Using Opentelemetry Java Agent

Using the OpenTelemetry Java agent with bytecode instrumentation is indeed a simpler and more comprehensive approach. It provides more out-of-the-box instrumentation and requires less code changes. Let's set this up for your Spring Boot application FutureXCourseCatalog.

https://github.com/jrquinte/part0-JaegerCourseCatalog

https://github.com/jrquinte/parte0-JaegerCourseApp

Here's how to implement this approach:

1. First, download the OpenTelemetry Java agent:

You can do this manually or add a step in your build process. Here's a command to download it:

Download OpenTelemetry Java Agent

```
https://github.com/open-telemetry/opentelemetry-java-instrumentation/releases/latest/download/opentelemetry-javaagent.jar
```

Place this JAR file in a known location, for example, in the root directory of your projects.

2. Modify your application.properties file:

Updated application.properties for services

```
# OpenTelemetry configuration
otel.exporter.otlp.endpoint=http://localhost:4317
otel.exporter.otlp.protocol=grpc
otel.metrics.exporter=none
otel.logs.exporter=none
otel.traces.exporter=otlp
```

3. Execute your Docker Compose file for Jaeger (if you haven't already):

Updated docker-compose.yml for Jaeger

```
version: '3'
services:
  jaeger:
    image: jaegertracing/all-in-one:latest
    ports:
      - "16686:16686"   # Web UI
      - "14250:14250"   # gRPC for Jaeger-to-Jaeger communication
      - "4317:4317"     # OTLP gRPC receiver
      - "4318:4318"     # OTLP HTTP receiver
    environment:
      - COLLECTOR_OTLP_ENABLED=true
```

Now, when you run your applications:

Start Jaeger using Docker Compose:

```
docker-compose up -d
```

4. To run your applications with the Java agent, use a command like this:

Run Spring Boot with OpenTelemetry Java Agent

```
java -javaagent:path/to/opentelemetry-javaagent.jar -jar your-application.jar
```

Replace `path/to/opentelemetry-javaagent.jar` with the actual path where you placed the agent JAR file, and `your-application.jar` with the name of your compiled Spring Boot JAR.

If you built your application from IntelliJ IDEA, you could use this example:

```
java -javaagent:opentelemetry-javaagent.jar -jar
out/artifacts/FutureXCourseCatalog_jar/FutureXCourseCatalog.jar
```

5. Your applications should now be automatically instrumented, and you should see traces in Jaeger UI (http://localhost:16686) when you make requests to your services.

This approach provides several advantages:

- It requires minimal code changes.
- It automatically instruments a wide range of libraries and frameworks.
- It can capture more detailed traces without you having to manually add tracing code.

Remember, you don't need to add any manual tracing code with this approach - the agent handles it automatically. If you find you need custom spans or metrics later, you can still add those manually using the OpenTelemetry API.