First Look: JDBC and InterSystems Databases

Version 2018.1
2018-10-22

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

First Look: JDBC and InterSystems Databases
InterSystems IRIS Data Platform Version 2018.1 2018-10-22

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

First Look: JDBC and InterSystems Databases

If you want to use Java with InterSystems IRIS™, this document provides an introduction to how to use JDBC.

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 1.7 or 1.8 installed, all you need to do is to add intersystems-jdbc-3.0.0.jar to your *CLASSPATH*. The driver name is com.intersystems.jdbc.IRISDriver, and it's typically installed under

<*install-dir*>`\dev\java\lib\`*JDK1x*

where *JDK1x* is either `JDK17` or `JDK18`.

The URL (connection string) is:

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

Plus, if you are connecting to a local Windows machine (either using a hostname of `localhost` or an IP address of `127.0.0.1`), then the connection automatically uses 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 API, 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 — InterSystems IRIS XEP, Hibernate, and also the InterSystems IRIS Spark Connector — 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 Java applications running on the same Windows 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.

If shared memory connections are available, InterSystems IRIS uses them by default — and you can easily disable them. For now, shared memory connections are available for JDBC only on Microsoft Windows. In future releases of InterSystems IRIS, you will be able to use shared memory connections on other platforms, including Linux.

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.

For instructions on how to install and license a development instance of InterSystems IRIS, see Quick Start: InterSystems IRIS Installation.

Here is the sample code:

```
import java.sql.*;

public class JDBCSample {
    public static void main(String[] str) throws Exception {
        String url = "jdbc:IRIS://127.0.0.1:51773/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.

A Note on the Sample

The connection string syntax is:

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

where:

- *ipAddress* — The IP address of the InterSystems IRIS instance. If you are connecting locally, use 127.0.0.1 instead of localhost.
- *superserverPort* — The superserver port number for the IRIS instance, which is not the same as the webserver port number. To find the superserver port number, in the Management Portal, go to **System Administration** > **Configuration** > **System Configuration** > **Memory and Startup**.
- *namespace* — An existing namespace in the InterSystems IRIS instance. In this demo, we connect to the **USER** namespace.

4 Learn More about JDBC

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

- First Look: XEP Object Persistence with InterSystems IRIS — InterSystems documentation: Java XEP First Look
- First Look: InterSystems IRIS Native API — InterSystems documentation: InterSystems IRIS Native API First Look
- "InterSystems Java Connectivity Options" in *Using Java JDBC with InterSystems IRIS* — overview of all InterSystems IRIS Java technologies enabled by the JDBC driver.
- Java Overview — InterSystems online learning: introductory video
- Using Java JDBC with InterSystems IRIS — InterSystems documentation: step-by-step instructions for using JDBC
- Persisting Java Objects with InterSystems IRIS XEP — InterSystems documentation: step-by-step instructions for using XEP
- InterSystems IRIS Implementation Reference for Java Third Party APIs — InterSystems documentation: connecting to InterSystems IRIS using JDBC, Hibernate, and Spark.
- Using the InterSystems Spark Connector — InterSystems documentation: using InterSystems IRIS as an Apache data source
- Hibernate and JDBC compared — Stack Overflow article