
Article

[Luis Angel Pérez...](#) · Feb 20, 2023 4m read

[Open Exchange](#)

Connecting to MySQL database from a Business Service using JDBC connections

Dear community members!

A very common problem of our users is to use an external database as data source in an IRIS production. As many of you already know, we have two ways to connect directly to an external database, the first one is using an ODBC connection, the second is using JDBC.

In our example we are going to create a connection using JDBC, and we are going to build a simple Docker's project, in this way you will be able to modify the example as you wish.

The code is available from this URL: <https://github.com/intersystems-ib/workshop-sql-jgw>

In our docker-compose.yml file we are going to configure the images that we use:

```
version: "2.2"
services:
  # mysql
  mysql:
    build:
      context: mysql
    container_name: mysql
    restart: always
    command: --default-authentication-plugin=mysql_native_password
    environment:
      MYSQL_ROOT_PASSWORD: SYS
      MYSQL_USER: testuser
      MYSQL_PASSWORD: testpassword
    volumes:
      - ./mysql/sql/dump.sql:/docker-entrypoint-initdb.d/dump.sql

  adminer:
    container_name: adminer
    image: adminer
    restart: always
    depends_on:
      - mysql
    ports:
      - 8080:8080
  # java gateway
  jgw:
    build:
      context: java
      dockerfile: Dockerfile
    depends_on:
      - mysql
    container_name: jgw
    restart: always
```

```
ports:
  - 44444:44444
environment:
  - PORT=44444
# iris
iris:
  init: true
  container_name: iris
  build:
    context: .
    dockerfile: iris/Dockerfile
  depends_on:
    - 'jgw'
  ports:
    - 52773:52773
    - 51773:51773
  command: --check-caps false
```

- MySQL: our database engine.
- Adminer: web application to administrate our MySQL instance.
- IRIS: with the latest community version of IRIS.
- JGW: Java Gateway to allow us to connect by JDBC to our database.

To deploy the containers we should run these commands from the terminal:

```
docker-compose build
...
docker-compose up -d
...
docker-compose start
```

Once the containers have been started you will be able to access to the database administrator from this [url](#) using the next parameters:

- System: MySQL
- Server: mysql
- User: root
- Date: SYS

You will find a new database configured named test if you access it you will find a table named patient with two rows. We are going to use this table to launch our query from the Business Service.

Language: English

Adminer 4.8.1

DB: test

SQL command [Import](#)
[Export](#) [Create table](#)

[select patient](#)

MySQL » mysql » test » Table: patient

Table: patient

[Select data](#) **Show structure** [Alter table](#) [New item](#)

Column	Type	Comment
Id	int <i>NULL</i>	
Name	varchar(225) <i>NULL</i>	
Lastname	varchar(225) <i>NULL</i>	

Indexes

[Alter indexes](#)

Foreign keys

[Add foreign key](#)

Triggers

[Add trigger](#)

Now, we can access to the Management Portal of our IRIS instance in order to check the production located in USER Namespace, Test.Production. This production is running by default, so you don't need to start it:

InterSystems
IRIS Data Platform

Management Portal

Home About Help Logout

Menu

Server 97f9cb8df140 Namespace USER [Switch](#) User SuperUser Licensed To InterSystems IRIS Community Instance IRIS

Interoperability > Production Configuration - (Test.Production)

Production Configuration Start Stop Sort: Name Status Number View: [List] [Grid] [Table]

Production Running

Category: All Legend Production Settings

Services

Processes

Operations

EnsLib.JavaGateway.Service

Test.PatientToString

EnsLib.File.PassthroughOperation

EnsLib.SQL.Service.GenericService

Production Settings

Settings Queue Log Messages Jobs Actions

Apply Search:

Informational Settings

Basic Settings

Actor Pool Size

2

Additional Settings

Alerting Control

Development and Debugging

These are the Production settings.
To view item settings, click on a configuration item.

Let's check the different elements that we can find on it:

- EnsLib.JavaGateway.Service: this Business Service is the responsible of the JDBC connection to our database, in this example it's necessary to deploy and specific container por the Java Gateway in which is located our JDK. The Java Gateway is listening in 44444 port. If you want to configure the Java Gateway in an IRIS instance in Windows or Linux you just need to add this type of Business Service to the production configure the path to the JAVA HOME and the path to the specific library (JAR) to connect with the database, the default port will be 55555, but you can change it.
- EnsLib.SQL.Service.GenericService: we should configure the following parameters.

- DSN: connection url to the database (jdbc:mysql://mysql:3306/test).
- Credentials: user and password to connect to our database (root/SYS).
- Target config names: the target component of the business service (a Business Process or a business Operation).
- Query: the query that we are going to execute in a loop in order to get the latest values recorded in our table.
- Message Class: the type of the object that we are going to use to save the result of the query (the properties of the object must match with the name of the fields used in the previous query).
- Java Gateway Service: the name of the JavaGateway service in our production.
- JDBC Driver: the name of the JDBC driver used to connect to our database (in this case: com.mysql.jdbc.Driver).
- Test.PatientToString: an example of a Business Process in which we get the properties of the object defined in the Business Service and are parsed to be sent to a Business Operation.
- EnsLib.File.PassthroughOperation: an standard Business Operation to write the properties of the patient in a file.

If we review the properties of the object that we are using to store the result of the query we can confirm that the name of the parameters match with the name of the fields used in the query:

```
Class Test.Patient Extends (%Persistent, %JSON.Adaptor, %XML.Adaptor, Ens.Request)
{

Property Id As %Integer;
Property Name As %String;
Property Lastname As %String;
}
```

This object will allow to us to map automatically our query in a object, no more transformations are required!.

If you have any question or suggestion don't hesitate to write a comment!

[#Docker](#) [#JDBC](#) [#InterSystems IRIS](#) [#InterSystems IRIS for Health](#) [#VSCode](#)
[Check the related application on InterSystems Open Exchange](#)

Source

URL:<https://community.intersystems.com/post/connecting-mysql-database-business-service-using-jdbc-connections>