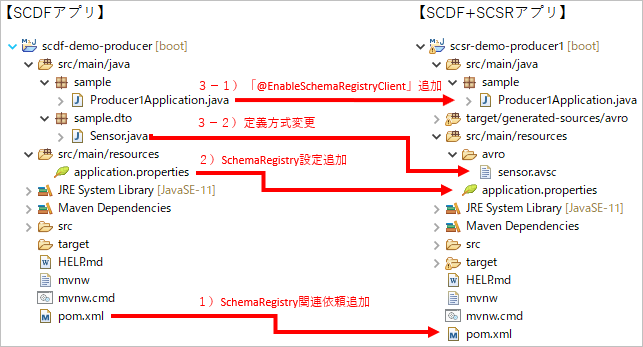
SpringCloudSchemaRegistryと組み合わせのために、カスタムAPPのソースコードの変更について

# 変更点の概要



# 各変更点の説明

## pomファイルの変更点

1. **「Spring Cloud Schema Registry Client」関連する依頼関係の追加**

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-schema-registry-client</artifactId>

</dependency>

<dependency>

<groupId>org.apache.avro</groupId>

<artifactId>avro</artifactId>

<version>1.10.2</version>

</dependency>

1. **「Spring Cloud Schema Registry Client」関連するプラグインの追加**

「avro-maven-plugin」を使用して、Mavenから直接Javaソースを生成します。

<plugin>

<groupId>org.apache.avro</groupId>

<artifactId>avro-maven-plugin</artifactId>

<version>1.9.2</version>

<executions>

<execution>

<phase>generate-sources</phase>

<goals>

<goal>schema</goal>

</goals>

<configuration>

<sourceDirectory>src/main/resources/avro</sourceDirectory>

<outputDirectory>${project.build.directory}/generated-sources/avro</outputDirectory>

</configuration>

</execution>

</executions>

</plugin>

いくつかの属性の役割

* 「sourceDirectory」:マファイルの Avro ソースディレクトリです。

デフォルト：${basedir}/src/main/avro

* 「outputDirectory」: Avroがコード生成されたソースを書き込むディレクトリです。

デフォルト： ${project.build.directory}/generated-sources/avro

## 設定ファイルapplication.propertiesの変更点

以下の設定が追加されました。

* 「spring.cloud.stream.bindings.output.contentType=application/\*+avro」：「Avro based schema registry client」を有効にします。この設定はアウトバウンドにスコープされているため、Sinkアプリはこの設定を構成する必要はありません。

補足:アウトバウンド変換中、メッセージコンバータは、SchemaRegistryClientを使用して、各アウトバウンドメッセージのスキーマ（タイプに基づく）を推論し、それを（ペイロードタイプに基づく）サブジェクトに登録しようとします。同一のスキーマが既に見つかっている場合は、そのスキーマへの参照が取得されます。見つからない場合は、スキーマが登録され、新しいバージョン番号が提供されます。メッセージは、次のスキームを使用した 「Content-Type」ヘッダとともに送信されます： 「application/[prefix].[subject].v[version]+avro」（「prefix」は設定可能で、「subject」はペイロードタイプから推測されます）。

* 「spring.cloud.schemaRegistryClient.endpoint=http://localhost:8990/」：スキーマレジストリの場所。これを設定するときは、プロトコル（httpまたはhttps）、ポート、コンテキストパスを含む完全なURLを使用してください。
* 「spring.cloud.schema.avro.schemaLocations=classpath:avro/sensor.avsc」：このプロパティに記載されているすべての「.avsc」ファイルをスキーマレジストリに登録します。

## Javaコードの変更点

* 「@EnableSchemaRegistryClient」を使用して、「Spring Cloud Stream schema registry client」を構成します。

@SpringBootApplication

@EnableSchemaRegistryClient

public static class AvroSinkApplication {

//...

}

* スキーマファイルの追加

「src\main\resources\avro」ディレクトリにスキーマファイル「sensor.avsc」を追加します。

スキーマファイル例：

「scsr-demo-consumer」のスキーマファイル：

{

"namespace" : "sample.dto",

"type" : "record",

"name" : "Sensor",

"fields" : [

{"name":"id","type":"string"},

{"name":"internalTemperature", "type":"float", "default":0.0, "aliases":["temperature"]},

{"name":"externalTemperature", "type":"float", "default":0.0},

{"name":"acceleration", "type":"float","default":0.0},

{"name":"velocity","type":"float","default":0.0}

]

}

「scsr-demo-producer1」のスキーマファイル：

{

"namespace" : "sample.dto",

"type" : "record",

"name" : "Sensor",

"fields" : [

{"name":"id","type":"string"},

{"name":"temperature", "type":"float", "default":0.0},

{"name":"acceleration", "type":"float","default":0.0},

{"name":"velocity","type":"float","default":0.0}

]

}

「scsr-demo-producer2」のスキーマファイル：

{

"namespace" : "sample.dto",

"type" : "record",

"name" : "Sensor",

"fields" : [

{"name":"id","type":"string"},

{"name":"internalTemperature", "type":"float", "default":0.0, "aliases":["temperature"]},

{"name":"externalTemperature", "type":"float", "default":0.0},

{"name":"acceleration", "type":"float","default":0.0},

{"name":"velocity","type":"float","default":0.0},

{"name":"accelerometer","type":[

"null",{

"type":"array",

"items":"float"

}

]},

{"name":"magneticField","type":[

"null",{

"type":"array",

"items":"float"

}

]}

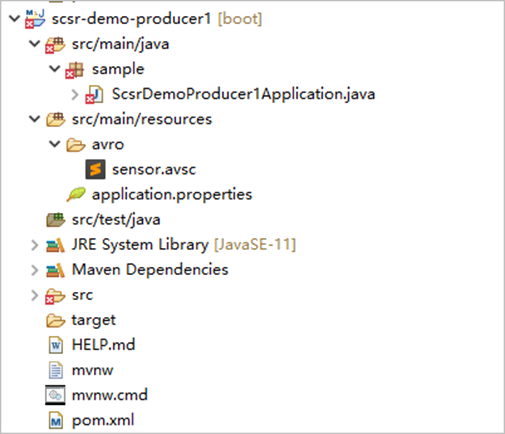
]

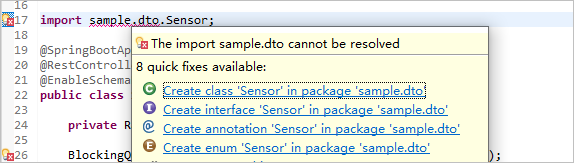
}

* [「avsc定義File→Javaソース」変換手順](#_「avsc定義File→Javaソース」変換手順)を参照して、スキーマファイルを使用して、Mavenから直接Javaソースを生成します。

# 「avsc定義File→Javaソース」変換手順

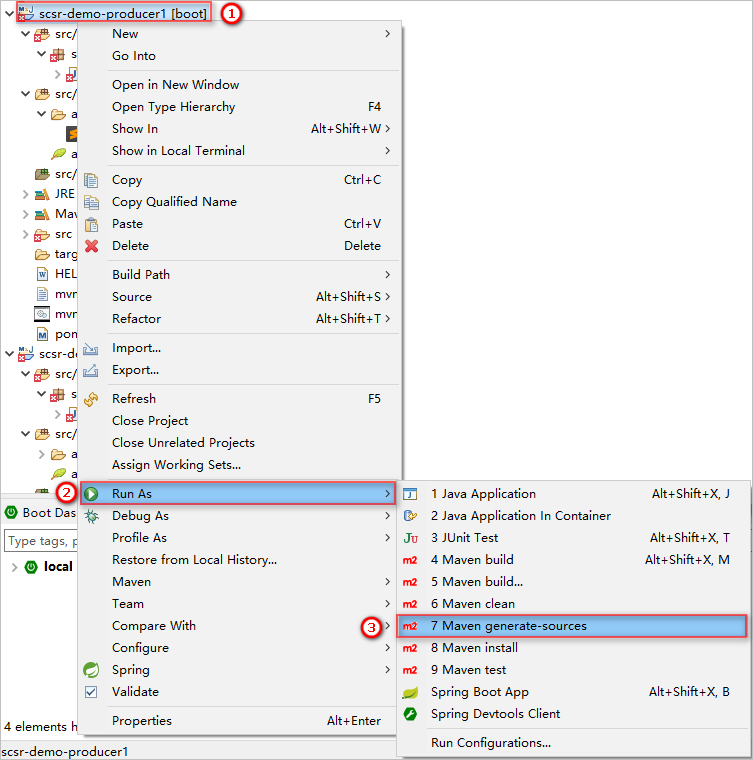
スキーマファイルを作成して使用すると、対応するJavaソースがないため、プロジェクトは次のようなエラーを報告します。



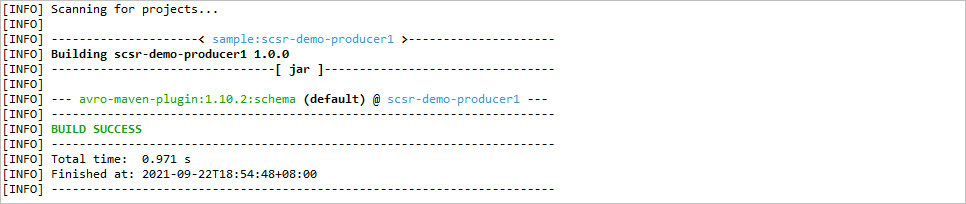


次の手順を使用して、Javaソースを生成し、ビルドパスに追加します。

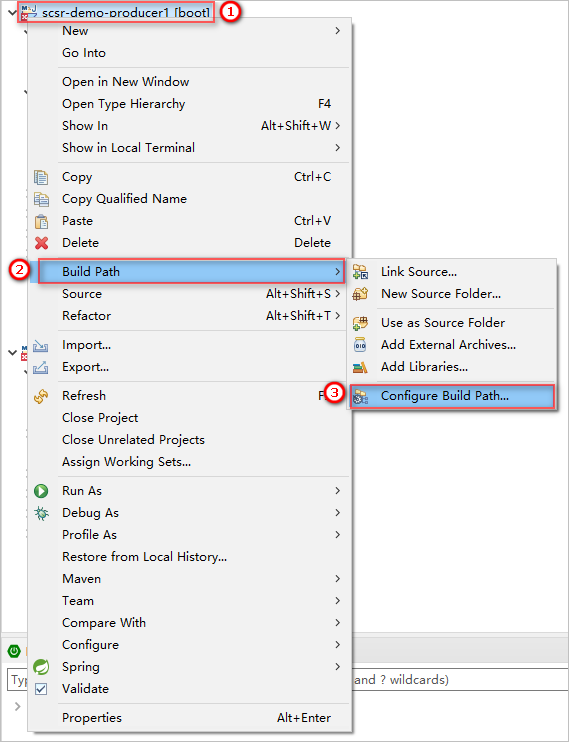
1. 指定した項目を選択して右クリックし、「Run As」をクリックしてから「Maven generate-sources」をクリックします。



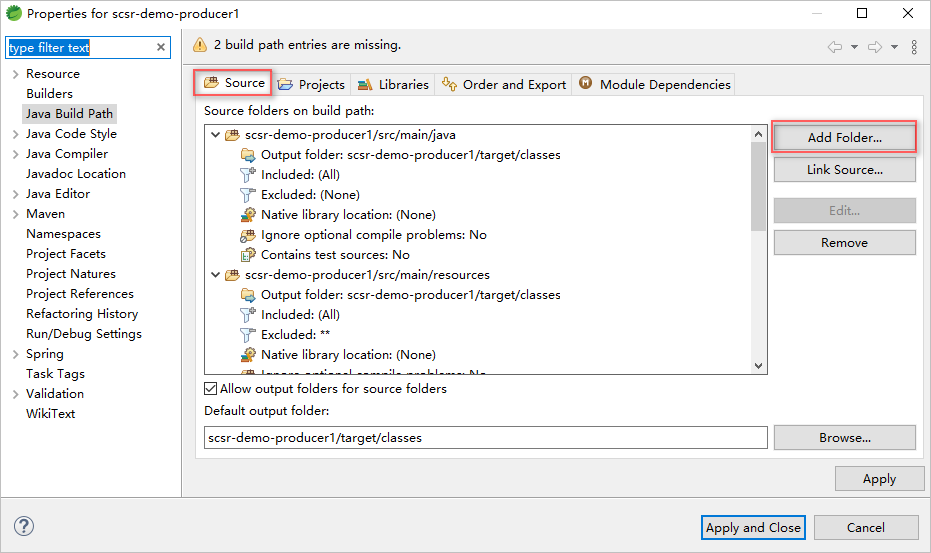
1. Mavenは次のように正常に実行されます。



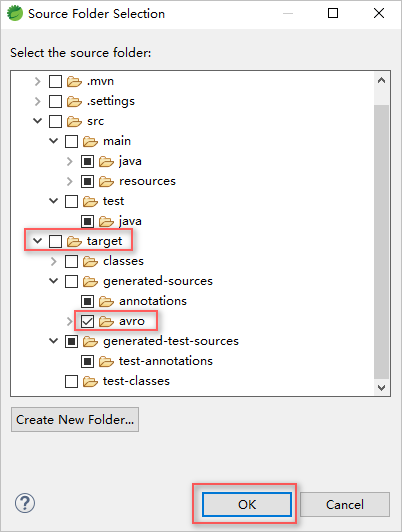
1. 指定した項目を選択して右クリックし、「Build Path」をクリックしてから「Configure Build Path...」をクリックします。



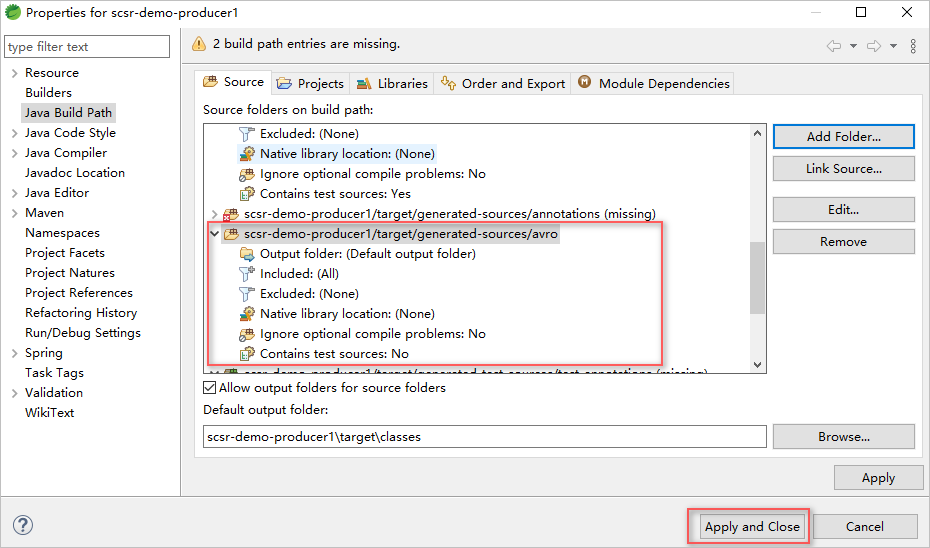
1. 「Source」をクリックして、「Add Folder...」をクリックします。



1. 「target/avro」ディレクトリを選択し、「OK」をクリックします。



1. 正常に追加されたことを確認し、「Apply and Close」をクリックします。



1. Javaソースを正常に生成し、ビルドパスで構成すると、プロジェクトはエラーを報告しなくなります

