**Practical Exercise: Data Integration**

Training Objective

Create a data service that exposes the information in a database, thereby decoupling the client and the database layer in the backend. This introduces students to the data integration

capabilities of WSO2 Micro Integrator.

High Level Steps

* Expose a datasource through a data service.
* Send a request to the data service.

Detailed Instructions

Data Integration

What you'll build

A **data service** provides a web service interface to access data that is stored in various datasources. The following sections describe how you can use WSO2 Integration Studio to work with data services' artifacts.

**Tip**

Note that this feature is currently supported in WSO2 Integration Studio for relational datasources and CSV files.

Let's get started!

Step 1: Set up the workspace

* Download the relevant [WSO2 Integration Studio](https://wso2.com/api-management/tooling/) based on your operating system. The path to the extracted/installed folder is referred to as MI\_TOOLING\_HOME throughout this tutorial.
* To demonstrate how data services work, we will use a MySQL database as the datasource. Follow the steps given below to set up a MySQL database:
  1. Install the MySQL server.
  2. Download the JDBC driver for MySQL from [here](http://dev.mysql.com/downloads/connector/j/). You will need this when you configure the MySQL server with the Micro Integrator.
  3. Create a database named Employees.

CREATE DATABASE Employees;

4. Create a user and grant the user access to the Database.

CREATE USER 'user'@'localhost' IDENTIFIED BY 'password'; GRANT ALL PRIVILEGES ON Employees.\* TO 'user'@'localhost';

* 1. Create the Employee table inside the Employees database:

USE Employees;

CREATE TABLE Employees (EmployeeNumber int(11) NOT NULL, FirstName varchar(255) NOT NULL, LastName varchar(255) DEFAULT NULL, Email varchar(255) DEFAULT NULL, Salary varchar(255));

INSERT INTO Employees (EmployeeNumber, FirstName, LastName, Email, Salary) values (3, "Edgar", "Code", "edgar@rdbms.com", 100000);

Step 2: Creating a data service

Follow the steps given below to create a new data service.

**Creating a Maven Multi Module project**

All the data services' artifacts that you create should be stored in a Data Service Module. Follow the steps given below to create a module:

1. Open **WSO2 Integration Studio** and click **New Maven Multi Module Project** in the **Getting Started** tab as shown below.
2. In the **Maven Modules Creation** dialog box that opens, give a name (artifactId) for the project.
3. If required, change the Maven information about the project.
4. Click **Finish**. The new project will be listed in the project explorer.

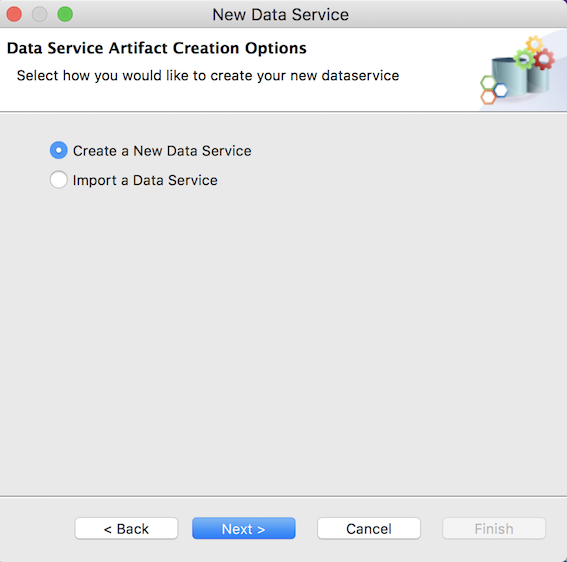
**Creating a data service module**

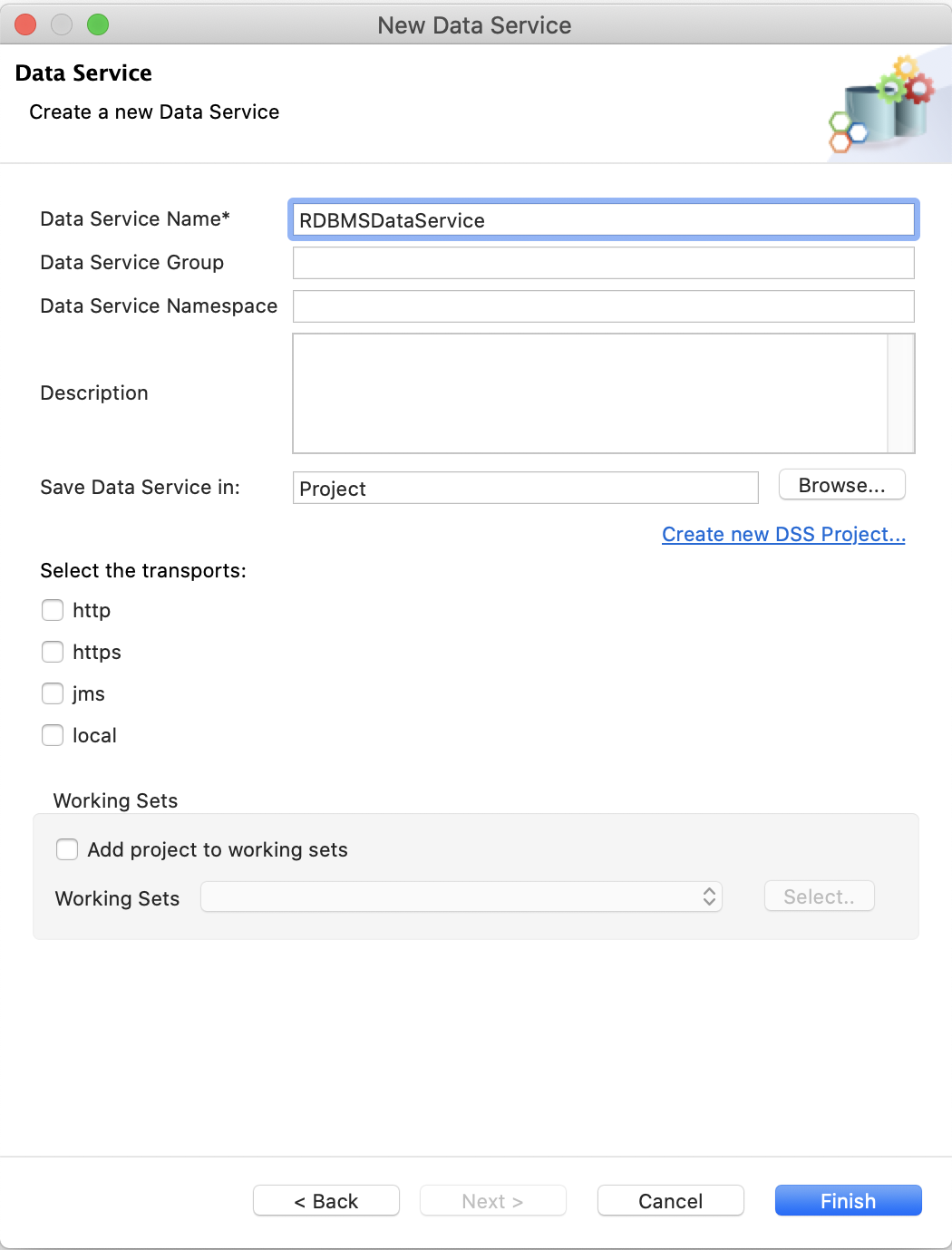
All the data services' artifacts that you create should be stored in a Data Service Module. Follow the steps given below to create a module:

1. Right click on the created **Maven Multi Module Project** and go to **New -> Data Service Configs**.
2. In the **New Data Service Configs** dialog box that opens, give a name for the config module and click **Next**.
3. If required, change the Maven information about the config module.
4. Click **Finish**. The new module will be listed in the project explorer.

**Creating the data service**

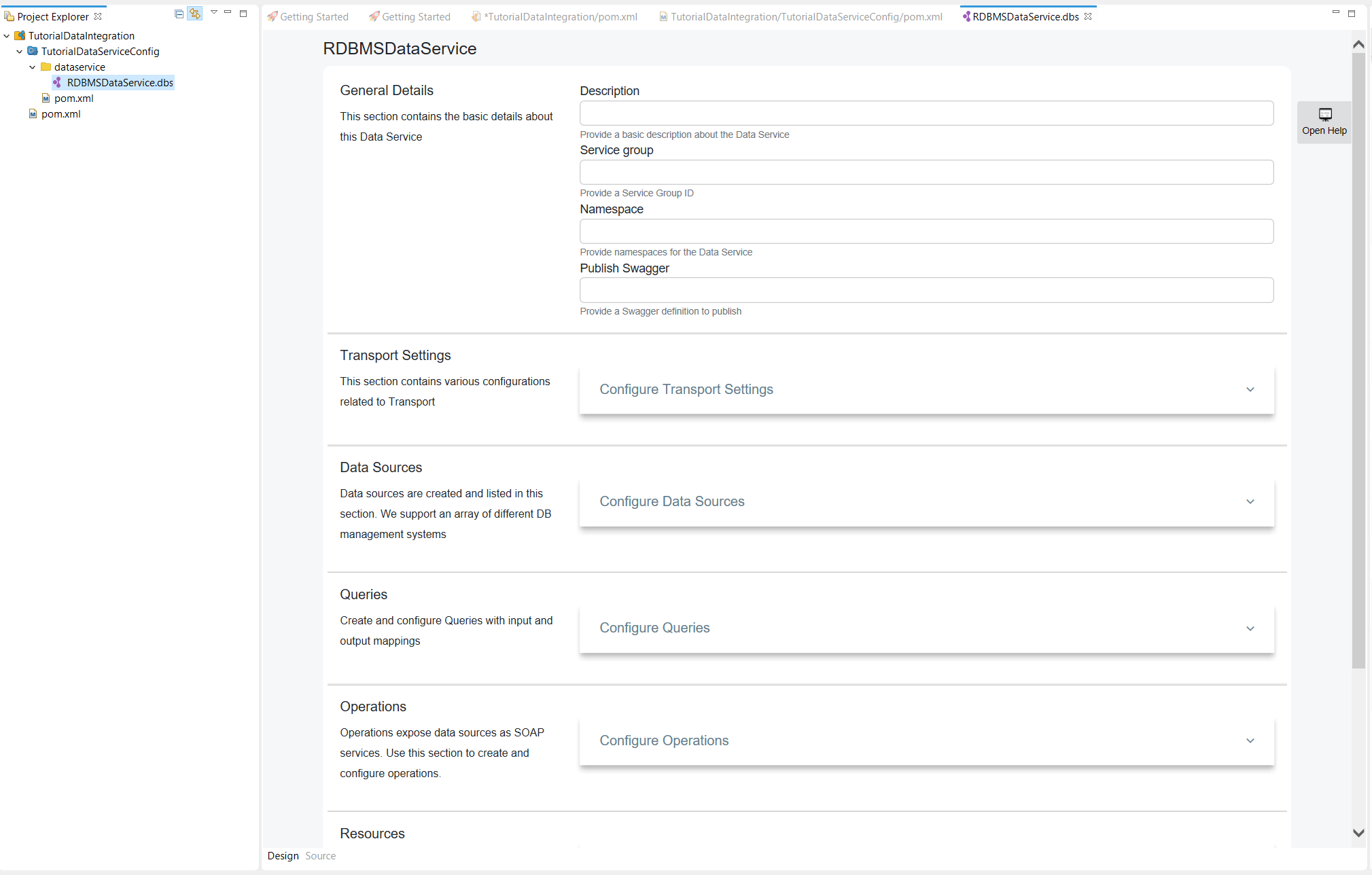
Follow the steps given below to create the data service file:

1. Select the already-created **Data Service Config** module in the project explorer, right-click and go to **New -> Data Service**.  
   The **New Data Service** window will open as shown below.  
   
2. To start creating a data service from scratch, select **Create New Data Service** and click **Next** to go to the next page.
3. Enter a name for the data service and click **Finish**:

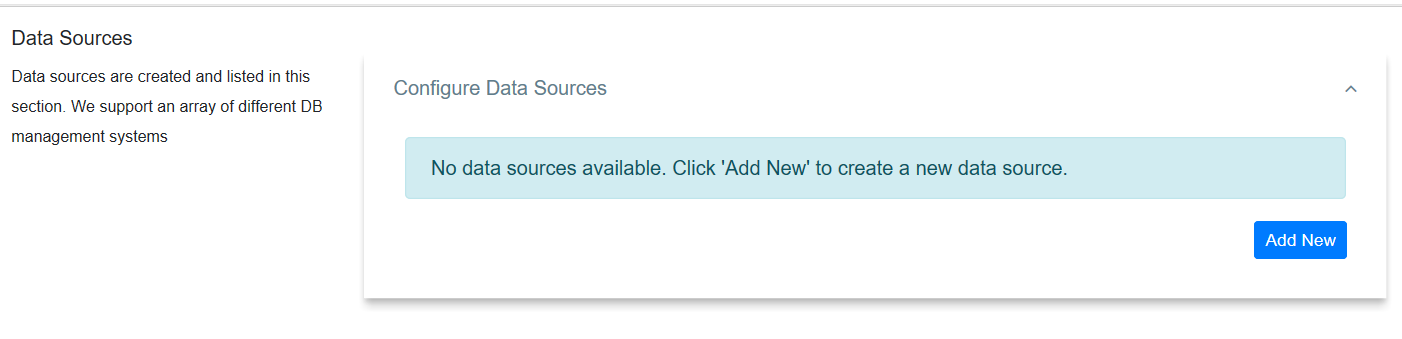
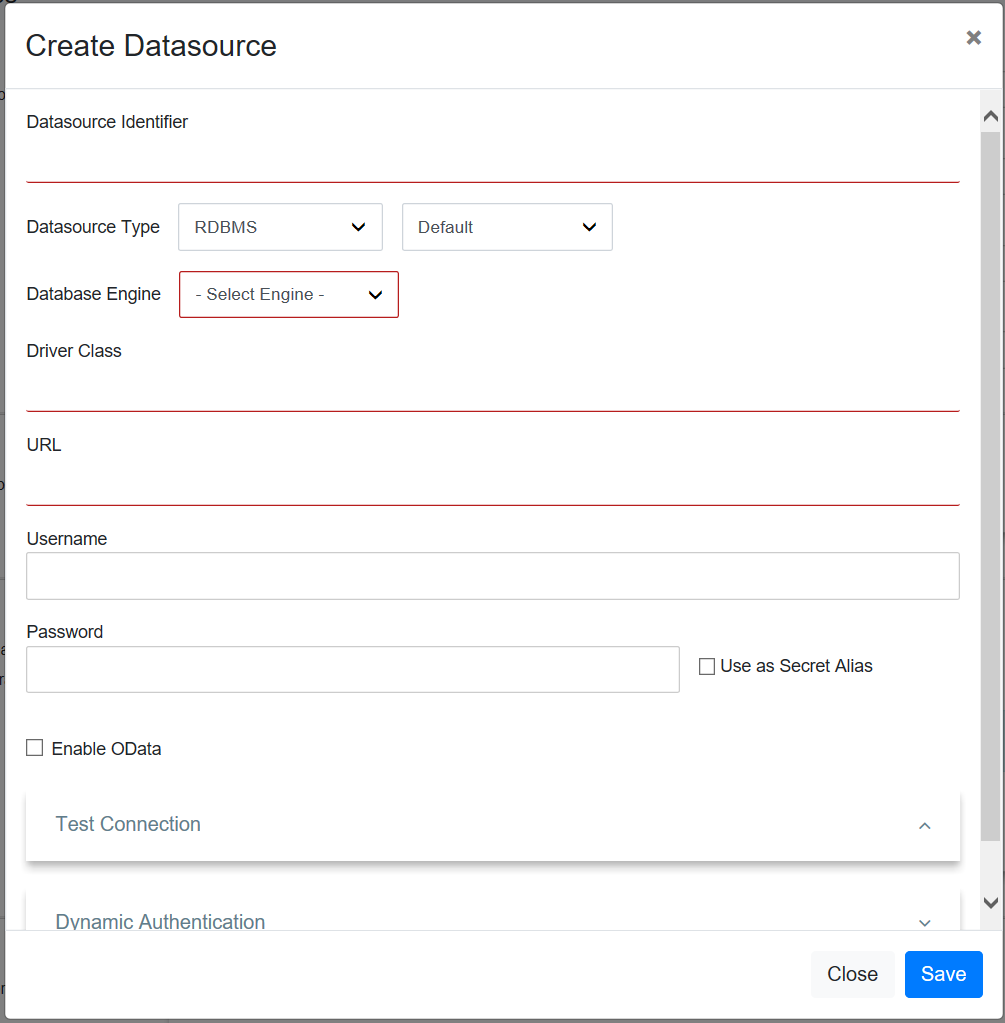


|  |  |
| --- | --- |
| Property | Description |
| Data Service Name | RDBMSDataService |

A data service file (DBS file) will now be created in your data service module as shown below.

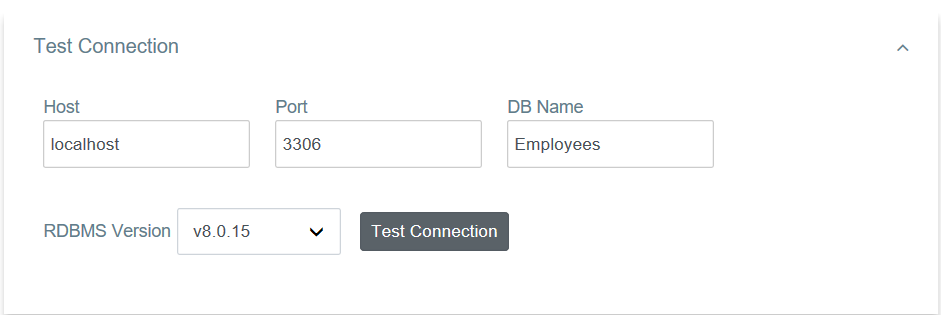


**Creating the datasource connection**

1. Click **Data Sources** to expand the section. 
2. Click **Add New** to open the **Create Datasource** page.
3. 
4. Enter the datasource connection details given below.

| Property | Description |
| --- | --- |
| Datasource ID | Datasource |
| Datasource Type | RDBMS |
| Datasource Type (Default/External) | Leave **Default** selected. |
| Database Engine | MySQL |
| Driver Class | com.mysql.jdbc.Driver |
| URL | jdbc:mysql://localhost:3306/Employees |
| User Name | user |
| Password | password |

1. Click **Test Connection** to expand the section.

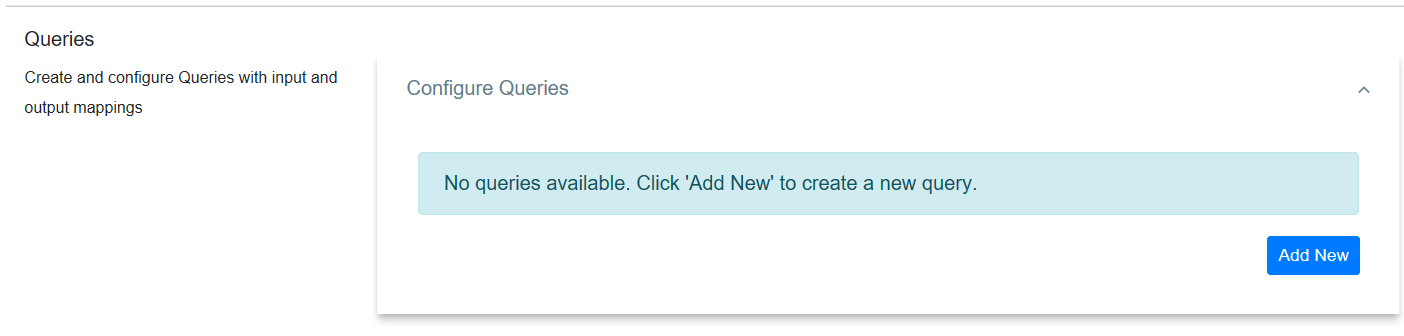


1. Click the **Test Connection** button to verify the connectivity between the MySQL datasource and the data service.
2. Save the data service.

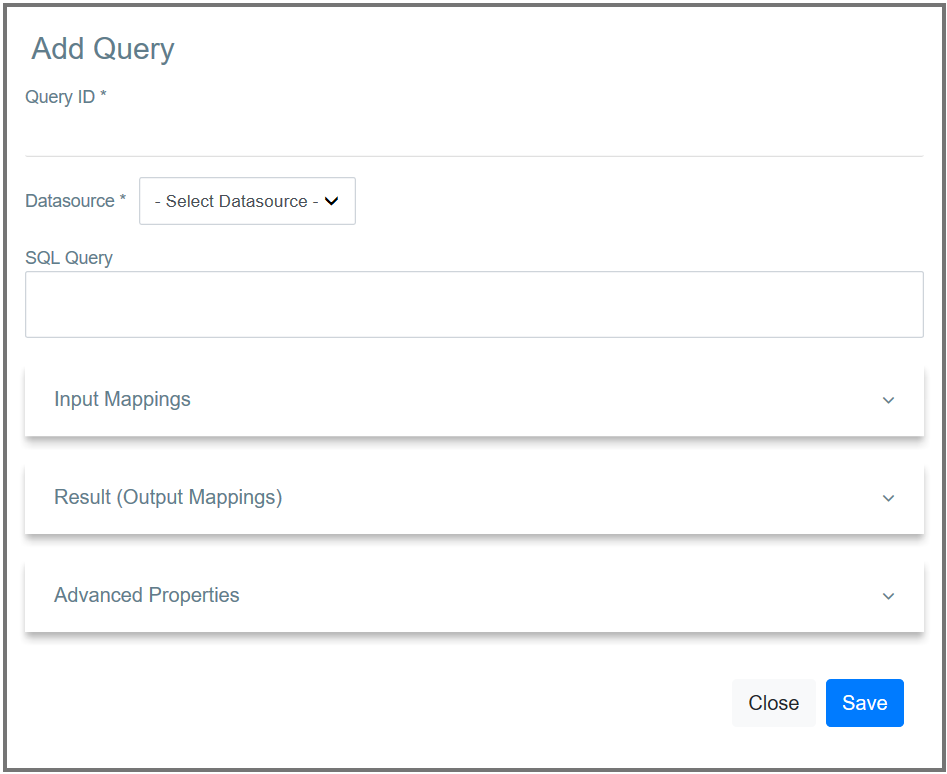
**Creating a query**

Let's write an SQL query to GET data from the MySQL datasource that you configured in the previous step:

1. Click **Queries** to expand the section.



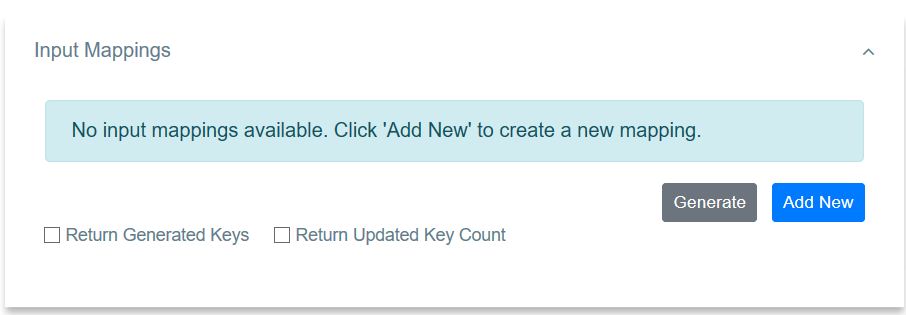
1. Click **Add New** to open the **Add Query** page.



1. Enter the following query details:

| Parameter | Description |
| --- | --- |
| Query ID | GetEmployeeDetails |
| Datasource | Datasource |
| SQL Query | select EmployeeNumber, FirstName, LastName, Email from Employees where EmployeeNumber=:EmployeeNumber |

1. Click **Input Mappings** to expand the section.



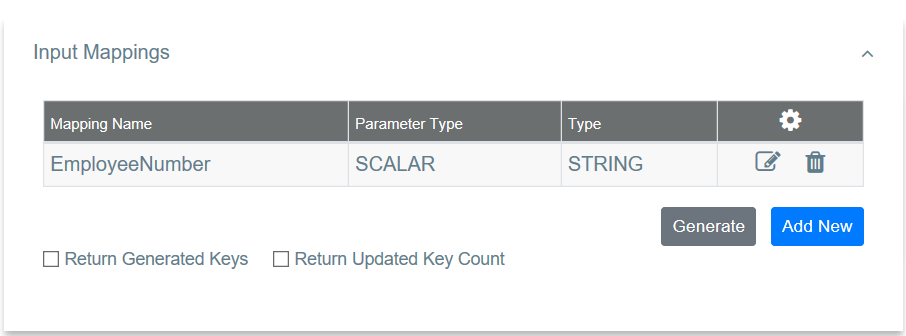
1. Click **Generate** to generate input mappings automatically.

**Tip**

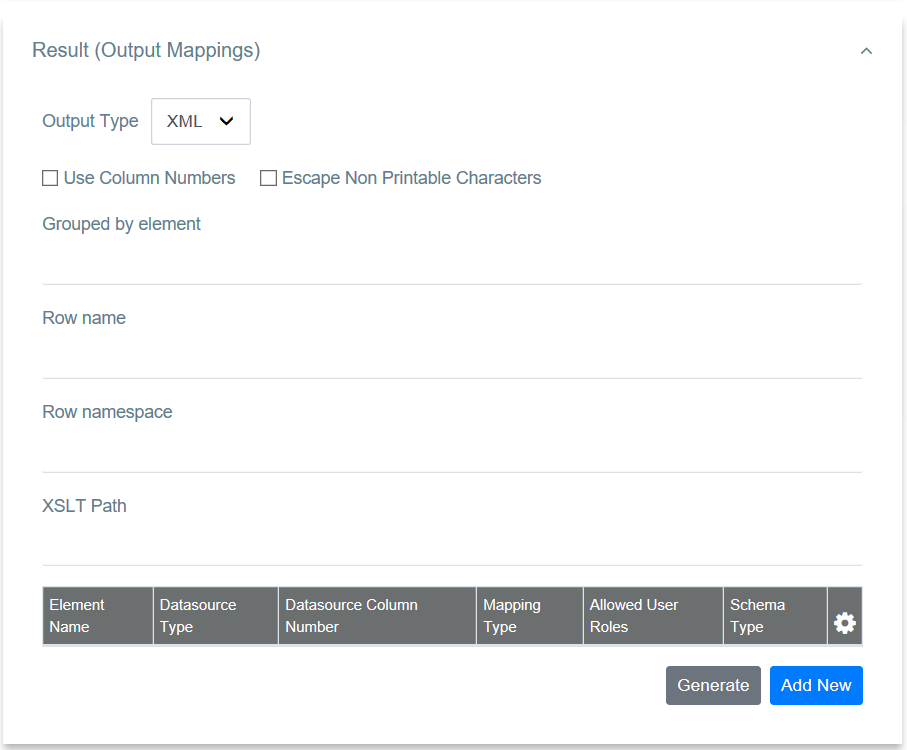
Alternatively, you can manually add the mappings: 1. Click **Add New** to open the **Add Input Mapping** page. 2. Enter the following input element details.

|  |  |
| --- | --- |
| Property | Description |
| Mapping Name | EmployeeNumber |
| Parameter Type | SCALAR |
| SQL Type | STRING |

1. Save the input mapping.



1. Click **Result (Output Mappings)** to expand the section.



1. Enter the following value to group the output mapping:

|  |  |
| --- | --- |
| Property | Description |
| Grouped by Element | Employees |

1. Click **Generate** to generate output mappings automatically.

**Tip**

Alternatively, you can manually add the mappings: 1. Click **Add New** to open the **Add Output Mapping** page. 2. Enter the following output element details.

|  |  |
| --- | --- |
| Property | Description |
| Datasource Type | column |
| Output Field Name | EmployeeNumber |
| Datasource Column Name | EmployeeNumber |
| Schema Type | String |

3. Save the element. 4. Follow the same steps to create the following output elements:

| Datasource Type | Output Field Name | Datasource Column Name | Schema Type |

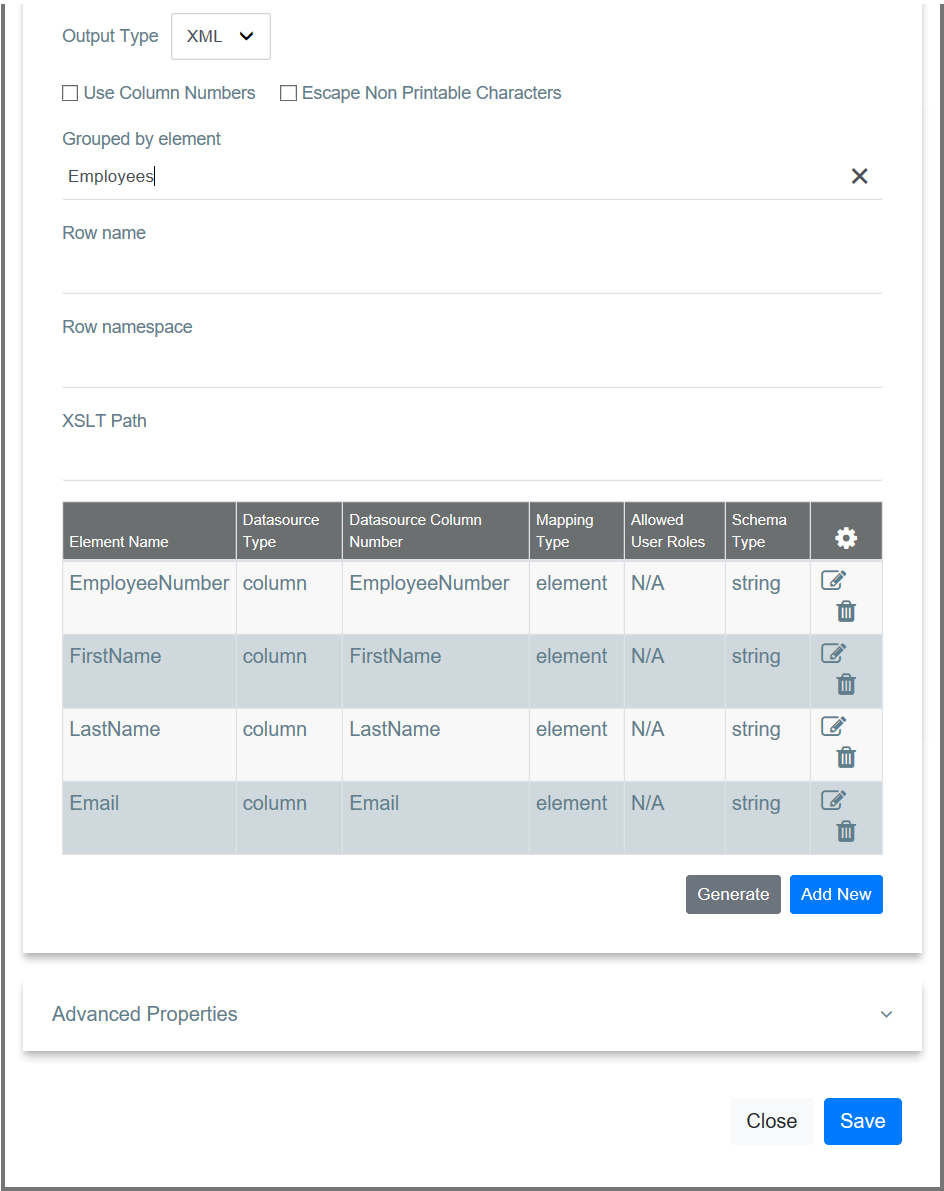
|-----------------|-------------------|------------------------|-------------|

| column | FirstName | FirstName | string |

| column | LastName | LastName | string |

| column | Email | Email | string |

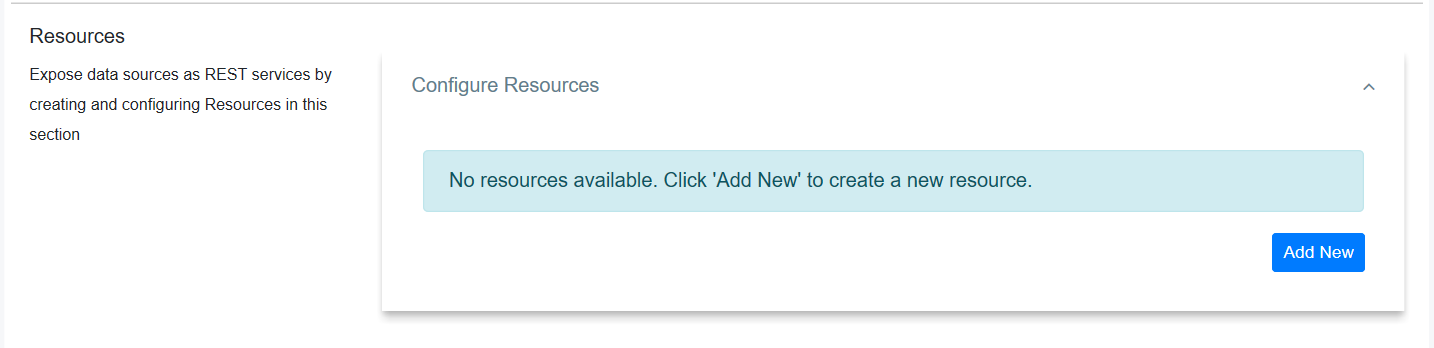
1. Click **Save** to save the query.



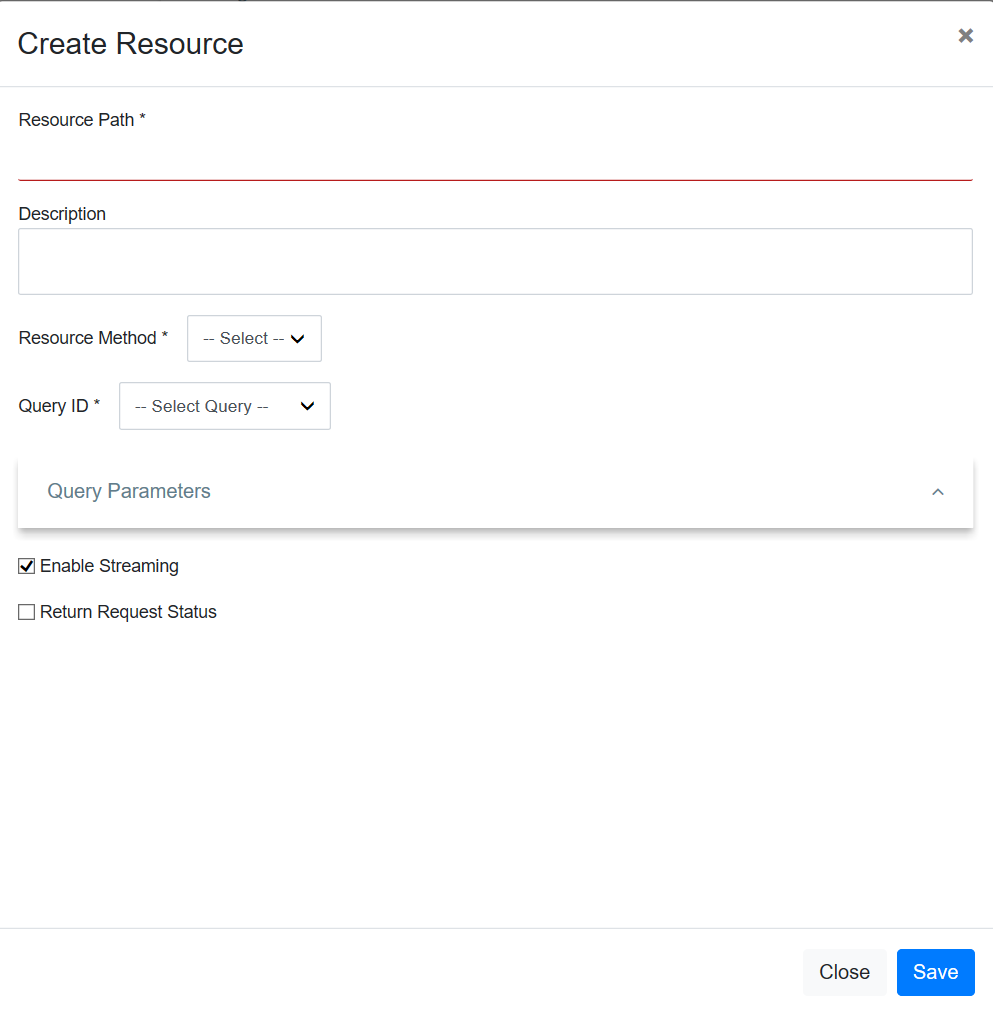
**Creating a resource to invoke the query**

Now, let's create a REST resource that can be used to invoke the query.

1. Click **Resources** to expand the section.



1. Click **Add New** to open the **Create Resource** page.



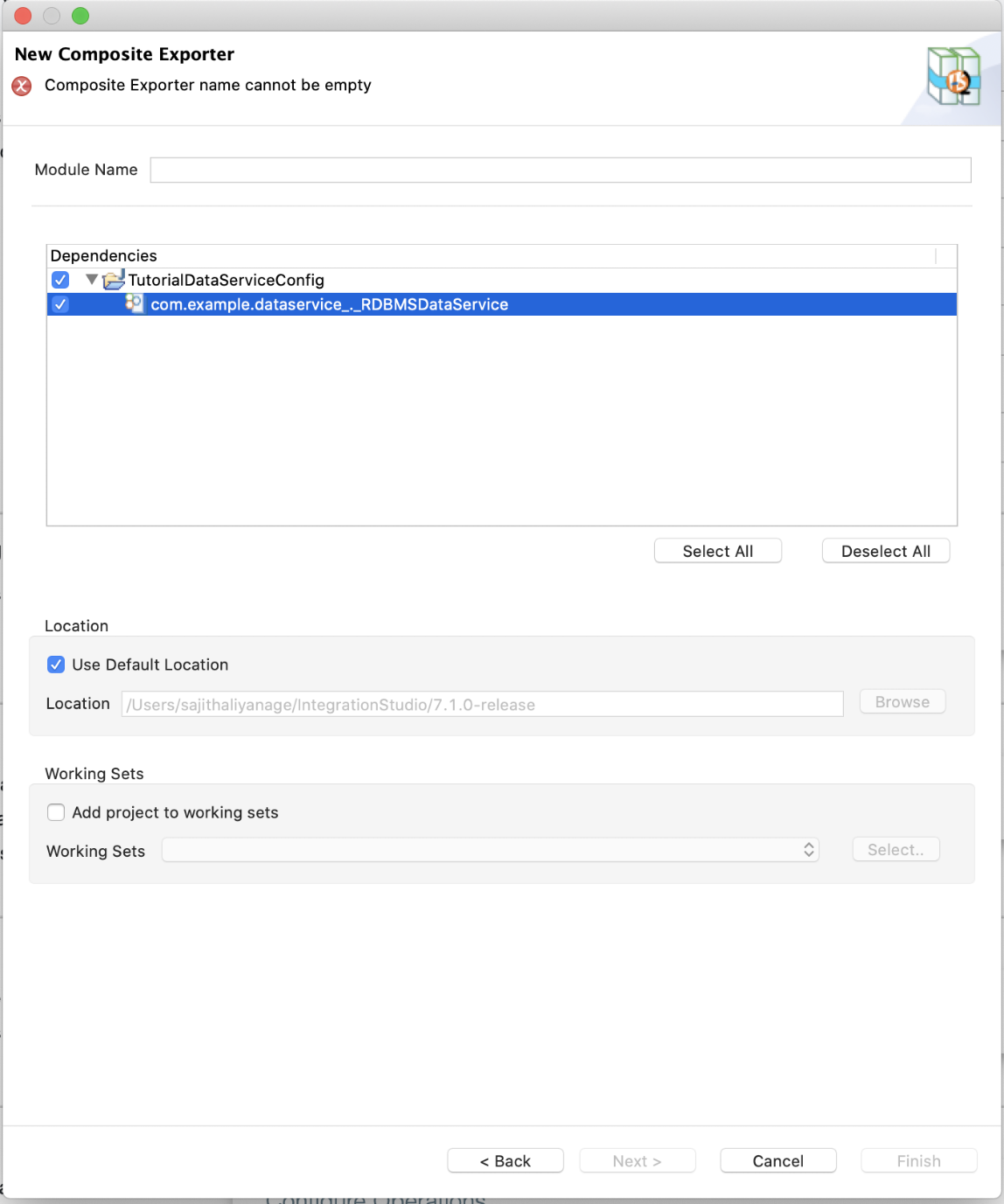
1. Enter the following resource details.

|  |  |
| --- | --- |
| Property | Description |
| Resource Path | Employee/{EmployeeNumber} |
| Resource Method | GET |
| Query ID | GetEmployeeDetails |

1. Save the resource.

Step 3: Package the artifacts

Create a new composite exporter module

1. Right-click the **Maven Multi Module Project** go to **New -> Composite Exporter**.
2. In the dialog box that opens, select the data service file, and click **Finish**.  
   

Package the artifacts in your composite exporter to be able to deploy the artifacts in the server.

1. Open the pom.xml file in the composite application.
2. Ensure that your data service file is selected in the POM file.
3. Save the file.

Step 4: Configure the Micro Integrator server

We will use the embedded Micro Integrator of WSO2 Integration Studio to run this solution.

To add the MySQL database driver to the server:

1. Click the **Embedded Micro Integrator Configuration ()** icon on the upper menu to open the dialog box.
2. Click the (https://apim.docs.wso2.com/en/4.0.0/assets/img/integrate/tutorials/common/plus-icon.png) icon to add the MySQL driver JAR (see [Setting up the Workspace](https://apim.docs.wso2.com/en/4.0.0/tutorials/integration-tutorials/sending-a-simple-message-to-a-datasource/#step-1-set-up-the-workspace)) to the /lib directory of the embedded Micro Integrator.

If the driver class does not exist in the relevant directory, you will get an exception such as Cannot load JDBC driver class com.mysql.jdbc.Driver when the Micro Integrator starts.

Step 5: Build and run the artifacts

To test the artifacts, deploy the [packaged artifacts](https://apim.docs.wso2.com/en/4.0.0/tutorials/integration-tutorials/sending-a-simple-message-to-a-datasource/#step-3-package-the-artifacts) in the embedded Micro Integrator:

1. Right-click the composite exporter module and click **Export Project Artifacts and Run**.
2. In the dialog box that opens, confirm that the required artifacts from the composite exporter module are selected.
3. Click **Finish**.

The artifacts will be deployed in the embedded Micro Integrator and the server will start.

* See the startup log in the **Console** tab.
* See the URLs of the deployed services and APIs in the **Runtime Services** tab.

Step 6: Testing the data service

Let's test the use case by sending a simple client request that invokes the service.

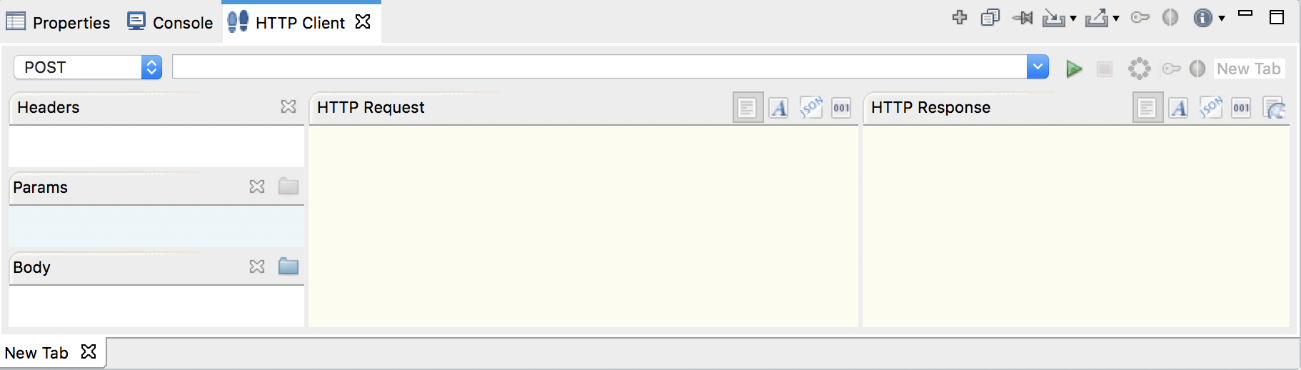
**Send the client request**

Let's send a request to the API resource to make a reservation. You can use the embedded **HTTP Client** of WSO2 Integration Studio as follows:

1. Open the **HTTP Client** of WSO2 Integration Studio.

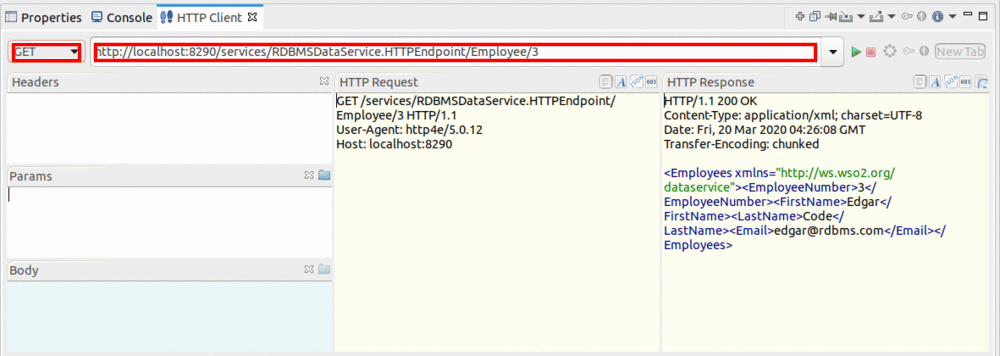
**Tip**

If you don't see the **HTTP Client** pane, go to **Window -> Show View - Other** and select **HTTP Client** to enable the client pane.



1. Enter the request information as given below and click the **Send** icon ().

|  |  |
| --- | --- |
| Method | GET |
| URL | http://localhost:8290/services/RDBMSDataService.HTTPEndpoint/Employee/3 |



If you want to send the client request from your terminal:

1. Install and set up [cURL](https://curl.haxx.se/" \t "_blank) as your REST client.
2. Execute the following command.

curl -X GET http://localhost:8290/services/RDBMSDataService.HTTPEndpoint/Employee/3

**Analyze the response**

You will see the following response received to your **HTTP Client**:

<**Employees** xmlns="http://ws.wso2.org/dataservice">

<**EmployeeNumber**>3</**EmployeeNumber**>

<**FirstName**>Edgar</**FirstName**>

<**LastName**>Code</**LastName**>

<**Email**>edgar@rdbms.com</**Email**>

</**Employees**>