

Project in mind

03 August 2025 00:05

>>> I am thinking to build AI based debugging and login system is spring project
That's a fantastic idea for a Spring project, Maver! Building an AI-powered debugging and logging system is really interesting and has a lot of potential. It's a very relevant and challenging project.

Here are some thoughts and areas you might want to consider as you develop it:

****Key Features & Technologies:****

- * ****Log Parsing:**** You'll need a robust system to parse logs (likely JSON or structured formats) to extract relevant information – error messages, timestamps, stack traces, user context, etc. Consider using libraries like Logstash, Fluentd, or even just regular expressions within Spring.
- * ****Anomaly Detection:**** This is where the AI comes in. You could use machine learning models (like anomaly detection algorithms, clustering, or even simple rule-based systems) to identify unusual patterns in the logs that might indicate problems.
- * ****Root Cause Analysis:**** The system could then attempt to correlate anomalies with other log data and application state to pinpoint the root cause of issues.
- * ****Spring Boot Integration:**** Using Spring Boot is excellent for creating a manageable and deployable application.
- * ****AI Model Training:**** You'll need a dataset of logs with known issues to train your AI model. You could generate synthetic data or leverage existing log data.
- * ****Alerting:**** Automatically send alerts when anomalies or potential problems are detected.

****Spring-Specific Considerations:****

- * ****Spring Cloud:**** Great for scalability and distributed logging.
- * ****Spring Data:**** Could be useful for querying logs stored in a database.
- * ****Spring Batch:**** For processing large log files.

****To help me give you more targeted advice, could you tell me a bit more about your project?****

- * What kind of applications are you targeting (e.g., web applications, microservices)?
- * What level of AI sophistication are you aiming for (e.g., simple rule-based, machine learning)?
- * What's your experience level with Spring and AI/ML?

Reference

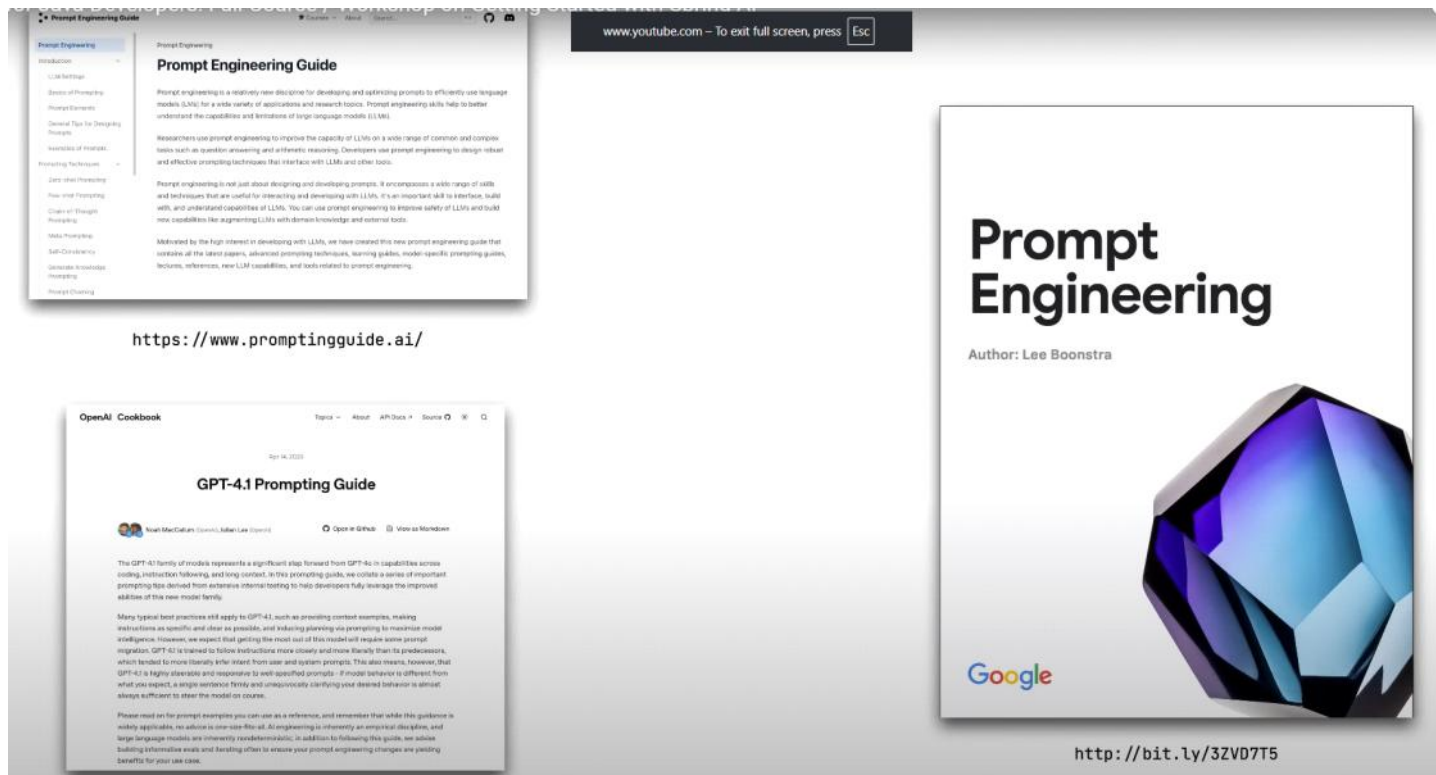
27 July 2025 15:53

AI for Java Developers: Full Course / Workshop on Getting Started with Spring AI

YouTube link - <https://www.youtube.com/watch?v=FzLABAppJfM>

Prompt Engineering -

1. Promptingguide.ai
2. Cookbook.openai.com



<https://platform.openai.com/api-keys>

SpringAI reference documentations -

<https://docs.spring.io/spring-ai/reference/index.html>

ModelContextProtocol -

<https://modelcontextprotocol.io/overview>

MCP Server -

<https://modelcontextprotocol.io/quickstart/server#java>

1. Introduction and Course Overview

27 July 2025 15:56

integrating AI into Java applications without deep expertise in machine learning or switching to Python. He highlights Spring AI, a newly released (version 1.0) framework designed to enable Java developers to build AI-powered applications by leveraging familiar tools and paradigms. The course aims to cover essential AI concepts, practical coding with Spring AI, and key features like prompt engineering, large language model (LLM) integration, observability, evaluation, and advanced topics such as retrieval augmented generation (RAG), tools, and model context protocol (MCP).

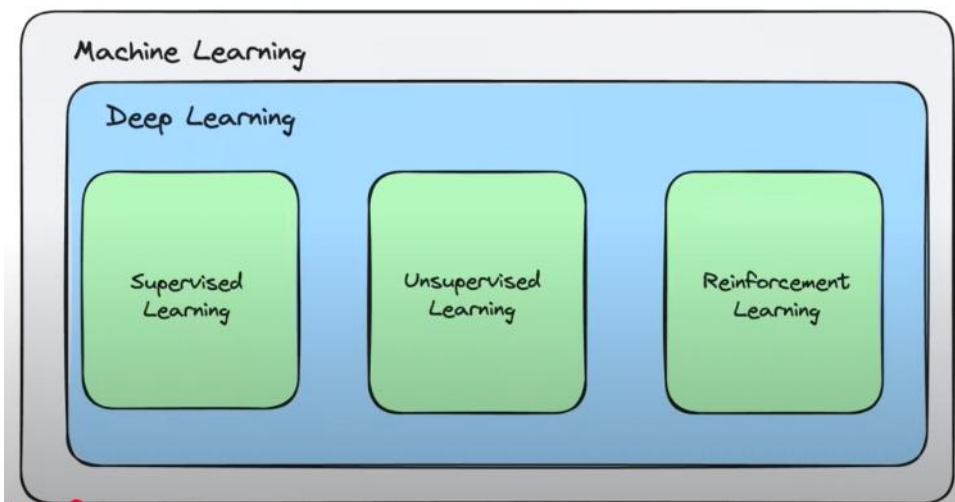
2. Fundamentals of AI, Machine Learning, and Large Language Models (LLMs)

27 July 2025 15:56

high-level overview of AI, clarifying that it is a broad umbrella term encompassing various techniques, including machine learning and deep learning. He explains:

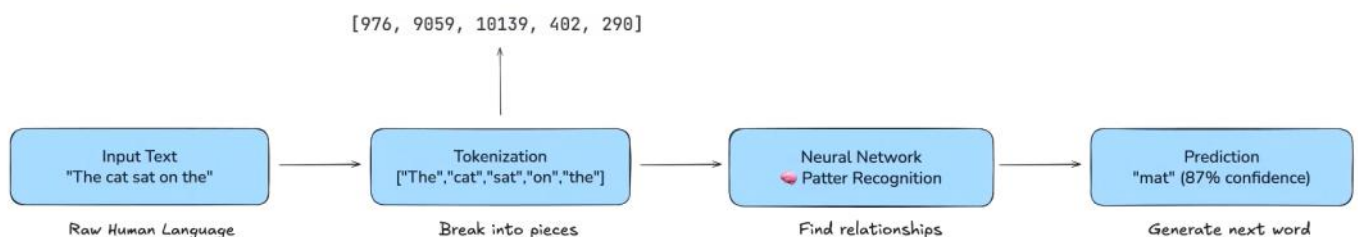
- **Machine Learning (ML):** The foundation of modern AI where systems learn patterns from data instead of explicit programming. ML includes:
 - Supervised learning (using labeled data for classification/regression)
 - Unsupervised learning (pattern discovery like clustering)
 - Reinforcement learning (learning through interaction and feedback)
- **Deep Learning:** A subset of ML using artificial neural networks inspired by the human brain, consisting of layers that progressively extract features (edges, shapes, complex parts) to perform tasks like image recognition.

Artificial Intelligence



- **Large Language Models (LLMs):** A breakthrough architecture introduced in 2017 by Google Brain, based on the Transformer model with attention mechanisms enabling models to understand context in text and predict the next word. LLMs are trained on massive datasets (books, web content, code) and can generate coherent, human-like text. Dan explains tokenization—breaking text into tokens (pieces of words), which are numerical representations processed by the model.

HOW DO LLMS ACTUALLY WORK?



Key Insight: LLMs are fundamentally *pattern matching machines* that predict the most likely next word based on patterns learned from billions of text examples.

The Training Process

1. Pre-training (Foundation)

Massive text datasets (books, web, code)
Learn general language patterns
Predict next word billions of times
Develops broad knowledge base

2. Fine-Tuning (Specialization)

Task-specific datasets
Human feedback training
Safety and alignment
Conversational abilities

Developer Takeaway: LLMs aren't programmed with rules - they learn patterns from examples. This is why they can be creative but also unpredictable.

He clarifies the acronyms behind GPT: Generative (creates content), Pre-trained (trained on large data sets), Transformer (the model architecture).

GENERATIVE AI

Generative Pre-trained Transformer (GPT)

- **Now that we understand:**

- Transformers provide the architecture (how they process language)
- Pre-training on vast amounts of data teaches language understanding
- Generative capability allows them to create new content

- **Key Capabilities:**

- Text generation (writing, translation, summarization)
- Code generation and analysis
- Complex reasoning and problem-solving

3. Prompt Engineering

27 July 2025 16:19

Prompt engineering is described as a vital skill to communicate effectively with AI models. Dan compares it to giving clear instructions to a junior developer instead of vague commands. Good prompts provide:

- Clear context and specific instructions
- Examples (one-shot, few-shot prompting)
- Structured input using delimiters or XML tags
- Task decomposition (breaking large problems into smaller steps)

He demonstrates how better prompts yield better AI responses, emphasizing iteration and saving effective prompts for reuse. Dan also discusses voice-based prompt input using dictation tools, improving ease of interaction.

Bad Prompt:

- *Explain database indexing*

Good Prompt:

As an experienced Java developer, explain database indexing in 3 parts:

1. What indexes are and why they matter for performance

2. When to use clustered vs non-clustered indexes

3. One practical example of creating an index in PostgreSQL

Keep each section to 2-3 sentences and include one code example.

ADVANCED TECHNIQUES

- Zero-shot Prompting: Will not contain examples or demonstrations
- One-shot prompting: Providing a single example
- Few-shot prompting: Providing Examples
- Chain-of-Thought: Think step by step
- XML Tags: help structure complex requests by clearly separating different types of information
- Task Decomposition: Breaking complex problems into steps

ZERO-SHOT PROMPTING

is asking the AI to perform a task without providing any examples. There is nothing wrong with this approach and there are times when this will give you exactly what you are looking for.

```
Create a Java class that implements the Observer pattern
for a stock price monitoring system.
```

ONE-SHOT PROMPTING

provides a single example to establish a pattern you would like the AI to follow:

```
// Here's an example of the coding style I prefer:
public Optional<User> findUserByEmail(String email) {
    if (email == null || email.trim().isEmpty()) {
        return Optional.empty();
    }
    return userRepository.findByEmail(email.toLowerCase());
}
```

FEW-SHOT PROMPTING

provides multiple examples to guide the AI's response.

Classify the sentiment of these customer reviews:

Example 1: "The software is intuitive and saves me hours of work" → Positive

Example 2: "Great documentation and excellent support team" → Positive

Example 3: "Buggy interface and crashes frequently" → Negative

Now classify: "The new features are helpful but the UI is confusing"

CHAIN-OF-THOUGHT PROMPTING

Bad Prompt:

•Optimize this code. [paste your code here]

Good Prompt:

I need to optimize this Java method that searches through a large dataset.

First, help me identify the current time complexity. Then suggest specific

improvements. Finally, show me the refactored code with comments explaining the performance gains.

[paste your code here]

XML TAGS

help structure complex requests by clearly separating different types of information.

```
<task>
  Create a RESTful API endpoint for user management
</task>

<requirements>
  - POST /users for creating users
  - Include validation for email and password
  - Return appropriate HTTP status codes
  - Use Spring Boot annotations
</requirements>

<constraints>
  - Java 17
  - Spring Boot 3.0
  - No external dependencies beyond Spring starter
</constraints>
```


TASK DECOMPOSITION

breaks down complex problems into manageable pieces.

I'm building a file processing system in Java. Help me break this down:

Phase 1: Design the file reading strategy (stream vs batch)

Phase 2: Create the data validation logic

Phase 3: Implement error handling and logging

Phase 4: Add unit tests

Start with Phase 1 - analyze the pros and cons of each approach for processing 1GB+ files.

PRACTICAL TIPS FOR IMMEDIATE IMPROVEMENT

- **Be Specific About Context:** Always provide relevant background information. If you're working on a Spring Boot application, mention it. If you're dealing with legacy code, say so.
- **Use Examples:** Show the AI what good output looks like. If you want code formatted a certain way, provide an example.
- **Give Context About Your Environment:** Mention your Java version, frameworks you're using, or constraints you're working within.
- **Specify a Role When Applicable:** Give the AI a specific persona or expertise to embody. For example, "Act as a senior Java architect reviewing this code" or "As a performance optimization expert, analyze this query." This helps frame the response with the appropriate level of detail and perspective.
- **Iterate and Refine:** Don't expect perfection on the first try. Use the AI's response to refine your prompt and get closer to your desired output.
- **Use AI to Improve Your Prompts:** When you're not getting the results you want, ask the AI to help you craft a better prompt. Try something like: "I'm trying to get you to help me debug this performance issue, but your response wasn't quite what I needed. Can you suggest how I should rephrase my request to get more specific debugging steps?"
- **Save Good Prompts:** When you craft a prompt that works well, save it. You'll likely need similar requests in the future.

4. Why Java and Spring AI?

27 July 2025 16:47

Dan explains the distinction between building AI models (often done in Python) and consuming AI capabilities, which any language, including Java, can do. Java's enterprise prevalence (90% of Fortune 500 companies rely on Java) makes it essential to integrate AI into Java applications. Spring AI offers a portable, vendor-agnostic API to consume AI services (chat, image, audio, video) without locking into a specific model or provider.

He demonstrates how simple it is to call AI APIs using curl or native Java HTTP clients but stresses that Spring AI goes beyond basic REST calls by solving real-world integration challenges in production applications, like model abstraction, streaming responses, and multiple model usage.

JAVA & AI

Leveraging Artificial Intelligence in Java Applications

- Why AI + Java?
 - The world of software is experiencing widespread adoption of Artificial Intelligence
 - Java is the language of enterprises, creating Java + AI apps is a new requirement
- Spring AI
 - Provides the necessary API access and components for developer AI applications
 - Abstraction similar to Spring Data
- Use Cases
 - Q&A Over docs
 - Documentation Summarization
 - Text, Code, Image, Audio and Video Generation



```
=====
```

**** Request and response using Bash ****

Request ->

```
#!/bin/bash
```

```
echo "Calling Open AI..."
```

```
MY_OPENAI_KEY="YOUR_API_KEY_HERE"
```

```
PROMPT="Tell me an interesting fact about Java"
```

```
curl https://api.openai.com/v1/chat/completions \
```

```
-H "Content-Type: application/json" \
```

```
-H "Authorization: Bearer $MY_OPENAI_KEY" \
```

```
-d '{"model": "gpt-4o", "messages": [{"role": "user", "content": "'"$PROMPT"'"}] }'
```

Response ->

```
{
  "id": "chatcmpl-ABNbjZ5oRbo720evnCX2arPufJCYK",
  "object": "chat.completion",
  "created": 1727275719,
  "model": "gpt-4o-2024-05-13",
  "choices": [
    {
      "index": 0,
      "message": {
        "role": "assistant",
        "content": "Sure! Did you know that Java was initially designed with interactive television in mind? James Gosling and his team at Sun Microsystems started the project in 1991 under the name \"Oak.\" The name was later changed to \"Java\" after discovering there was already a programming language called Oak. Java's versatility has made it one of the most popular programming languages for a wide range of applications, far beyond its initial intended use for TV set-top boxes!",
        "refusal": null
      },
      "logprobs": null,
      "finish_reason": "stop"
    }
  ],
  "usage": {
    "prompt_tokens": 14,
    "completion_tokens": 90,
    "total_tokens": 104,
    "completion_tokens_details": {
      "reasoning_tokens": 0
    }
  }
},
"system_fingerprint": "fp_e375328146"
```

=====

** Request and response using Bash **
Request ->

```
public static void main(String[] args) throws IOException, InterruptedException {
    var apiKey = "YOUR_API_KEY_HERE";
    var body = ""
        {
            "model": "gpt-4o",
            "messages": [
                {
                    "role": "user",
                    "content": "Tell me an interesting fact about Java"
                }
            ]
        }
    };

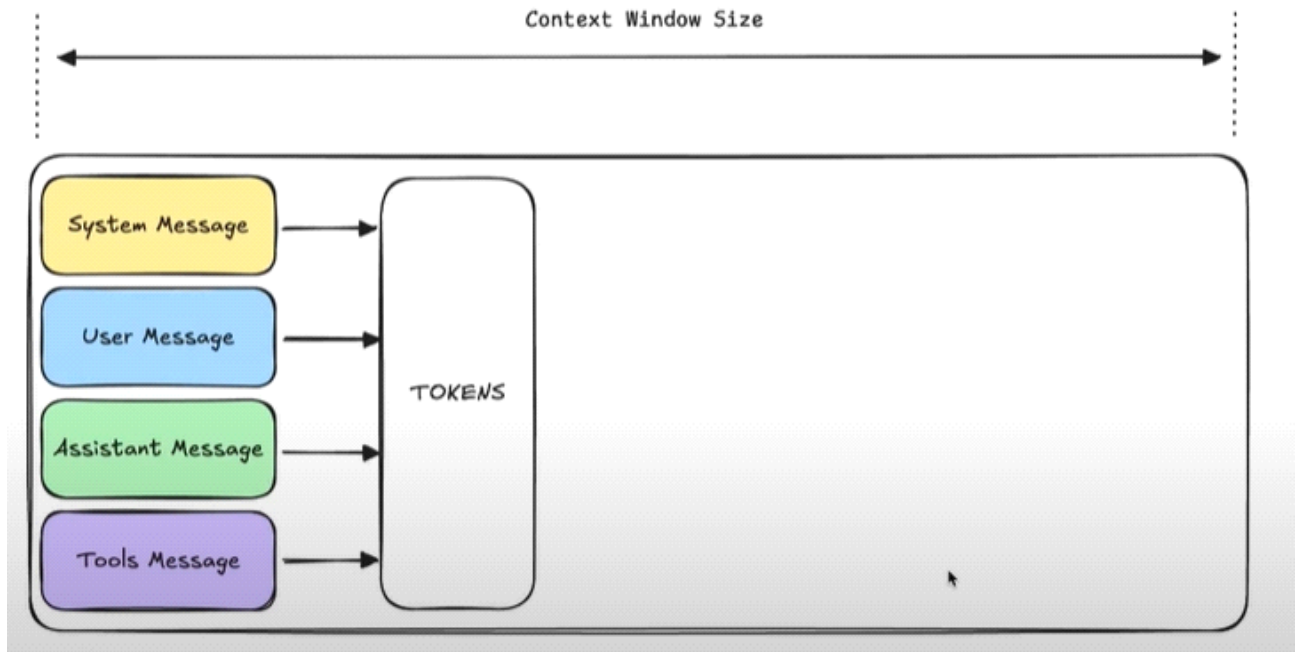
    HttpRequest request = HttpRequest.newBuilder()
        .uri(URI.create("https://api.openai.com/v1/chat/completions"))
        .header("Content-Type", "application/json")
        .header("Authorization", "Bearer " + apiKey)
        .POST(HttpRequest.BodyPublishers.ofString(body))
        .build();

    var client = HttpClient.newHttpClient();
    var response = client.send(request, HttpResponse.BodyHandlers.ofString());
    System.out.println(response.body());
}
```

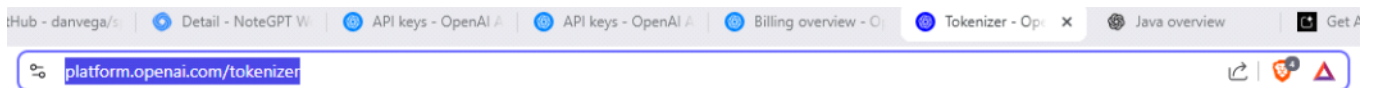
API key and tokens

27 July 2025 17:11

Context Window



<https://platform.openai.com/tokenizer>



You can use the tool below to understand how a piece of text might be tokenized by a language model, and the total count of tokens in that piece of text.

GPT-4o & GPT-4o mini GPT-3.5 & GPT-4 GPT-3 (Legacy)

tell me about java

Clear Show example

Tokens Characters
4 18

tell me about java

Text Token IDs

A helpful rule of thumb is that one token generally corresponds to ~4 characters of text for common English text. This translates to roughly ¾ of a word (so 100 tokens ~= 75 words).

platform.openai.com/tokenizer



You can use the tool below to understand how a piece of text might be tokenized by a language model, and the total count of tokens in that piece of text.

GPT-4o & GPT-4o mini GPT-3.5 & GPT-4 GPT-3 (Legacy)

tell me about java

Clear

Show example

Tokens	Characters
4	18

[145692, 668, 1078, 1557]

Text Token IDs

Flash

application.properties x

```
1 spring.application.name=flash
2 spring.ai.openai.chat.base-url=https://generativelanguage.googleapis.com
3 spring.ai.openai.chat.completions-path=/v1beta/openai/chat/completions
4 spring.ai.openai.api-key=
5 spring.ai.openai.chat.options.model=gemini-2.0-flash
```

5. Getting Started with Spring AI

27 July 2025 17:04

Dan walks through the practical steps to get started:

- Obtaining API keys from providers like OpenAI or Anthropic.
- Creating a new Spring Boot project via Spring Initializr with Spring AI dependencies.
- Configuring API keys securely using environment variables.
- Building simple REST endpoints that send prompts to chat models and receive responses.
- Demonstrating both blocking and streaming response modes for chat clients.
- Handling multiple AI models in one application using Spring's dependency injection and qualifiers.

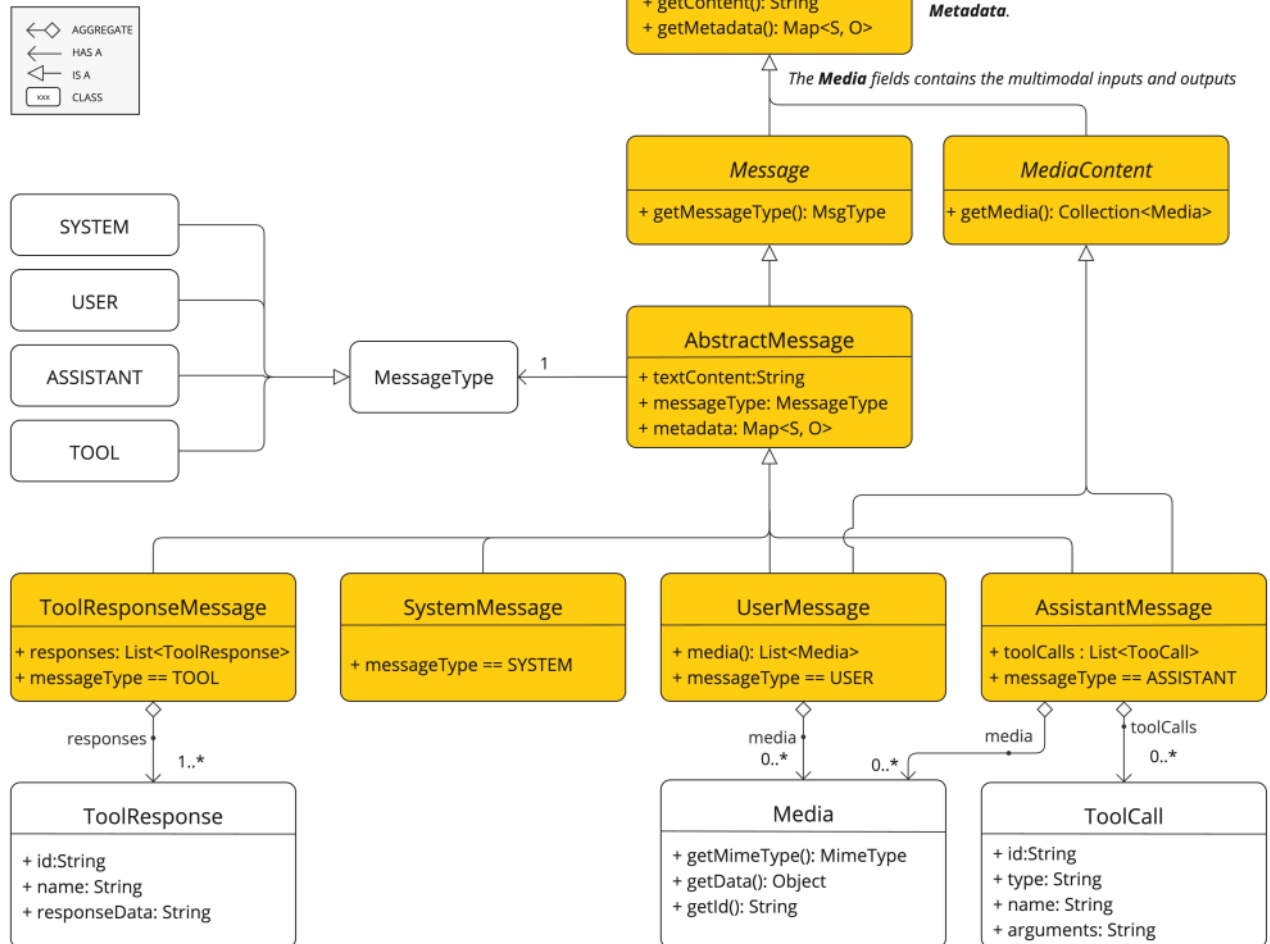
He also covers pricing considerations, explaining token usage and context window sizes, which significantly affect costs in high-throughput applications.

6. Spring AI Features: Prompts, Structured Output, Multimodality, and Chat Memory

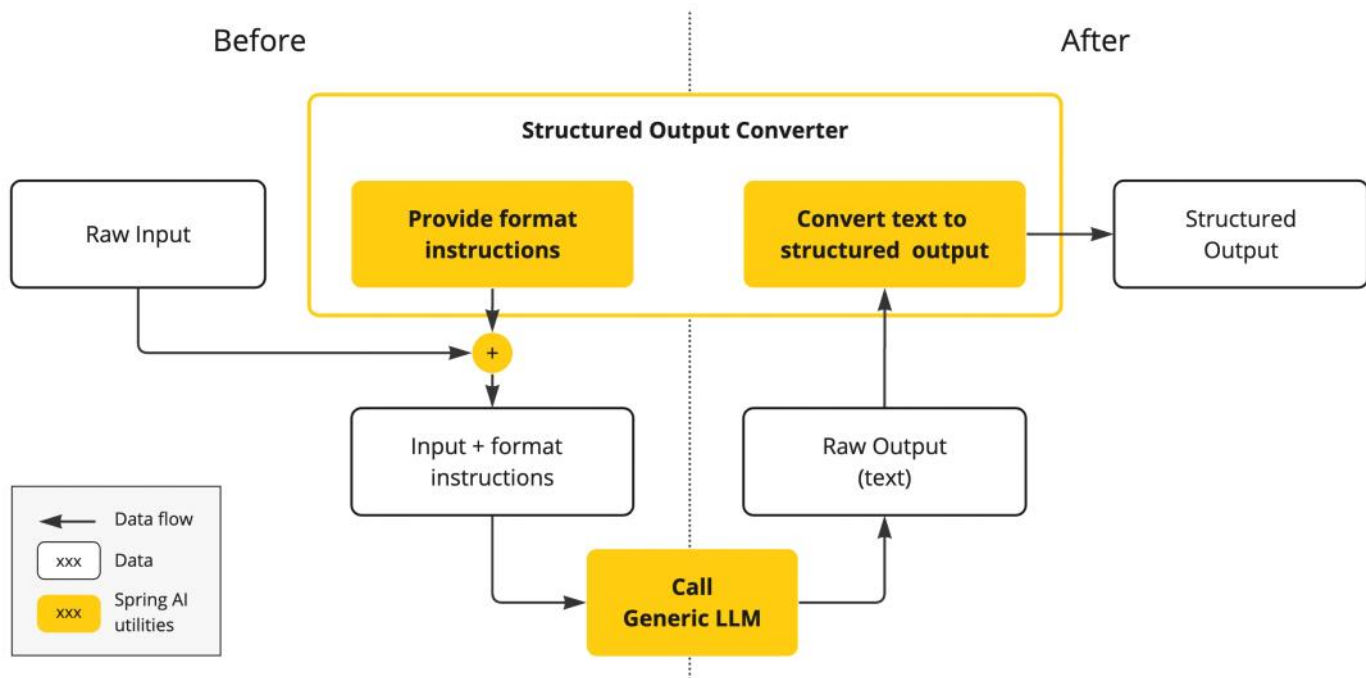
28 July 2025 00:21

- **Prompt Templates and System Messages:** shows how to use system messages to set the AI's role, establish guardrails, or provide consistent context for multiple user queries, improving control over AI behavior.

Spring AI Message API



- **Structured Output:** Instead of receiving freeform text, developers can instruct AI to produce structured JSON or Java objects. Spring AI automates serialization/deserialization, enabling reliable downstream processing.













- **Multimodality:** Spring AI supports not only text but images and audio. demonstrates how to send images as input for AI to describe and how to generate images and audio (text-to-speech) with minimal code.
- **Chat Memory:** Since AI models are stateless, Spring AI offers chat memory abstractions to maintain conversational state across requests (in-memory or persistent repositories). demonstrates a simple chat memory example to recall user information.





7. Overcoming Limitations of Large Language Models (LLMs)

28 July 2025 16:15

LLM limitations such as hallucinations (confidently incorrect answers), stale knowledge (fixed training data cutoff), bias, non-determinism, privacy risks, and token cost. To address these, he presents a “Swiss Army Knife” approach:

Limitation	Quick Note – “what this looks like”
 Hallucinate	Invents facts or API names with total confidence
 Stale Data	Knowledge cutoff means “today’s stock price?” → “_(ツ)_/”
 Bias & Safety	Outputs stereotypes, toxic language, or policy violations
 Domain Gaps	Uses generic wording where niche jargon is required
 Context Window	Long threads get truncated; model “forgets” earlier details
 Non-Determinism	Same prompt, different answer → flaky tests, review churn
 Privacy Leak	Proprietary/PII text could leave your trusted boundary
 Cost & Latency	High-token chains drain budget and slow UX
 Weak Reasoning	Multi-step calculations or logical deductions fail
 Low Explainability	Hard to audit: “Why did you choose that answer?”

- **Prompt Guarding:** Using system messages to restrict AI’s scope (e.g., only banking-related queries).
- **Prompt Stuffing:** Injecting static but relevant context into prompts to ground answers.
- **Retrieval Augmented Generation (RAG):** Using vector databases to fetch relevant documents dynamically, adding precise, up-to-date context.
- **Tools and Function Calling:** Allowing AI to invoke external APIs or services to get fresh data or perform actions.
- **Model Context Protocol (MCP):** A protocol to build modular AI “agents” that integrate tools and data sources in complex workflows.

#	Lever	Purpose
1	 Prompt Guarding	Encode rules that constrain the model’s behavior (tone, honesty, refusal policy).
2	 Prompt Stuffing / RAG	Inject fresh, task-specific context so the model quotes facts instead of guessing.
3	 Tools / Function Calling	Let the model invoke code or APIs for real-time data, calculations, or business logic.
4	 MCP (Resources + Tools)	Package those tools as reusable, versioned endpoints every client can share.

8. Retrieval Augmented Generation (RAG) with Spring AI

29 July 2025 00:13

explains RAG as a two-phase process:

1. **Ingestion:** Splitting documents or data into chunks, embedding them into vector representations, and storing them in vector databases.
2. **Query:** Transforming user queries into embeddings, performing similarity searches in the vector store to retrieve relevant chunks, and appending this context to prompts sent to the LLM.

Spring AI provides built-in support for vector stores (in-memory, pgvector, Neo4j, Pinecone, Milvus, etc.) and advisors that automatically handle RAG workflows. how to configure and use this in a Spring application.

9. Tools and Function Calling

29 July 2025 00:14

Tools enable models to extend capabilities by calling external APIs or running business logic:

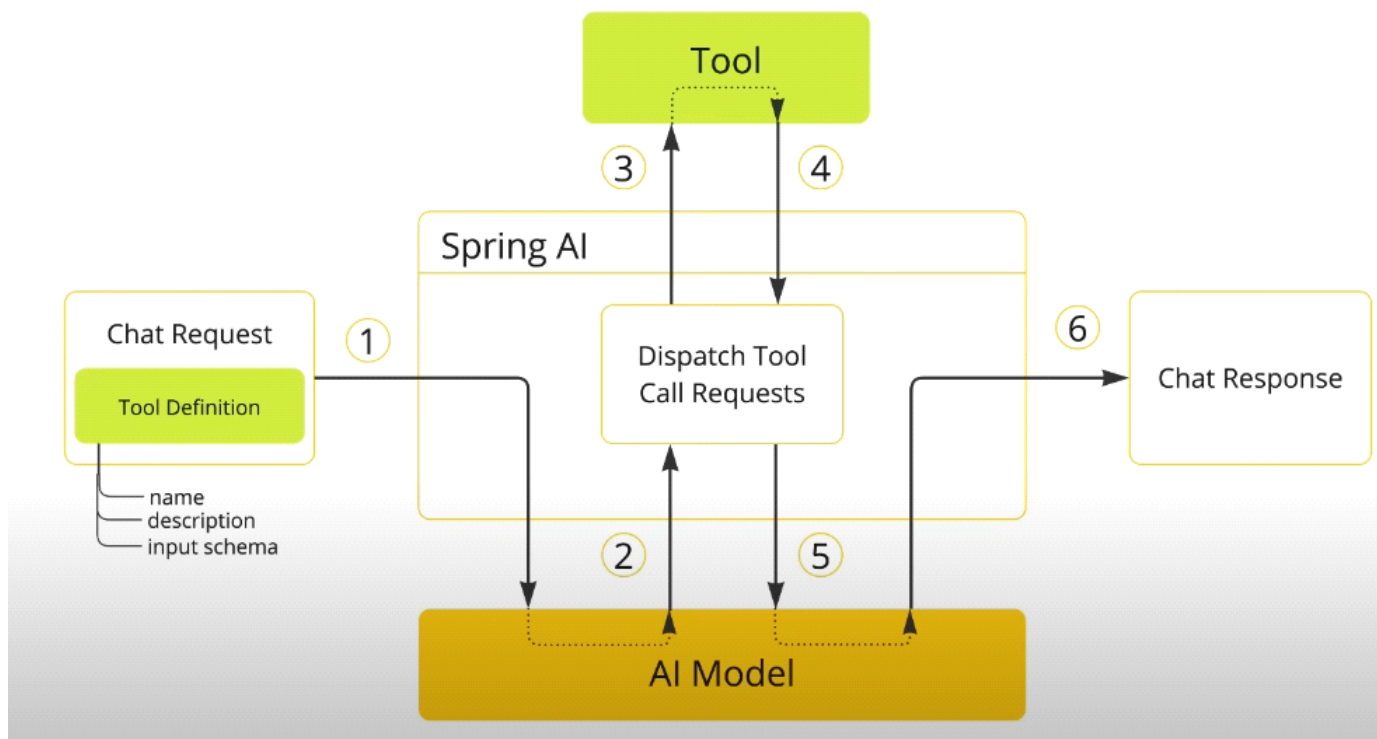
- Tools can retrieve real-time data (weather, stock prices) or perform actions (create tasks, send emails).
- Spring AI provides the `@Tool` annotation to define tools with descriptive metadata.
- The model requests tool invocation with input parameters; the application executes and returns results.
- Dan demonstrates tools for date/time, task management, and weather data, showing how AI uses them to provide accurate, up-to-date responses.

Information Retrieval

Tools in this category can be used to retrieve information from external sources, such as a database, a web service, a file system, or a web search engine. The goal is to augment the knowledge of the model, allowing it to answer questions that it would not be able to answer otherwise. As such, they can be used in Retrieval Augmented Generation (RAG) scenarios. For example, a tool can be used to retrieve the current weather for a given location, to retrieve the latest news articles, or to query a database for a specific record.

Taking Action

Tools in this category can be used to take action in a software system, such as sending an email, creating a new record in a database, submitting a form, or triggering a workflow. The goal is to automate tasks that would otherwise require human intervention or explicit programming. For example, a tool can be used to book a flight for a customer interacting with a chatbot, to fill out a form on a web page, or to implement a Java class based on an automated test (TDD) in a code generation scenario.



10. Model Context Protocol (MCP)

29 July 2025 22:11

MCP is a client-server protocol designed to orchestrate complex AI workflows and share tools across clients and servers. Features include:

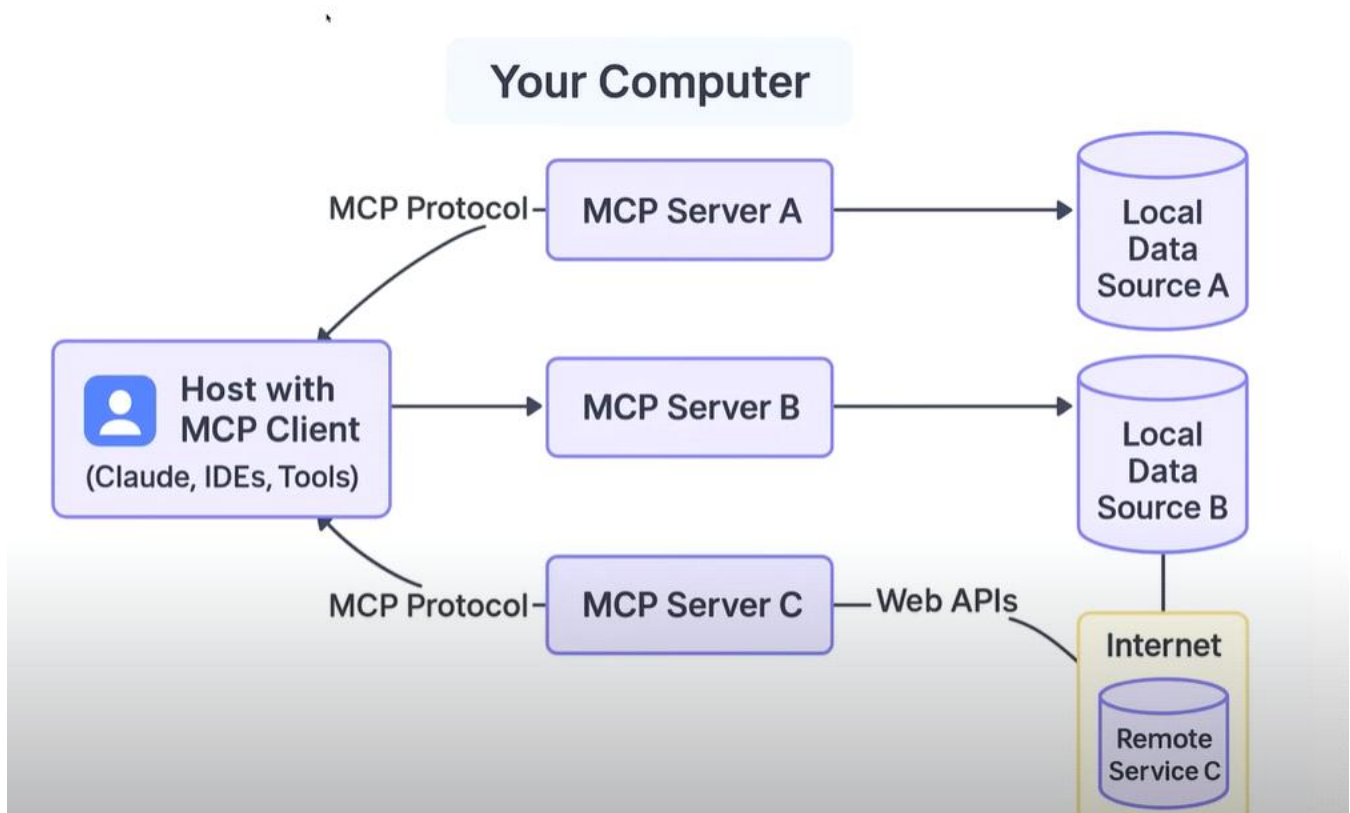
- Language-agnostic architecture enabling interoperability.
- Multiple transports (stdio, server-sent events).
- Security support via OAuth2.
- Pre-built integrations with GitHub, Slack, Google Maps, and others.
- Dan builds two example MCP servers exposing conference session data, demonstrating tool registration, JSON data loading, and client interaction via MCP Inspector and Claude Desktop.

He also shares a real-world MCP server for managing his Beehive newsletter, showcasing how AI can generate markdown from CMS content, automate workflows, and integrate with various clients.

WHY MCP?

MCP helps you build agents and complex workflows on top of LLMs. LLMs frequently need to integrate with data and tools, and MCP provides:

- A growing list of pre-built integrations that your LLM can directly plug into
- The flexibility to switch between LLM providers and vendors
- Best practices for securing your data within your infrastructure



TRANSPORTS

Standard Input/Output (stdio)

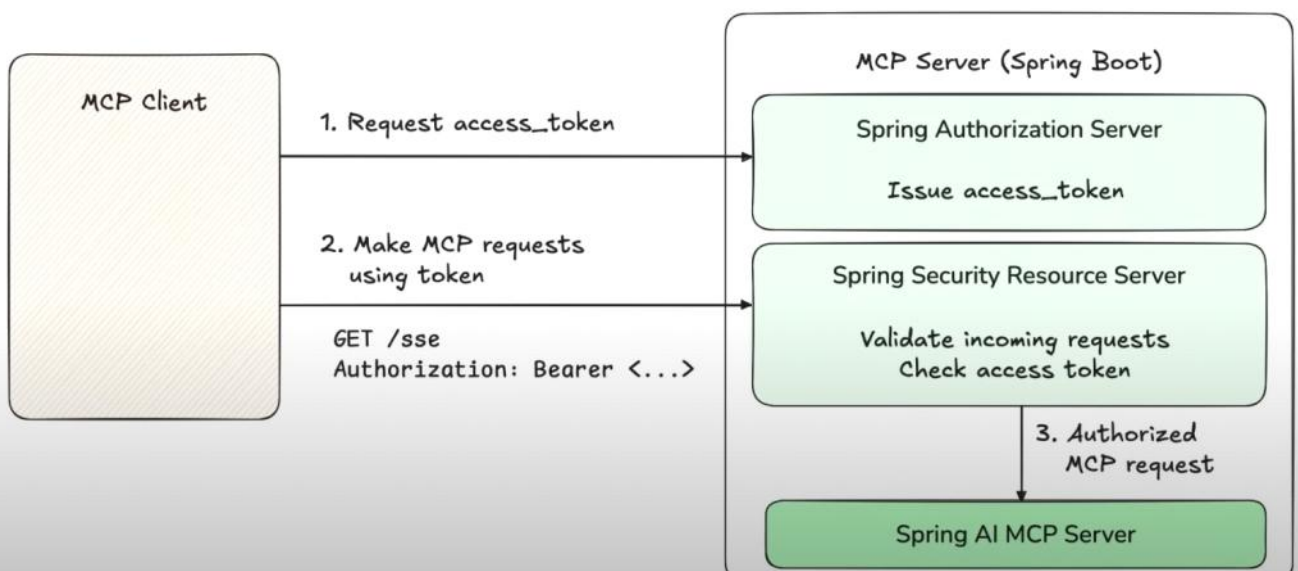
- Use stdio when:
 - Building command-line tools
 - Implementing local integrations
 - Needing simple process communication
 - Working with shell scripts

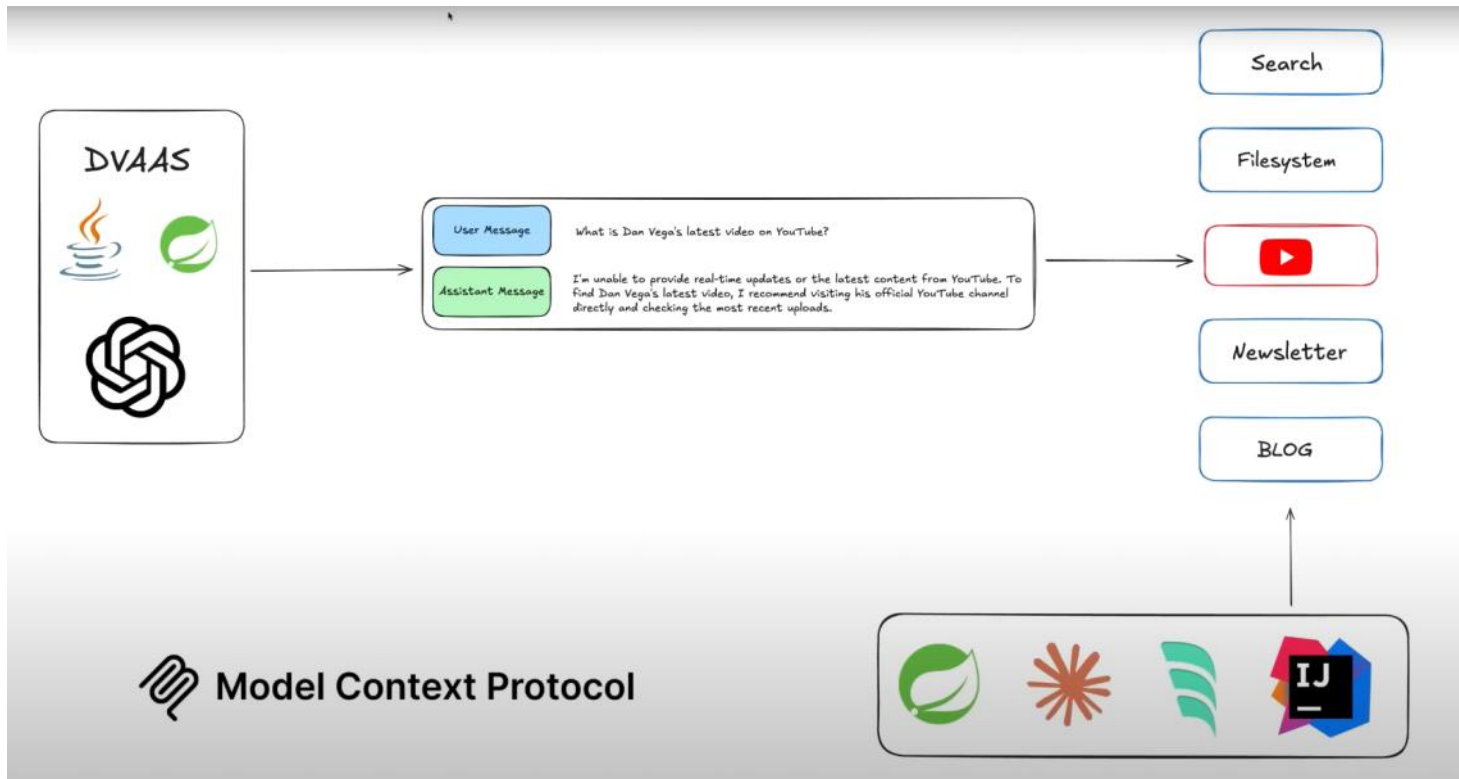
Server-Sent Events (SSE)

- Use SSE when:
 - Only server-to-client streaming is needed
 - Working with restricted networks
 - Implementing simple updates

- **Security Challenge:** While local MCP servers (STDIO transport) may not need authentication, enterprise HTTP deployments require robust security and permission management
- **OAuth2 Integration:** New MCP spec (2025-03-26) leverages OAuth2 framework - MCP server acts as both Resource Server (validates tokens) and Authorization Server (issues tokens)
- **Implementation Requirements:**
 - Add Spring Security & Spring Authorization Server Dependencies
 - Configure OAuth2 client credentials in application.properties
 - Create SecurityFilterChain to handle authentication and token validation

SECURING MCP SERVERS





=====
Application.properties

STDIO -

```

pStdioServerApplication.java  pom.xml (spring-io-mcp-stdio-server)
1  spring.application.name=spring-io-mcp-stdio-server
2  spring.main.web-application-type=none
3  spring.ai.mcp.server.stdio=true
4  spring.ai.mcp.server.name=spring-io-sessions-mcp
5  spring.ai.mcp.server.version=0.0.1
6
7  #disable banner and logging for stdio transport
8  spring.main.banner-mode=off
9  logging.level.root=OFF

```

SSE -

Project **spring-io-mcp-stdio** ~/Downloads/spring-io-mcp-stdio

application.yaml

```
1 spring:
2   ai:
3     mcp:
4       server:
5         enabled: true
6         name: spring-io-2025
7         version: 1.0.0
8         type: SYNC
9         instructions: "This server provides session catalog data for Spring I/O 2025"
10        capabilities:
11          resource: true
12          tool: true
13          prompt: false
14          completion: false
15
16      server:
17        port: 8085
18
19      management:
20        endpoints:
21          web:
22            exposure:
23              include: "*"
24
25      logging:
26        level:
27          org.springframework.ai.mcp: DEBUG
28          io.modelcontextprotocol: DEBUG
29          root: INFO
```

Project **spring-io-mcp-stdio** ~/Downloads/spring-io-mcp-stdio

application.yaml

```
1 spring:
2   ai:
3     mcp:
4       server:
5         enabled: true
6         name: spring-io-2025
7         version: 1.0.0
8         type: SYNC
9         instructions: "This server provides session catalog data for Spring I/O 2025"
10        capabilities:
11          resource: false
12          tool: true
13          prompt: false
14          completion: false
15
16      server:
17        port: 8085
18
19      logging:
20        level:
21          org.springframework.ai.mcp: DEBUG
22          io.modelcontextprotocol: DEBUG
23          root: INFO
```


11. Open Source vs Proprietary Models

02 August 2025 23:38

Dan contrasts proprietary models (OpenAI, Anthropic, Google Gemini) with open-source alternatives (Llama, Gemma, Mistral):

- **Open Source Models:**

- Offer transparency, auditability, and control over data privacy.
- Can be self-hosted locally or on private cloud, avoiding data leakage and API costs.
- Require more maintenance, infrastructure, and expertise.
- Often trail proprietary models on complex tasks but rapidly improve.
- Community-driven with faster iteration on bugs and improvements.
- Examples of tools to run open-source models: Olama CLI, Docker Model Runner, Open Web UI, LM Studio, Hugging Face inference endpoints.

demonstrates running a local Gemma 3 model with Olama CLI and integrating it into Spring AI by configuring the base URL and model name.

=====

Open weights and open source are distinct approaches in AI model development, particularly for large language models (LLMs). Open source provides full transparency, sharing the model's architecture, training code, datasets (when available), and weights.

Open weights, on the other hand, only release the trained model parameters (weights), without necessarily sharing the underlying code, training data, or full architecture details.

An AI model is open source, it means more than just free to use. We borrow this software uh open source definition. Anyone can run, study, modify, and share all the artifacts under an OSI approved license.

=====

OPEN SOURCE MODELS - KEY CHALLENGES

- Performance Gap vs Frontier Models
- Up-Front Hardware & Ops Overhead
- No Vendor-Grade Support or SLA
- Safety & Quality Risks (Hallucinations/Bias)
- Security & Supply-Chain Exposure
- License & IP Ambiguity
- Benchmark / Eval Burden Rests on You
- Rapid-Fire Releases - Maintenance Churn

HOW TO EVALUATE OPEN SOURCE MODELS

- Licence & "Openness" – OSI-approved? any commercial-use limits?
- Capability Benchmarks – MMLU, HellaSwag, GSM-8K scores; check the Open LLM Leaderboard
- Tool / Function-Calling Support – built-in schemas for agents & Spring AI's callable() interface
- Size vs Hardware Budget – parameter count, VRAM need, quantised variants
- Context Window – 8 K, 128 K... even 1 M tokens for long-doc RAG
- Latency & Throughput – tokens/sec locally vs on-prem GPU or HF Endpoint
- Community & Release Cadence – active issues, model-card updates, patch frequency
- Security & Supply Chain – reproducible weights, SBOM, no suspicious commits

AI-local Model usage

03 August 2025 00:01

1. Ollama

Ollama created by meta and used for running AI model locally.

Steps -

1. Download ollama from it site. - <https://ollama.com/>
2. We can run it on CLI and also on webUI (interface like chatGPT) - <https://docs.openwebui.com/>

Commands for CLI -

1. ollama list
2. ollama run gemma3:4b

a.

```
C:\Users\gauty>ollama list
NAME          ID          SIZE      MODIFIED
gemma3:4b     a2af6cc3eb7f 3.3 GB    45 seconds ago

C:\Users\gauty>ollama run gemma3:4b
>>> Hi What is my name ?
As an AI, I don't know your name! You haven't told me. 😊
```

3.

openWebUI -

1. It requires docker desktop so that its container can connect to localhost.
2. Docker also supports and provide functionality for local-AI-model connections - <https://docs.docker.com/ai/model-runner/>

12. Observability in Spring AI

03 August 2025 01:14

the importance of observability for AI applications, especially in production:

- Spring Boot 3 revamped observability with Micrometer and OpenTelemetry support.
- Key observability pillars: **Metrics** (token usage, latency, costs), **Logs** (prompt and response tracking with PII redaction), and **Traces** (end-to-end request flows).
- Spring AI exposes Micrometer metrics for AI usage (token counts, latency, success rates).
- Dan walks through a sample app with Prometheus and Grafana using Docker Compose to monitor AI metrics visually.
- He demonstrates how to collect, query, and visualize token usage, response times, and error rates in dashboards.

WHY OBSERVABILITY

- Cost can spike → need token & \$ metrics
- • LLMs are nondeterministic → traces & logs for debug
- • Safety / legal → need evidence when things go wrong

THE 3+1 PILLARS OF OBSERVABILITY

- Metrics – latency, token counts, cost
- Logs – structured prompt / response (PII-redacted)
- Traces – end-to-end view of RAG pipeline
- Evaluations – automated quality checks (FactCheckingEvaluator)

WHAT SPRING GIVES YOU FOR FREE

- Micrometer metric names (gen_ai.usage.*, gen_ai.client.latency) (docs.spring.io)
- • Observation events → Micrometer Tracing / OpenTelemetry
- • Pluggable log masking (PII filter)
- • GenerationListener hook for custom metrics

13. Model Evaluation and Testing

03 August 2025 17:41

Testing AI outputs is challenging due to non-determinism. Dan introduces Spring AI's evaluation framework:

- **Deterministic tests:** For tasks like sentiment analysis or intent detection with predictable outputs.
- **Non-deterministic evaluation:** Using AI to evaluate AI outputs based on relevancy and factual accuracy.
- Key evaluators:
 - **Relevancy Evaluator:** Checks if the response matches the user query context, useful in RAG scenarios.
 - **Fact-checking Evaluator:** Verifies claims against source documents to spot hallucinations.
- Dan demonstrates writing unit and integration tests for sentiment classification, structured output, relevancy, and fact-checking using Spring Boot Test and Testcontainers (for running local LLMs like Llama).

SPRING AI EVALUATION FRAMEWORK

Core Interface

```
@FunctionalInterface
public interface Evaluator {
    EvaluationResponse evaluate(EvaluationRequest evaluationRequest);
}
```

Evaluation Request Components

- **userText** - Original user input
- **dataList** - Context data (e.g., from RAG)
- **responseContent** - AI model's response

TWO KEY EVALUATORS

Relevancy Evaluator

- Purpose: Is there AI response relevant to the user's question?
- Best for:
 - RAG (Retrieval Augmented Generation)
 - Ensuring responses stay on topic
 - Quality Control for chatbots

Fact Checking Evaluator

- Purpose: Is the AI response factually accurate?
- Best for:
 - Detecting hallucinations
 - Verifying claims against source material
 - Content validation

FACT CHECKING EVALUATOR DEEP DIVE

What it does:

- Verifies claims against provided documents
- Detects factual inaccuracies and hallucinations
- Can use specialized models like *Bespoke-Minichack*
 - Accurate, Small, Fast & Cost-effective

Evaluation Format:

```
Document: {context}  
Claim: {ai_response}
```

TESTING DETERMINISTIC AI TASKS

Classification Tasks

More predictable outcomes = Traditional testing approaches

Examples:

- Sentiment analysis (positive/negative/neutral)
- Content Moderation (safe/unsafe)
- Intent detection (question/request/complaint)

TESTING DETERMINISTIC AI TASKS

@Test

```
void testSentimentClassification() {  
    String positiveText = "I love this product!";  
    String result = classifySentiment(positiveText);  
  
    assertThat(result).isEqualTo("POSITIVE");  
}
```

Key Difference: Expected outcomes are known and consistent

14. Conclusion: Key Takeaways and Resources

03 August 2025 18:03

key lessons:

- AI is a powerful tool, but success depends on effective prompt engineering—clear, contextual, and iterative communication.
- Spring AI provides a portable, extensible Java API for integrating AI without vendor lock-in.
- Understand and mitigate LLM limitations through prompt guarding, RAG, tools, and MCP.
- Observability and evaluation are critical for production readiness.
- Start building simple projects immediately to learn by doing.

He shares valuable resources including:

- The Spring AI workshop GitHub repo with course code.
- Spring AI reference documentation.
- Spring AI community projects and curated lists.
- Podcasts, newsletters, and upcoming books on Spring AI.
- His personal website and social media for ongoing support.

KEY TAKEAWAYS

- **AI is a Tool to be Mastered:** Large Language Models are powerful but not magical. They are pattern-matching machines that predict the next word. Your success in using them depends on your ability to communicate effectively through well-crafted prompts.
- **Prompt Engineering is a Developer Skill:** Don't just command the AI; teach it what you want. Providing context, examples, and a clear structure to your prompts will dramatically improve the quality of the response. Start saving your effective prompts as you create them.
- **Spring AI is Your Gateway:** Spring AI simplifies the process of integrating AI into your Java applications by providing a portable API that can work across different AI providers. This means you can build with AI without being locked into a single vendor.
- **Don't Trust, Verify:** LLMs can "hallucinate" and provide incorrect information with confidence. You must build guardrails. Use techniques like Retrieval Augmented Generation (RAG) to ground the model with your specific data, and implement evaluation checks to validate the accuracy and relevance of AI responses.