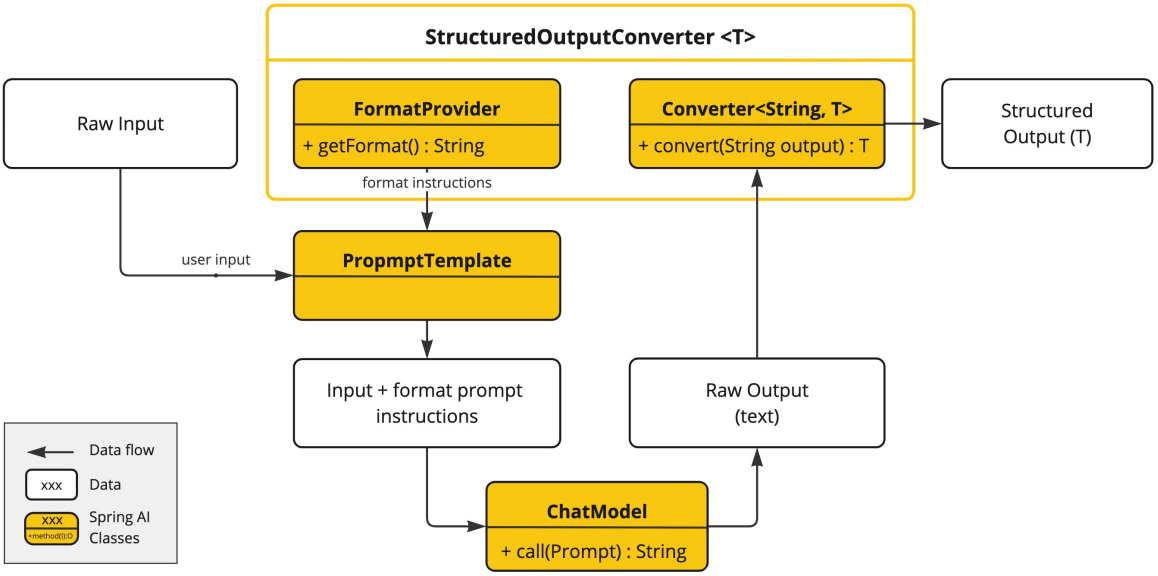
Day14-結構化輸出轉換器

先說個題外話，Spring 1.0.0 M1 最大的改變應該就是命名吧XD，`StructuredOutputConverter` 在 0.8 版稱為 `OutputParser`，Spring 認為這功能沒任何解析的作用，只是做了轉換，所以跟其他模組的轉換器採用同樣的命名原則，所以改為 `StructuredOutputConverter` ，當初應該是為了搶快，所以名稱都抄 LangChain4j 的吧

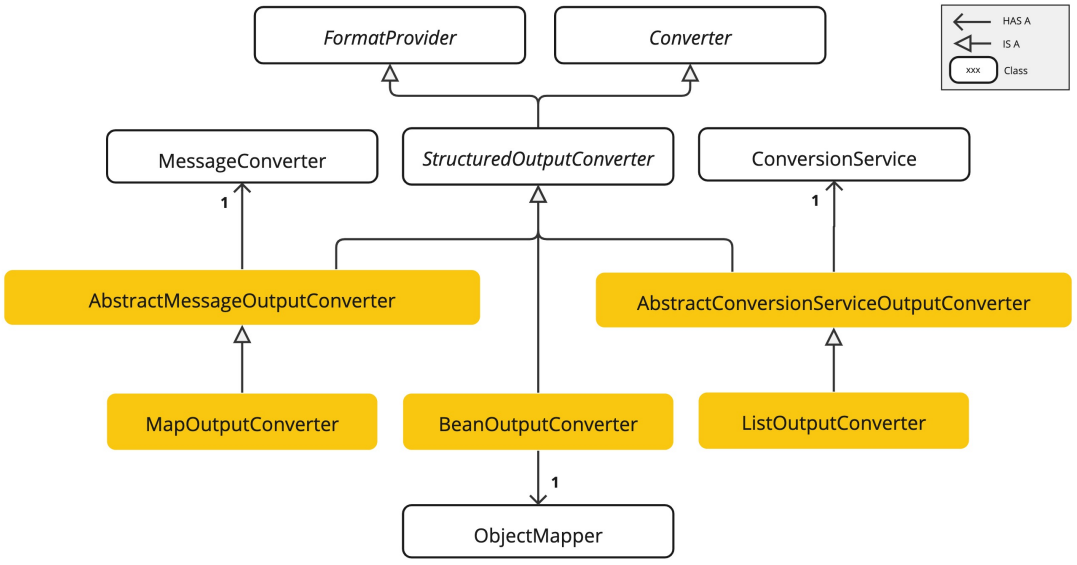
對一般系統而言，要看懂 ChatGPT 回應的文字並不容易，所以 OpenAI 也提供了許多方式將回應轉為不同型態，JSON、XML 或是 Markdown 的語法，最近 OpenAI 還更新了新的功能，透過強制約束，讓 OpenAI 的輸出可以與定義的 JSON 格式完全一致，有了這功能對非 AI 系統的整合才能讓失誤率降到最低

下面是目前 `StructuredOutputConverter` 的資料流程，不過我相信馬上也會針對 OpenAI 的新功能進行改版



可以看到流程中主要還是透過 Prompt 的方式讓 AI 依據描述的內容輸出成需要的格式，凱文大叔猜測這一塊之後只需放上要輸出的 Class，就能引導 AI 照格式來輸出，不過對我們來說，這都是封裝在轉換器的內容，只需要知道以後的回應會是更精準的內容即可

Spring AI 目前提供的轉換器有三個 `BeanOutputConverter` 、 `MapOutputConverter` 和 `ListOutputConverter` （抽象類別就不看了）



`BeanOutputConverter`：AI 產生的資料主要是以 JSON 為主，這個轉換器就是將 JSON 轉為 Java 程式需要的 Bean，背後用到的就是寫 Spring MVC 最常用到的 ObjectMapper

`MapOutputConverter`：這個轉換器是將資料使用 Map 的方式轉出，對未知的格式最常使用的處理方式

`ListOutputConverter`：這個轉換器顧名思義就是將結果轉為 List，不過這裡主要以字串的 List 為主，例如請 AI 提供最受歡迎的五種冰淇淋口味，若是結構化的複數資料則還是使用 `BeanOutputConverter` 進行轉換，只是將 Bean 的類別改為 `ParameterizedTypeReference`

程式案例

Bean Output Converter

我們將寫一隻電影百科機器人，當詢問影星時還要列出其作品，資料結構如下

```
record ActorsFilms(String actor, List<String> movies) {
}
```

以下是`BeanOutputConverter` 的使用方式

```
BeanOutputConverter<ActorsFilms> beanOutputConverter =
    new BeanOutputConverter<>(ActorsFilms.class);
```

```
String format = beanOutputConverter.getFormat();

String actor = "Tom Hanks";

String template = """
        Generate the filmography of 5 movies for {actor}.
        {format}
        """;

Generation generation = chatModel.call(
    new Prompt(new PromptTemplate(template, Map.of("actor", a

ActorsFilms actorsFilms = beanOutputConverter.convert(generat
```

可以看出轉換器就是將 ActorsFilms 轉成 JSON 格式，再放入 Prompt 要求 AI 將結果轉為此格式

如果要問多位影星，程式就改成如下

```
BeanOutputConverter<List<ActorsFilms>> outputConverter = new
        new ParameterizedTypeReference<List<ActorsFilms>>() {

String format = outputConverter.getFormat();
String template = """
        Generate the filmography of 5 movies for Tom Hanks an
        {format}
        """;

Prompt prompt = new Prompt(new PromptTemplate(template, Map.o

Generation generation = chatModel.call(prompt).getResult();

List<ActorsFilms> actorsFilms = outputConverter.convert(gener
```

Map Output Converter

```
MapOutputConverter mapOutputConverter = new MapOutputConverte

String format = mapOutputConverter.getFormat();
String template = """
        Provide me a List of {subject}
        {format}
        """;
PromptTemplate promptTemplate = new PromptTemplate(template,
        Map.of("subject", "an array of numbers from 1 to 9 un
Prompt prompt = new Prompt(promptTemplate.createMessage());
Generation generation = chatModel.call(prompt).getResult();

Map<String, Object> result = mapOutputConverter.convert(gener
```

List Output Converter

```
ListOutputConverter listOutputConverter = new ListOutputConve

String format = listOutputConverter.getFormat();
String template = """
        List five {subject}
        {format}
        """;
PromptTemplate promptTemplate = new PromptTemplate(template,
        Map.of("subject", "ice cream flavors", "format", form
Prompt prompt = new Prompt(promptTemplate.createMessage());
Generation generation = this.chatModel.call(prompt).getResult

List<String> list = listOutputConverter.convert(generation.ge
```