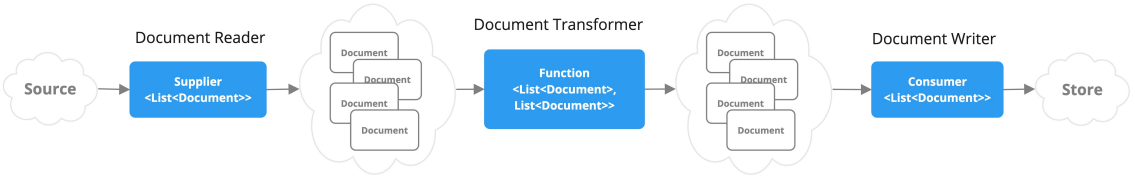
Day25 - ETL pipeline(上)

為什麼 ETL 後面要加個 pipeline？看下面官方的圖片就一目了然



上圖藍色部分分別對應三個介面

- 產出 `Supplier<List<Document>>` 的 `DocumentReader`
- 產出 `Function<List<Document>, List<Document>>` 的 `DocumentTransformer`
- 產出 `Consumer<List<Document>>` 的 `DocumentWriter`

將這些動作串在一起就是一個 ETL Pipeline，假設要從 pdf 讀取資料，最後存入向量資料庫，程式可以用下面方式串在一起

```
vectorStore.accept(tokenTextSplitter.apply(pdfReader.get()));
```

為了更容易理解，上述類別還提供不同的方法名稱

```
vectorStore.write(tokenTextSplitter.split(pdfReader.read()));
```

要注意的是 Java 程式的 Pipeline 方向是從內到外，Unix 或是 Python 則是從左到右，或許未來 Spring AI 會提供 Builder 的流式寫法程式碼會更容易解讀

接下來就介紹 Spring AI 各種實作的類別，由於需要用到向量資料庫，程式可從 Day21 延伸

JsonReader :

用來讀取 JSON 格式的檔案，由於 JSON 都是一筆一個 Document，所以不用再切割成小塊，使用 JsonReader 讀取後可直接寫入向量資料庫
讀取 JSON 通常用於跨系統整合，別的系統無法開放 DB 讀取時會提供匯出的 JSON，這時就可使用這個 ETL 工具

下面是程式實作

EtlService.java: 主要邏輯程式

```
@Service
@RequiredArgsConstructor
public class EtlService {
    private final VectorStore vectorStore;

    @Value("classpath:students.json")  //準備匯入的JSON
    private Resource resource;

    //將json轉為List<Document>,若資料是array,每筆資料會轉成一個node
    public List<Document> loadJsonAsDocuments() {
    JsonReader jsonReader = new JsonReader(resource, new Jsonl
            @Override
            public Map<String, Object> generate(Map<String, O
                return Map.of("Type", "Student grades");
                //Metadata,可放多組,之後可用來篩選資料
            }
        }, "name","grade"); //設定JSON要匯入那些屬性,沒寫的話就是整
    return jsonReader.get();
    }

    //將List<Document>寫入向量資料庫,在寫入前vectorStore會先計算Emb
    public void importStudentsAsJson() {
        vectorStore.write(loadJsonAsDocuments());
    }

    //向量資料庫的近似查詢
    public List<Document> search(String queryStr){
        return vectorStore.similaritySearch(
                SearchRequest.query(queryStr)
                );
    }
}
```

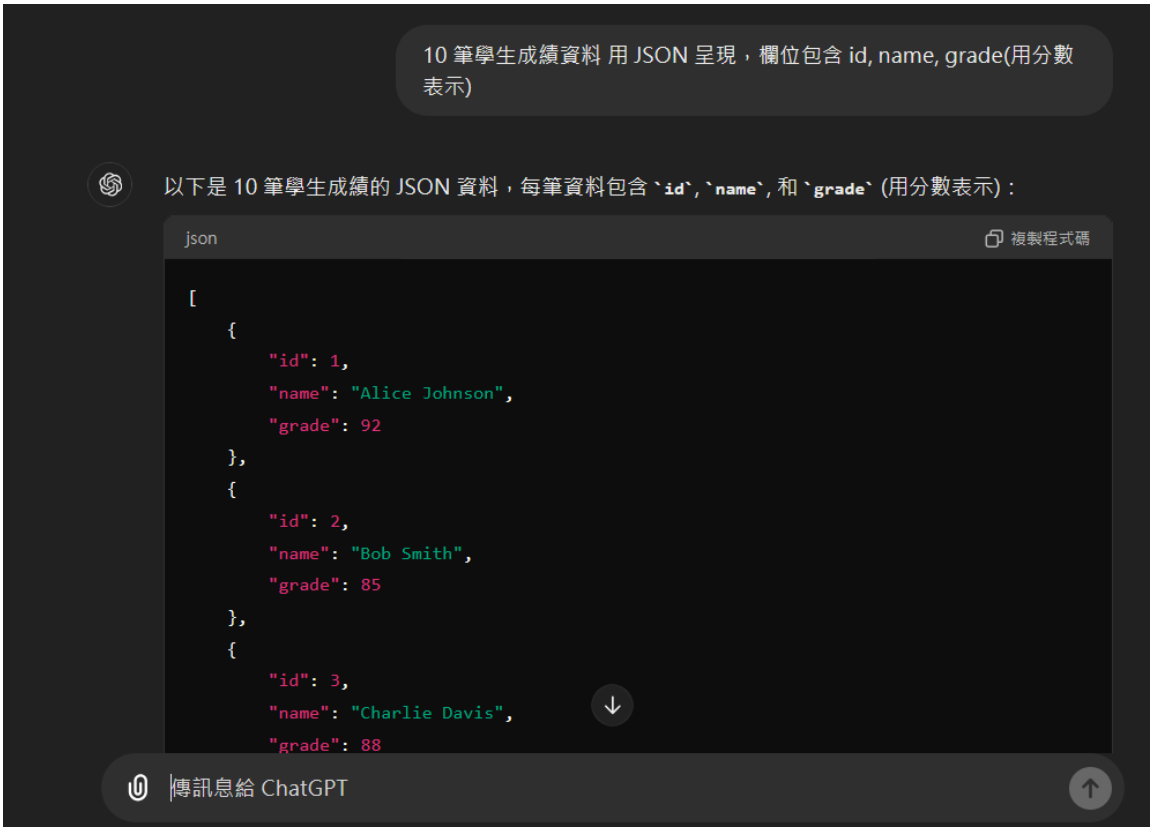
EtlController.java: 提供 API 供外部呼叫

```
@RestController
@RequestMapping("/ai/etl")
@RequiredArgsConstructor
```

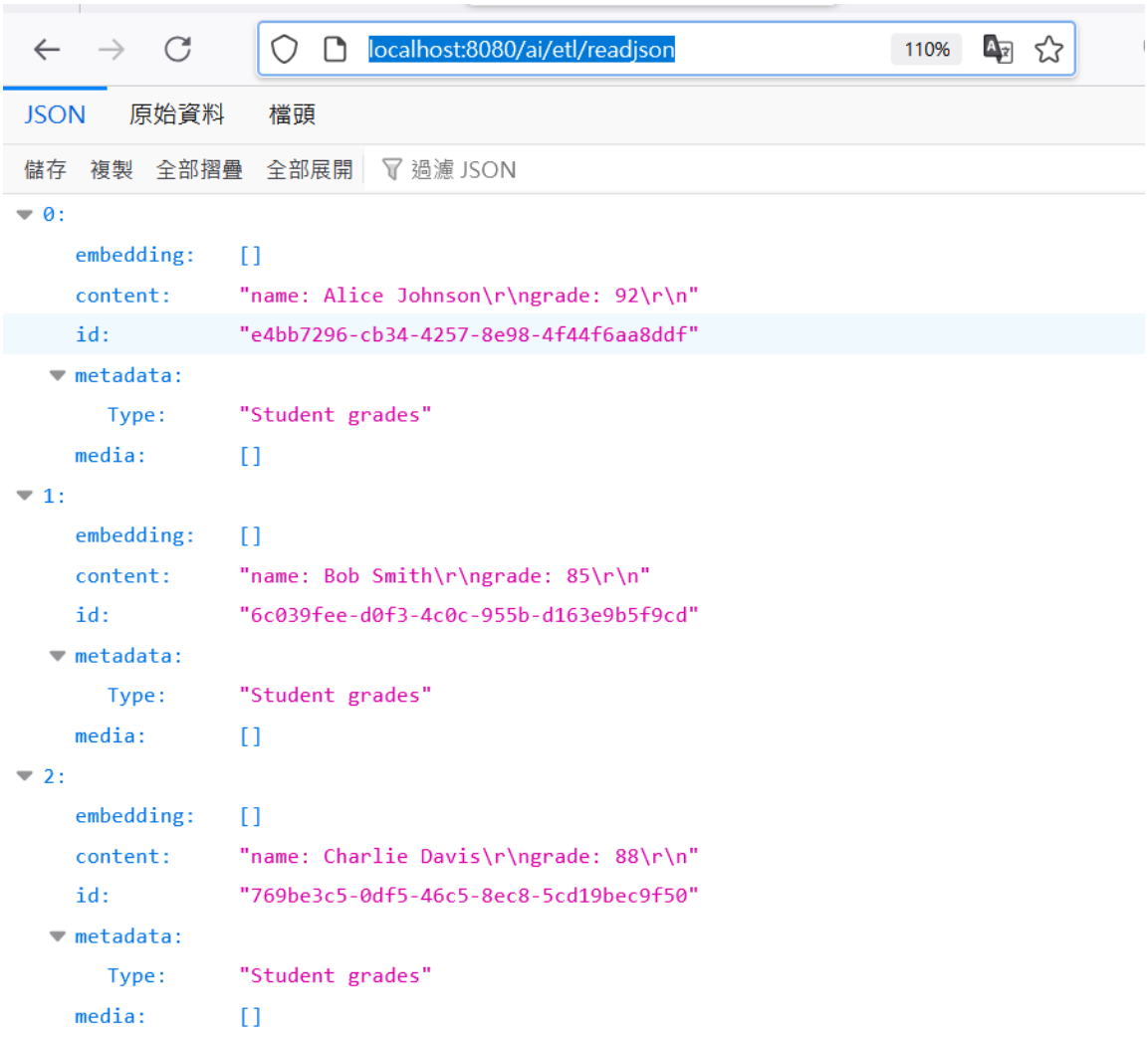
```
public class EtlController {

    private final EtlService etlService;

    //將JSON轉為List<Document>,為了測試方便直接將JSON檔案放在 resou
    @GetMapping("readjson")
    public List<Document> readJsonFile(){
        return etlService.loadJsonAsDocuments();
    }

    //將List<Document>存入向量資料庫
    @GetMapping("importjson")
    public void importJson(){
        etlService.importStudentsAsJson();
    }

    //使用近似值查詢匯入的內容
    @GetMapping("search")
    public List<Document> search(String query){
        return etlService.search(query);
    }
}
```

驗收程式

1. 準備JSON資料，可以請 ChatGPT 依特定格式隨機產生資料，並存在 resources/students.json 內

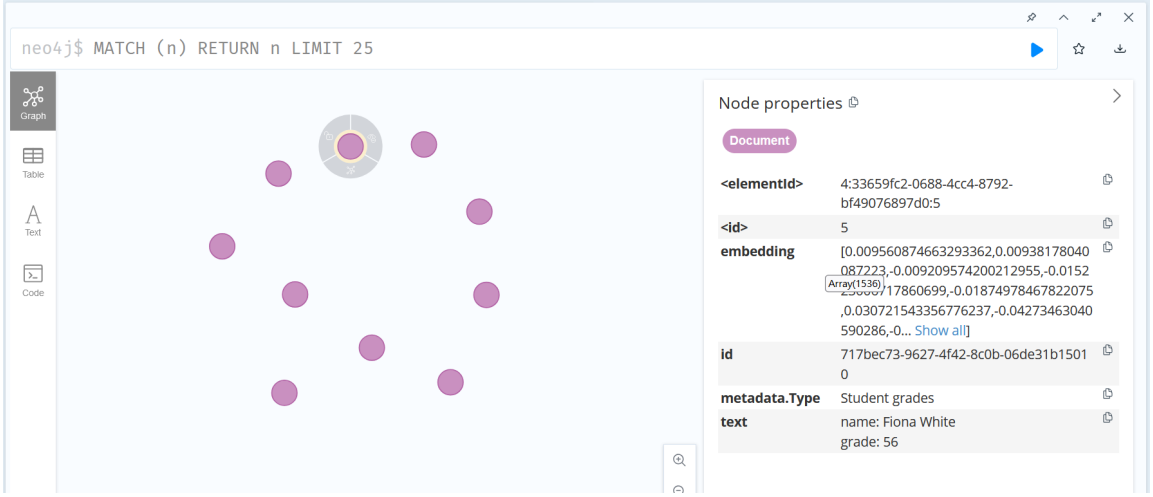


2. 透過URL執行匯入並印出資料: http://localhost:8080/ai/etl/readjson



可以看出程式指定的 type 放在 metadata 下，embedding 還沒寫入向量資料庫所以是空的

3. 透過URL讀取JSON後直接寫入向量資料庫:
   http://localhost:8080/ai/etl/importjson



可以看到除了要寫入的內容外，還多了 embedding，由於 embedding 都需要透過 AI 算出，所以資料越多執行時間就越久

4. 近似查詢測試: http://localhost:8080/ai/etl/search?query=xxx

可以看出透過近似查詢 metadata 還會多一個 distance，這就是使用近似查詢將問題與向量資料庫內的資料計算後所求出的內容

TextReader:

用來讀取文字格式的資料，每個檔案會存入一個 Document 中，metadata 預設會包含檔名跟字元集=UTF8

由於預設會將整個檔案存入，之後要做 RAG 搜索時很容易超過 Token 上限，所以需要進行分塊

Spring AI 提供一個將大檔案分割成小塊的工具`TokenTextSplitter`，它可協助幫我們切塊後還保留原本的 metadata

程式實作

EtlService.java: 跟上個範例一樣提供三個 Function，分別針對讀取，寫入以及查詢，另外在寫入向量資料庫前還會將 Document 在拆分成小塊

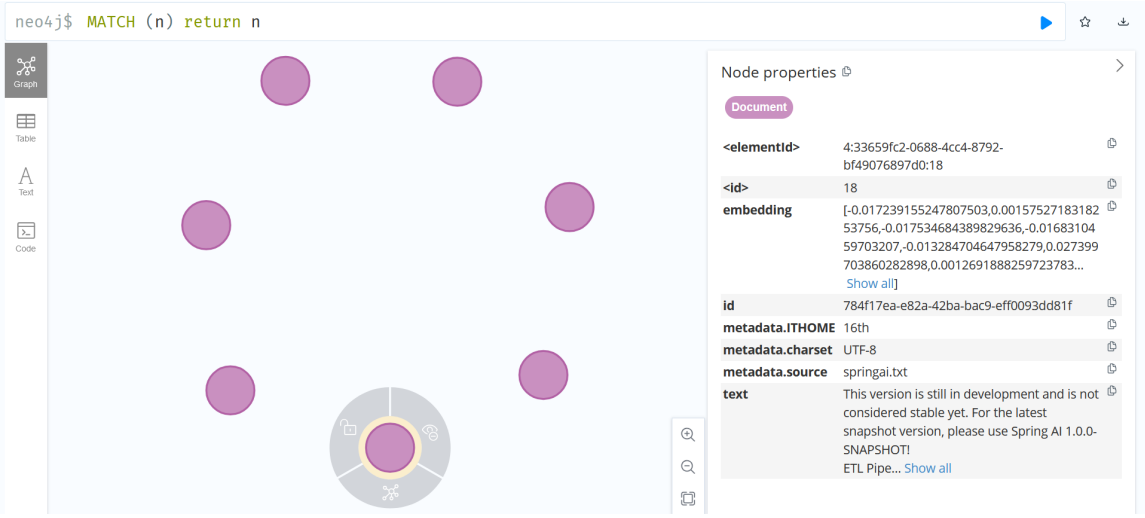
```
@Value("classpath:springai.txt")
private Resource textResource;

public List<Document> loadTextAsDocuments() {
        TextReader textReader = new TextReader(textResource);
        textReader.getCustomMetadata().put("filename", textRe
        return textReader.get();
    }

    public void importText() {
        TokenTextSplitter splitter = new TokenTextSplitter();
        vectorStore.write(splitter.split(loadTextAsDocuments(
    }

    public List<Document> searchTextData(String queryStr){
        return vectorStore.similaritySearch(
                SearchRequest.query(queryStr)
                );
    }
```

EtlController.java: 這裡一樣提供三個對應的 API

```
private final EtlService etlService;
@GetMapping("readtext")
public List<Document> readTextFile(){
    return etlService.loadTextAsDocuments();
}

@GetMapping("importtext")
public void importText(){
    etlService.importText();
}
@GetMapping("searchtext")
public List<Document> searchText(String query){
    return etlService.searchTextData(query);
}
```

驗收成果

前面步驟都一樣就不依依執行了，直接看一下切塊後進入向量資料庫的結果



TokenTextSplitter 將 Document 拆成了七分，metadata 也都完整保留下來，下面是預設大小，若需調整可透過建構子修改

```
// The target size of each text chunk in tokens
private int defaultChunkSize = 800;
```

```
// The minimum size of each text chunk in characters
private int minChunkSizeChars = 350;

// Discard chunks shorter than this
private int minChunkLengthToEmbed = 5;

// The maximum number of chunks to generate from a text
private int maxNumChunks = 10000;
```

明天會再繼續說明 **Markdown、PDF 還有 Office 等相關的文件如何處理**