Spring Boot 2 with Apache Camel and SQL component.

code-addict.pl/spring-boot-2-camel

10 czerwca 2018

How to run Apache Camel route with SQL component in Spring Boot 2.1

2 minute read

Photo credit: Markus Spiske
In this post I would like to introduce a way to configure Apache Camel version 2.24.1 with the SQL component and Spring Boot 2.1. During the configuration of the project with such technology I encountered few problems, so I would like to describe it in a simple way.

This is root project `build.gradle` file:

```
plugins {
        id 'org.springframework.boot' version '2.1.7.RELEASE'
        id 'io.spring.dependency-management' version '1.0.8.RELEASE'
        id 'java'
}

group = 'pl.codeaddict'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
        mavenCentral()
}

dependencies {
        implementation 'org.springframework.boot:spring-boot-starter-jdbc'
        implementation 'org.apache.camel:camel-spring-boot-starter:2.24.0'
        runtimeOnly 'com.microsoft.sqlserver:mssql-jdbc'
        testImplementation 'org.springframework.boot:spring-boot-starter-test'
        implementation 'org.apache.camel:camel-jdbc-starter:2.24.0'
        implementation 'org.apache.camel:camel-sql-starter:2.24.0'
        compileOnly 'org.projectlombok:lombok:1.18.8'
        annotationProcessor 'org.projectlombok:lombok:1.18.8'
}
```

As you can see when using Spring Boot, we should use Apache Camel starters and not standard Camel dependencies:

```
implementation 'org.apache.camel:camel-jdbc-starter:2.24.0'
implementation 'org.apache.camel:camel-sql-starter:2.24.0'
```

Below is the configuration from the `applications.properties` file:

```
spring.datasource.url=jdbc:sqlserver://localhost;databaseName=master
spring.datasource.username=sa
spring.datasource.password=QWE45rty
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.SQLServer2012Dialect
logging.level.root=DEBUG
camel.springboot.main-run-controller=true
```

I added configuration of the connection to the MS SQL database. I wanted to show how the Apache Camel SQL component works. An unusual thing is the parameter `camel.springboot.main-run-controller=true`, it allows you to keep the application alive if you do not use Spring's web components (`spring-boot-starter-web`).

The whole application consists basically of one Apache Camel route which is to fetch rows from the database, map them to objects and log them in the console:

```
@Component
public class QueueSelectRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {
        from("timer://dbQueryTimer?period=10s")
                .routeId("DATABASE_QUERY_TIMER_ROUTE")
                .to("sql:SELECT * FROM event_queue?dataSource=#dataSource")
                .process(xchg -> {
                    List<Map<String, Object>> row =
xchg.getIn().getBody(List.class);
                    row.stream()
                            .map((x) -> {
                                EventQueue eventQueue = new EventQueue();
                                eventQueue.setId((Long)x.get("id"));
                                eventQueue.setData((String)x.get("data"));
                                return eventQueue;
                            }).collect(Collectors.toList());
                })
                .log(LoggingLevel.INFO,"******Database query executed -
body:${body}******");

    }

}
```

Note the use of `?dataSource=#dataSource`. The `dataSource` name points to the DataSource object configured by Spring, it can be changed to another one and thus use different DataSource in different routes. Adding this parameter is required for proper operation of the application.

I added simple Liquibase scripts to the project to simplify project launch. You can also run docker container with MS SQL using below command:

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=QWE45rty' --name camel_example_db -p
1433:1433 -d mcr.microsoft.com/mssql/server:2017-CU8-ubuntu
```

When your container is up you can run Liquibase scripts which will add simple tables and a few rows of data:

```
./gradlew -p db-schema update
```

Remember that this should be run from the main project directory and not from the db-schema catalog.

You can now start the applications by running `./gradlew bootRun` command in project directory.

This is it! You can find all the source code in my repository GitHub account. Have fun and thanks for reading!

You May Also Enjoy

Authentication and authorization in Flutter For Web using Keycloak and Spring Boot 2 application as resource server

17 minute read

Authenticate in Keycloak and authorize with returned JWT token in Spring Boot 2 Web Flux API

How to write the first Kotlin code in Java project

8 minute read

Adding Kotlin to an existing multi-module Java project

How to bring image to life using machine learing

5 minute read

Playing with ML model to animate image using Pytorch, Python and Jupyter

Spring Boot 2 vs Quarkus - comparison

12 minute read

Playing with Quarkus using Kotlin and comparing it with same application written using Spring Boot 2