LangChain for Java in Quadim

on **noviembre 29, 2023** in **LangChain4j** with **No hay comentarios.**

LangChain for Java: Supercharge your Java application with the power of LLMs

At Quadim we have worked with langchain4j. The goal of this project is to simplify

the integration of AI/LLM capabilities into your Java application.

Below we are going to show you some simple examples of using this library so that you can use it in your projects.

Source Code: https://github.com/joedayz/Quadim-AI-Assisted-Functionality-Service

Prerequisite

You need an OpenAI API Key. In this link you will find detailed information to obtain your API Key.

In the source code, you will find the ApiKeys class, which is where you will use your API KEY.

```
public class ApiKeys {

 public static final String MY_OPENAI_API_KEY ="<YOU MUST HERE PUT YOUR API KEY>";

}
```

Assistant Mode

In the **AiAssistedHRAssistantTest** we are going to use a prompt that will answer as if it were a virtual HR assistant in Quadim.

Below I will explain the source code in parts:

```
@Test
public void testChatWithHRAI() throws Exception {
 Random r = new Random();
 int userNo = r.nextInt(100);
 // Plan
 ChatLanguageModel model = OpenAiChatModel.builder()
   .apiKey(ApiKeys.MY_OPENAI_API_KEY)
   .modelName(OpenAiModelName.GPT_3_5_TURBO)
```

```
    .timeout(ofSeconds(900))
    .temperature(0.9)
    .build();
```

1. **OpenAiChatModel.builder()**: This creates a new builder for the OpenAiChatModel class.
2. **.apiKey(ApiKeys.MY_OPENAI_API_KEY)**: Sets the API key for the OpenAI model. You need to replace ApiKeys.MY_OPENAI_API_KEY with your actual OpenAI API key.
3. **.modelName(OpenAiModelName.GPT_3_5_TURBO)**: Sets the model name to GPT-3.5 Turbo. This specifies the version of the OpenAI language model you want to use.
4. **.timeout(ofSeconds(900))**: Sets the timeout for the API call to 900 seconds (15 minutes). This means that if the API call takes longer than 15 minutes, it will be aborted.
5. **.temperature(0.9)**: Sets the temperature parameter for sampling. A higher temperature (e.g., 0.9) makes the output more random, while a lower temperature (e.g., 0.2) makes the output more focused and deterministic.
6. **.build()**: Builds the ChatLanguageModel instance with the specified configurations.

After executing this code, you will have a ChatLanguageModel instance named model configured with the specified parameters. You can then use this model to generate language-based responses using the LangChain library in Java.

```
Assistant assistant = AiServices.builder(Assistant.class)
    .chatLanguageModel(model)
    .chatMemoryProvider(memoryId ->
MessageWindowChatMemory.withMaxMessages(10))
    .build();
```

1. **AiServices.builder(Assistant.class)**: This creates a builder for the Assistant class within the AiServices utility.
2. **.chatLanguageModel(model)**: Sets the ChatLanguageModel for the assistant. The model here is the ChatLanguageModel instance that you created in the previous code block.

3. **.chatMemoryProvider(memoryId -> MessageWindowChatMemory.withMaxMessages(10))**: Sets the chat memory provider for the assistant. It uses a lambda expression to create a MessageWindowChatMemory with a maximum of 10 messages. This means that the assistant will keep track of the conversation history, and in this case, it will retain the last 10 messages.
4. **.build()**: Builds the Assistant instance with the specified configurations.

After executing this code, you will have an Assistant instance named assistant configured with the specified language model (model) and chat memory provider. This Assistant can then be used to interact with the language model and manage conversation history.

```
// a) create types for retrieving skills objects from responses
SkillExtractor skillExtractor =
AiServices.create(SkillExtractor.class, model);
```

```
static class SkillReference {

  @Description("the name of this skill")
  private String name;
  @Description("description of this skill. please make it selling
and not more than 10 lines of text")
  private String description;

  //...
}
```

---

1. **AiServices.create(SkillExtractor.class, model)**: This creates an instance of the SkillExtractor class using the AiServices utility. It takes the SkillExtractor class as a parameter and the model (presumably the ChatLanguageModel instance) as another parameter.
2. **static class SkillReference**: This declares a static nested class named SkillReference. This class has two fields (name and description) with corresponding @Description annotations. These annotations might be used for

documentation or metadata purposes.

The SkillExtractor instance (skillExtractor) is likely to be used for extracting skills or features from the language model (model). The SkillReference class appears to be a structure for holding information about a skill, with name and description attributes.

```
// b) simulate a chat
String appendPrompt = "Answer acting as a friendly HR Consultant helping the user with his/her competence mapping, focussing on skills and projects."+
    "Structure the answer friendly and selling with bullets for discovered or suggested supporting skills and potential typical projects"+
    "where the user may have used those skills. " +
    "Limit answer to the most relevant 5 skills and top 8 projects";
```

```
String q1 = "Yes, I do work with Java and java microservices on the backend ";
System.out.println("me: " + q1);
String res1 = assistant.chat(userNo, q1 + appendPrompt);
System.out.println(res1);
Skill extractedSkills1 = skillExtractor.extractSkillFrom(res1);
System.out.println("\n\n1. Skill mapped:" + mapper.writerWithDefaultPrettyPrinter().writeValueAsString(extractedSkills1) + "\n\n");
```

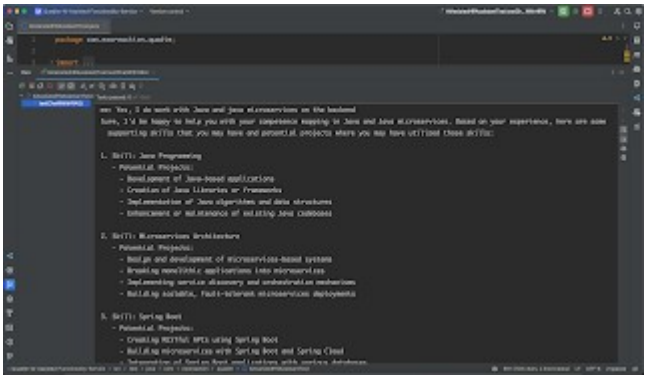
The assistant is defined this way:

```
interface Assistant {
```

```
    String chat(@MemoryId int memoryId, @UserMessage
String userMessage);
 }
```

So if we run the test, the AI assistant will tell us **what skills we need to have to develop with Java microservices**.

Result:



Below I show you the test log:

> *me: Yes, I do work with Java and java microservices on the backend*
> *Skills:*
> *- Java programming: Experience in Java programming is essential for working with Java microservices on the backend. This includes a deep understanding of object-oriented programming concepts, data structures, and algorithms.*
> *- Spring Framework: Knowledge of the Spring Framework is vital for developing Java microservices. This includes proficiency in Spring Boot, Spring Data, and Spring Cloud.*
> *- RESTful API development: Understanding how to design and develop RESTful APIs is necessary for creating microservices. This involves knowledge of*

HTTP, JSON, and API documentation tools like Swagger.
- Containerization and orchestration: Proficiency in containerization technologies like Docker and container orchestration platforms like Kubernetes is crucial for scaling and managing microservices in a distributed environment.
- Database management: Having experience with relational databases like MySQL or PostgreSQL, as well as NoSQL databases like MongoDB or Redis, is important for storing and retrieving data in microservices.

Projects:
1. Building a microservices-based e-commerce platform: Developing a scalable and fault-tolerant e-commerce platform using Java microservices, Spring Boot, and containerization technologies like Docker. Implementing RESTful APIs for product catalog management, order processing, and payment integration.

2. Creating a social media analytics system: Designing a system to analyze and process large volumes of social media data using Java microservices, Spring Cloud, and Apache Kafka for stream processing. Implementing sentiment analysis, trend detection, and user engagement metrics.

3. Developing a microservices-based banking application: Building a secure and highly available banking application using Java microservices, Spring Boot, and container orchestration with Kubernetes. Creating APIs for account management, transaction processing, and fraud detection.

4. Building a document management

system: Creating a system to store and manage documents using Java microservices, Spring Data, and Elasticsearch for full-text search capabilities. Implementing features like document versioning, access control, and document tagging.

5. Designing a real-time chat application: Developing a real-time messaging application using Java microservices, Spring Boot, and WebSocket technology. Implementing features like chat rooms, private messaging, and message history.

6. Building a microservices-based healthcare platform: Designing a platform for managing patient records, appointments, and healthcare providers using Java microservices, Spring Cloud, and a combination of relational and NoSQL databases. Implementing secure authentication, data encryption, and integration with external healthcare systems.

7. Creating a recommendation system: Developing a recommendation engine using Java microservices, Spring Boot, and machine learning algorithms. Implementing personalized recommendations based on user preferences, purchase history, and browsing behavior.

8. Building a microservices-based travel booking system: Designing a system for managing travel bookings, flight reservations, and hotel accommodations using Java microservices, Spring Cloud, and messaging queues like RabbitMQ. Implementing features like real-time availability updates, payment processing, and itinerary generation.

Note: These suggested projects are not exhaustive but provide a range of examples where the mentioned skills could be utilized.

1. Skill mapped:{
"name" : "Java programming",
"description" : "Java programming is a crucial skill for working with Java microservices on the backend. With a deep understanding of object-oriented programming concepts, data structures, and algorithms, you'll be able to develop robust and efficient microservices. Java's versatility and extensive libraries make it a popular choice among developers worldwide, ensuring ample resources and community support.",
"listOfCandidateSkillDefinitions" : [{
"name" : null,
"description" : null
}, {
"name" : null,
"description" : null
}, {
"name" : null,
"description" : null
}]
}

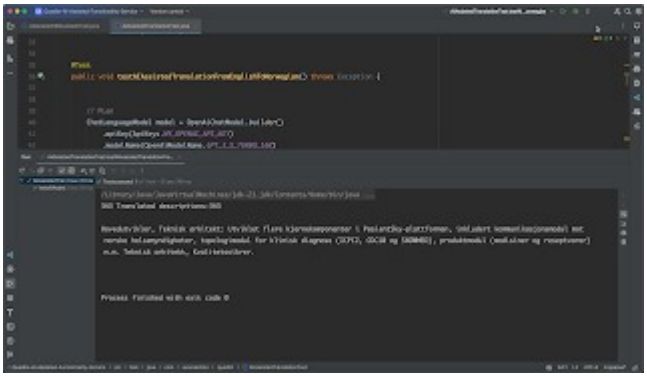
Translator Mode

In the **AiAssistedTranslationTest** we will see how to test the language translation of our virtual assistant.

```
@Test
  public void testAIAssistedTranslationFromEnglishToNorwegian() throws
```

```
Exception {


    // PLan
    ChatLanguageModel model = OpenAiChatModel.builder()
      .apiKey(ApiKeys.MY_OPENAI_API_KEY)
      .modelName(OpenAiModelName.GPT_3_5_TURBO_16K)
      .timeout(ofSeconds(900))
      .temperature(0.2)
      .build();


    String initialProjectDescription = "Chief Developer, Technical Architect:
Developed several core modules in PasientSky's platform, " +
      "including communication module against Norwegian public health
authorities, topology module for clinical diagnosis " +
      "(ICPC2, CDC10 and SNOWMED), product module (medicines and
prescription goods) m.m. Technical architect, Quality assurer. ";



    int n = 343;

    try {

      String res0 = model.generate( "Translate " + initialProjectDescription + " from
English to Norwegian");
      System.out.println(n + " Translated descriptions:" + n++ + "\n\n" + res0 +
"\n\n");
    } catch (Exception e) {
      System.out.println("Exception handling  - Stacktrace:" +
Arrays.toString(e.getStackTrace()));
    }
  }
```

In this example, we will see **how it can translate from English to Norwegian**. The result is:



Below I show you the test log:

> *Hovedutvikler, Teknisk arkitekt: Utviklet flere kjernekomponenter i PasientSky-plattformen, inkludert kommunikasjonsmodul mot norske helsemyndigheter, topologimodul for klinisk diagnose (ICPC2, CDC10 og SNOWMED), produktmodul (medisiner og reseptvarer) m.m. Teknisk arkitekt, Kvalitetssikrer.*

Parse PDF Resume and get skills

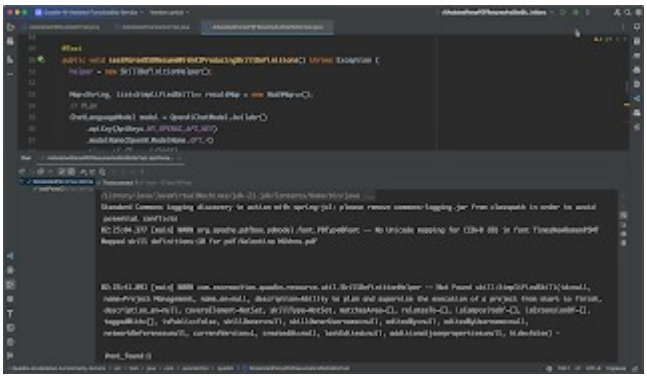
In this last example we search a directory for CVs in PDF format, we ask you to extract the skills in a summarized format and then compare it with a skills base found in skilldefinitions.json to see which skills you have and which you don't.

```
@Test
 public void
testParseOSDResumeWithAIProducingSkillDefinitions()
throws Exception {
   helper = new SkillDefinitionHelper();

   Map<String, List<SimplifiedSkill>> resultMap = new
HashMap<>();
   // PLan
   ChatLanguageModel model = OpenAiChatModel.builder()
      .apiKey(ApiKeys.MY_OPENAI_API_KEY)
      .modelName(OpenAiModelName.GPT_4)
      .timeout(ofSeconds(900))
```

```
        .temperature(0.6)
        .build();
    Assistant assistant = AiServices.builder(Assistant.class)
        .chatLanguageModel(model)
        .chatMemoryProvider(memoryId ->
MessageWindowChatMemory.withMaxMessages(10))
        .build();


    // a) Get list of PDF Resumes
    List<Resource> resourceList = getPDFResources();


    String appendPrompt = "Extract the users skills from this
resume. Present the result as structured " +
        "json data in the following json format " +
jsonSkillDefinition + " keep the name of the skill short";

    int n = 0;
    boolean RUN_FULL_REGRESSON = true;
    int found_and_swapped = 0;
    int not_found = 0;
    if (RUN_FULL_REGRESSON) {
      for (Resource resource : resourceList) {
        try {
          File file = resource.getFile();
          PDDocument document = Loader.loadPDF(file);
          PDFTextStripper stripper = new PDFTextStripper();
          String text = stripper.getText(document);
          //System.out.println(n + " Input data extracted from pdf
resume:\n" + text + "\n\n");
          String res0 = assistant.chat(n, text + appendPrompt);
          //System.out.println(n + " Generated JSON
SkillDefinitions:" + n++ + "\n\n" + res0 + "\n\n");
          List<SimplifiedSkill> simplifiedSkills =
mapper.readValue(res0, new
TypeReference<List<SimplifiedSkill>>() {
          });
          System.out.println("Mapped skill definitions:" +
simplifiedSkills.size() + " for pdf:" + resource.getFilename() +
"\n\n\n");
          List<SimplifiedSkill> enhancedSkillList =
helper.getEnhancedSkillDefinitions(simplifiedSkills);
          resultMap.put(resource.getFilename(), simplifiedSkills);
          resultMap.put(resource.getFilename() + "-enhanced",
```

```
enhancedSkillList);

      } catch (Exception e) {
        System.out.println("Exception handling " +
resource.getFilename() + " - Stacktrace:" +
Arrays.toString(e.getStackTrace()));
      }
    }
  }
  System.out.println(resultMap);

 }
```

As you can see, the virtual assistant returns the information as indicated. Amazing.



The log:

```
[
 {
  "name": "C language",
  "description": "Proficiency in C language, a popular
programming language in software development.",
  "isPublic": false,
  "currentVersion": 1,
  "additionaljsonproperties": null,
  "public": false
 },
 {
  "name": "Javascript",
  "description": "Knowledge in Javascript, a programming
language used primarily for web development.",
```

```
    "isPublic": false,
    "currentVersion": 1,
    "additionaljsonproperties": null,
    "public": false
  },
  {
    "name": "HTML",
    "description": "Proficiency in HTML, a standard language for
designing and creating websites.",
    "isPublic": false,
    "currentVersion": 1,
    "additionaljsonproperties": null,
    "public": false
  },
  {
    "name": "CSS",
    "description": "Knowledge in CSS, a style sheet language used
for describing the look and formatting of a document written
in HTML.",
    "isPublic": false,
    "currentVersion": 1,
    "additionaljsonproperties": null,
    "public": false
  },
  {
    "name": "Microsoft Excel",
    "description": "Skills in Microsoft Excel, a spreadsheet
program used to store and process data.",
    "isPublic": false,
    "currentVersion": 1,
    "additionaljsonproperties": null,
    "public": false
  },
  {
    "name": "Microsoft Word",
    "description": "Proficiency in Microsoft Word, a word
processing software used to create, edit, and print documents.",
    "isPublic": false,
    "currentVersion": 1,
    "additionaljsonproperties": null,
    "public": false
  },
  {
    "name": "PowerPoint",
    "description": "Skills in PowerPoint, a presentation program
```

used for creating slideshow presentations.",
"isPublic": false,
"currentVersion": 1,
"additionaljsonproperties": null,
"public": false
},
{
"name": "Graphic Design",
"description": "Experience in freelance graphic design, including creating social media, logos, and image editing.",
"isPublic": false,
"currentVersion": 1,
"additionaljsonproperties": null,
"public": false
},
{
"name": "Customer Service",
"description": "Experience in customer service roles, including working as a waiter, cashier, and attendant.",
"isPublic": false,
"currentVersion": 1,
"additionaljsonproperties": null,
"public": false
}
]

Conclusion

We can integrate this library into our programs and enjoy the power of LLMS. The possibilities are endless.

Enjoy!

Jose

Share:

Related Posts:

LangChain for Java
in Quadim

Entrada más reciente Página Principal Entrada antigua

0 COMENTARIOS:

PUBLICAR UN COMENTARIO

JOEDAYZ.PE

ABOUT

code igniter (2)

code2cloud (1)

codeigniter (1)

comparabien.com (1)

computacion (1)

continuos integration (1)

CoronaVirus (1)

cuba (1)

cursos (1)

darkside (1)

datagrip (1)

deltaspike (1)

dew (1)

docker (1)

DulceAmorPeru (1)

e-commerce (1)

English (1)

EntityFramework (2)

EPEUPC (3)

eureka (2)

evangelios (1)

eventos (3)

exoreaction (2)

facebook (1)

familia (2)

farmaciaperuanas (1)

firebase (2)

► **2018** (22)

► **2017** (23)

► **2014** (5)

► **2013** (21)

► **2012** (28)

► **2011** (44)

► **2010** (28)

► **2009** (27)

► **2008** (20)

► **2007** (16)

► **2006** (11)

► **2005** (6)

firebase-admin (1)

flutter (2)

functions (1)

gcp (1)

git (1)

github (2)

golang (1)

google-format (1)

google-style (1)

grails (5)

groovy (3)

hangouts (1)

highchart-export-server (2)

huacho (1)

hudson (1)

hyperledger-composer (1)

hyperledger-fabric (1)

i-educa (1)

iBATIS (2)

icescrum (1)

informatica (1)

Intigas (1)

ITP_JAVA (1)

jakartaee (4)

jakartaee10 (1)

JasperReports (1)

java (3)

JavaCard (1)

JavaDayUNI (1)

JavaOne (1)

jhipster (2)

jmeter (1)

joedayz (46)

JOERP (4)

jpa (1)

jquery (1)

kafka (3)

kotlin (2)

Kubernetes (3)

LangChain4j (1)

lombok (1)

m2eclipse (1)

mac (2)

Matt Raible (1)

Maven (3)

microprofile (6)

microprofile-jwt jakartaee (2)

microprofile-jwt jdbc-realm jakartaee (1)

microprofile-jwt jdbc-realm payara (1)

microservicios (1)

Ministerio del Interior (1)

MJN (5)

móvil (1)

mysql (1)

namespaces (1)

navidad (1)

NET (4)

Nextel (1)

Novell (1)

ocjp (1)

Opentaps (2)

Oracle (1)

oraclecloud (1)

oraclefunctions (1)

oracleopenworld (1)

OSUM (1)

OSX (1)

p6spy (1)

Payara (5)

personal (1)

perujug jconfperu joedayz (2)

php (1)

play (1)

PMP (1)

podcasts (1)

PostgreSQL (8)

programacion (1)

pubsub (1)

PUCP (4)

quadim (2)

quarkus (1)

rackspace (1)

rails (2)

redis (1)

refactoring (2)

Reniec (1)

renovatebot (2)

Rider (1)

ruby (4)

rust (1)

scala (1)

SCD2010 (1)

SCJP (1)

Scrum (3)

Scrum evaluacion (1)

seminarios (1)

Setup (1)

SourceRepo (1)

spring (13)

spring 3.1 (1)

spring android (1)

spring mobile (1)

spring social (1)

spring-boot (9)

spring-boot-admin (2)

spring-cloud (1)

spring-cloud-config (2)

SpringCommunityDay (1)

SpringRoo (2)

springsource (1)

sqlserver (2)

start-up (1)

STS (1)

Subclipse (1)

Subversion (1)

SUN (1)

SUNAT (2)

synergyj (2)

Syscom (1)

Talleres (21)

Telefonica (1)

thedevconf (1)

thymeleaf (1)

Trac (1)

try-with-resources (1)

twitter (1)

Tye (1)

ubuntu (3)

UNI (3)

UNMSM (1)

UPC (1)

videos (1)

vimeo (1)

weblogic (2)

Workspace (1)

WPF (1)

xorcery (9)

xsd (1)

YaRetail (1)

YoMeQuedoEnCasa (1)

zuul (2)