# Camel SQL Component Example

kswaughs.com/2016/08/camel-sql-component-example.html

Camel SQL Component is used to perform database operations using JDBC queries.

As a developer, You have to write only a mapper class to transform input/output parameters to your pojo class.

In this example, we will see how to use this component for insert and read books from an embedded derby database.

### Step 1: DataSource Setup

db-schema.sql

```
CREATE TABLE books (
  BookId VARCHAR(10) NOT NULL,
  BookName VARCHAR(100) NOT NULL,
  author VARCHAR(50) NOT NULL,
  price VARCHAR(20),
  CreateDate VARCHAR(50) NOT NULL
);
```

### Step 2: Externalize the sql queries in a file

sql.properties

```
sql.insertBook=INSERT INTO books(BookId, BookName, Author, Price, CreateDate)
VALUES (:#BookId, :#BookName, :#Author, :#Price, :#CreateDate)

sql.getAllBooks=select * from books
```

### Step 3: Application Context Configuration

database-context.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:jdbc="http://www.springframework.org/schema/jdbc"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:cxf="http://camel.apache.org/schema/cxf"
    xmlns:jaxrs="http://cxf.apache.org/jaxrs"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/cxf
       http://camel.apache.org/schema/cxf/camel-cxf.xsd
       http://cxf.apache.org/jaxrs
       http://cxf.apache.org/schemas/jaxrs.xsd
       http://www.springframework.org/schema/jdbc
       http://www.springframework.org/schema/jdbc/spring-jdbc-3.0.xsd
       http://camel.apache.org/schema/spring
       http://camel.apache.org/schema/spring/camel-spring.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd ">

    <!-- this is the JDBC data source which uses an in-memory only Apache Derby
database -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
        destroy-method="close">
        <property name="driverClassName"
value="org.apache.derby.jdbc.EmbeddedDriver" />
        <property name="url" value="jdbc:derby:memory:orders;create=true" />
        <property name="username" value="" />
        <property name="password" value="" />
    </bean>

    <jdbc:initialize-database data-source="dataSource" enabled="true">
        <jdbc:script location="classpath:db-schema.sql" />
    </jdbc:initialize-database>

    <!-- configure the Camel SQL component to use the JDBC data source -->
    <bean id="sqlComponent" class="org.apache.camel.component.sql.SqlComponent">
        <property name="dataSource" ref="dataSource" />
    </bean>

    <bean id="bookMapper" class="com.kswaughs.db.util.BookMapper" />

    <bean id="bookRouter" class="com.kswaughs.router.BookRouter" />

    <camelContext id="bookCtx" xmlns="http://camel.apache.org/schema/spring">

        <!-- use Camel property placeholder loaded from the given file -->
        <propertyPlaceholder id="placeholder" location="classpath:sql.properties"
/>

        <routeBuilder ref="bookRouter" />

    </camelContext>

</beans>
```

## Step 4: Define Routers for Insert & Read using Java DSL

BookRouter.java

```
package com.kswaughs.router;

import org.apache.camel.builder.RouteBuilder;

public class BookRouter extends RouteBuilder {

    @Override
    public void configure() throws Exception {

        from("direct:insert")
            .log("Inserted new Book")
            .bean("bookMapper", "getMap")
            .to("sqlComponent:{{sql.insertBook}}");

        from("direct:select")
            .to("sqlComponent:{{sql.getAllBooks}}")
            .bean("bookMapper", "readBooks")
            .log("${body}");
    }

}
```

If you want to configure routes using spring DSL, replace camelContext section from database-context.xml with below configuration.

```
<camelContext id="bookCtx" xmlns="http://camel.apache.org/schema/spring">

        <!-- use Camel property placeholder loaded from the given file -->
        <propertyPlaceholder id="placeholder" location="classpath:sql.properties"
/>

        <!-- route that generate new orders and insert them in the database -->
        <route id="insertBook-route">
            <from uri="direct:insert" />
            <log message="Inserted new Book" />
            <transform>
                <method ref="bookMapper" method="getMap" />
            </transform>
            <to uri="sqlComponent:{{sql.insertBook}}" />
        </route>

        <route id="getAllBooks-route">
            <from uri="direct:select" />
            <to uri="sqlComponent:{{sql.getAllBooks}}" />
            <to uri="bean:bookMapper?method=readBooks" />
            <log message="${body}" />
        </route>

</camelContext>
```

### Step 5: Create POJO class & Row mapper class

Book.java

```
package com.kswaughs.beans;

public class Book {

    private String bookId;
    private String bookName;
    private String author;
    private String price;
    private String createDate;

    public String getBookId() {
        return bookId;
    }

    public void setBookId(String bookId) {
        this.bookId = bookId;
    }

    public String getBookName() {
        return bookName;
    }

    public void setBookName(String bookName) {
        this.bookName = bookName;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public String getPrice() {
        return price;
    }
    public void setPrice(String price) {
        this.price = price;
    }

    public String getCreateDate() {
        return createDate;
    }

    public void setCreateDate(String createDate) {
        this.createDate = createDate;
    }

    @Override
    public String toString() {
        StringBuilder builder = new StringBuilder();
        builder.append("Book [bookId=");
        builder.append(bookId);
        builder.append(", bookName=");
        builder.append(bookName);
        builder.append(", author=");
        builder.append(author);
        builder.append(", price=");
```

```
        builder.append(price);
        builder.append(", createDate=");
        builder.append(createDate);
        builder.append("]");
        return builder.toString();
    }
}
```

BookMapper.java

```
package com.kswaughs.db.util;

import com.kswaughs.beans.Book;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class BookMapper {

    public Map<String, Object> getMap(Book book) {
        Map<String, Object> answer = new HashMap<String, Object>();
        answer.put("BookId", book.getBookId());
        answer.put("BookName", book.getBookName());
        answer.put("Author", book.getAuthor() );
        answer.put("Price", book.getPrice());
        answer.put("CreateDate", book.getCreateDate());
        return answer;
    }

   public List<Book> readBooks( List<Map<String, String>>  dataList) {

        System.out.println("data:"+dataList);

        List<Book> books = new ArrayList<Book>();

        for (Map<String, String> data : dataList) {

            Book book = new Book();

            book.setBookId(data.get("BookId"));
            book.setBookName(data.get("BookName"));
            book.setAuthor(data.get("Author"));
            book.setPrice(data.get("Price"));
            book.setCreateDate(data.get("CreateDate"));

            books.add(book);
        }

       return books;
    }
}
```

### Step 6: Test the application

CamelBookApp.java

```
package com.kswaughs.app;

import com.kswaughs.beans.Book;

import java.util.Date;
import java.util.List;

import org.apache.camel.CamelContext;
import org.apache.camel.ProducerTemplate;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.kswaughs.db.util.Book;

public class CamelBookApp {

    public static void main(String[] args) {

        try {
            ApplicationContext springCtx = new ClassPathXmlApplicationContext(
                    "database-context.xml");

            CamelContext context = springCtx.getBean("bookCtx",
CamelContext.class);

            context.start();

            ProducerTemplate producerTemplate = context.createProducerTemplate();

            // Insert book 1
            Book book1 = buildBook1();
            String resp = producerTemplate.requestBody("direct:insert",  book1,
String.class);
            System.out.println("resp:"+resp);

            // Insert book 2
            Book book2 = buildBook2();
            resp = producerTemplate.requestBody("direct:insert",  book2,
String.class);
            System.out.println("resp:"+resp);

            // Read all books
            List<Book> resp1 = producerTemplate
                     .requestBody("direct:select",  null, List.class);
            System.out.println("resp1:"+resp1);

        } catch (Exception e) {

            e.printStackTrace();

        }
    }

    private static Book buildBook1() {

        Book book = new Book();

        book.setBookId("FICT1");
        book.setBookName("Rogue Lawyer");
```

```
        book.setAuthor("John Grisham");
        book.setPrice("$10");
        book.setCreateDate(new Date().toString());
        return book;

    }

    private static Book buildBook2() {

        Book book = new Book();

        book.setBookId("FICT2");
        book.setBookName("Doctor Sleep");
        book.setAuthor("Stephen King");
        book.setPrice("$9");
        book.setCreateDate(new Date().toString());
        return book;

    }
}
```

## Console Logs

```
INFO|08/30/2016 11:05:26 918|Refreshing
org.springframework.context.support.ClassPathXmlApplicationContext@67fc85d:
startup date [Tue Aug 30 11:05:26 IST 2016]; root of context hierarchy
INFO|08/30/2016 11:05:26 966|Loading XML bean definitions from class path resource
[database-context.xml]
INFO|08/30/2016 11:05:29 385|Executing SQL script from class path resource [db-
schema.sql]
INFO|08/30/2016 11:05:29 529|Executed SQL script from class path resource [db-
schema.sql] in 144 ms.
INFO|08/30/2016 11:05:29 806|Apache Camel 2.16.0 (CamelContext: bookCtx) is
starting
INFO|08/30/2016 11:05:29 807|JMX is enabled
INFO|08/30/2016 11:05:29 961|Loaded 198 type converters
INFO|08/30/2016 11:05:30 005|Runtime endpoint registry is in extended mode
gathering usage statistics of all incoming and outgoing endpoints (cache limit:
1000)
INFO|08/30/2016 11:05:30 127|AllowUseOriginalMessage is enabled. If access to the
original message is not needed, then its recommended to turn this option off as it
may improve performance.
INFO|08/30/2016 11:05:30 127|StreamCaching is not in use. If using streams then
its recommended to enable stream caching. See more details at
http://camel.apache.org/stream-caching.html
INFO|08/30/2016 11:05:30 211|Route: route1 started and consuming from:
Endpoint[direct://insert]
INFO|08/30/2016 11:05:30 213|Route: route2 started and consuming from:
Endpoint[direct://select]
INFO|08/30/2016 11:05:30 213|Total 2 routes, of which 2 is started.
INFO|08/30/2016 11:05:30 214|Apache Camel 2.16.0 (CamelContext: bookCtx) started
in 0.408 seconds
INFO|08/30/2016 11:05:30 217|Apache Camel 2.16.0 (CamelContext: bookCtx) is
starting
INFO|08/30/2016 11:05:30 217|Total 2 routes, of which 2 is started.
INFO|08/30/2016 11:05:30 217|Apache Camel 2.16.0 (CamelContext: bookCtx) started
in 0.000 seconds
INFO|08/30/2016 11:05:30 228|Inserted new Book
resp:{BookName=Rogue Lawyer, Price=$10, Author=John Grisham, BookId=FICT1,
CreateDate=Tue Aug 30 11:05:30 IST 2016}
INFO|08/30/2016 11:05:30 297|Inserted new Book
resp:{BookName=Doctor Sleep, Price=$9, Author=Stephen King, BookId=FICT2,
CreateDate=Tue Aug 30 11:05:30 IST 2016}
data:[{BOOKID=FICT1, BOOKNAME=Rogue Lawyer, AUTHOR=John Grisham, PRICE=$10,
CREATEDATE=Tue Aug 30 11:05:30 IST 2016}, {BOOKID=FICT2, BOOKNAME=Doctor Sleep,
AUTHOR=Stephen King, PRICE=$9, CREATEDATE=Tue Aug 30 11:05:30 IST 2016}]
INFO|08/30/2016 11:05:30 328|[Book [bookId=FICT1, bookName=Rogue Lawyer,
author=John Grisham, price=$10, createDate=Tue Aug 30 11:05:30 IST 2016], Book
[bookId=FICT2, bookName=Doctor Sleep, author=Stephen King, price=$9,
createDate=Tue Aug 30 11:05:30 IST 2016]]
resp1:[Book [bookId=FICT1, bookName=Rogue Lawyer, author=John Grisham, price=$10,
createDate=Tue Aug 30 11:05:30 IST 2016], Book [bookId=FICT2, bookName=Doctor
Sleep, author=Stephen King, price=$9, createDate=Tue Aug 30 11:05:30 IST 2016]]
```

## Maven dependencies

pom.xml

```
<properties>
    <spring.version>4.1.6.RELEASE</spring.version>
    <camelspring.version>2.16.0</camelspring.version>
</properties>

<dependencies>
    <!-- Camel Dependencies -->
    <dependency>
        <groupId>org.apache.camel</groupId>
        <artifactId>camel-core</artifactId>
        <version>${camelspring.version}</version>
    </dependency>
     <dependency>
        <groupId>org.apache.camel</groupId>
        <artifactId>camel-cxf</artifactId>
        <version>${camelspring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.camel</groupId>
        <artifactId>camel-spring</artifactId>
        <version>${camelspring.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.camel</groupId>
        <artifactId>camel-sql</artifactId>
        <version>2.17.1</version>
    </dependency>
    <!-- End of Camel Dependencies -->

    <dependency>
        <groupId>commons-dbcp</groupId>
        <artifactId>commons-dbcp</artifactId>
        <version>1.2.2</version>
    </dependency>

    <dependency>
        <groupId>commons-pool</groupId>
        <artifactId>commons-pool</artifactId>
        <version>1.6</version>
    </dependency>
    <dependency>
        <groupId>org.apache.derby</groupId>
        <artifactId>derby</artifactId>
        <version>10.11.1.1</version>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
    </dependency>
</dependencies>
```

### Related Links

How to call Stored procedures using Camel SQL Stored Component