Published in Think Integration

Arunan Sugunakumar Follow

Apr 30, 2021 · 5 min read · Listen

Building a CRUD RESTful API with WSO2 Micro Integrator

With WSO2 Micro Integrator Data Services, users can integrate with different Data Sources and decouple the data from its infrastructure. In other words, when you create a data service in WSO2 Micro Integrator, the data that is stored in a storage system (such as the RDBMS) can be exposed in the form of a service. This allows users (that may be any application or system) to access the data without interacting with the original source of the data.

In this blog, we are going to create a CRUD RESTful API with Data Services in WSO2 Micro Integrator. We'll create a table called *City* in the database, and develop a data service so that we can add and retrieve cities.

Creating a database

We'll use MySQL as our database. Let's use a mysql docker container. (If you have your own database that would be fine too.)

```
docker run --name mysql_db -p 3308:3306 -e MYSQL_ROOT_PASSWORD=root -
d mysql:8.0.23
```

Go inside the docker container and create the database

```
docker exec -i -t mysql_db /bin/bash
```

Login into database. (Enter password as *root*)

```
mysql -u root -p
```

Create database.

```
create database city_db character set latin1;
```

Run the following script.

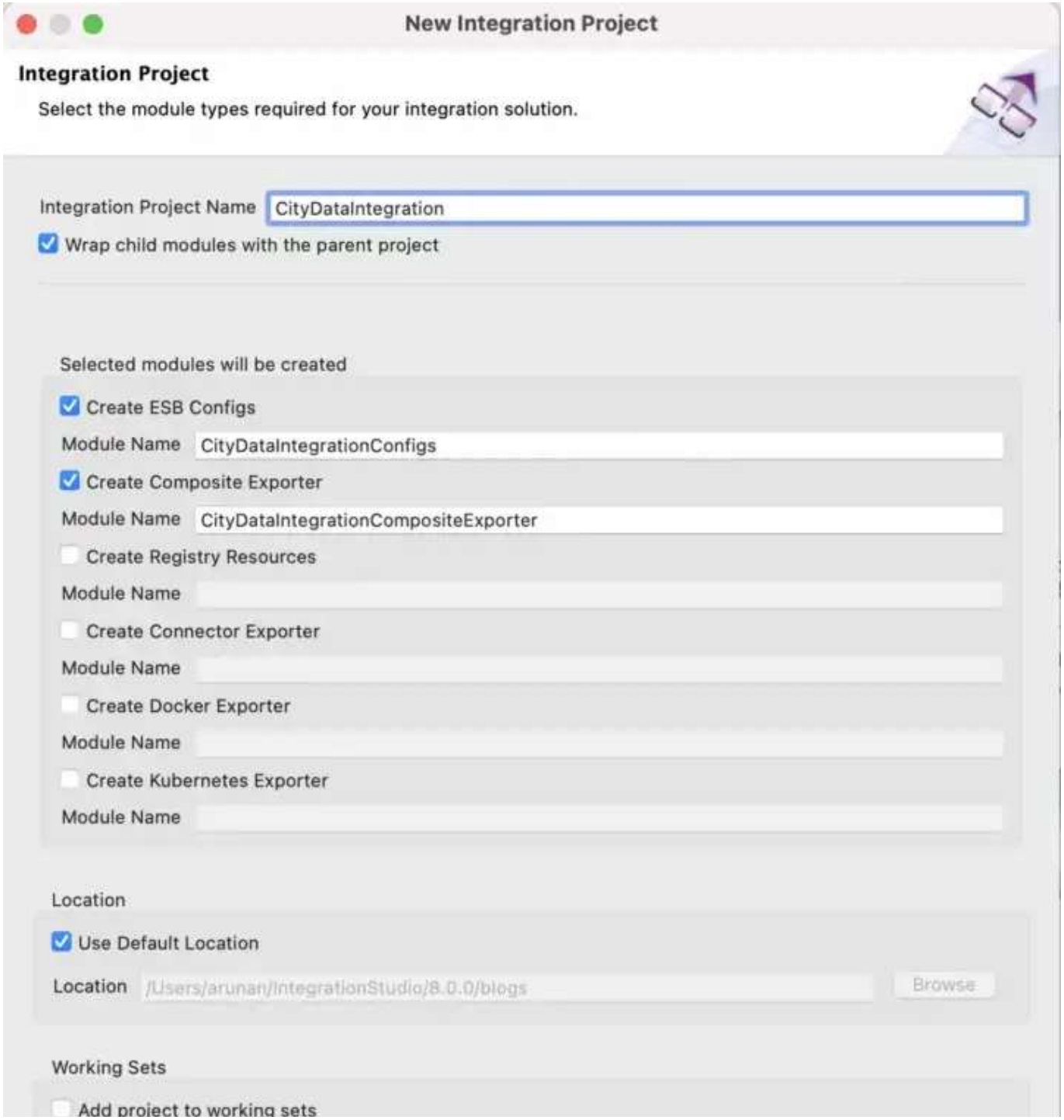
```
use city_db

CREATE TABLE `city` (
`id` int(10) unsigned NOT NULL AUTO_INCREMENT,
`city` varchar(50) DEFAULT NULL,
PRIMARY KEY (`id`)
);
```

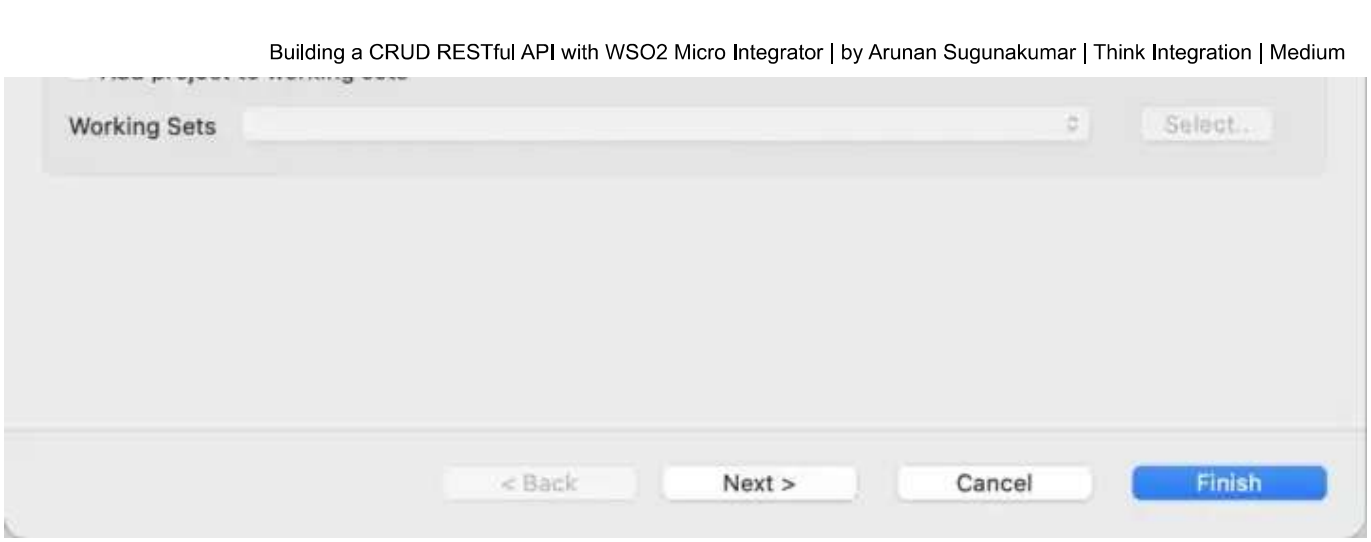
Now that we have created the database, let's create the DataService using Integration Studio.

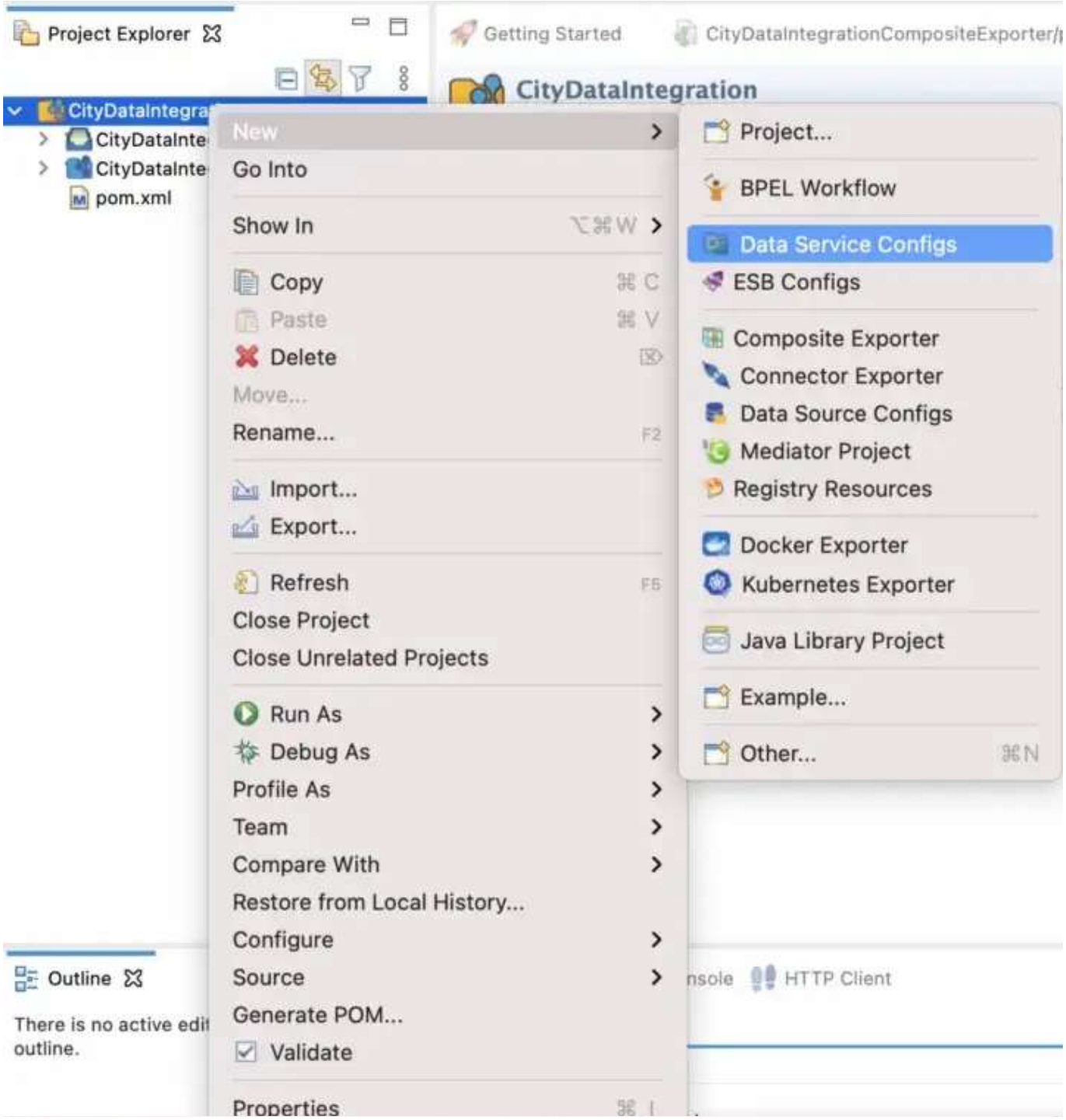
> *I will be using Integration Studio 8.0.0 for demonstration purposes. It contains an embedded Micro Integrator 4.0.0. To download the Integration Studio, please visit https://wso2.com/integration/integration-studio/. Micro Integrator is the Integration runtime of API Manager 4.0.0*

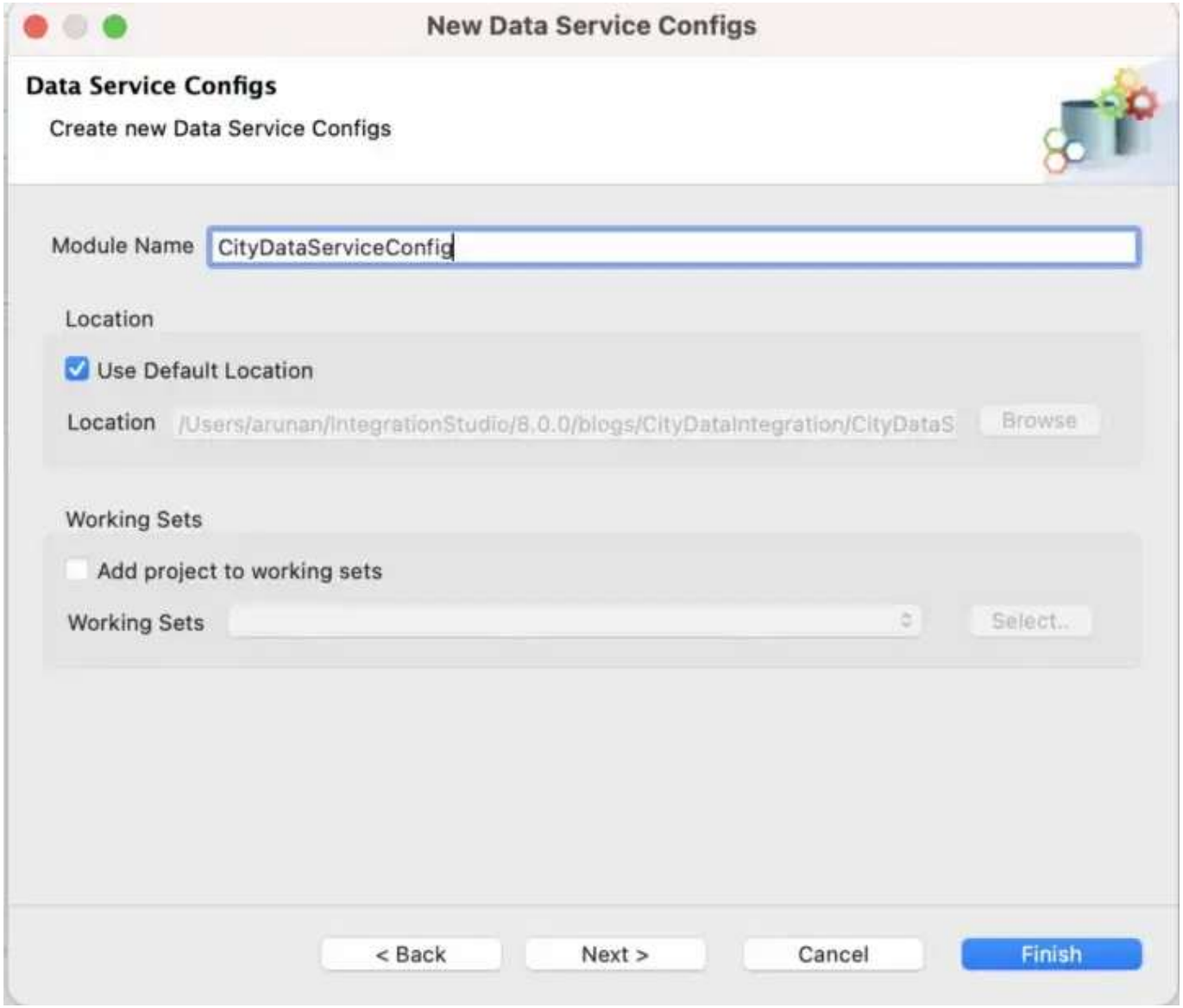
Creating Data Service using Integration Studio

1. Open Integration Studio and create an *Integration Project*.

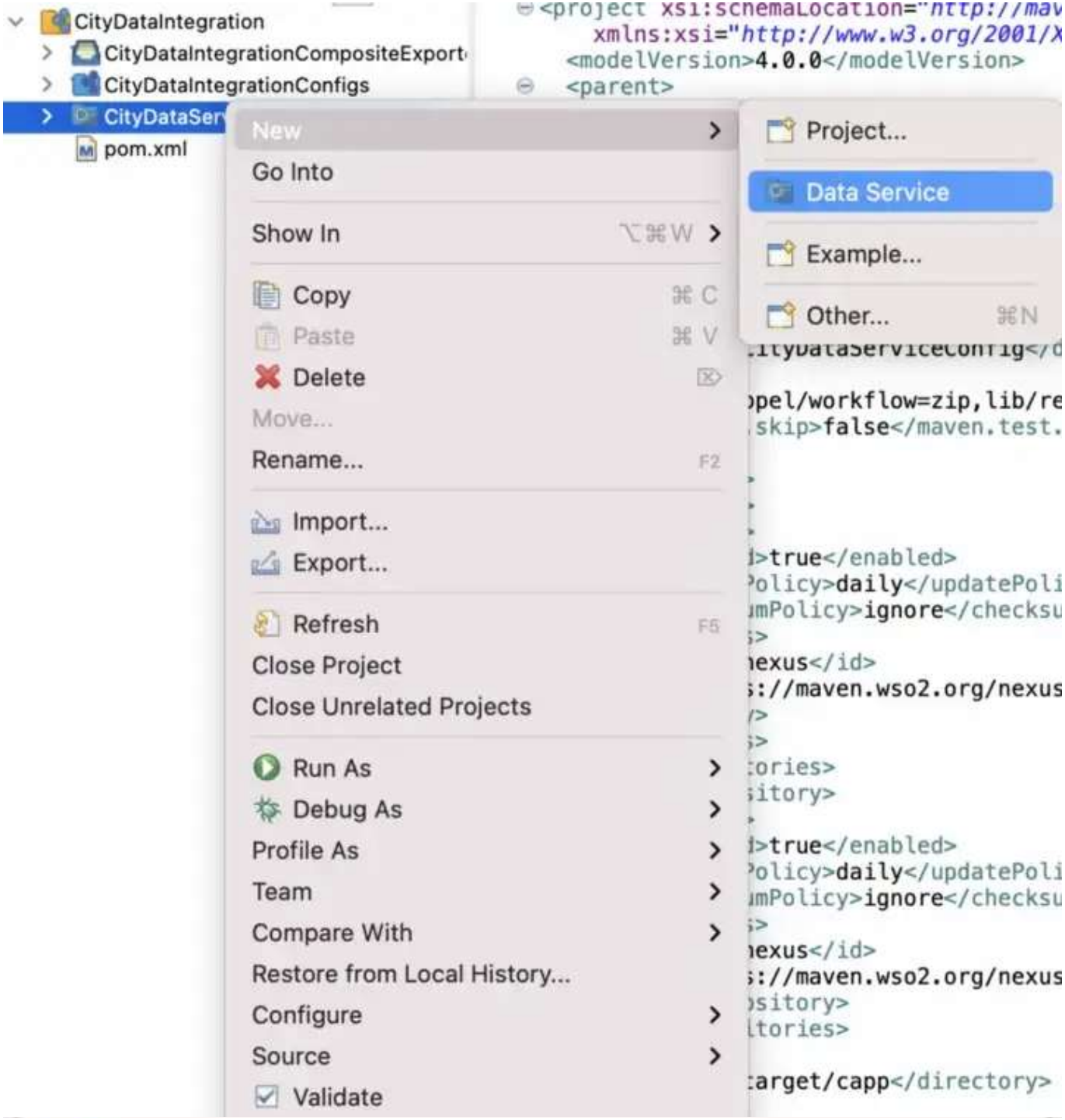

New Integration Project
Integration Project
Select the module types required for your integration solution.
Integration Project Name
CityDataIntegration
Wrap child modules with the parent project
Selected modules will be created
Create ESB Configs
Module Name
CityDataIntegrationConfigs
Create Composite Exporter
Module Name
CityDataIntegrationCompositeExporter
Create Registry Resources
Module Name
Create Connector Exporter
Module Name
Create Docker Exporter
Module Name
Create Kubernetes Exporter
Module Name
Location
Use Default Location
Location
/Users/arunan/IntegrationStudio/8.0.0/blogs
Browse
Working Sets
Add project to working sets

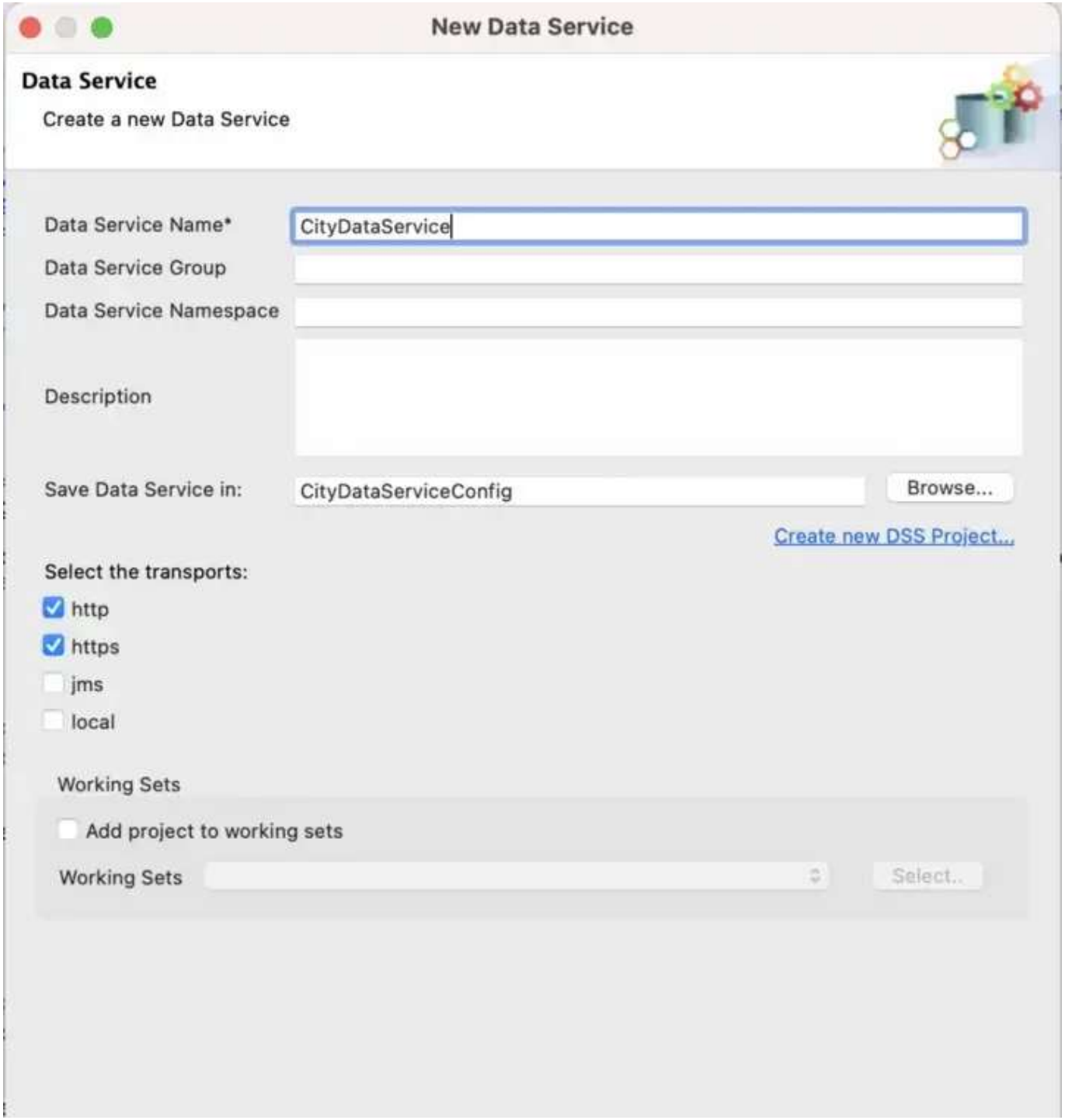

Working Sets
Select...
< Back
Next >
Cancel
Finish


Project Explorer
Getting Started
CityDataIntegrationCompositeExporter/
CityDataIntegration
CityDataIntegra
CityDataInte
CityDataInte
pom.xml
New
Go Into
Show In
Copy
Paste
Delete
Move...
Rename...
Import...
Export...
Refresh
Close Project
Close Unrelated Projects
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
Configure
Source
Generate POM...
Validate
Properties
Project...
BPEL Workflow
Data Service Configs
ESB Configs
Composite Exporter
Connector Exporter
Data Source Configs
Mediator Project
Registry Resources
Docker Exporter
Kubernetes Exporter
Java Library Project
Example...
Other...
Outline
There is no active edi
outline.
nsole
HTTP Client

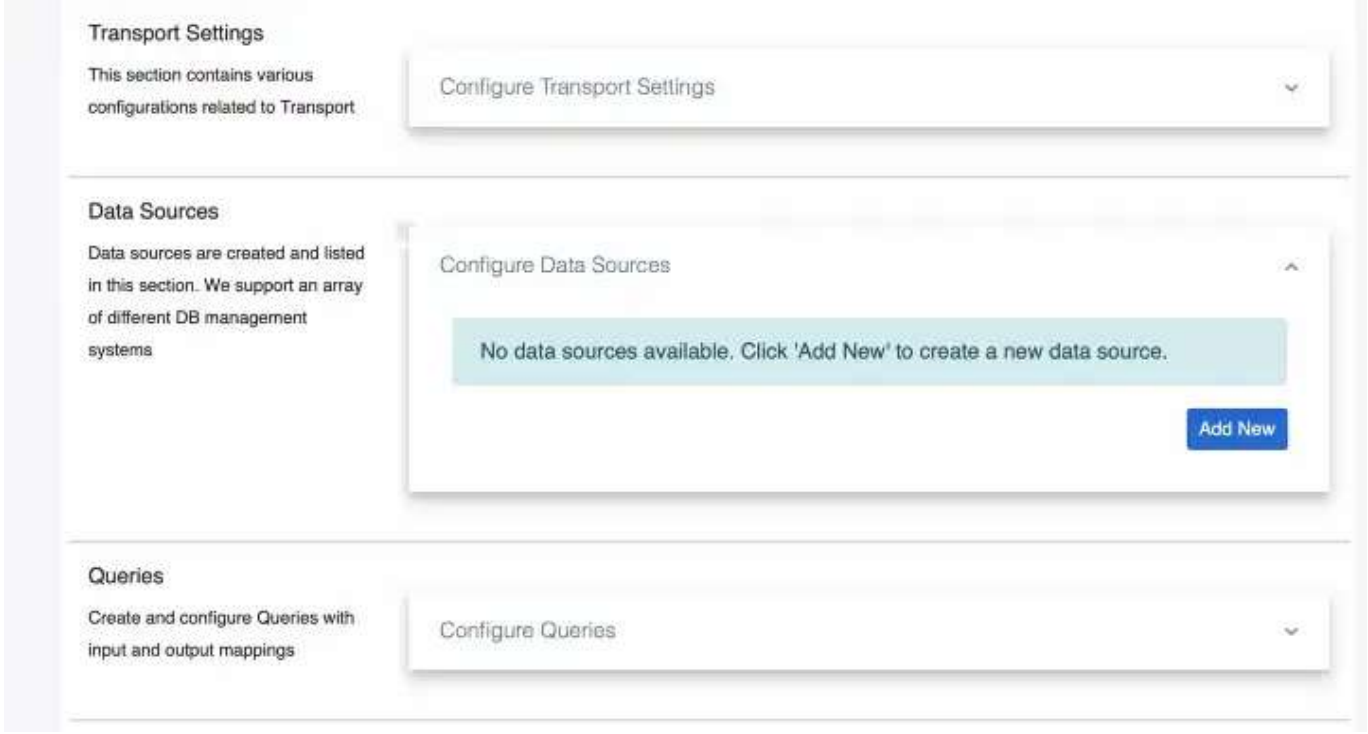


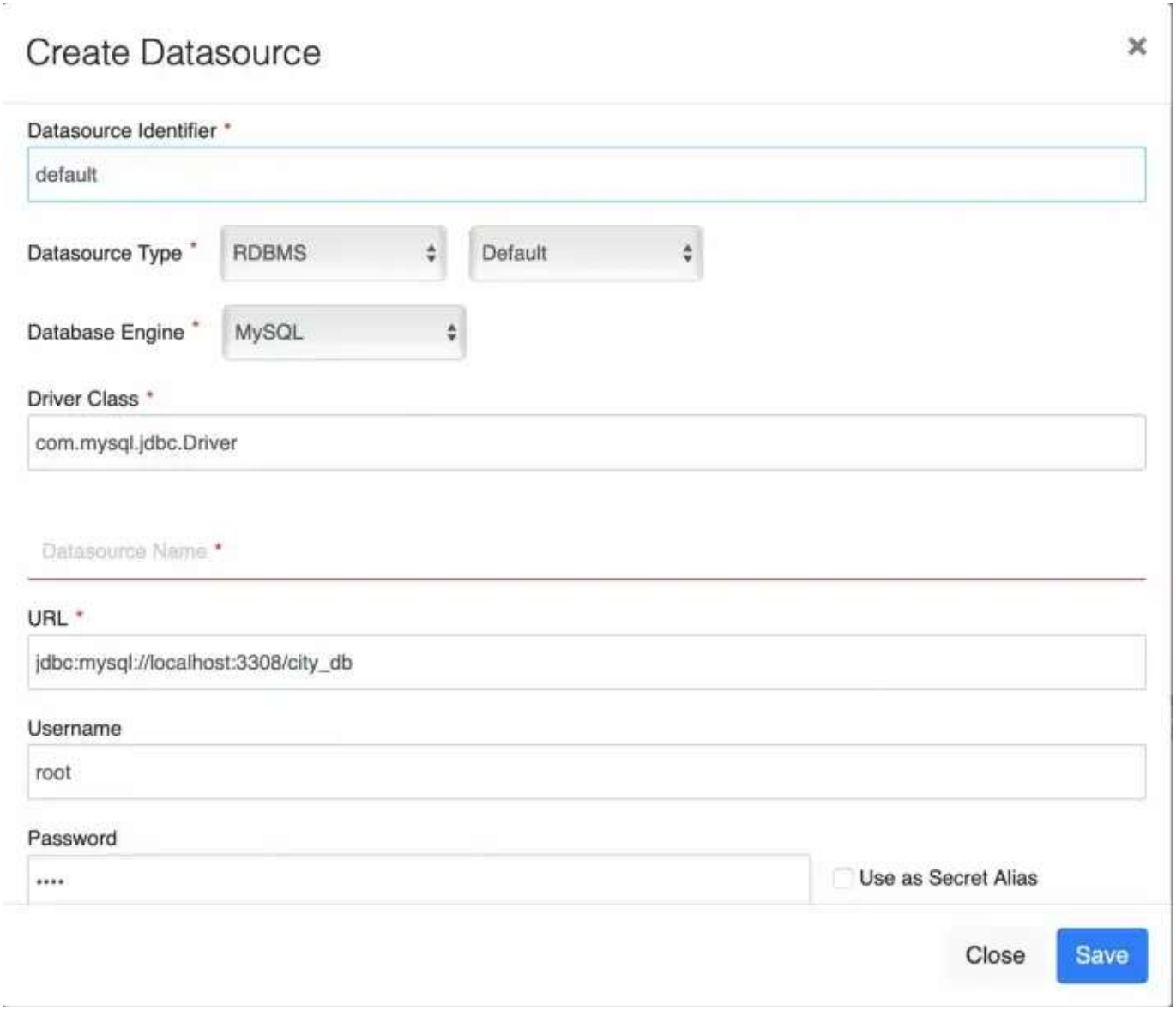
3. Inside the newly created Data Service Config project, create a *Data Service*.


CityDataIntegration
CityDataIntegrationCompositeExport
CityDataIntegrationConfigs
CityDataSer
pom.xml
New
Go Into
Show In
Copy
Paste
Delete
Move...
Rename...
Import...
Export...
Refresh
Close Project
Close Unrelated Projects
Run As
Debug As
Profile As
Team
Compare With
Restore from Local History...
Configure
Source
Validate
Project...
Data Service
Example...
Other...
<project xsi:schemaLocation="http://mav
xmlns:xsi="http://www.w3.org/2001/X
<modelVersion>4.0.0</modelVersion>
<parent>
opel/workflow=zip,lib/re
skip>false</maven.test.
>true</enabled>
Policy>daily</updatePoli
mPolicy>ignore</checksu
nexus</id>
s://maven.wso2.org/nexus
tories>
sitory>
>true</enabled>
Policy>daily</updatePoli
mPolicy>ignore</checksu
nexus</id>
s://maven.wso2.org/nexus
sitory>
tories>
target/capp</directory>


New Data Service
Data Service
Create a new Data Service
Data Service Name*
CityDataService
Data Service Group
Data Service Namespace
Description
Save Data Service in:
CityDataServiceConfig
Browse...
Create new DSS Project...
Select the transports:
http
https
jms
local
Working Sets
Add project to working sets
Working Sets
Select...

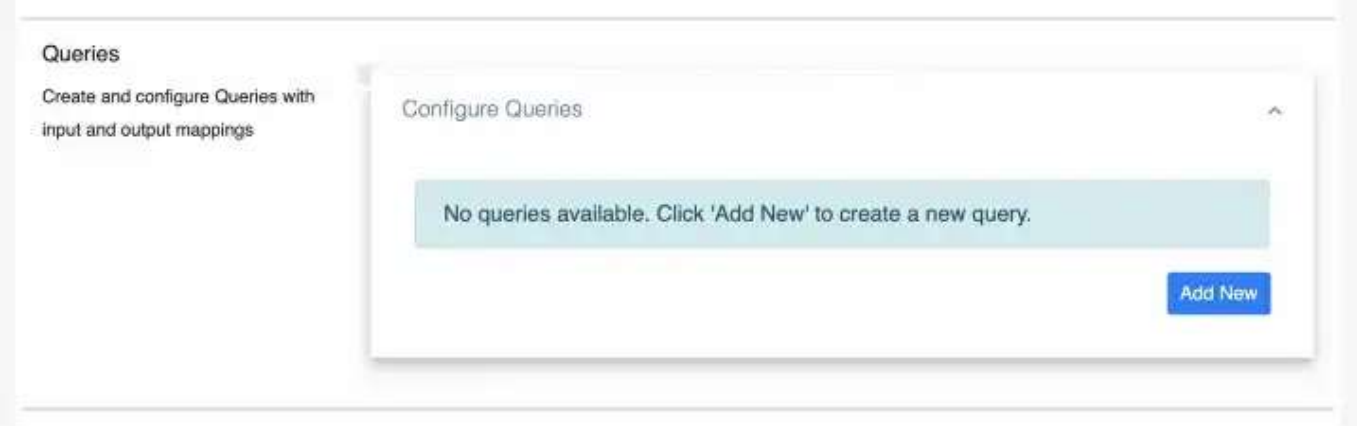

< Back
Next >
Cancel
Finish


Transport Settings
This section contains various configurations related to Transport
Configure Transport Settings
Data Sources
Data sources are created and listed in this section. We support an array of different DB management systems
Configure Data Sources
No data sources available. Click 'Add New' to create a new data source.
Add New
Queries
Create and configure Queries with input and output mappings
Configure Queries

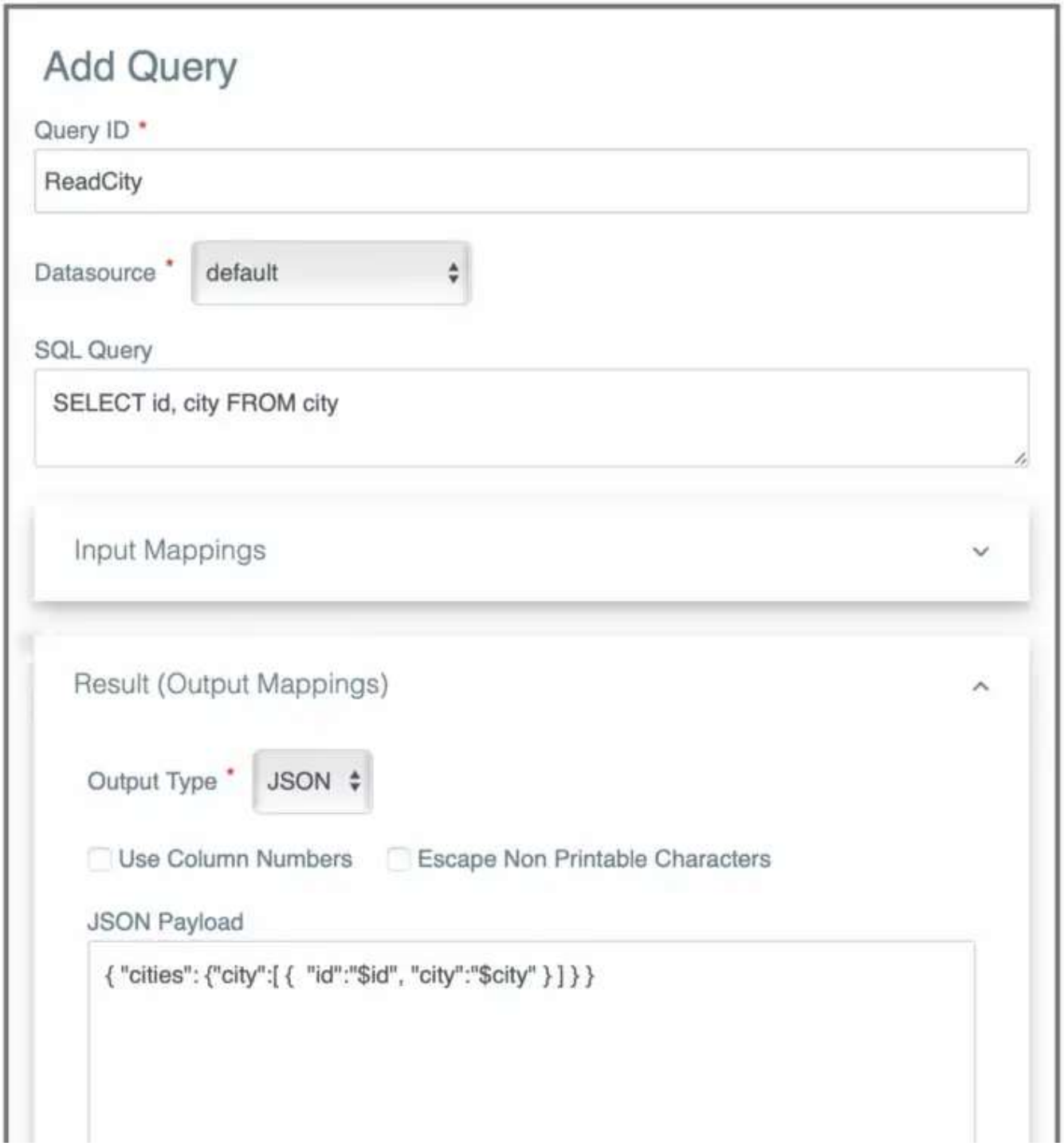


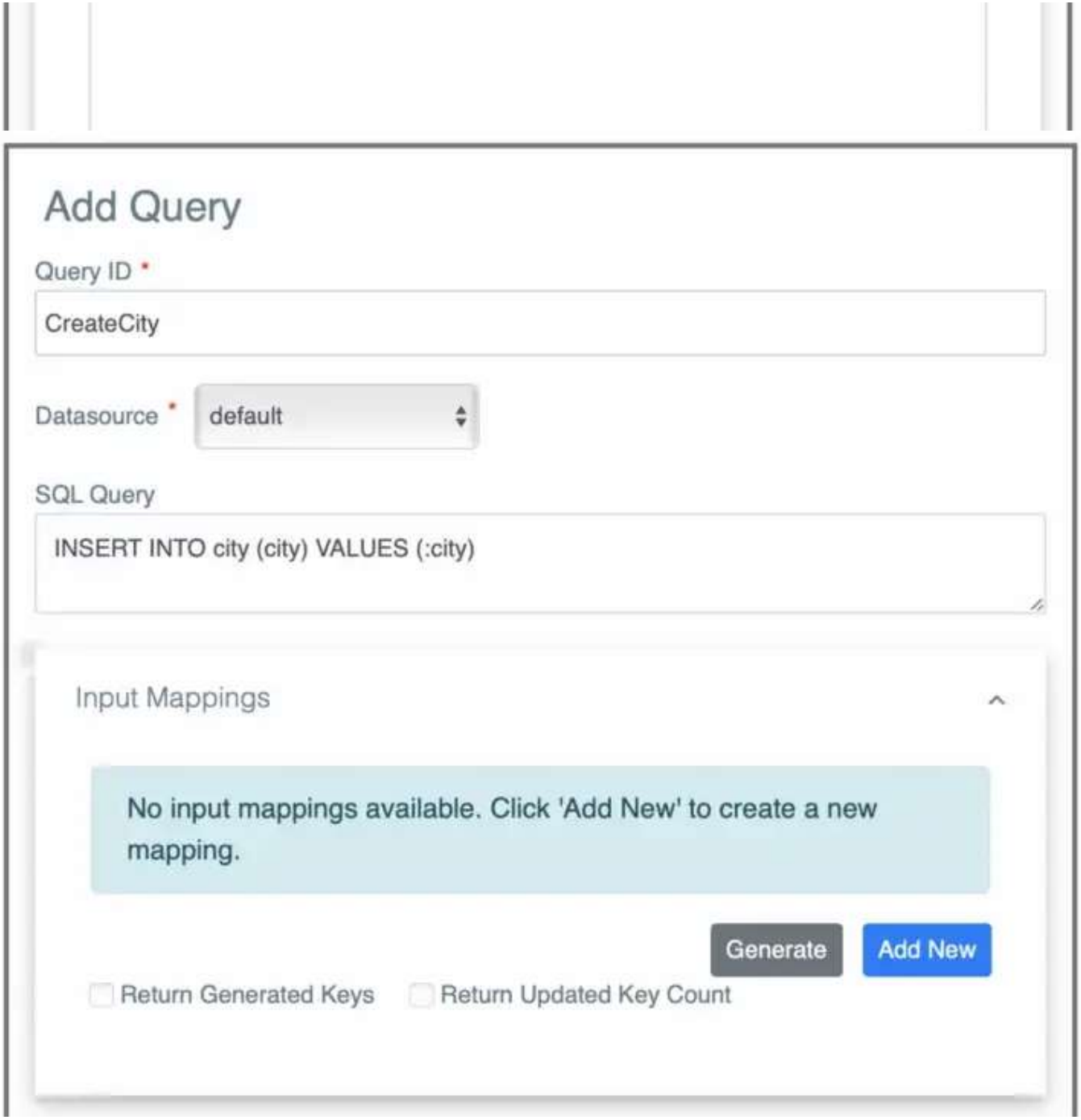
5. Next we need to create the queries against the datasource. Here I am going to create 3 queries : for reading the city list, adding a new city and deleting a

city.

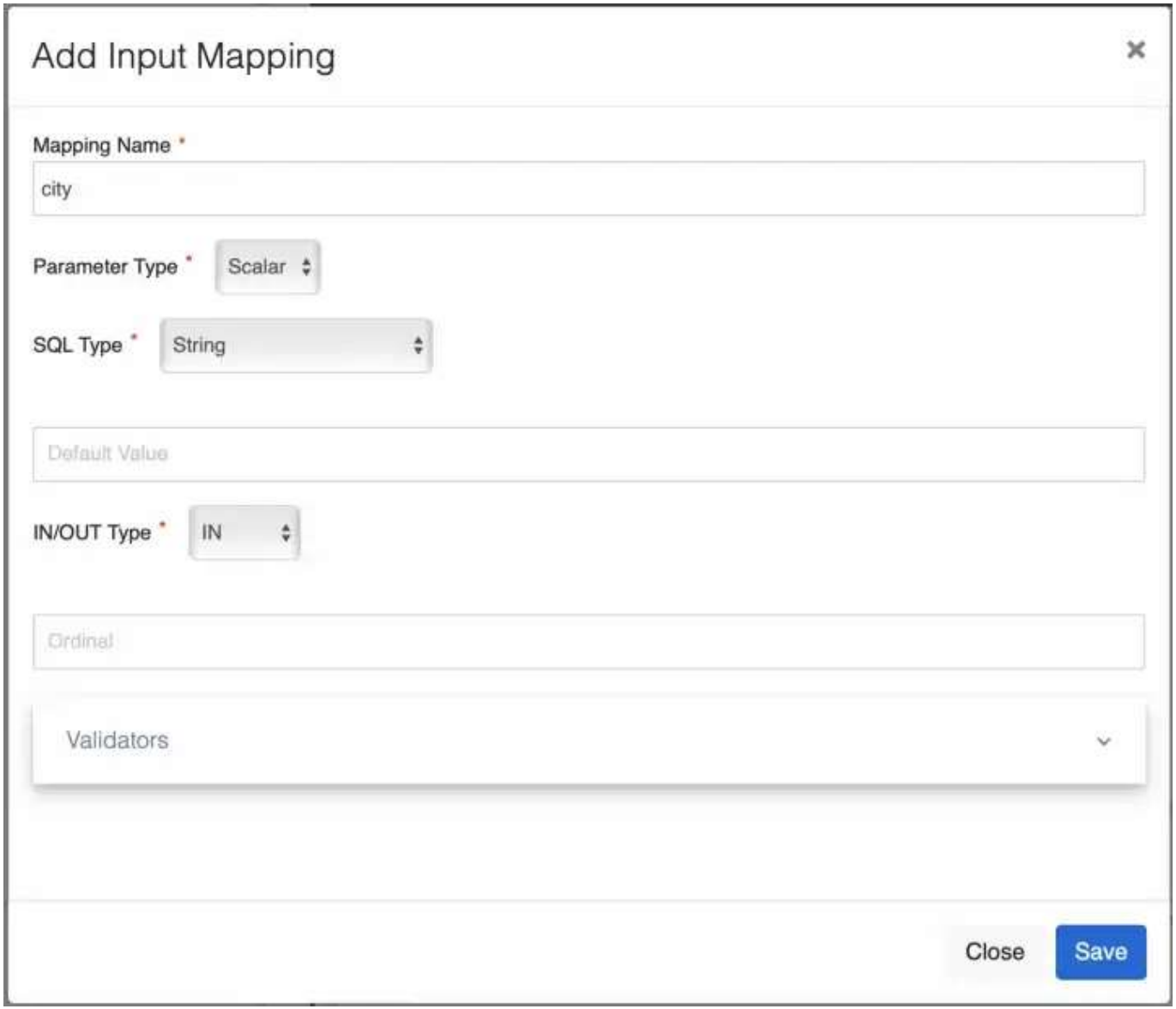


Let’s create the query for reading the list.

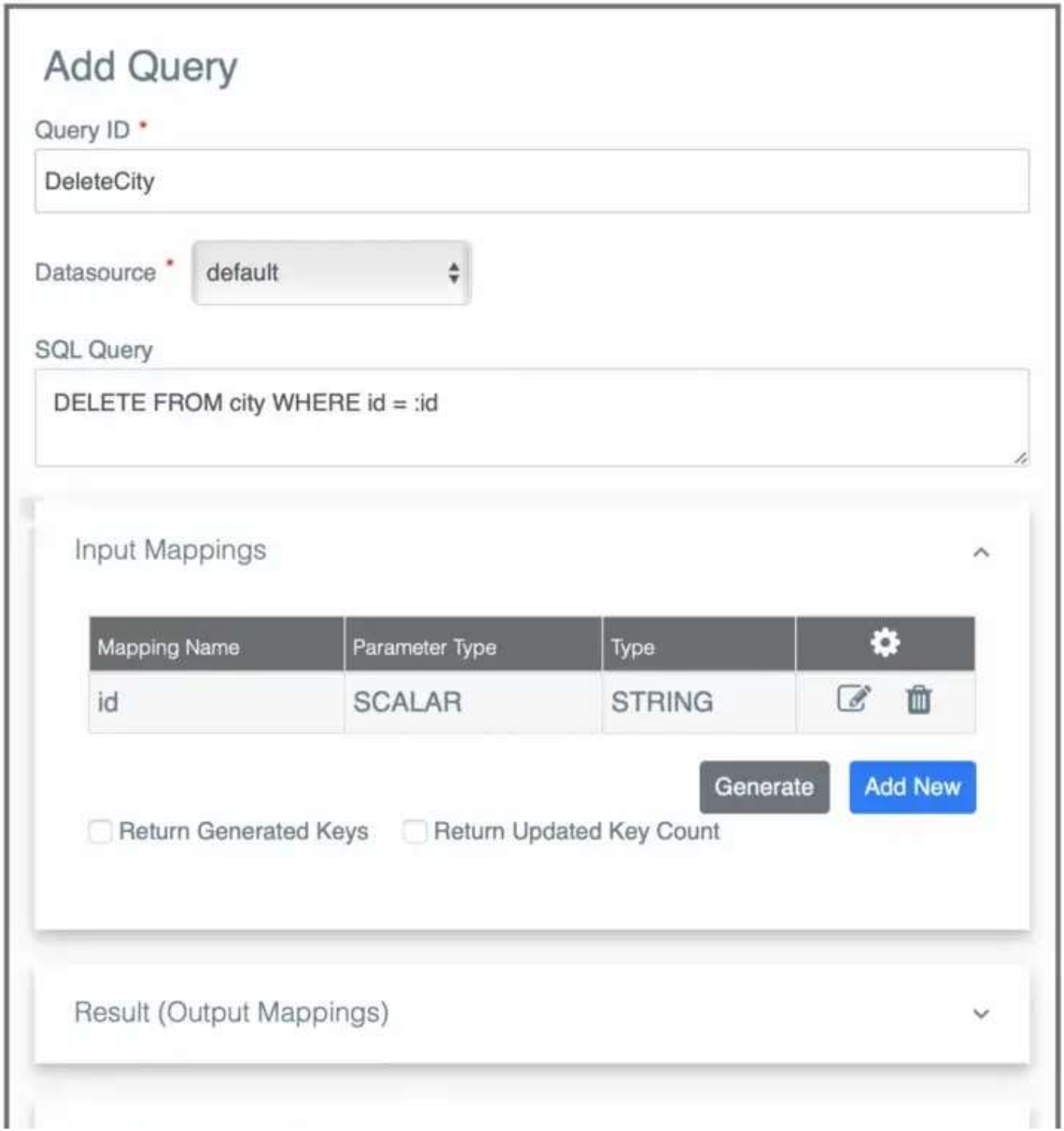

Add Query
Query ID *
ReadCity
Datasource * default
SQL Query
SELECT id, city FROM city
Input Mappings
Result (Output Mappings)
Output Type * JSON
Use Column Numbers
Escape Non Printable Characters
JSON Payload
{ "cities": {"city":[{ "id":"$id", "city":"$city" }] } }


Add Query
Query ID *
CreateCity
Datasource *
default
SQL Query
INSERT INTO city (city) VALUES (:city)
Input Mappings
No input mappings available. Click 'Add New' to create a new mapping.
Generate
Add New
Return Generated Keys
Return Updated Key Count

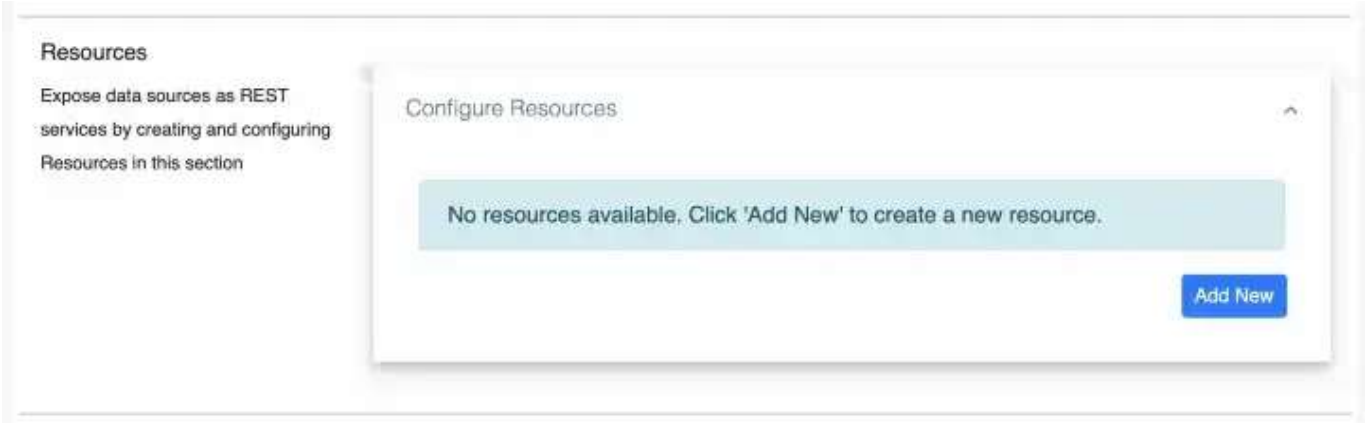
In the above query, we need to specify an input mapping since we need to pass the City name.



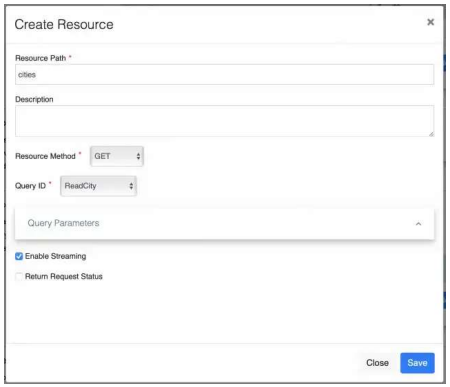
Similarly, create a query for deleting a city too.

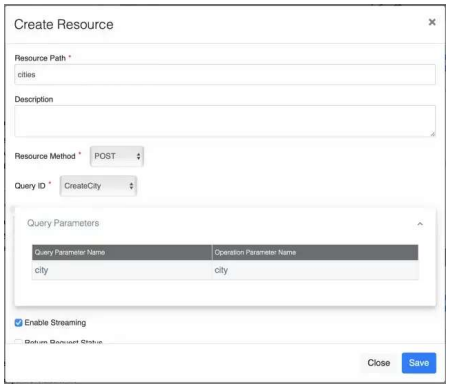

Add Query
Query ID *
DeleteCity
Datasource *
default
SQL Query
DELETE FROM city WHERE id = :id
Input Mappings
Mapping Name
Parameter Type
Type
id
SCALAR
STRING
Generate
Add New
Return Generated Keys
Return Updated Key Count
Result (Output Mappings)

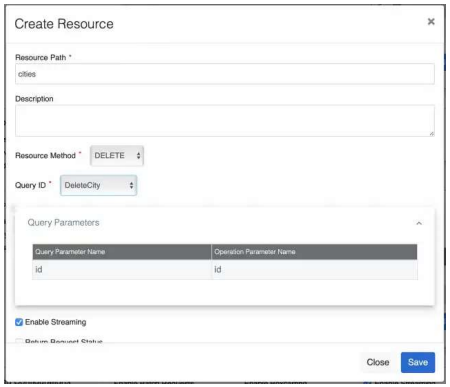
6. Next we need to create resources, which will execute these queries.



Create three resources, which will execute the three queries that we created above.





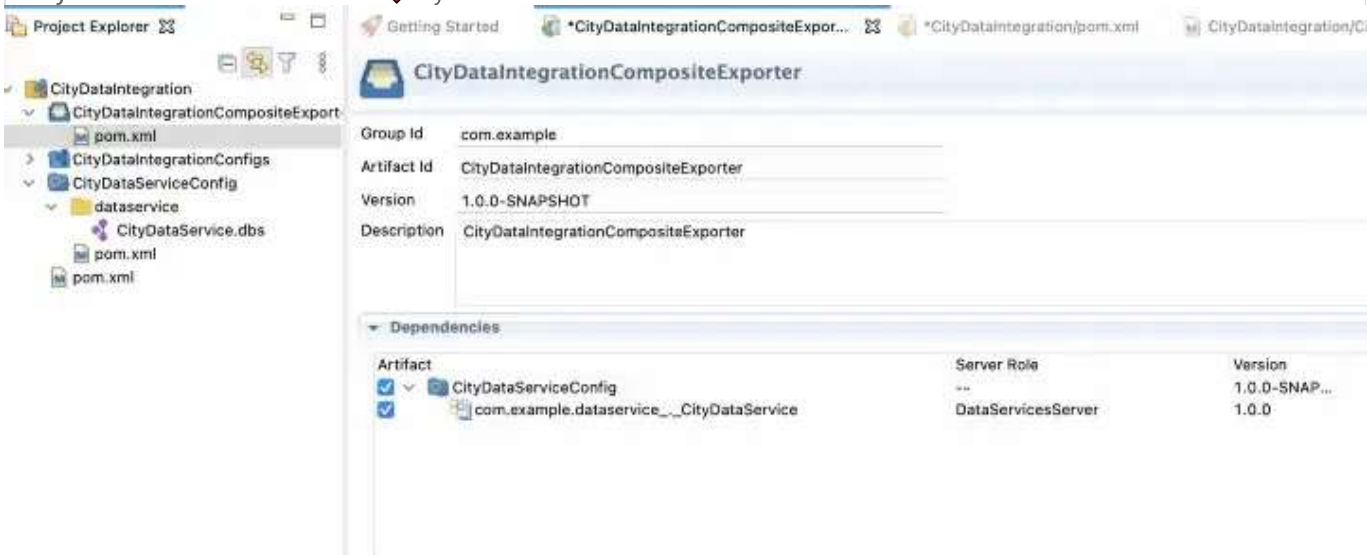


That’s it. You have created a Data Service for Micro Integrator successfully. If you check the source view of the Data Service, it should be like below.

```
<data name="CityDataService" serviceNamespace="" serviceGroup="" transports="http https">
  <description />
  <config id="default">
    <property name="driverClassName">com.mysql.jdbc.Driver</property>
    <property name="url">jdbc:mysql://localhost:3308/city_db</property>
    <property name="username">root</property>
    <property name="password">root</property>
  </config>
  <resource method="GET" path="cities">
    <description />
    <call-query href="ReadCity" />
  </resource>
  <resource method="POST" path="cities">
    <description />
    <call-query href="CreateCity">
      <with-param name="city" query-param="city" />
    </call-query>
  </resource>
  <resource method="DELETE" path="cities">
    <description />
    <call-query href="DeleteCity">
      <with-param name="id" query-param="id" />
    </call-query>
  </resource>
  <query id="ReadCity" useConfig="default">
    <sql>SELECT id, city FROM city</sql>
    <result outputType="json">{"cities":{ "city":[  {  "id":"$id","city":"$city" }  ] }}
</result>
  </query>
  <query id="CreateCity" useConfig="default">
    <sql>INSERT INTO city (city) VALUES (:city)</sql>
    <param name="city" paramType="SCALAR" sqlType="STRING" type="IN" optional="false" />
  </query>
```

```
33     <query id="DeleteCity" useConfig="default">
34       <sql>DELETE FROM city WHERE id = :id</sql>
35       <param name="id" paramType="SCALAR" sqlType="STRING" type="IN" optional="false" />
36     </query>
37   </data>
```

CityDataService.dbs hosted with ❤ by GitHub view raw



Download the mysql-connector-java-8.0.19.jar and add to the Micro Integrator *lib* folder.

Start the Micro Integrator instance and make sure that the Data Service is deployed successfully.

Once started, you can invoke the following curl command and add a city to the database.

```
curl --location --request POST
'http://localhost:8290/services/CityDataService/cities' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data-raw '{
"_postcities": {
"city" : "New York"
}
}'
```

The following curl command will retrieve the list of cities.

```
curl --location --request GET
'http://localhost:8290/services/CityDataService/cities' \
--header 'Accept: application/json'
```

Micro Integrator

Low-code Integration with Integration Studio Integration Studio, a graphical drag-and-drop integration flow designer...

wso2.com

WSO2 Integration Studio

WSO2 Integration Studio is a drag-and-drop graphical development environment for WSO2 Enterprise Integrator. It...

wso2.com

About Data Services - WSO2 API Manager Documentation 4.0.0

The data in your organization can be a complex pool of information that is stored in heterogeneous systems (such as...

apim.docs.wso2.com

The Love story of Integration Studio and Micro Integrator

Starting off with Micro Integrator — What is Integration Studio

medium.com

Life of MI : How an ESB went on to serve Microservices?

Starting off with Micro Integrator — What is WSO2 Micro Integrator

medium.com

Integration Wso 2 Micro Integrator Data Services Rest