

Day26 - ETL pipeline(中)

延續昨天的主題，今天要處理的文件內容比昨天複雜，分別是

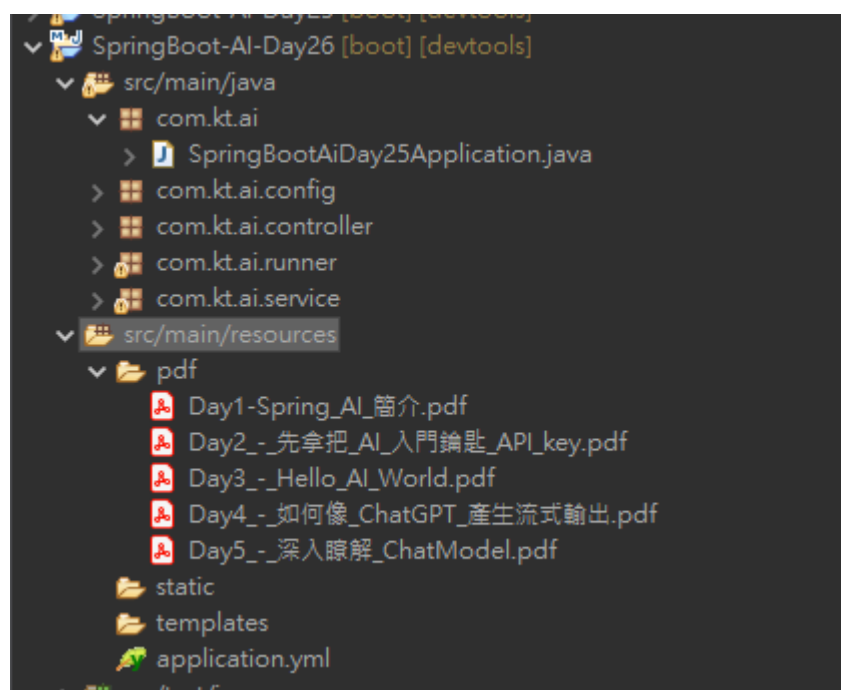
PDF： `PagePdfDocumentReader` 、 `ParagraphPdfDocumentReader`

DOCX, PPTX, HTML...： `TikaDocumentReader`

以上都是 apache 的專案，PDF 是 pdfbox，而 `TikaDocumentReader` 就像它的名稱是 Tike 的專案，Tike 支援的檔案類型可參考[官方文件](#)

PDF:

`PagePdfDocumentReader` 與 `ParagraphPdfDocumentReader` 只差在一個是以 Page 為單位，一個是以目錄的章節為單位，顯然以章節為單位拆分內容就不會被截斷，不過不是所有的 pdf 文件都有目錄，我們在程式中可先使用 `ParagraphPdfDocumentReader` 失敗時在改用 `PagePdfDocumentReader`，這樣會最大程度優化文件向量化的結果，另外企業將資料向量化時不可能一份一份操作，凱文大叔特地將鐵人賽前幾天的文章轉成 pdf 檔，並存在 resources/pdf 目錄下，來看看程式如何批次處理這麼多份文件



程式實作

首先需要先引入依賴才能使用，編輯 pom.xml 加入下面內容

```

<dependency>
  <groupId>org.springframework.ai</groupId>
  <artifactId>spring-ai-pdf-document-reader</artifactId>
</dependency>

```

EtlService.java : 在 Service 中加入以下函式

```

public List<Document> loadPdfAsDocuments() throws IOException
    ResourcePatternResolver resolver = new PathMatchingRe
    Resource[] resources = new Resource[0];
    resources = resolver.getResources("./pdf/*.pdf");
    //透過上面方式可將指定目錄下所有的pdf檔案載入,後面在針對每份檔案
    List<Document> docs = new ArrayList<>();
    for (Resource pdfResource : resources) {
        try {
            //先使用目錄分段讀取方式
            ParagraphPdfDocumentReader pdfReader = new Paragr
            docs.addAll(pdfReader.read());
        } catch (IllegalArgumentException e) {
            //沒有目錄會產生Exception,在改用分頁方式拆分
            PagePdfDocumentReader pdfReader = new PagePdf
            docs.addAll(pdfReader.read());
        }
    }
    return docs;
}

public void importPdf() throws IOException {
    //與Text一樣需在使用TokenTextSplitter切塊
    TokenTextSplitter splitter = new TokenTextSplitter();
    vectorStore.write(splitter.split(loadPdfAsDocuments())
}

```

EtlController.java : Controller 加入對應的 API

```

@GetMapping("readpdf")
public List<Document> readPdfFile() throws IOException{
    return etlService.loadPdfAsDocuments();
}

```

```

    }

    @GetMapping("importpdf")
    public void importPdf() throws IOException{
        etlService.importPdf();
    }

```

程式重點

1. 將整個目錄檔案一次讀取可使用 `PathMatchingResourcePatternResolver`，透過 `getResources("*.pdf")`，就能讀取所有的 pdf 檔
2. 程式先使用 `ParagraphPdfDocumentReader` 分段拆分，失敗時再改用 `PagePdfDocumentReader` 分頁拆分
3. 文件類資料都會有內容過長問題，需要再使用 `TokenTextSplitter` 分割成更小塊

成果驗收

直接看最後寫入向量資料庫的成果，可以看到 metadata 預設會保存檔名跟第幾頁，如果使用段落拆分則會顯示哪個段落

The screenshot shows the Neo4j Desktop interface. On the left, a graph visualization displays several purple nodes connected by lines. On the right, the 'Node properties' panel is open, showing details for a selected 'Document' node. The properties include:

- `<elementId>`: 4:33659fc2-0688-4cc4-8792-bf49076897d0:12
- `<id>`: 12
- `embedding`: [-0.012953020632266998,-0.0011170142097398639,0.0016558879287913442,-0.040797337889671326,-0.008882644586265087,0.012070771306753159,-0.018099473789334... [Show all](#)]
- `id`: dc014778-cfcd-4194-87e3-fee5b09e82ab
- `metadata.file_name`: Day2_先拿把_AI_入門鑰匙_API_key.pdf
- `metadata.page_number`: 5
- `text`: 竟然直接報錯，看一下錯誤的 log

Below the properties, a stack trace is visible, indicating an error related to Spring Boot's automatic configuration for OpenAI:

```

org.springframework.beans.factory.BeanCreationException:
Error creating bean
    with name 'openAiChatModel' defined in
    class path resource
    [org/springframework/ai/autoconfigure/openai/
    OpenAiAutoConfiguration.class]:
    Failed to instantiate
    [org.springframework.ai.openai.OpenAiChatModel]:
    Factory
        method 'openAiChatModel' threw
    exception with message: OpenAI API key
    must be set

原來是 Spring Boot 的自動配置找不到 API
key，導致扣圖剛創建的 key 沒存在 application.yml

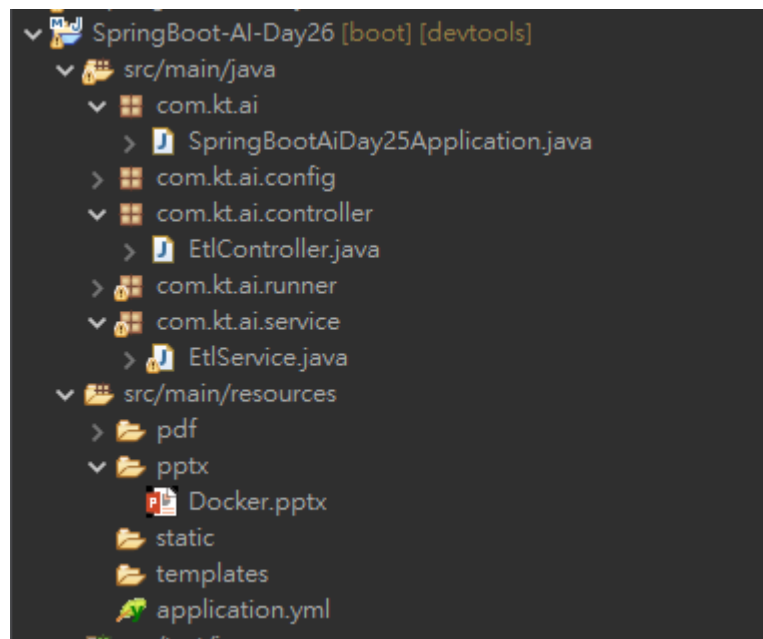
```

Tike:

Tike 支援的檔案類型非常豐富，有興趣可自行到[官網查看](#)，另外 Tike 也有支援 pdf 檔案，不過核心也是使用 pdfbox，使用上個範例能控制更多細節

下面凱文大叔就實作如何讀入 pptx 資料

一樣先把 pptx 集中在一個目錄，這裡我就先放一個檔案



程式實作

pom.xml 一樣需要引入依賴，內容如下

```
<dependency>
  <groupId>org.springframework.ai</groupId>
  <artifactId>spring-ai-tika-document-reader</artifactId>
</dependency>
```

EtlService.java : 在 Service 中加入以下函式

```
public List<Document> loadPptxAsDocuments() throws IOException {
    ResourcePatternResolver resolver = new PathMatchingResourcePatternResolver(this.getClass().getClassLoader());
    Resource[] resources = new Resource[0];
    resources = resolver.getResources("./pptx/*.pptx");
    List<Document> docs = new ArrayList<>();
    for (Resource pptxResource : resources) {
        TikaDocumentReader pptxReader = new TikaDocumentReader(pptxResource);
        docs.addAll(pptxReader.read());
    }
}
```

```

    return docs;
}

public void importPptx() throws IOException {
    TokenTextSplitter splitter = new TokenTextSplitter();
    vectorStore.write(splitter.split(loadPptxAsDocuments()));
}

```

EtlController.java：同樣的 Controller 也加入對應的 API

```

@GetMapping("readpptx")
public List<Document> readPptxFile() throws IOException{
    return etlService.loadPptxAsDocuments();
}

@GetMapping("importpptx")
public void importPptx() throws IOException{
    etlService.importPptx();
}

```

成果驗收

可以看到 metadata 預設一樣會有檔名，若要再添加一些資訊記得在

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 graph with several purple circular nodes. One node is highlighted with a larger, more detailed view on the right. This view shows the 'Node properties' for a 'Document' type. The properties include:

- <elementId>**: 4:33659fc2-0688-4cc4-8792-bf49076897d0:35
- <id>**: 35
- embedding**: [0.010403040796518326, 0.0021142750047147274, 0.01673913188278675, -0.02076568268239498, -0.045517511665821075, 0.014503656886518002, -0.012544249184429646, ...]
- id**: 431d9301-96a7-4e37-a7a9-77e1ef3ba186
- metadata.source**: Docker.pptx
- text**: 挂载/root/mysql/data到容器内的/var/lib/mysql目录
挂载/root/mysql/init到容器内的/docker-entrypoint-initdb.d目录，携带课前资料准备的SQL脚本
挂载/root/mysql/conf到容器内的/etc/mysql/conf.d目录，携带课前资料准备的配置文件
本地目录必须以"/"或"/"开头，如果直接以名称开头，会被识别为数据卷而非本地目录
-v mysql:/var/lib/mysql 会被识别为一个数据卷叫mysql
-v ./mysql:/var/lib/mysql 会被识别为当前目录下的mysql目录

At the bottom of the text field, there is a link labeled '案例' (Case).

大家應該能發現 Spring 封裝後的工具操作起來都差不多，程式碼的差別就只是 Reader 的類別不一樣，大家可以將 Tike 支援的類別都測試看看