Rapport Projet Cloud Integration | TADJER Badr & BOIRARD Cédric

Introduction

This project is designed to transfer data from a source A (CSV file) to a source B (Database or JSON file).

Follow this link to see the source code :
https://github.com/ImBadr/ST2DCCC_Project_M2_APP

Execution

requirements:

- Java 17
- Java IDE
- Web Browser

This project can be run in two different ways.

- You can run the XML configuration `json_channels.xml` (csv -> json).
- You can run the XML configuration `db_channels.xml` (csv -> database).

json_channel.xml

To execute the XML configuration `json_channels.xml` , you need to comment the lines below.

```
@SpringBootApplication
//@ImportResource("classpath:db_channels.xml")
public class Main {
    public static void main(String[] args) {
        //SpringApplication.run(Main.class, args);
        new ClassPathXmlApplicationContext("json_channels.xml");
    }
}
```

db_channel.xml

To execute the XML configuration `db_channels.xml` , you need to comment the line below.

```
@SpringBootApplication
@ImportResource("classpath:db_channels.xml")
public class Main {
    public static void main(String[] args) {
        SpringApplication.run(Main.class, args);
        //new ClassPathXmlApplicationContext("json_channels.xml");
    }
}
```

Results obtained

input files: CSV

`movies.csv`

```
IdMovie,Title,ReleaseDate,Author
1,Interstellar,2014,Christopher Nolan
2,Gladiator,2000,Ridley Scott
3,Snatch,2000,Guy Ritchie
4,American Gangster,2007,Ridley Scott
```

`actors.csv`

```
IdActor,Name,BirthYear,IdMovie
1,Matthew McConaughey,1969,1
2,Russell Crowe,1964,2
3,Brad Pitt,1963,3
4,Denzel Washington,1954,4
```

output files : JSON | Database

JSON files results

`movies.json`

```
[
  {"id":1,"title":"Interstellar","releaseDate":2014,"author":"Christopher
```

```
 Nolan"},
   {"id":2,"title":"Gladiator","releaseDate":2000,"author":"Ridley Scott"},
   {"id":3,"title":"Snatch","releaseDate":2000,"author":"Guy Ritchie"},
   {"id":4,"title":"American Gangster","releaseDate":2007,"author":"Ridley
 Scott"}
 ]
```

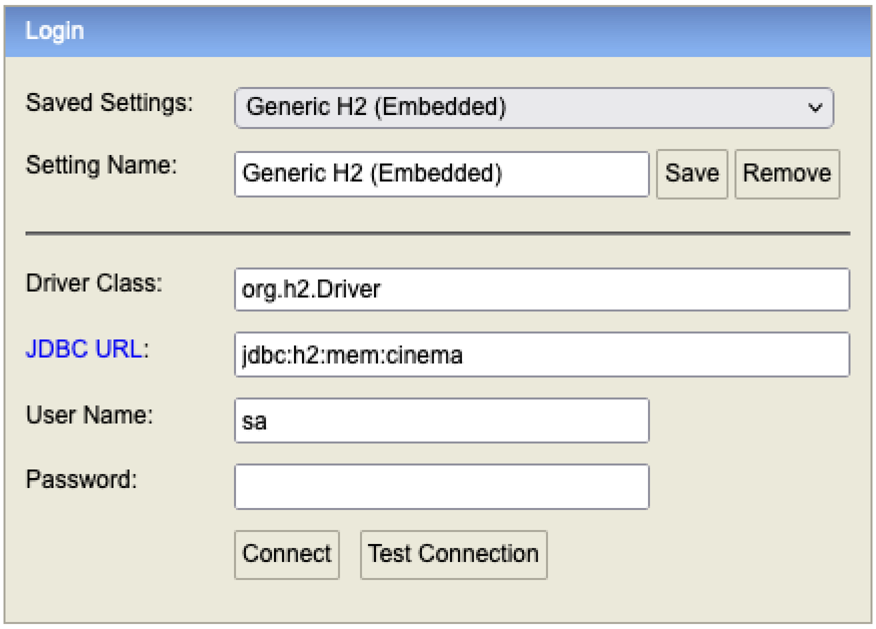
```
actors.json
```

```
 [
   {"id":1,"name":"Matthew McConaughey","birthYear":1969,"idMovie":1},
   {"id":2,"name":"Russell Crowe","birthYear":1964,"idMovie":2},
   {"id":3,"name":"Brad Pitt","birthYear":1963,"idMovie":3},
   {"id":4,"name":"Denzel Washington","birthYear":1954,"idMovie":4}
 ]
```

Database results

To see the result of the execution :

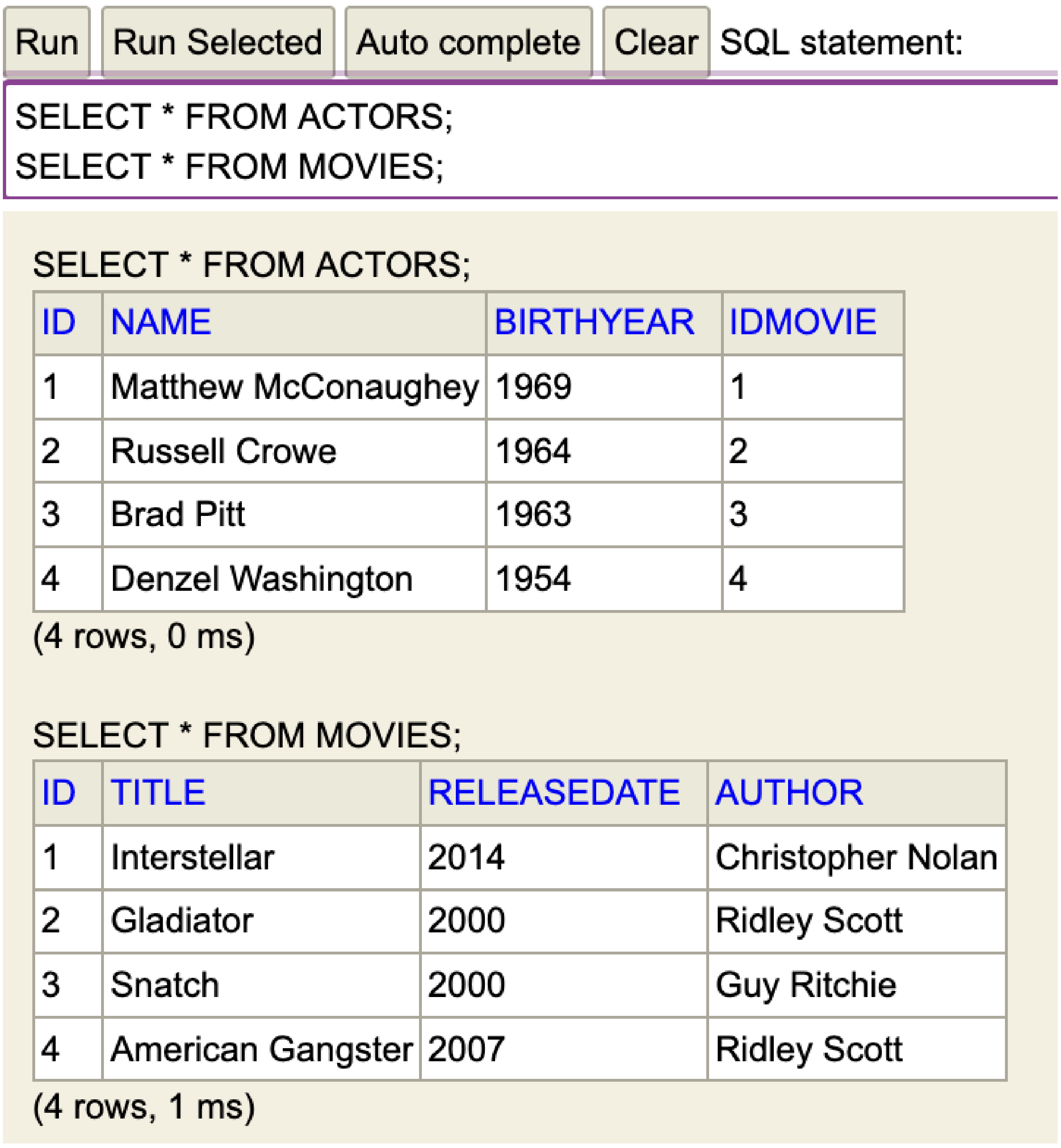
- launch browser to : `localhost:9006/h2-console/`
- database name : `cinema`
- user : `sa`
- leave blank password


Login

Saved Settings:	Generic H2 (Embedded)
Setting Name:	Generic H2 (Embedded)
Driver Class:	org.h2.Driver
JDBC URL:	jdbc:h2:mem:cinema
User Name:	sa
Password:	

execute SQL requests :

```
SELECT * FROM MOVIES;
SELECT * FROM ACTORS;
```



SQL statement:

```
SELECT * FROM ACTORS;
SELECT * FROM MOVIES;
```

SELECT * FROM ACTORS;

ID	NAME	BIRTHYEAR	IDMOVIE
1	Matthew McConaughey	1969	1
2	Russell Crowe	1964	2
3	Brad Pitt	1963	3
4	Denzel Washington	1954	4

(4 rows, 0 ms)

SELECT * FROM MOVIES;

ID	TITLE	RELEASEDATE	AUTHOR
1	Interstellar	2014	Christopher Nolan
2	Gladiator	2000	Ridley Scott
3	Snatch	2000	Guy Ritchie
4	American Gangster	2007	Ridley Scott

(4 rows, 1 ms)

XML Configuration files explanations

json_channels.xml

```
<!--  read data from dataIn  with a 2 seconds delay per read  -->
<int-file:inbound-channel-adapter id="CsvInboundChannelAdapter"
```

```
                                    directory="./dataIn"
                                    filename-pattern="*.csv"
                                    channel="CsvInputChannel">
    <int:poller fixed-delay="2000"/>
</int-file:inbound-channel-adapter>



<!--  split a single message into multiple messages  -->
<int-file:splitter id="splitter" apply-sequence="false" charset="UTF-8"
first-line-as-header="true"
                   input-channel="CsvInputChannel" output-
channel="SplitCsvOutputChannel"
                   auto-startup="true"/>



<!--  route messages based on the value of the header  -->
<int:header-value-router id="router" input-channel="SplitCsvOutputChannel"
header-name="file_name" resolution-required="false">
    <int:mapping value="movies.csv" channel="MovieChannel"/>
    <int:mapping value="actors.csv" channel="ActorChannel"/>
</int:header-value-router>



<!--  convert the payload message from one CSV to Object by using mapMovie
function from MapToObject class-->
<int:transformer input-channel="MovieChannel" output-
channel="OutputTransformerChannel" ref="mapToObject" method="mapMovie"/>



<!--  convert the payload message from one CSV to Object by using mapActor
function from MapToObject class-->
<int:transformer input-channel="ActorChannel" output-
channel="OutputTransformerChannel" ref="mapToObject" method="mapActor"/>



<!--  convert the Java Objects created above to Json objects  -->
<int:object-to-json-transformer input-channel="OutputTransformerChannel"
output-channel="JsonOutputChannel"/>



<!--  channel to transfer JSON Objects  -->
<int:channel id="JsonOutputChannel"/>



<!--  write data JSON Objects into dataOut folder using the NameGenerator
class to create the JSON file with the right name  -->
```

```
<int-file:outbound-channel-adapter id="CsvOutboundChannelAdapter"
channel="JsonOutputChannel" filename-generator="nameGenerator"
directory="./dataOut" append-new-line="true" mode="APPEND"/>
```

db_channels.xml

```
<!-- expose an embedded database instance as a bean in a Spring
ApplicationContext  -->
<!-- This will create automatically the database named "cinema"  -->
<!-- This will read the SQL script inside the "script.sql" file and run it
-->
<jdbc:embedded-database id="cinema" type="H2">
    <jdbc:script location="classpath:script.sql"/>
</jdbc:embedded-database>



<!--  read data from dataIn  -->
<int-file:inbound-channel-adapter id="CsvInboundChannelAdapterToDB"
                                  directory="./dataIn"
                                  filename-pattern="*.csv"
                                  channel="CsvInputChannel">
    <int:poller id="poller" fixed-delay="2000"/>
</int-file:inbound-channel-adapter>



<!--  split a single message into multiple messages  -->
<int-file:splitter id="splitterDB" apply-sequence="false" charset="UTF-8"
first-line-as-header="true"
                   input-channel="CsvInputChannel" output-
channel="SplitCsvOutputChannelDB"
                   auto-startup="true"/>



<!--  route messages based on the value of the header  -->
<int:header-value-router id="routerDB" input-
channel="SplitCsvOutputChannelDB" header-name="file_name" resolution-
required="false">
    <int:mapping value="movies.csv" channel="MovieChannel"/>
    <int:mapping value="actors.csv" channel="ActorChannel"/>
</int:header-value-router>



<!--  convert the payload message from one CSV to Object by using mapMovie
function from MapToObject class-->
```

```
<int:transformer input-channel="MovieChannel" output-
channel="JsonMoviesChannel" ref="mapToObject" method="mapMovie"/>


<!--  convert the payload message from one CSV to Object by using mapActor
function from MapToObject class-->
<int:transformer input-channel="ActorChannel" output-
channel="JsonActorsChannel" ref="mapToObject" method="mapActor"/>


<!--  Transfer JSON Objects through this channel  -->
<int:channel id="JsonMoviesChannel"/>


<!--  Transfer JSON Objects through this channel  -->
<int:channel id="JsonActorsChannel"/>


<!--  write data to h2 database  -->
<int-jdbc:outbound-channel-adapter
        query="INSERT INTO MOVIES (id, title, releaseDate, author)
        values (:payload.id, :payload.title, :payload.releaseDate,
:payload.author)"
        data-source="cinema"
        channel="JsonMoviesChannel"/>


<!--  write data to h2 database  -->
<int-jdbc:outbound-channel-adapter
        query="INSERT INTO ACTORS (id, name, birthYear, idMovie)
        values (:payload.id, :payload.name, :payload.birthYear,
:payload.idMovie)"
        data-source="cinema"
        channel="JsonActorsChannel"/>
```