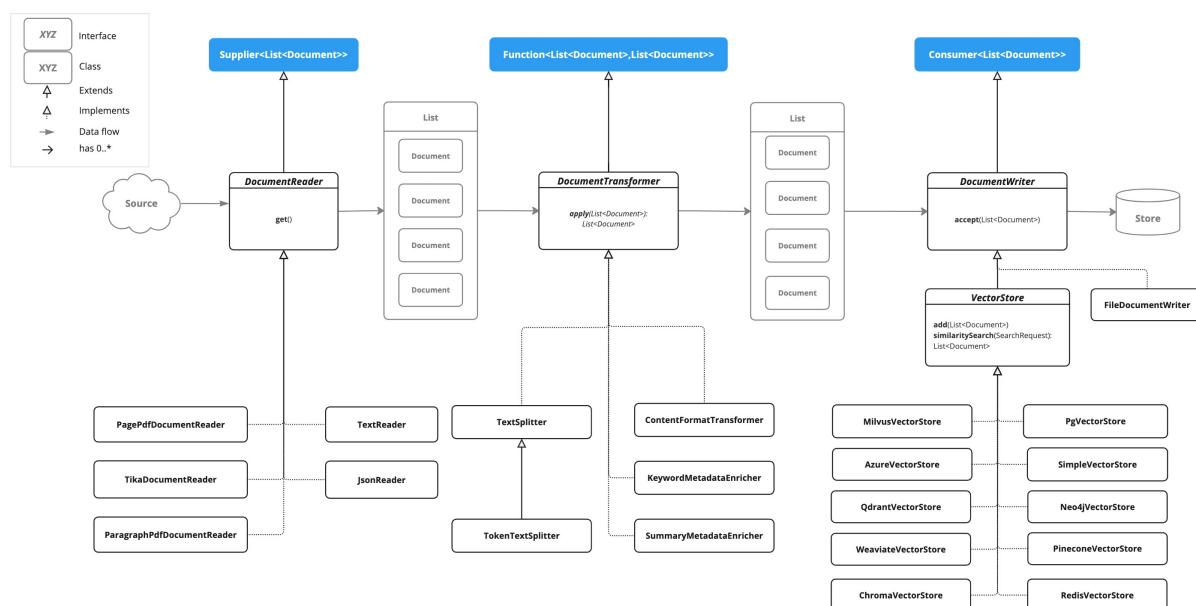


Day27 - 給向量化加上Buff-ETL(下)

嵌套俄羅斯娃娃

回顧一下 ETL Pipeline 中間那一塊



前面只用到 `TokenTextSplitter` 將大檔案切成小塊，今天來詳細的介紹這些工具有甚麼功用以及詳細的參數設定

TokenTextSplitter

前面用過預設的標準拆分，他還有個帶參數的建構子

```
TokenTextSplitter(int
    defaultChunkSize, int minChunkSizeChars, int minChunkLengthT
    maxNumChunks, boolean keepSeparator)
```

`defaultChunkSize`：每個文字區塊的目標大小（以 Token 為單位）（預設值：800）。

`minChunkSizeChars`：每個文字區塊的最小字元數（預設值：350）。

`minChunkLengthToEmbed`：要嵌入的區塊的最小長度（預設值：5）。

`maxNumChunks`：從內容產生的最大區塊數（預設值：10000）。

`keepSeparator`：是否在區塊中保留分隔符號（如換行符）（預設值：true）。

一般使用預設切割即可，想優化查詢結果在自行調整參數測試

KeywordMetadataEnricher

它能使用 AI 模型從文件內容中提取關鍵字，並將其添加為 metadata。剛開始以為這是甚麼黑科技，竟然能自己列出關鍵字，沒想到只是一個提示詞就能做到，下面是原始碼使用的提示詞

```
public static final String KEYWORDS_TEMPLATE = ""
    {context_str}. Give %s unique keywords for this
    document. Format as comma separated. Keywords: "";
```

修改昨天匯入 pdf 的程式碼

EtlService.java

```
private final ChatModel chatModel;
//從Document中取得10個關鍵字
List<Document> keywordDocuments(List<Document> documents) {
    KeywordMetadataEnricher keywordEnricher = new KeywordMeta
    return keywordEnricher.apply(documents);
}

public List<Document> loadPdfAsDocuments() throws IOException
    ResourcePatternResolver resolver = new PathMatchingResour
    Resource[] resources = new Resource[0];
    resources = resolver.getResources("./pdf/*.pdf");
    List<Document> docs = new ArrayList<>();
    for (Resource pdfResource : resources) {
        try {
            ParagraphPdfDocumentReader pdfReader = new ParagraphP
            docs.addAll(pdfReader.read());
        } catch (IllegalArgumentException e) {
            PagePdfDocumentReader pdfReader = new PagePdfDocu
            docs.addAll(pdfReader.read());
        }
    }
}
```

```

    }
}
return docs;
}

public void importPdf() throws IOException {
    TokenTextSplitter splitter = new TokenTextSplitter();
    vectorStore.write(splitter.split(keywordDocuments(loadPdf,
}

```

這一個串一個的程式有沒有像嵌套俄羅斯娃娃XD

看看測試結果

The screenshot shows the Neo4j web interface. On the left, there's a sidebar with icons for Graph, Table, Text, and Code. The main area displays a Cypher query: `neo4j$ MATCH (n) RETURN n`. Below the query, there's a graph visualization with several purple nodes. On the right, the 'Node properties' panel is open, showing the details of a selected node. The node is a 'Document' with the following properties:

<elementId>	4:33659fc2-0688-4cc4-8792-bf49076897d0:6
<id>	6
embedding	[-0.004015991929918528,-0.017162011936306953,-0.006384918931871653,-0.038770418614149094,-0.0279797725379467,0.026922397315502167,-0.026190368458628654... Show all]
id	80992573-b611-4740-91ed-89e7a45a7c57
metadata.excerpt_keywords	Spring, IDE, Spring Tools 4, Eclipse, Spring Starter Project, Java Version, Spring Boot, Maven, pom.xml, API key
metadata.file_name	Day2_ - 先拿把 AI 入門鑰匙_API_key.pdf
metadata.page_number	3
text	與 Open AI 一樣沒複製就查不到了

The 'text' property contains a snippet of text from a document, which is partially obscured by a red box in the original image.

這功能其實就是進階 RAG 的方法之一，透過關鍵字強化 embedding 數值，讓近似搜尋更為精準，另外也可作為推薦字或是進階篩選的功能

SummaryMetadataEnricher

它使用 AI 模型為文件建立摘要並新增為 metadata，它可以為當前 Document 以及相鄰 Document（上一個和下一個）產生摘要。

一樣先看一下原始碼中如何寫提示詞

```

public static final String DEFAULT_SUMMARY_EXTRACT_TEMPLATE =
    Here is the content of the section:
    {context_str}

```

```
Summarize the key topics and entities of the section.  
Summary:"";
```

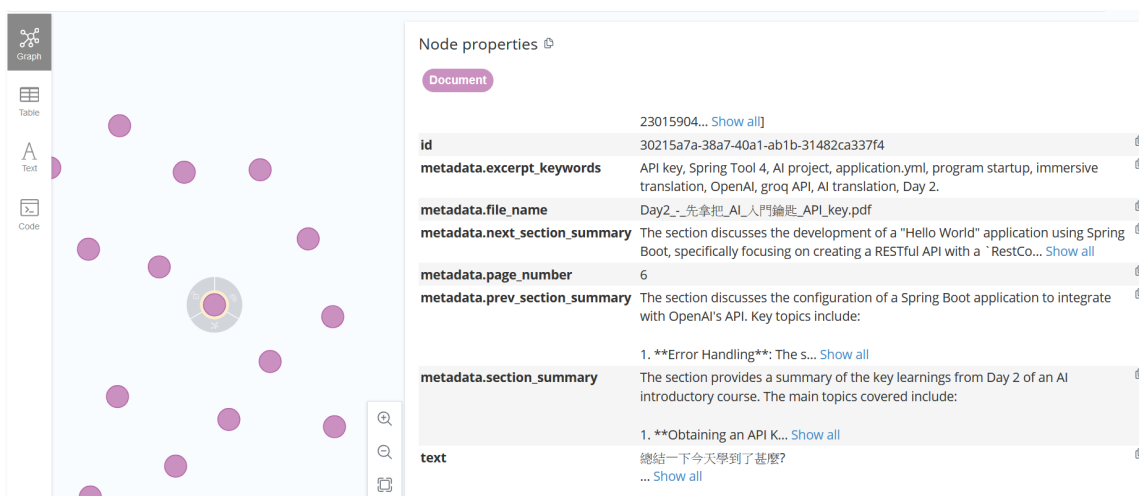
看起來也沒多複雜，學習程式內如何使用模板也是不錯的方式

EtlService.java: 一樣寫一個 summaryDocuments 函式，在嵌套進拆分前的程式中

```
List<Document> summaryDocuments(List<Document> documents) {  
    SummaryMetadataEnricher summaryEnricher = new SummaryMetadataEnricher(  
        List.of(SummaryType.PREVIOUS, SummaryType.CURRENT, SummaryType.NEXT)  
    );  
    return summaryEnricher.apply(documents);  
}  
  
public void importPdf() throws IOException {  
    TokenTextSplitter splitter = new TokenTextSplitter();  
    vectorStore.write(splitter.split(summaryDocuments(keyword)).iterator());  
}
```

老實說這樣寫有點陷入嵌套地獄了，之後應該就會改成 Advisor 的寫法

一樣直接看結果



The screenshot shows a document viewer interface. On the left, there's a sidebar with icons for Graph, Table, Text, and Code. The main area displays a graph with several purple nodes. On the right, a 'Node properties' panel is open, showing details for a 'Document' node. The properties include:

- id**: 23015904... [Show all](#)
- metadata.excerpt_keywords**: API key, Spring Tool 4, AI project, application.yml, program startup, immersive translation, OpenAI, groq API, AI translation, Day 2.
- metadata.file_name**: Day2_先把_AI_入門鑰匙_API_key.pdf
- metadata.next_section_summary**: The section discusses the development of a "Hello World" application using Spring Boot, specifically focusing on creating a RESTful API with a RestCo... [Show all](#)
- metadata.page_number**: 6
- metadata.prev_section_summary**: The section discusses the configuration of a Spring Boot application to integrate with OpenAI's API. Key topics include:
- metadata.section_summary**: The section provides a summary of the key learnings from Day 2 of an AI introductory course. The main topics covered include:
 - 1. **Error Handling**: The S... [Show all](#)
- text**: 總結一下今天學到了甚麼? [Show all](#)

除了上個範例的 metadata.excerpt_keywords，還多了三個摘要，內容分別如下

metadata.section_summary: The section provides a summary of the key learnings from Day 2 of an AI introductory course. The main topics covered include:

1. **Obtaining an API Key:** Instructions on how to acquire an API key necessary for accessing AI services.
2. **Creating an AI Project in Spring Tool 4:** Guidance on setting up an AI project using the Spring Tool 4 development environment.
3. **Configuring the API Key in application.yml:** Steps to integrate the obtained API key into the project's configuration file (application.yml) for successful program startup.
4. **Immersive Translation:** A discussion on custom immersive translation features compatible with OpenAI's API, specifically mentioning the ability to use the groq API for free AI translation services.

Overall, the section emphasizes practical skills for working with AI projects and utilizing APIs effectively.

metadata.prev_section_summary: The section discusses the configuration of a Spring Boot application to integrate with OpenAI's API. Key topics include:

1. **Error Handling:** The section begins with an error log indicating a `BeanCreationException` due to a missing OpenAI API key, which is necessary for the application to function properly.

2. **Configuration Setup:** It explains how to set the API key in the `application.yml` file, detailing the necessary structure for the configuration:

- The key should be placed under `spring: ai: openai: api-key: {your_api_key}`.
- It also mentions setting the chat model to `gpt-4o-mini` for cost-effective performance.

3. **Successful Initialization:** After configuring the API key, the application starts successfully without errors, demonstrating that the automatic configuration and bean creation for `openAiChatModel` is functioning correctly.

4. **Alternative Integration with Groq:** The section suggests that users who may not have credits for OpenAI can utilize Groq as an alternative. It provides a similar configuration structure but uses Groq's API key and base URL (`https://api.groq.com/openai`), along with a different model (`llama-3.1-70b-versatile`).

Overall, the key entities include Spring Boot, OpenAI, API keys, application configuration, and the models used for AI interaction.

metadata.next_section_summary: The section discusses the development of a "Hello World" application using Spring Boot, specifically focusing on creating a RESTful API with a `RestController`. Key topics include:

- **Spring Boot:** A framework used for building web applications in Java.

- **Hello World:** A common introductory example in programming.
- **RestController:** An annotation that indicates the class is a controller for handling RESTful web services.
- **API Development:** The process of creating an application programming interface to enable communication between different software components.
- **ChatClient:** A component presumably used to interact with a chat service.
- **自動注入 (Automatic Injection):** A mechanism in Spring for automatically injecting dependencies into classes.
- **錯誤 (Error):** An issue encountered when the application fails to find a required Bean during execution.
- **Bean:** An object managed by the Spring IoC (Inversion of Control) container.
- **Controller:** A component that handles incoming requests and returns responses.

The section emphasizes the simplicity of starting a Spring Boot application and highlights an error encountered related to dependency management.

老實說這摘要還蠻長的，若之後想使用這功能建議可以自訂提示詞，並在提示詞中限制字數上限，自訂提示詞須改用下面建構子

```
SummaryMetadataEnricher(ChatModel chatModel, List<SummaryType
```

參數說明如下

`chatModel`：用於產生摘要的 AI 模型。

`summaryTypes`：一個 `SummaryType` 枚舉值列表，表示要產生 Document 的**前一個(PREVIOUS)**、**目前(CURRENT)**、**下一個(NEXT)**摘要，預設是全部都產生。

摘要與關鍵字都是用來加強 embedding 的準確性，也都是進階 RAG 常用的方法，另外其結果使用英文也會讓 AI 更容易判讀