            .defaultAdvisors(new MessageChatMemoryAdvisor(new InMemoryChatMemory()))
            .build();
    var answer = client.prompt()
            .advisors(a -> a.param(CHAT_MEMORY_CONVERSATION_ID_KEY, "session-abc"))
            .user("What is the capital of Germany?")
            .call()
            .content();
    System.out.println(answer);

    answer = client.prompt()
            .advisors(a -> a.param(CHAT_MEMORY_CONVERSATION_ID_KEY, "session-abc"))
            .user("And of France?")
            .call()
            .content();
    System.out.println(answer);
}
```

```
private void multimodality(ChatClient.Builder clientBuilder) {
    var client = clientBuilder.build();

    var answer = client.prompt()
            .user(u -> u
                    .text("What do you see in this picture?")
                    .media(IMAGE_JPEG, new ClassPathResource("/image.jpg")))
            .call()
            .content();

    System.out.println(answer);
}
```

```
public void rag(ChatClient.Builder clientBuilder,
                VectorStore store,
                @Value("classpath:/offices.json") Resource resource) {
    var documents = new JsonReader(resource, "zip", "city", "address").get();
    store.add(documents);

    var question = "Where exactly is the INNOQ office located in Hamburg?";

    var contextDocuments = store.similaritySearch(question).stream()
            .map(Document::getContent)
            .collect(joining("\n"));

    var answer = clientBuilder
            .build()
            .prompt()
            .system(s -> s
                    .text("""
                            You are a virtual assistant.
                            You are answering questions regarding the INNOQ offices provided within the DOCUMENTS paragraph.
                            You are only allowed to use information from the DOCUMENTS paragraph and no other information.
                            If you are not sure or don't know honestly state that you don't know.

                            DOCUMENTS:
                            {question_answer_context}""")
                    .param("question_answer_context", contextDocuments))
            .user(question)
            .call()
            .content();

    System.out.println(answer);
}
```

```
public void rag(ChatClient.Builder clientBuilder,
                VectorStore store,
                @Value("classpath:/offices.json") Resource resource) {
    var documents = new JsonReader(resource, "zip", "city", "address").get();
    store.add(documents);

    var question = "Where exactly is the INNOQ office located in Hamburg?";

    var answer = client
            .build()
            .prompt()
            .advisors(new QuestionAnswerAdvisor(store, defaults(),
            """
                You are a virtual assistant.
                You are answering questions regarding the INNOQ offices provided within the DOCUMENTS paragraph.
                You are only allowed to use information from the DOCUMENTS paragraph and no other information.
                If you are not sure or don't know honestly state that you don't know.

                DOCUMENTS:
                {question_answer_context}"""))
            .user(question)
            .call()
            .content();

    System.out.println(answer);
}
```

```
$ curl http://localhost:11434/api/chat \
    -d '
{
  "model": "llama3.1:8b",
  "stream": false,
  "messages": [
    {
      "role": "user",
      "content": "Which employees knows Java?"
    }
  ],
  "tools": [
    {
      "type": "function",
      "function": {
        "name": "employeesForSkill",
        "description": "Get the list of employees that have the given skill",
        "parameters": {
          "type": "object",
          "properties": {
            "skill": {
              "description": "The skill to search employees for, e.g. Java or JavaScript",
              "type": "string"
            }
          },
          "required": [
            "skill"
          ]
        }
      }
    }
  ]
}'
```

```
{
  "model": "llama3.1:8b",
  "created_at": "...",
  "message": {
    "role": "assistant",
    "content": "",
    "tool_calls": [
      {
        "function": {
          "name": "employeesForSkil
          "arguments": {
            "skill": "Java"
          }
        }
      }
    ]
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 15440747648,
  "load_duration": 4607826715,
  "prompt_eval_count": 177,
  "prompt_eval_duration": 792260600
  "eval_count": 18,
  "eval_duration": 2908405000
}
```

```
$ curl http://localhost:11434/api/chat \
    -d '
{
  "model": "llama3.1:8b",
  "stream": false,
  "messages": [
    {
      "role": "user",
      "content": "Which employees knows Java?"
    },
    {
      "role": "assistant",
      "content":"",
      "tool_calls": [
        ...
      ]
    },
    {
      "role": "tool",
      "name": "employeesForSkill",
      "content": "[\"Marco\", \"Michael\", \"Stefan\"]"
    }
  ],
  "tools": [
    ...
  ]
}'
```

```
{
  "model": "llama3.1:8b",
  "created_at": "2024-09-24T13:45:06.824552Z",
  "message": {
    "role": "assistant",
    "content": "Based on the tool call response, it appears that Marco,
Michael, and Stefan are the employees who know Java."
  },
  "done_reason": "stop",
  "done": true,
  "total_duration": 12736926568,
  "load_duration": 4090846558,
  "prompt_eval_count": 100,
  "prompt_eval_duration": 4850322000,
  "eval_count": 24,
  "eval_duration": 3790006000
}
```

```java
public static class EmployeeSkillFunction implements Function<EmployeeSkillRequest, EmployeeSkillResponse> {

    @JsonClassDescription("Get the list of employees that have the given skill")
    public record EmployeeSkillRequest(
            @JsonProperty(required = true) @JsonPropertyDescription("The skill to …") String skill) {
    }

    public record EmployeeSkillResponse(
            List<String> employeeNames) {
    }

    @Override
    public EmployeeSkillResponse apply(EmployeeSkillRequest request) {
        System.out.println("EmployeeSkillFunction.apply");
        System.out.println("request = " + request);

        return new EmployeeSkillResponse(of("Michael"));
    }
}
```

```java
@Bean
public Function<EmployeeSkillRequest, EmployeeSkillResponse> employeesForSkill() {
    return new EmployeeSkillFunction();
}
```

```java
public void functionCall(ChatClient client) {
    var answer = client.prompt()
            .user("Which employees knows Java?")
            .functions("employeesForSkill")
            .call()
            .content();
    System.out.println(answer);
}
```

```java
public static class WeatherFunction
        implements Function<WeatherFunction.WeatherRequest, WeatherFunction.WeatherResponse> {

    @JsonClassDescription("Returns the weather for the given location at the given date")
    public record WeatherRequest(
            @JsonProperty(required = true)
            @JsonPropertyDescription("The latitude of the city")
            String latitude,
            @JsonProperty(required = true)
            @JsonPropertyDescription("The longitude of the city")
            String longitude,
            @JsonProperty(required = true)
            @JsonPropertyDescription("The date to retrieve the weather for, e.g. 2024-02-29")
            String date) {
    }

    public record WeatherResponse(
            String weather) {
    }

    @Override
    public WeatherResponse apply(WeatherRequest request) {
        System.out.println("WeatherFunction.apply");
        System.out.println("request = " + request);

        return new WeatherResponse("lots of rain");
    }
}
```

```
@Bean
public Function<WeatherFunction.WeatherRequest, WeatherFunction.WeatherResponse>
        getWeatherForGivenLocationAndDate() {
    return new WeatherFunction();
}


public void functionCall(ChatClient client) {
    var answer = client.prompt()
            .user("How is today's weather in Berlin?")
            .functions("getWeatherForGivenLocationAndDate")
            .call()
            .content();
    System.out.println(answer);
}
```

```
public static class DatabaseFunction
        implements Function<DatabaseRequest, DatabaseResponse> {

    @JsonClassDescription("""
            Use this function to answer questions about employee skills.
            Input should be a fully formed SQL query.
            """)
    public record DatabaseRequest(
            @JsonProperty(required = true)
            @JsonPropertyDescription("""
                    SQL query for extracting data to answer the question.
                    SQL should be written using the given schema:
                      Table: employees
                      Columns: id, name

                      Table: skills
                      Columns: id, name

                      Table: employee_skills
                      Columns: employee_id, skill_id

                    The query should be plain text and not JSON.""")
            String query) {
    }

    public record DatabaseResponse(
            List<Map<String, String>> result) {
    }

    @Override
    public DatabaseResponse apply(DatabaseRequest request) {
        System.out.println("DatabaseFunction.apply");
        System.out.println("request = " + request);

        return new DatabaseResponse(of(
                Map.of("e.name", "Michael"),
                Map.of("e.name", "Stefan")));
    }
```

```java
@Bean
public Function<DatabaseRequest, DatabaseResponse> retrieveFromDatabase() {
    return new DatabaseFunction();
}
```

```java
public void functionCall(ChatClient client) {
    var answer = client.prompt()
            .user("Which employees knows Java?")
            .functions("retrieveFromDatabase")
            .call()
            .content();
    System.out.println(answer);
}
```

Trace

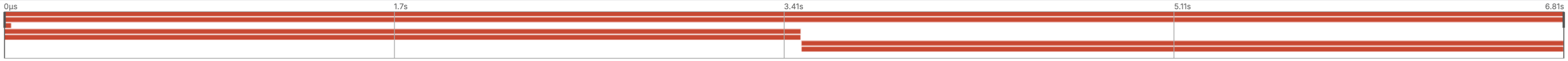
spring-ai: http get / 6.81s

2024-09-24 16:02:54.808 /

Give feedback | Trace ID | Export

Span Filters

8 spans | Prev | Next



Service & Operation	0µs	1.7s	3.41s	5.11s	6.81s
spring-ai http get / (6.81s)					
spring_ai chat_client (6.81s)					
chat_client_advisor message_chat_memory_advisor_be (411.04µs)	411.04µs				
chat llama3.1:8b (3.47s)			3.47s		
http post (3.47s)			3.47s		
chat llama3.1:8b (3.33s)			3.33s		
http post (3.32s)			3.32s		
chat_client_advisor message_chat_memory_advisor_af (502.75µs)					502.75µs

Conclusion

Ollama

- Allows running LLMs locally
- Feels like Docker for LLMs
- Takes a lot of resources
- Results are sufficient for local development

Spring AI

- Offers specific, higher level API for LLM/VectorStore interactions
- Abstracts from specific LLM vendors
- Abstraction may leak (not every vendor supports every feature)
- Looks like the Spring code you are used to
- Is not final yet and moves at fast pace (like the whole AI space)

Alternatives to Spring AI

- Quarkus LangChain4j
- LangChain4j
- pick whatever floats your boat

Resources

- Spring AI Reference Documentation
- Prebuild Ollama Testcontainers
- How to Use Hugging Face Models
- Leverage the Power of 45k, free, Hugging Face Models

Thanks! Questions?

Michael Vitz

Mail	michael.vitz@innoq.com
X	@michaelvitz
Mastodon	@michaelvitz@innoq.social
LinkedIn	michaelvitz

https://www.innoq.com/de/talks/2024/10/llms-mit-spring-ai-integrieren-10-24/