OpenAI

OpenAI stands as a pioneering AI research organization, famous for its groundbreaking Large Language Models (LLMs) like GPT-3 and GPT-4, setting new benchmarks in natural language understanding and generation.

OpenAI's LLMs offer extensive support for:

- Tools facilitating seamless interaction between the LLM and applications.
- Document retrievers enabling the transmission of pertinent content to the LLM.

Using OpenAI Models

To employ OpenAI LLMs, integrate the following dependency into your project:

```
<dependency>
    <groupId>io.quarkiverse.langchain4j</groupId>
    <artifactId>quarkus-langchain4j-openai</artifactId>
    <version>0.14.1</version>
</dependency>
```

If no other LLM extension is installed, AI Services will automatically utilize the configured OpenAI model.

Configuration

Configuring OpenAI models mandates an API key, obtainable by creating an account on the OpenAI platform.

The API key can be set in the `application.properties` file:

```
quarkus.langchain4j.openai.api-key=sk-...
```

TIP

Alternatively, leverage the `QUARKUS_LANGCHAIN4J_OPENAI_API_KEY` environment variable.

Several configuration properties are available:

🔒 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
🔒 `quarkus.langchain4j.openai.chat-model.enabled` Whether the model should be enabled Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_ENABLED`	boolean	`true`
🔒 `quarkus.langchain4j.openai.embedding-model.enabled` Whether the model should be enabled Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_EMBEDDING_MODEL_ENABLED`	boolean	`true`
🔒 `quarkus.langchain4j.openai.moderation-model.enabled` Whether the model should be enabled Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_MODERATION_MODEL_ENABLED`	boolean	`true`
🔒 `quarkus.langchain4j.openai.image-model.enabled` Whether the model should be enabled Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_ENABLED`	boolean	`true`
`quarkus.langchain4j.openai.base-url` Base URL of OpenAI API Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_BASE_URL`	string	`https://api.openai.com/v1/`
`quarkus.langchain4j.openai.api-key` OpenAI API key Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_API_KEY`	string	`dummy`
`quarkus.langchain4j.openai.organization-id` OpenAI Organization ID (https://platform.openai.com/docs/api-reference/organization-optional) Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_ORGANIZATION_ID`	string	
`quarkus.langchain4j.openai.timeout` Timeout for OpenAI calls Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_TIMEOUT`	Duration ❓	`10s`

`quarkus.langchain4j.openai.max-retries` The maximum number of times to retry. 1 means exactly one attempt, with retrying disabled. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_MAX_RETRIES`	int	`1`
`quarkus.langchain4j.openai.log-requests` Whether the OpenAI client should log requests Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai.log-responses` Whether the OpenAI client should log responses Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai.enable-integration` Whether to enable the integration. Defaults to `true`, which means requests are made to the OpenAI provider. Set to `false` to disable all requests. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_ENABLE_INTEGRATION`	boolean	`true`
`quarkus.langchain4j.openai.chat-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_MODEL_NAME`	string	`gpt-3.5-turbo`
`quarkus.langchain4j.openai.chat-model.temperature` What sampling temperature to use, with values between 0 and 2. Higher values means the model will take more risks. A value of 0.9 is good for more creative applications, while 0 (argmax sampling) is good for ones with a well-defined answer. It is recommended to alter this or topP, but not both. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_TEMPERATURE`	double	`1.0`
`quarkus.langchain4j.openai.chat-model.top-p` An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with topP probability mass. 0.1 means only the tokens comprising the top 10% probability mass are considered. It is recommended to alter this or topP, but not both. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_TOP_P`	double	`1.0`

`quarkus.langchain4j.openai.chat-model.max-tokens` The maximum number of tokens to generate in the completion. The token count of your prompt plus max_tokens can't exceed the model's context length. Most models have a context length of 2048 tokens (except for the newest models, which support 4096). Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_MAX_TOKENS`	int	
`quarkus.langchain4j.openai.chat-model.presence-penalty` Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_PRESENCE_PENALTY`	double	`0`
`quarkus.langchain4j.openai.chat-model.frequency-penalty` Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_FREQUENCY_PENALTY`	double	`0`
`quarkus.langchain4j.openai.chat-model.log-requests` Whether chat model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai.chat-model.log-responses` Whether chat model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai.chat-model.response-format` The response format the model should use. Some models are not compatible with some response formats, make sure to review OpenAI documentation. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_CHAT_MODEL_RESPONSE_FORMAT`	string	
`quarkus.langchain4j.openai.embedding-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_EMBEDDING_MODEL_MODEL_NAME`	string	`text-embedding-ada-002`

`quarkus.langchain4j.openai.embedding-model.log-requests` Whether embedding model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_EMBEDDING_MODEL_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai.embedding-model.log-responses` Whether embedding model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_EMBEDDING_MODEL_LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai.embedding-model.user` A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_EMBEDDING_MODEL_USER`	string	
`quarkus.langchain4j.openai.moderation-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_MODERATION_MODEL_MODEL_NAME`	string	`text-moderation-latest`
`quarkus.langchain4j.openai.moderation-model.log-requests` Whether moderation model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_MODERATION_MODEL_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai.moderation-model.log-responses` Whether moderation model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_MODERATION_MODEL_LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai.image-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_MODEL_NAME`	string	`dall-e-3`
`quarkus.langchain4j.openai.image-model.persist` Configure whether the generated images will be saved to disk. By default, persisting is disabled, but it is implicitly enabled when `quarkus.langchain4j.openai.image-mode.directory` is set and this property is not to `false` Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_PERSIST`	boolean	`false`

`quarkus.langchain4j.openai.image-model.persist-directory` The path where the generated images will be persisted to disk. This only applies of `quarkus.langchain4j.openai.image-mode.persist` is not set to `false`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_PERSIST_DIRECTORY`	path	`${java.io.tmpdir}/dall-e-images`
`quarkus.langchain4j.openai.image-model.response-format` The format in which the generated images are returned. Must be one of `url` or `b64_json` Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_RESPONSE_FORMAT`	string	`url`
`quarkus.langchain4j.openai.image-model.size` The size of the generated images. Must be one of `1024x1024`, `1792x1024`, or `1024x1792` when the model is `dall-e-3`. Must be one of `256x256`, `512x512`, or `1024x1024` when the model is `dall-e-2`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_SIZE`	string	`1024x1024`
`quarkus.langchain4j.openai.image-model.quality` The quality of the image that will be generated. `hd` creates images with finer details and greater consistency across the image. This param is only supported for when the model is `dall-e-3`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_QUALITY`	string	`standard`
`quarkus.langchain4j.openai.image-model.number` The number of images to generate. Must be between 1 and 10. When the model is dall-e-3, only n=1 is supported. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_NUMBER`	int	`1`

`quarkus.langchain4j.openai.image-model.style` The style of the generated images. Must be one of `vivid` or `natural`. Vivid causes the model to lean towards generating hyper-real and dramatic images. Natural causes the model to produce more natural, less hyper-real looking images. This param is only supported for when the model is `dall-e-3`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_STYLE`	string	`vivid`
`quarkus.langchain4j.openai.image-model.user` A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_USER`	string	
`quarkus.langchain4j.openai.image-model.log-requests` Whether image model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai.image-model.log-responses` Whether image model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI_IMAGE_MODEL_LOG_RESPONSES`	boolean	`false`
Named model config	Type	Default
`quarkus.langchain4j.openai."model-name".base-url` Base URL of OpenAI API Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__BASE_URL`	string	`https://api.openai.com/v1/`
`quarkus.langchain4j.openai."model-name".api-key` OpenAI API key Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__API_KEY`	string	`dummy`
`quarkus.langchain4j.openai."model-name".organization-id` OpenAI Organization ID (https://platform.openai.com/docs/api-reference/organization-optional) Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__ORGANIZATION_ID`	string	

`quarkus.langchain4j.openai."model-name".timeout` Timeout for OpenAI calls Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__TIMEOUT`	Duration ❓	`10s`
`quarkus.langchain4j.openai."model-name".max-retries` The maximum number of times to retry. 1 means exactly one attempt, with retrying disabled. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__MAX_RETRIES`	int	`1`
`quarkus.langchain4j.openai."model-name".log-requests` Whether the OpenAI client should log requests Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai."model-name".log-responses` Whether the OpenAI client should log responses Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai."model-name".enable-integration` Whether to enable the integration. Defaults to `true`, which means requests are made to the OpenAI provider. Set to `false` to disable all requests. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__ENABLE_INTEGRATION`	boolean	`true`
`quarkus.langchain4j.openai."model-name".chat-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_MODEL_NAME`	string	`gpt-3.5-turbo`
`quarkus.langchain4j.openai."model-name".chat-model.temperature` What sampling temperature to use, with values between 0 and 2. Higher values means the model will take more risks. A value of 0.9 is good for more creative applications, while 0 (argmax sampling) is good for ones with a well-defined answer. It is recommended to alter this or topP, but not both. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_TEMPERATURE`	double	`1.0`

`quarkus.langchain4j.openai."model-name".chat-model.top-p` An alternative to sampling with temperature, called nucleus sampling, where the model considers the results of the tokens with topP probability mass. 0.1 means only the tokens comprising the top 10% probability mass are considered. It is recommended to alter this or topP, but not both. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_TOP_P`	double	`1.0`
`quarkus.langchain4j.openai."model-name".chat-model.max-tokens` The maximum number of tokens to generate in the completion. The token count of your prompt plus max_tokens can't exceed the model's context length. Most models have a context length of 2048 tokens (except for the newest models, which support 4096). Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_MAX_TOKENS`	int	
`quarkus.langchain4j.openai."model-name".chat-model.presence-penalty` Number between -2.0 and 2.0. Positive values penalize new tokens based on whether they appear in the text so far, increasing the model's likelihood to talk about new topics. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_PRESENCE_PENALTY`	double	`0`
`quarkus.langchain4j.openai."model-name".chat-model.frequency-penalty` Number between -2.0 and 2.0. Positive values penalize new tokens based on their existing frequency in the text so far, decreasing the model's likelihood to repeat the same line verbatim. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_FREQUENCY_PENALTY`	double	`0`
`quarkus.langchain4j.openai."model-name".chat-model.log-requests` Whether chat model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai."model-name".chat-model.log-responses` Whether chat model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_LOG_RESPONSES`	boolean	`false`

`quarkus.langchain4j.openai."model-name".chat-model.response-format` The response format the model should use. Some models are not compatible with some response formats, make sure to review OpenAI documentation. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__CHAT_MODEL_RESPONSE_FORMAT`	string	
`quarkus.langchain4j.openai."model-name".embedding-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__EMBEDDING_MODEL_MODEL_NAME`	string	`text-embedding-ada-002`
`quarkus.langchain4j.openai."model-name".embedding-model.log-requests` Whether embedding model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__EMBEDDING_MODEL_LOG_REQUESTS`	boolean	`false`
`quarkus.langchain4j.openai."model-name".embedding-model.log-responses` Whether embedding model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__EMBEDDING_MODEL_LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai."model-name".embedding-model.user` A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__EMBEDDING_MODEL_USER`	string	
`quarkus.langchain4j.openai."model-name".moderation-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__MODERATION_MODEL_MODEL_NAME`	string	`text-moderation-latest`
`quarkus.langchain4j.openai."model-name".moderation-model.log-requests` Whether moderation model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__MODERATION_MODEL_LOG_REQUESTS`	boolean	`false`

`quarkus.langchain4j.openai."model-name".moderation-model.log-responses` Whether moderation model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__MODERATION_MODEL_LOG_RESPONSES`	boolean	`false`
`quarkus.langchain4j.openai."model-name".image-model.model-name` Model name to use Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_MODEL_NAME`	string	`dall-e-3`
`quarkus.langchain4j.openai."model-name".image-model.persist` Configure whether the generated images will be saved to disk. By default, persisting is disabled, but it is implicitly enabled when `quarkus.langchain4j.openai.image-mode.directory` is set and this property is not to `false` Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_PERSIST`	boolean	`false`
`quarkus.langchain4j.openai."model-name".image-model.persist-directory` The path where the generated images will be persisted to disk. This only applies of `quarkus.langchain4j.openai.image-mode.persist` is not set to `false`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_PERSIST_DIRECTORY`	path	`${java.io.tmpdir}/dall-e-images`
`quarkus.langchain4j.openai."model-name".image-model.response-format` The format in which the generated images are returned. Must be one of `url` or `b64_json` Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_RESPONSE_FORMAT`	string	`url`
`quarkus.langchain4j.openai."model-name".image-model.size` The size of the generated images. Must be one of `1024x1024`, `1792x1024`, or `1024x1792` when the model is `dall-e-3`. Must be one of `256x256`, `512x512`, or `1024x1024` when the model is `dall-e-2`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_SIZE`	string	`1024x1024`

`quarkus.langchain4j.openai."model-name".image-model.quality` The quality of the image that will be generated. `hd` creates images with finer details and greater consistency across the image. This param is only supported for when the model is `dall-e-3`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_QUALITY`	string	`standard`
`quarkus.langchain4j.openai."model-name".image-model.number` The number of images to generate. Must be between 1 and 10. When the model is dall-e-3, only n=1 is supported. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_NUMBER`	int	`1`
`quarkus.langchain4j.openai."model-name".image-model.style` The style of the generated images. Must be one of `vivid` or `natural`. Vivid causes the model to lean towards generating hyper-real and dramatic images. Natural causes the model to produce more natural, less hyper-real looking images. This param is only supported for when the model is `dall-e-3`. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_STYLE`	string	`vivid`
`quarkus.langchain4j.openai."model-name".image-model.user` A unique identifier representing your end-user, which can help OpenAI to monitor and detect abuse. Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_USER`	string	
`quarkus.langchain4j.openai."model-name".image-model.log-requests` Whether image model requests should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_LOG_REQUESTS`	boolean	`false`

`quarkus.langchain4j.openai."model-name".image-model.log-responses` Whether image model responses should be logged Environment variable: `QUARKUS_LANGCHAIN4J_OPENAI__MODEL_NAME__IMAGE_MODEL_LOG_RESPONSES`	boolean	`false`

NOTE

About the Duration format

To write duration values, use the standard `java.time.Duration` format. See the Duration#parse() Java API documentation for more information.

You can also use a simplified format, starting with a number:

- If the value is only a number, it represents time in seconds.
- If the value is a number followed by `ms`, it represents time in milliseconds.

In other cases, the simplified format is translated to the `java.time.Duration` format for parsing:

- If the value is a number followed by `h`, `m`, or `s`, it is prefixed with `PT`.
- If the value is a number followed by `d`, it is prefixed with `P`.

Document Retriever

When utilizing OpenAI models, the recommended practice involves leveraging the `OpenAiEmbeddingModel`. If no other LLM extension is installed, retrieve the embedding model as follows:

```
@Inject EmbeddingModel model; // Injects the OpenAIEmbeddingModel
```

IMPORTANT

The `OpenAIEmbeddingModel` transmits the document to OpenAI for embedding computation.

Azure OpenAI

Applications can leverage the Azure's version of OpenAI services simply by using the `quarkus-langchain4j-azure-openai` extension instead of the `quarkus-langchain4j-openai` extension.

When this extension is used, the following configuration properties are required:

```
quarkus.langchain4j.azure-openai.resource-name=
quarkus.langchain4j.azure-openai.deployment-name=

# And one of the below depending on your scenario
quarkus.langchain4j.azure-openai.api-key=
quarkus.langchain4j.azure-openai.ad-token=
```

In the case of Azure, the `api-key` and `ad-token` properties are mutually exclusive. The `api-key` property should be used when the Azure OpenAI service is configured to use API keys, while the `ad-token` property should be used when the Azure OpenAI service is configured to use Azure Active Directory tokens.

In both cases, the key will be placed in the Authorization header when communicating with the Azure OpenAI service.

Advanced usage

`quarkus-langchain4j-openai` and `quarkus-langchain4j-azure-openai` extensions use a REST Client under the hood to make the REST calls required by LangChain4j. This client is however available for use in a Quarkus application in the same way as any other REST client.

An example usage is the following:

```
import java.net.URI;
import java.net.URISyntaxException;

import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;

import org.jboss.resteasy.reactive.RestStreamElementType;

import dev.ai4j.openai4j.chat.ChatCompletionChoice;
import dev.ai4j.openai4j.chat.ChatCompletionResponse;
import dev.ai4j.openai4j.chat.Delta;
import dev.ai4j.openai4j.chat.Message;
import dev.ai4j.openai4j.completion.CompletionChoice;
import dev.ai4j.openai4j.completion.CompletionResponse;
import io.quarkiverse.langchain4j.openai.OpenAiRestApi;
import io.quarkus.rest.client.reactive.QuarkusRestClientBuilder;
import io.smallrye.mutiny.Multi;


@Path("restApi")
@ApplicationScoped
```

```
public class QuarkusRestApiResource {

    private final OpenAiRestApi restApi;
    private final String token;

    public QuarkusRestApiResource() throws URISyntaxException {
        this.restApi = QuarkusRestClientBuilder.newBuilder()
                .baseUri(new URI("https://api.openai.com/v1/"))
                .build(OpenAiRestApi.class);
        this.token = "sometoken";

    }

    @GET
    @Path("language/streaming")
    @RestStreamElementType(MediaType.TEXT_PLAIN)
    public Multi<String> languageStreaming() {
        return restApi.streamingCompletion(
                createCompletionRequest("Write a short 1 paragraph funny poem about
Enterprise Java"), token, null)
                .map(r -> {
                    if (r.choices() != null) {
                        if (r.choices().size() == 1) {
                            CompletionChoice choice = r.choices().get(0);
                            var text = choice.text();
                            if (text != null) {
                                return text;
                            }
                        }
                    }
                    return "";
                });
    }
}
```

This example allows for streaming OpenAIs response back to the user’s browser as Server Sent Events.

NOTE

We used `null` as the `apiVersion` parameter in the call to `streamingCompletion` as this parameter is optional when using the vanilla OpenAI APIs. This parameter is only required when using Azure OpenAI.

Copyright (C) 2020-2024 Red Hat and individual contributors to Quarkiverse.