InterSystems IRIS Demo: JDBC and InterSystems Databases

Version 2024.2
2024-09-05

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

InterSystems IRIS Demo: JDBC and InterSystems Databases
PDF generated on 2024-09-05
InterSystems IRIS® Version 2024.2

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Response Center (WRC)
Tel: +1-617-621-0700
Tel: +44 (0) 844 854 2917
Email: support@InterSystems.com

Table of Contents

InterSystems IRIS Demo: JDBC and InterSystems Databases

This article provides an introduction to how to use the InterSystems JDBC driver to connect to an InterSystems IRIS® data platform instance so you can use Java with InterSystems IRIS.

1 JDBC: How to Use It with InterSystems IRIS

InterSystems provides a fully compliant (JDBC 4.2), pure Java, type 4 JDBC driver, which is a single standalone JAR file with no dependencies. If you are already familiar with JDBC, and have a JDK 1.8 installed, all you need to do is to add the JDBC driver to your local *CLASSPATH* (see JDBC: Exploring It). The JDBC URL (connection string) is:

```
jdbc:IRIS://ipAddress:superserverPort/namespace
```

where the variables represent the InterSystems IRIS instance host's IP address, the instance's superserver port, and a namespace on the instance.

If you are connecting to an instance on the local machine (either using a hostname of `localhost` or an IP address of `127.0.0.1`), the connection can use a special, high-performance local connection called a *shared memory connection*. For more information about shared memory connections, see "JDBC: What's Unique about Shared Memory Connections".

The point of this document is to give you a taste of using JDBC with InterSystems IRIS without bogging you down in details, so we've kept this exploration simple. When you bring InterSystems IRIS to your production systems, though, there are many things you will need to do differently, such as in regard to (but not limited to) security. So be sure not to confuse this exploration of InterSystems IRIS with the real thing! The sources provided at the end of this document will give you a good idea of what's involved in using JDBC with InterSystems IRIS in production.

2 JDBC: Part of InterSystems IRIS Java Connectivity Options

The InterSystems IRIS JDBC driver is the core InterSystems IRIS Java component, and supports traditional relational (SQL) access. It also provides the connection mechanism for Java calls that use the InterSystems IRIS Native SDK for Java, which accesses the data in its natively stored format. For Java integration based on objects, InterSystems IRIS also provides a separate feature — the InterSystems IRIS XEP component.

Taken all together, InterSystems IRIS provides a unique set of capabilities to use the same physical connection and transaction context to manipulate data using multiple paradigms: native, relational, and object-oriented. For more complex applications, InterSystems fully supports Hibernate. Enabling all these forms of connectivity is the InterSystems IRIS JDBC driver.

2.1 JDBC: What's Unique about Shared Memory Connections

As with other database platforms, a JDBC connection to a remote InterSystems IRIS instance is over TCP/IP. To maximize performance, InterSystems IRIS also offers a Java *shared memory connection*. Shared memory connections are available to many Java applications running on the same machine as an InterSystems IRIS instance.

A shared memory connection is a temporary device, backed virtual memory, which is shared by a JDBC client and an instance of InterSystems IRIS running on the same physical machine. Further, these connections do not require potentially expensive calls into the kernel network stack. By using a channel directly from the JDBC client to InterSystems IRIS, they provide the ultimate in low latency and high throughput for JDBC operations.

For detailed information on shared memory, see "Shared Memory Connections" in Using Java with the InterSystems JDBC Driver.

3 JDBC: Exploring It

We've developed a demo that shows you how to work with JDBC and InterSystems IRIS — and how straightforward that is.

Please note that this code does not demonstrate the improved performance power of the InterSystems Java shared memory connection, because it does not deal with the large volumes of data that the shared memory connection can handle so efficiently.

3.1 Before You Begin

To use the procedure, you will need a system to work on, with version 1.8 of the JDK and a Java IDE of your choice installed, and a running InterSystems IRIS instance to connect to. Your choices for InterSystems IRIS include several types of licensed and free evaluation instances; the instance need not be hosted by the system you are working on (although they must have network access to each other). For information on how to deploy each type of instance if you do not already have one to work with, see Deploying InterSystems IRIS in *InterSystems IRIS Basics: Connecting an IDE*. Connect your IDE to your InterSystems IRIS instance using the information in InterSystems IRIS Connection Information and Java IDEs in the same document. For this demo, you can connect to the **USER** namespace, as shown in the code that follows, or you can specify another one that you have created in your installed instance.

You will also need to add the InterSystems IRIS JDBC driver, intersystems-jdbc-*version*.jar, to your local *CLASSPATH*. You can download this file from the InterSystems IRIS Driver Packages page. If you have installed InterSystems IRIS on your local machine or another you have access to, you can find the file in *install-dir*\dev\java\lib\JDK18, where *install-dir* is the InterSystems IRIS installation directory.

3.2 Try the Sample Code

Cut and paste the sample code into your IDE, updating the *url* and *connection* variables and the username and password with the connection settings described for your instance in *InterSystems IRIS Basics: Connecting an IDE*.

Java

```
import java.sql.*;

public class JDBCSample {
    public static void main(String[] str) throws Exception {
        String url = "jdbc:IRIS://127.0.0.1:1972/USER";

        Class.forName("com.intersystems.jdbc.IRISDriver");
        Connection connection = DriverManager.getConnection(url,"_SYSTEM","SYS");
```

```
        // Replace _SYSTEM and SYS with a username and password on your system

       String createTable = "CREATE TABLE People(ID int, FirstName varchar(255), LastName varchar(255))";

        String insert1 = "INSERT INTO People VALUES (1, 'John', 'Smith')";
        String insert2 = "INSERT INTO People VALUES (2, 'Jane', 'Doe')";
        String query = "SELECT * FROM People";

        Statement statement = connection.createStatement();
        statement.executeUpdate(createTable);
        statement.executeUpdate(insert1);
        statement.executeUpdate(insert2);
        ResultSet resultSet = statement.executeQuery(query);
        System.out.println("Printing out contents of SELECT query: ");
        while (resultSet.next()) {
            System.out.println(resultSet.getString(1) + ", " + resultSet.getString(2) + ", " +
resultSet.getString(3));
        }
        resultSet.close();
        statement.close();
        connection.close();
    }
}
```

If the connection and queries have completed successfully, you should see a console window containing the results of the SELECT query.

4 Learn More about JDBC

To learn more about JDBC, other Java interoperability technologies in InterSystems IRIS, and other related topics, see:

- "InterSystems Java Connectivity Options" in *Using Java JDBC with InterSystems IRIS* — overview of all InterSystems IRIS Java technologies enabled by the JDBC driver.

  Using Java with the InterSystems JDBC Driver — InterSystems documentation: step-by-step instructions for using JDBC.

- Java Overview — InterSystems online learning: introductory video
- InterSystems IRIS Demo: Java Object Persistence with XEP — InterSystems documentation: Java XEP demo
- Persisting Java Objects with InterSystems XEP — InterSystems documentation: step-by-step instructions for using XEP
- Using the InterSystems Native SDK for Java — InterSystems documentation: InterSystems Native SDK
- InterSystems Implementation Reference for Java Third Party APIs — InterSystems documentation: connecting to InterSystems IRIS using JDBC and Hibernate
- Hibernate and JDBC compared — Stack Overflow article