به نام خدا

web programming

تحقیق پایانی درس برنامه سازی وب

اعضای گروه : سبا شامخی - زهرا علیپور - کمیل یحیی زاده

Structured Output Converter و Spring AI

بیان مسئله

با پیشرفت فناوری‌های هوش مصنوعی و به‌ویژه مدل‌های زبانی بزرگ (LLMs)، نیاز به تبدیل خروجی‌های متنی به داده‌های ساختارمند به‌طور فزاینده‌ای احساس می‌شود. این تبدیل برای برنامه‌های کاربردی مختلف، از جمله تجزیه و تحلیل داده‌ها، تولید محتوا و اتوماسیون فرآیندها ضروری است. خروجی‌های متنی معمولاً غیرساختارمند هستند و نمی‌توان به راحتی از آن‌ها در سیستم‌های اطلاعاتی استفاده کرد. Spring AI با ویژگی‌هایی مانند Structured Output Converter به توسعه‌دهندگان کمک می‌کند تا این چالش را حل کنند.

معرفی Spring AI

Spring AI یک فریم‌ورک مدرن است که به توسعه‌دهندگان این امکان را می‌دهد تا از قابلیت‌های مدل‌های زبانی بزرگ بهره‌برداری کنند. این ابزار به‌ویژه برای برنامه‌نویسان جاوا طراحی شده و امکاناتی برای تعامل با مدل‌های هوش مصنوعی فراهم می‌کند. یکی از ویژگی‌های کلیدی آن، Structured Output Converter است که خروجی‌های تولید شده توسط مدل‌ها را به فرمت‌های ساختارمند مانند XML، JSON یا YAML تبدیل می‌کند.

ویژگی‌ها و مزایای Structured Output Converter

- تبدیل مستقیم خروجی‌ها: این ابزار امکان تبدیل خروجی متن آزاد به اشیاء جاوا را فراهم می‌آورد.
- پشتیبانی از فرمت‌های مختلف: قابلیت تولید خروجی در فرمت‌هایی مانند XML، JSON و YAML.
- سازگاری با سایر ماژول‌های Spring: این ابزار به راحتی با دیگر اجزای Spring Boot یکپارچه می‌شود.

نحوه عملکرد Structured Output Converter

معماری کلی

Structured Output Converter بخشی از کتابخانه Spring AI است که به صورت زیر عمل می‌کند:

1. تعریف قالب مورد نظر توسط توسعه‌دهنده.
2. نگاشت متن تولید شده توسط مدل زبانی بزرگ به قالب مشخص.
3. تولید خروجی به صورت داده ساختارمند.

مثال عملی: تولید لیست فیلم‌های یک بازیگر

در این بخش، یک مثال عملی برای نمایش نحوه استفاده از Structured Output Converter ارائه می‌شود.

1. تعریف کلاس داده

ابتدا یک رکورد برای نگهداری اطلاعات بازیگر و فیلم‌ها تعریف می‌کنیم:

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

2. پیاده‌سازی منطق تبدیل

از `BeanOutputConverter` برای تبدیل متن خام به نوع داده‌ای مشخص استفاده می‌کنیم:

```
@Bean
public BeanOutputConverter<ActorsFilms> actorsFilmsConverter() {
    return new BeanOutputConverter<>(ActorsFilms.class);
}
```

3. فراخوانی مدل و تبدیل خروجی

کد زیر نشان‌دهنده نحوه فراخوانی مدل و استفاده از Structured Output Converter است:

```
@Autowired
private ChatModel chatModel;

@Autowired
private BeanOutputConverter<ActorsFilms> actorsFilmsConverter;

public ActorsFilms getFilmography(String actorName) {
    String format = actorsFilmsConverter.getFormat();
    String template = """
        Generate a list of 5 movies for the actor {actor}.
        {format}
        """;

    Generation generation = chatModel.call(
        new PromptTemplate(template, Map.of("actor", actorName,
"format", format)).create()
    ).getResult();
```

```
    return
actorsFilmsConverter.convert(generation.getOutput().getContent()
);
}
```

4. اجرای برنامه

با اجرای متد بالا و ارسال نام بازیگر (مثلاً "Tom Hanks")، خروجی زیر تولید خواهد شد:

```
{
  "actor": "Tom Hanks",
  "movies": [
    "Forrest Gump",
    "Cast Away",
    "Saving Private Ryan",
    "The Green Mile",
    "Toy Story"
  ]
}
```

مثال دیگر: تجزیه و تحلیل نظرات مشتریان

در این مثال، فرض کنید که ما نظرات مشتریان را جمع‌آوری کرده‌ایم و می‌خواهیم آن‌ها را به دسته‌بندی‌های مثبت، منفی و خ中خالی تقسیم کنیم.

1. تعریف کلاس داده برای نظرات

```
record CustomerReview(String review, String sentiment) {}
```

2. پیاده‌سازی منطق تجزیه و تحلیل نظرات

```
@Bean
public BeanOutputConverter<CustomerReview>
customerReviewConverter() {
```

```
    return new BeanOutputConverter<>(CustomerReview.class);
}
```

3. فراخوانی مدل برای تجزیه و تحلیل نظرات

```
public List<CustomerReview> analyzeReviews(List<String> reviews)
{
    List<CustomerReview> results = new ArrayList<>();

    for (String review : reviews) {
        String format = customerReviewConverter.getFormat();
        String template = """
            Analyze the sentiment of the following review:
{review}.
            {format}
            """;

        Generation generation = chatModel.call(
            new PromptTemplate(template, Map.of("review",
review, "format", format)).create()
        ).getResult();


results.add(customerReviewConverter.convert(generation.getOutput
().getContent()));
    }

    return results;
}
```

4. اجرای برنامه

با اجرای متد بالا و ارسال لیستی از نظرات مشتریان، خروجی‌ای شامل نظرات همراه با احساسات آن‌ها (مثبت یا منفی) دریافت خواهیم کرد.

نتیجه‌گیری

Spring AI و ابزار Structured Output Converter یکی از راهکارهای قدرتمند برای مدیریت خروجی مدل‌های زبانی بزرگ هستند. این ابزار نه تنها فرآیند تبدیل داده‌ها را ساده‌تر می‌کند، بلکه امکان یکپارچگی آسان با سایر اجزای برنامه را نیز فراهم می‌آورد. با استفاده از این ابزار، توسعه‌دهندگان می‌توانند داده‌هایی قابل اعتماد و ساختارمند تولید کنند که در برنامه‌های مختلف قابل استفاده باشند.

مراجع

1. Spring AI Documentation - Structured Output Converter
2. Baeldung - Introduction to Spring Boot
3. Spring Framework Documentation