Day24 - RAG 簡介

之前的章節幾乎涵蓋了 Spring AI 的基本功能，是時候向下一個里程邁進了
先來看看 Spring AI 對 RAG 的說明:

一種被稱為"檢索增強生成"（RAG）的技術應運而生，用來將相關數據加入提示詞，以獲得準確的回覆內容。

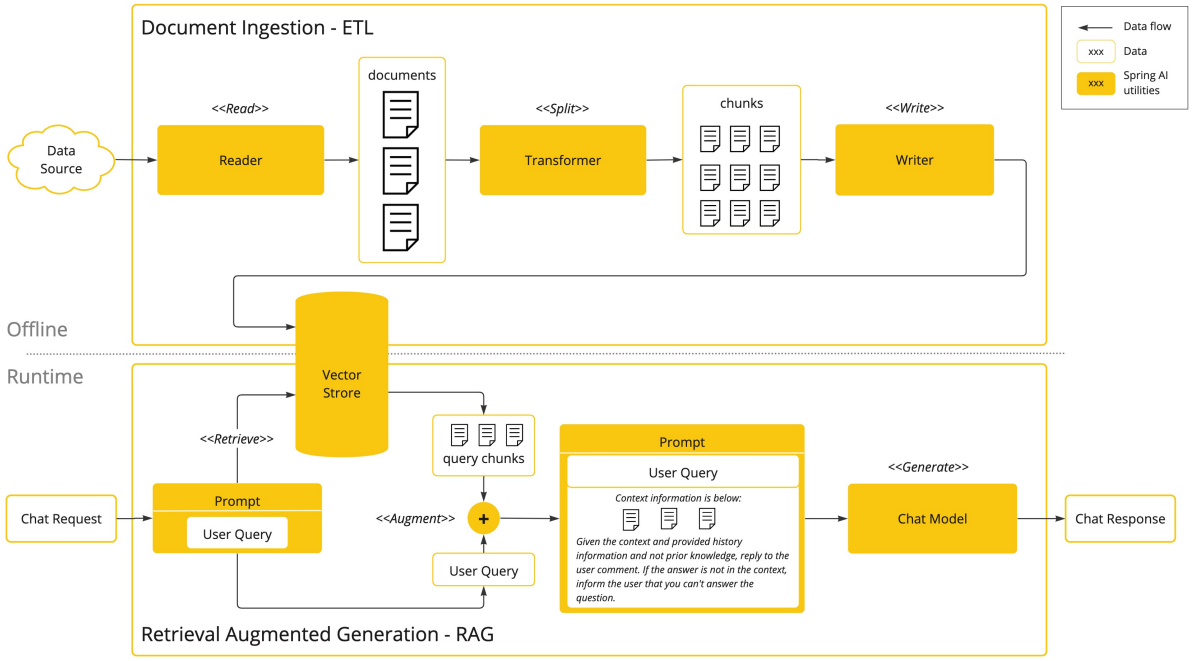
RAG ETL階段

這種方法採用批次程式設計模型，從檔案中讀取非結構化數據，對其進行Embedding，然後將內容與 Embeddings 一起寫入向量資料庫。
整體來看，這是一個ETL（提取、轉換和載入）流程。向量資料庫會被用於 RAG 技術的檢索部分。
將非結構化資料載入向量資料庫的過程中，最重要的轉換之一就是將原始文件分割成小塊。
分割的過程有兩個重要步驟：

1. 在保留語意邊界的前提下將檔案分割成若干分 Documents。例如，對於包含段落和表格的文檔，應避免在段落或表格中間分割文檔。對於程式碼，應避免在方法中分割程式碼。
2. 將 Documents 進一步拆分成若干 Chunks，Chunks 的大小取決於 AI 模型的單次請求 Token 上限。

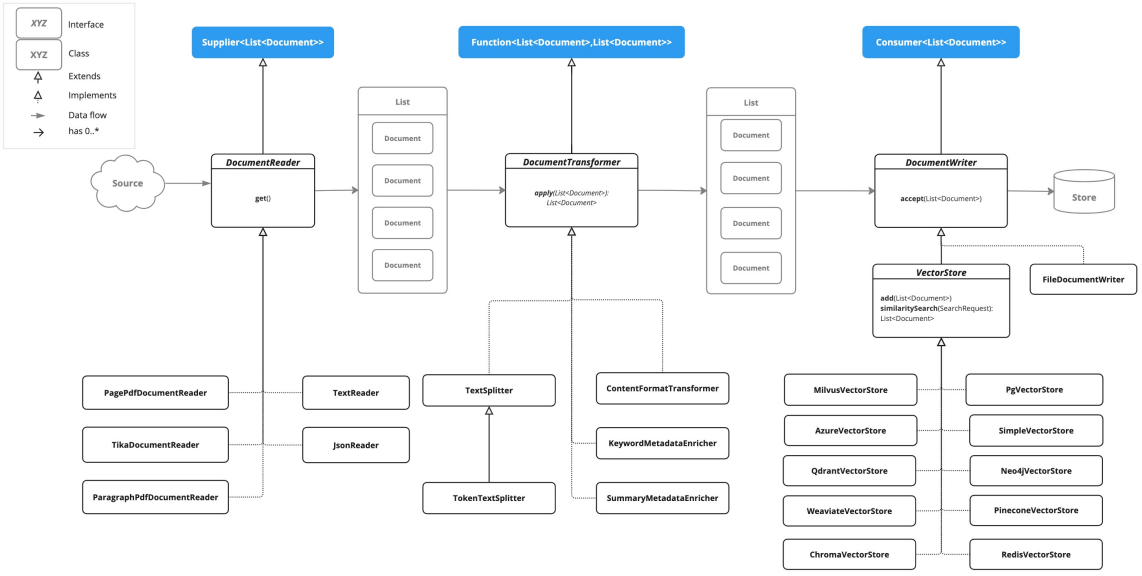
RAG執行階段

RAG 的執行階段是處理使用者輸入。當 AI 模型要回答使用者的問題時，問題和所有"相似"的 Chunks 都會被放入提示詞中。
這就是使用向量資料庫的原因。它能很快速的找到相似內容。



前面幾天介紹的內容基本上都是 Runtime 的範圍，Offline 的部分就是將企業內的資料向量化，向量資料庫則是串起這兩功能的核心

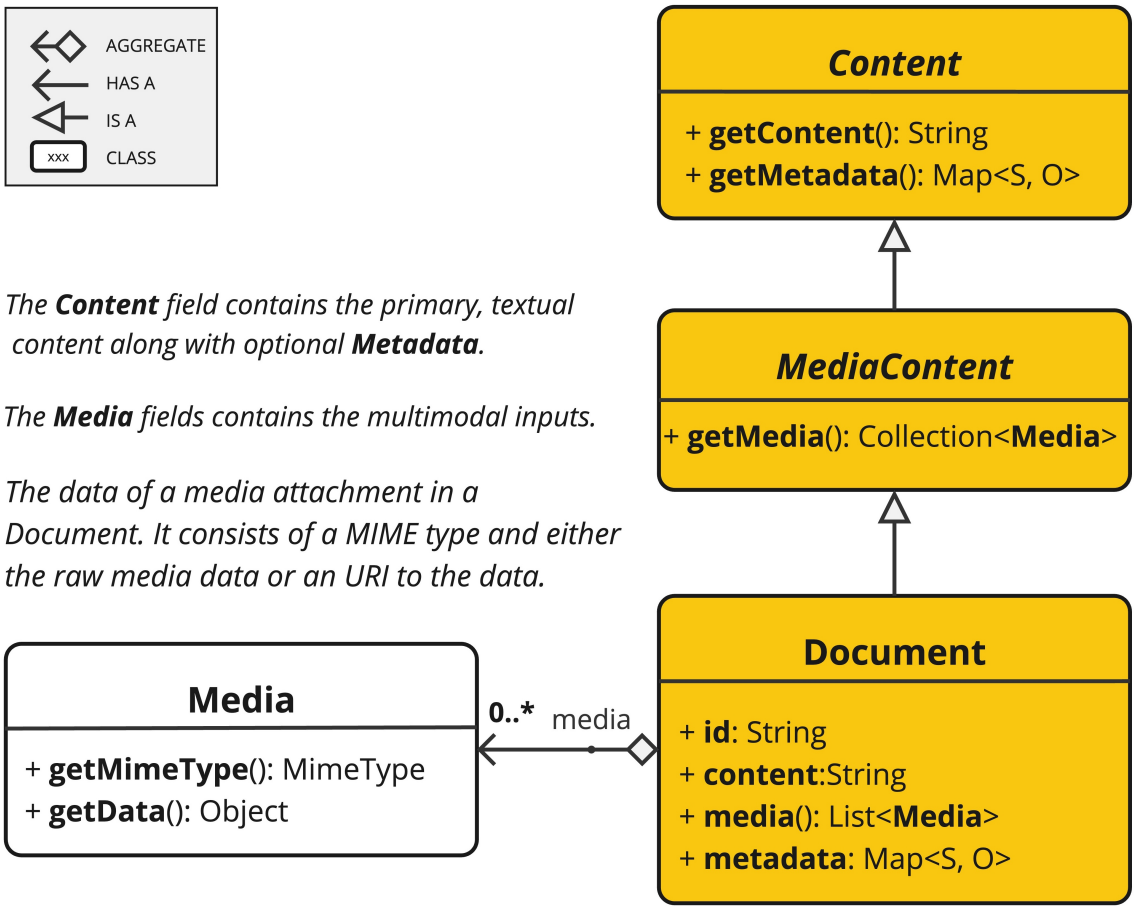
下面我們先來看一下 Spring AI 在 ETL 的部分提供了哪些類別



這三個類別分別對應到 Offline 三個動作

1. **Reader**: DocumentReader 實作了多種介面，主要用來讀取不同類型的檔案，並將這些內容轉為 List<Document>，Document 的部分前面我們已經看過有資料

內容外，還會記錄相關的 metadata，另外還有一個很重要的內容 Media，這表示 ETL 除了能讀取一般文件外，還能讀取多媒體檔案




```
public interface DocumentReader extends Supplier<List<Docu
    default List<Document> read() {
            return get();
        }
}
```

2. Transformer: DocumentTransformer 的動作很簡單，就是將大的 Document 拆成更小的 Chunks，而轉換後的類別依然是 Document，切塊的大小取決於 AI 同時能吞吐的 Tokens 數量

```
public interface DocumentTransformer extends Function<List
    default List<Document> transform(List<Document> transf
            return apply(transform);
```

```
        }
}
```

3. Writer: DocumentWriter 這個動作我們之前有操作過，就是將對話內容寫入向量資料庫的動作

```
public interface DocumentWriter extends Consumer<List<Docu
    default void write(List<Document> documents) {
            accept(documents);
        }
}
```

接下來幾天凱文大叔就會一步步帶著大家實作