MySQL Connector/MXJ Developer Guide

MySQL Connector/MXJ Developer Guide

Abstract

This manual describes MySQL Connector/MXJ.

For release notes detailing the changes in each release of Connector/MXJ, see Chapter 9, *MySQL Connector/MXJ Release Notes*.

Support EOL for MySQL Connector/MXJ. Due to very low demand, MySQL has stopped development and support for Connector/MXJ. Source and binaries for previously released versions will continue to be available from archives.

Document generated on: 2014-01-21 (revision: 37458)

Table of Contents

Preface and Legal Notices v
1 Overview of Connector/MXJ 1
2 Connector/MXJ Versions 3
3 Connector/MXJ Installation 5
    3.1 Supported Platforms 5
    3.2 Connector/MXJ Base Installation 6
    3.3 Connector/MXJ Quick Start Guide 7
    3.4 Deploying Connector/MXJ Using Driver Launch 8
    3.5 Deploying Connector/MXJ within JBoss 8
    3.6 Verifying Installation Using JUnit 8
        3.6.1 JUnit Test Requirements 9
        3.6.2 Running the JUnit Tests 9
4 Connector/MXJ Configuration 11
    4.1 Running as part of the JDBC Driver 11
    4.2 Running within a Java Object 12
    4.3 Setting Server Options 13
5 Connector/MXJ Reference 15
    5.1 MysqldResource Constructors 15
    5.2 MysqldResource Methods 15
6 Connector/MXJ Notes and Tips 17
    6.1 Creating your own Connector/MXJ Package 17
    6.2 Deploying Connector/MXJ with a pre-configured database 19
    6.3 Running within a JMX Agent (custom) 19
    6.4 Deployment in a Standard JMX Agent Environment (JBoss) 20
7 Connector/MXJ Samples 23
    7.1 Using Connector/MXJ with JSP 23
8 Connector/MXJ Support 27
9 MySQL Connector/MXJ Release Notes 29
    9.1 Changes in MySQL Connector/MXJ 5.0.12 (2011-07-07) 29
    9.2 Changes in MySQL Connector/MXJ 5.0.11 (2009-11-24) 29
    9.3 Changes in MySQL Connector/MXJ 5.0.10 (Not released) 30
    9.4 Changes in MySQL Connector/MXJ 5.0.9 (2008-08-19) 30
    9.5 Changes in MySQL Connector/MXJ 5.0.8 (2007-08-06) 30
    9.6 Changes in MySQL Connector/MXJ 5.0.7 (2007-05-27) 31
    9.7 Changes in MySQL Connector/MXJ 5.0.6 (2007-05-04) 31
    9.8 Changes in MySQL Connector/MXJ 5.0.5 (2007-03-14) 32
    9.9 Changes in MySQL Connector/MXJ 5.0.4 (2007-01-28) 33
    9.10 Changes in MySQL Connector/MXJ 5.0.3 (2006-06-24) 34
    9.11 Changes in MySQL Connector/MXJ 5.0.2 (2006-06-15) 34
    9.12 Changes in MySQL Connector/MXJ 5.0.1 (Not released) 35
    9.13 Changes in MySQL Connector/MXJ 5.0.0 (2005-12-09) 35

Preface and Legal Notices

This manual describes MySQL Connector/MXJ.

Legal Notices

only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this material is subject to the terms and conditions of your Oracle Software License and Service Agreement, which has been executed and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle or as specifically provided below. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

This documentation is NOT distributed under a GPL license. Use of this documentation is subject to the following terms:

You may create a printed copy of this documentation solely for your own personal use. Conversion to other formats is allowed as long as the actual content is not altered or edited in any way. You shall not publish or distribute this documentation in any form or on any media, except if you distribute the documentation in a manner similar to how Oracle disseminates it (that is, electronically for download on a Web site with the software) or on a CD-ROM or similar medium, provided however that the documentation is disseminated together with the software on the same medium. Any other use, such as any dissemination of printed copies or use of this documentation, in whole or in part, in another publication, requires the prior written consent from an authorized representative of Oracle. Oracle and/or its affiliates reserve any and all rights to this documentation not expressly granted above.

For more information on the terms of this license, or for details on how the MySQL documentation is built and produced, please visit MySQL Contact & Questions.

For help with using MySQL, please visit either the MySQL Forums or MySQL Mailing Lists where you can discuss your issues with other MySQL users.

For additional documentation on MySQL products, including translations of the documentation into other languages, and downloadable versions in variety of formats, including HTML and PDF formats, see the MySQL Documentation Library.

Chapter 1 Overview of Connector/MXJ

> **Support EOL for MySQL Connector/MXJ**
>
> Due to very low demand, MySQL has stopped development and support for Connector/MXJ. Source and binaries for previously released versions will continue to be available from archives.

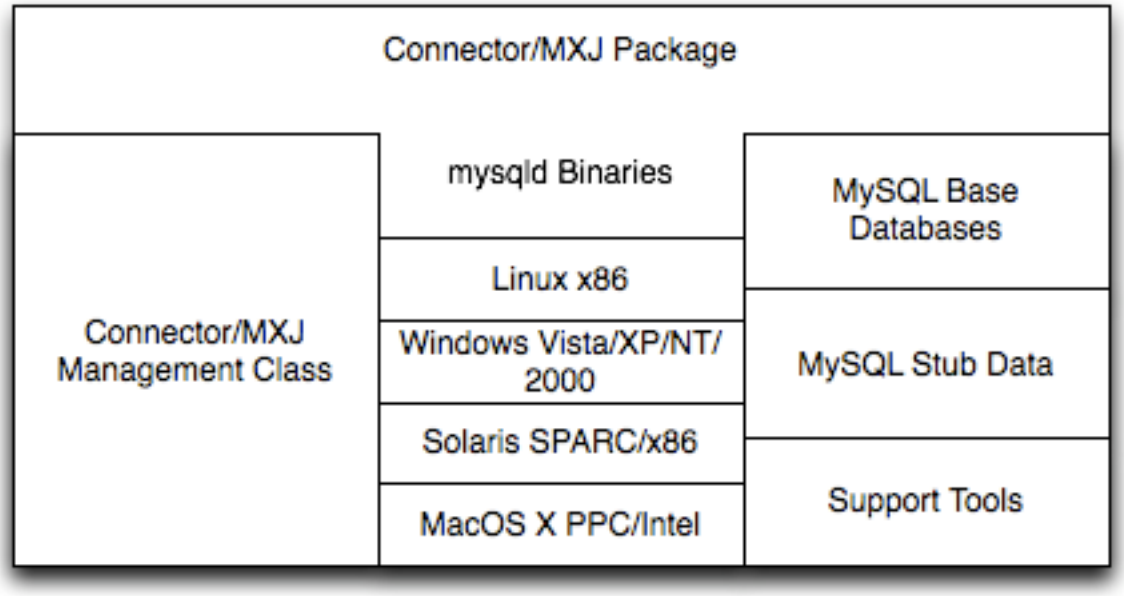
MySQL Connector/MXJ is a Java Utility package for deploying and managing a MySQL database. Deploying and using MySQL can be as easy as adding another parameter to the JDBC connection URL, which will result in the database being started when the first connection is made. This makes it easy for Java developers to deploy applications which require a database by reducing installation barriers for their end-users.

MySQL Connector/MXJ makes the MySQL database appear to be a Java-based component. It does this by determining what platform the system is running on, selecting the appropriate binary, and launching the executable. It will also optionally deploy an initial database, with any specified parameters.

Included are instructions for use with a JDBC driver and deploying as a JMX MBean to JBoss.

You can download sources and binaries from the download archives under http://downloads.mysql.com/archives.php?p=mysql-connector-mxj-5.0.

Connector/MXJ consists of a Java class, a copy of the `mysqld` binary for a specific list of platforms, and associated files and support utilities. The Java class controls the initialization of an instance of the embedded `mysqld` binary, and the ongoing management of the `mysqld` process. The entire sequence and management can be controlled entirely from within Java using the Connector/MXJ Java classes. You can see an overview of the contents of the Connector/MXJ package in the figure below.



It is important to note that Connector/MXJ is not an embedded version of MySQL, or a version of MySQL written as part of a Java class. Connector/MXJ works through the use of an embedded, compiled binary of `mysqld` as would normally be used when deploying a standard MySQL installation.

It is the Connector/MXJ wrapper, support classes and tools, that enable Connector/MXJ to appear as a MySQL instance.

When Connector/MXJ is initialized, the corresponding `mysqld` binary for the current platform is extracted, along with a pre-configured data directed. Both are contained within the Connector/MXJ JAR file. The `mysqld` instance is then started, with any additional options as specified during the initialization, and the MySQL database becomes accessible.

Because Connector/MXJ works in combination with Connector/J, you can access and integrate with the MySQL instance through a JDBC connection. When you have finished with the server, the instance is terminated, and, by default, any data created during the session is retained within the temporary directory created when the instance was started.

Connector/MXJ and the embedded `mysqld` instance can be deployed in a number of environments where relying on an existing database, or installing a MySQL instance would be impossible, including CD-ROM embedded database applications and temporary database requirements within a Java-based application environment.

Chapter 2 Connector/MXJ Versions

Connector/MXJ 5.x, which is in beta status, includes `mysqld` version 5.x and binaries for Linux x86, Mac OS X PPC, Windows XP/NT/2000 x86 and Solaris SPARC. Connector/MXJ 5.x requires the Connector/J 5.x package.

A summary of the different MySQL versions supplied with each Connector/MXJ release are shown in the table.

Connector/MXJ Version	MySQL Version(s)
5.0.11	5.1.40
5.0.9	5.0.51a (CS), 5.0.54 (ES)
5.0.8	5.0.45 (CS), 5.0.46 (ES)
5.0.7	5.0.41 (CS), 5.0.42 (ES)
5.0.6	5.0.37 (CS), 5.0.40 (ES)
5.0.5	5.0.37 (CS), 5.0.36 (ES)
5.0.4	5.0.27 (CS), 5.0.32 (ES)
5.0.3	5.0.22
5.0.2	5.0.19

This guide provides information on the Connector/MXJ 5.x release. For information on using the older releases, please see the documentation included with the appropriate distribution.

Chapter 3 Connector/MXJ Installation

Table of Contents

3.1 Supported Platforms 5
3.2 Connector/MXJ Base Installation 6
3.3 Connector/MXJ Quick Start Guide 7
3.4 Deploying Connector/MXJ Using Driver Launch 8
3.5 Deploying Connector/MXJ within JBoss 8
3.6 Verifying Installation Using JUnit 8
    3.6.1 JUnit Test Requirements 9
    3.6.2 Running the JUnit Tests 9

Connector/MXJ does not have an installation application or process, but there are some steps you can follow to make the installation and deployment of Connector/MXJ easier.

Before you start, there are some baseline requirements for

- Java Runtime Environment (v1.4.0 or newer) if you are only going to deploy the package.
- Java Development Kit (v1.4.0 or newer) if you intend to build Connector/MXJ from source.
- Connector/J 5.0 or newer.

Depending on your target installation/deployment environment you may also require:

- JBoss - 4.0rc1 or newer
- Apache Tomcat - 5.0 or newer
- Sun's JMX reference implementation version 1.2.1 (from http://java.sun.com/products/JavaManagement/)

3.1 Supported Platforms

Connector/MXJ is compatible with any platform supporting Java and MySQL. By default, Connector/MXJ incorporates the `mysqld` binary for a select number of platforms which differs by version. The following platforms have been tested and working as deployment platforms. Support for all the platforms listed below is not included by default.

- Linux (i386)
- FreeBSD (i386)
- Windows NT (x86), Windows 2000 (x86), Windows XP (x86), Windows Vista (x86)
- Solaris 8, SPARC 32-bit (compatible with Solaris 8, Solaris 9 and Solaris 10 on SPARC 32-bit and 64-bit platforms)
- Mac OS X (PowerPC and Intel)

The Connector/MXJ 5.0.8 release includes `mysqld` binaries for the following platforms:

- Linux (i386)

- Windows (x86), compatible with Windows NT, Windows 2000, Windows XP , Windows Vista
- Solaris 8, SPARC 32-bit (compatible with Solaris 8, Solaris 9 and Solaris 10 on SPARC 32-bit and 64-bit platforms)
- Mac OS X (PowerPC and Intel)

For more information on packaging your own Connector/MXJ with the platforms you require, see Section 6.1, “Creating your own Connector/MXJ Package”

3.2 Connector/MXJ Base Installation

Because there is no formal installation process, the method, installation directory, and access methods you use for Connector/MXJ are entirely up to your individual requirements.

To perform a basic installation, choose a target directory for the files included in the Connector/MXJ package. On Unix/Linux systems, you might use a directory such as `/usr/local/connector-mxj`. On Windows, you might install the files in the base directory, `C:\Connector-MXJ`, or within the `Program Files` directory.

To install the files, for a Connector/MXJ 5.0.4 installation:

1. Download the Connector/MXJ package, either in Tar/Gzip format (ideal for Unix/Linux systems) or Zip format (Windows).
2. Extract the files from the package. This will create a directory `mysql-connector-mxj-gpl-[ver]`. Copy and optionally rename this directory to your desired location.
3. For best results, update your global `CLASSPATH` variable with the location of the required `jar` files.

   Within Unix/Linux you can do this globally by editing the global shell profile, or on a user by user basis by editing their individual shell profile.

   On Windows 2000, Windows NT and Windows XP, you can edit the global `CLASSPATH` by editing the `Environment Variables` configured through the `System` control panel.

For Connector/MXJ 5.0.6 and later you need the following JAR files in your `CLASSPATH`:

- `mysql-connector-mxj-gpl-[ver].jar`: contains the main Connector/MXJ classes.
- `mysql-connector-mxj-gpl-[ver]-db-files.jar`: contains the embedded `mysqld` and database files.
- `aspectjrt.jar`: the AspectJ runtime library, located in `lib/aspectjrt.jar` in the Connector/MXJ package.
- `mysql-connector-java-[ver]-bin.jar`: Connector/J, see MySQL Connector/J Developer Guide.

For Connector/MXJ 5.0.4 and later you need the following JAR files in your `CLASSPATH`:

- `connector-mxj.jar`: contains the main Connector/MXJ classes.
- `connector-mxj-db-files.jar`: contains the embedded `mysqld` and database files.
- `aspectjrt.jar`: the AspectJ runtime library, located in `lib/aspectjrt.jar` in the Connector/MXJ package.
- `mysql-connector-mxj-gpl-[ver].jar`: Connector/J, see MySQL Connector/J Developer Guide.

For Connector/MXJ 5.0.3 and earlier, you need the following JAR files:

- `connector-mxj.jar`
- `aspectjrt.jar`: the AspectJ runtime library, located in `lib/aspectjrt.jar` in the Connector/MXJ package.
- `mysql-connector-mxj-gpl-[ver].jar`: Connector/J, see MySQL Connector/J Developer Guide.

3.3 Connector/MXJ Quick Start Guide

Once you have extracted the Connector/MXJ and Connector/J components you can run one of the sample applications that initiates a MySQL instance. You can test the installation by running the `ConnectorMXJUrlTestExample`:

```
shell> java ConnectorMXJUrlTestExample
jdbc:mysql:mxj://localhost:3336/our_test_app?server.basedir»
    =/var/tmp/test-mxj&createDatabaseIfNotExist=true&server.initialize-user=true
[/var/tmp/test-mxj/bin/mysqld][--no-defaults][--port=3336][--socket=mysql.sock]»
    [--basedir=/var/tmp/test-mxj][--datadir=/var/tmp/test-mxj/data]»
    [--pid-file=/var/tmp/test-mxj/data/MysqldResource.pid]
[MysqldResource] launching mysqld (driver_launched_mysqld_1)
InnoDB: The first specified data file ./ibdata1 did not exist:
InnoDB: a new database to be created!
080220  9:40:20  InnoDB: Setting file ./ibdata1 size to 10 MB
InnoDB: Database physically writes the file full: wait...
080220  9:40:20  InnoDB: Log file ./ib_logfile0 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile0 size to 5 MB
InnoDB: Database physically writes the file full: wait...
080220  9:40:20  InnoDB: Log file ./ib_logfile1 did not exist: new to be created
InnoDB: Setting log file ./ib_logfile1 size to 5 MB
InnoDB: Database physically writes the file full: wait...
InnoDB: Doublewrite buffer not found: creating new
InnoDB: Doublewrite buffer created
InnoDB: Creating foreign key constraint system tables
InnoDB: Foreign key constraint system tables created
080220  9:40:21  InnoDB: Started; log sequence number 0 0
080220  9:40:21 [Note] /var/tmp/test-mxj/bin/mysqld: ready for connections.
Version: '5.0.51a'  socket: 'mysql.sock'  port: 3336  MySQL Community Server (GPL)
[MysqldResource] mysqld running as process: 2238
-----------------------
SELECT VERSION()
-----------------------
5.0.51a
-----------------------
[MysqldResource] stopping mysqld (process: 2238)
080220  9:40:27 [Note] /var/tmp/test-mxj/bin/mysqld: Normal shutdown

080220  9:40:27  InnoDB: Starting shutdown...
080220  9:40:29  InnoDB: Shutdown completed; log sequence number 0 43655
080220  9:40:29 [Note] /var/tmp/test-mxj/bin/mysqld: Shutdown complete

[MysqldResource] shutdown complete
```

The above output shows an instance of MySQL starting, the necessary files being created (log files, InnoDB data files) and the MySQL database entering the running state. The instance is then shutdown by Connector/MXJ before the example terminates.

> **Warning**
>
> Avoid running your Connector/MXJ application as the `root` user, because this will cause the `mysqld` to also be executed with root privileges. For more information, see How to Run MySQL as a Normal User.

3.4 Deploying Connector/MXJ Using Driver Launch

Connector/MXJ and Connector/J work together to enable you to launch an instance of the `mysqld` server through the use of a keyword in the JDBC connection string. Deploying Connector/MXJ within a Java application can be automated through this method, making the deployment of Connector/MXJ a simple process:

1. Download and unzip Connector/MXJ. Add `mysql-connector-mxj-gpl-[ver].jar` to the `CLASSPATH`.

   If you are using Connector/MXJ v5.0.4 or later, also add the `mysql-connector-mxj-gpl-[ver]-db-files.jar` file to your `CLASSPATH`.

2. To the JDBC connection string, embed the `mxj` keyword, for example: `jdbc:mysql:mxj://localhost:PORT/DBNAME`.

For more details, see Chapter 4, *Connector/MXJ Configuration*.

3.5 Deploying Connector/MXJ within JBoss

For deployment of Connector/MXJ within a JBoss environment, you must configure the JBoss environment to use the Connector/MXJ component within the JDBC parameters:

1. Download Connector/MXJ and copy the `mysql-connector-mxj-gpl-[ver].jar` file to the `$JBOSS_HOME/server/default/lib` directory.

   If you are using Connector/MXJ v5.0.4 or later, also copy the `mysql-connector-mxj-gpl-[ver]-db-files.jar` file to `$JBOSS_HOME/server/default/lib`.

2. Download Connector/J and copy the `mysql-connector-java-5.1.5-bin.jar` file to the `$JBOSS_HOME/server/default/lib` directory.

3. Create an MBean service xml file in the `$JBOSS_HOME/server/default/deploy` directory with any attributes set, for instance the `datadir` and `autostart`.

4. Set the JDBC parameters of your web application to use:

```
String driver = "com.mysql.jdbc.Driver";
String url = "jdbc:mysql:///test?propertiesTransform="+
             "com.mysql.management.jmx.ConnectorMXJPropertiesTransform";
String user = "root";
String password = "";
Class.forName(driver);
Connection conn = DriverManager.getConnection(url, user, password);
```

You might create a separate users and database table spaces for each application, rather than using "root and test".

We highly suggest having a routine backup procedure for backing up the database files in the `datadir`.

3.6 Verifying Installation Using JUnit

The best way to ensure that your platform is supported is to run the JUnit tests. These will test the Connector/MXJ classes and the associated components.

3.6.1 JUnit Test Requirements

The first thing to do is make sure that the components will work on the platform. The `MysqldResource` class is really a wrapper for a native version of MySQL, so not all platforms are supported. At the time of this writing, Linux on the i386 architecture has been tested and seems to work quite well, as does Mac OS X 10.3. There has been limited testing on Windows and Solaris.

Requirements:

1. JDK-1.4 or newer (or the JRE if you aren't going to be compiling the source or JSPs).
2. MySQL Connector/J version 5.0 or newer (from http://dev.mysql.com/downloads/connector/j/) installed and available using your CLASSPATH.
3. The `javax.management` classes for JMX version 1.2.1, these are present in the following application servers:
   - JBoss - 4.0rc1 or newer.
   - Apache Tomcat - 5.0 or newer.
   - Sun's JMX reference implementation version 1.2.1 (from http://java.sun.com/products/JavaManagement/).
4. JUnit 3.8.1 (from http://www.junit.org/).

If building from source, All of the requirements from above, plus:

1. Ant version 1.5 or newer (download from http://ant.apache.org/).

3.6.2 Running the JUnit Tests

1. The tests attempt to launch MySQL on the port 3336. That MySQL instance is unlikely to conflict with another MySQL server on the same machine, because the default port for MySQL is 3306. If you do encounter a port conflict, either set the `c-mxj_test_port` Java property to a port of your choosing, or shut down any instances of MySQL you have running on the target machine.

   The tests suppress output to the console by default. For verbose output, you may set the `c-mxj_test_silent` Java property to `false`.

2. To run the JUnit test suite, the `$CLASSPATH` must include the following:
   - JUnit
   - JMX
   - Connector/J
   - MySQL Connector/MXJ

3. If `connector-mxj.jar` is not present in your download, unzip MySQL Connector/MXJ source archive.

   ```
   cd mysqldjmx
   ant dist
   ```

   Then add `$TEMP/cmxj/stage/connector-mxj/connector-mxj.jar` to the `CLASSPATH`.

4. If you have `junit`, execute the unit tests. From the command line, type:

```
java com.mysql.management.AllTestsSuite
```

The output should look something like this:

```

.....................................
.....................................
.........
Time: 259.438

OK (101 tests)
```

Note that the tests are a bit slow near the end, so please be patient.

Chapter 4 Connector/MXJ Configuration

Table of Contents

4.1 Running as part of the JDBC Driver 11
4.2 Running within a Java Object 12
4.3 Setting Server Options 13

4.1 Running as part of the JDBC Driver

A feature of the MySQL Connector/J JDBC driver is the ability to specify a connection to an embedded Connector/MXJ instance through the use of the mxj keyword in the JDBC connection string.

In the following example, we have a program which creates a connection, executes a query, and prints the result to the System.out. The MySQL database will be deployed and started as part of the connection process, and shutdown as part of the finally block.

You can find this file in the Connector/MXJ package as `src/ConnectorMXJUrlTestExample.java`.

```
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;

import com.mysql.management.driverlaunched.ServerLauncherSocketFactory;
import com.mysql.management.util.QueryUtil;

public class ConnectorMXJUrlTestExample {
  public static String DRIVER = "com.mysql.jdbc.Driver";

  public static String JAVA_IO_TMPDIR = "java.io.tmpdir";

  public static void main(String[] args) throws Exception {
    File ourAppDir = new File(System.getProperty(JAVA_IO_TMPDIR));
    File databaseDir = new File(ourAppDir, "test-mxj");
    int port = Integer.parseInt(System.getProperty("c-mxj_test_port", "3336"));
    String dbName = "our_test_app";

    String url = "jdbc:mysql:mxj://localhost:" + port + "/" + dbName //
      + "?" + "server.basedir=" + databaseDir //
      + "&" + "createDatabaseIfNotExist=true"//
      + "&" + "server.initialize-user=true" //
    ;

    System.out.println(url);

    String userName = "alice";
    String password = "q93uti0opwhkd";

    Class.forName(DRIVER);
    Connection conn = null;
    try {
      conn = DriverManager.getConnection(url, userName, password);
      String sql = "SELECT VERSION()";
      String queryForString = new QueryUtil(conn).queryForString(sql);

      System.out.println("------------------------");
      System.out.println(sql);
      System.out.println("------------------------");
      System.out.println(queryForString);
      System.out.println("------------------------");
```

```
      System.out.flush();
      Thread.sleep(100); // wait for System.out to finish flush
    } finally {
      try {
        if (conn != null)
           conn.close();
      } catch (Exception e) {
        e.printStackTrace();
      }

      ServerLauncherSocketFactory.shutdown(databaseDir, null);
    }
  }
}
```

To run the above program, be sure to have `connector-mxj.jar` and Connector/J in the CLASSPATH. Then type:

```
java ConnectorMXJUrlTestExample
```

4.2 Running within a Java Object

To “embed” a MySQL database within a Java application, use the `com.mysql.management.MysqldResource` class directly. This class may be instantiated with the default (no argument) constructor, or by passing in a `java.io.File` object representing the directory into which to unzip the server. It may also be instantiated with printstreams for `stdout` and `stderr` for logging.

Once instantiated, a `java.util.Map`, the object will be able to provide a `java.util.Map` of server options appropriate for the platform and version of MySQL which you will be using.

The `MysqldResource` enables you to "start" MySQL with a `java.util.Map` of server options which you provide, as well as "shutdown" the database. The following example shows a simplistic way to embed MySQL in an application using plain java objects.

You can find this file in the Connector/MXJ package as `src/ConnectorMXJObjectTestExample.java`.

```
import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.HashMap;
import java.util.Map;

import com.mysql.management.MysqldResource;
import com.mysql.management.MysqldResourceI;
import com.mysql.management.util.QueryUtil;

public class ConnectorMXJObjectTestExample {
    public static final String DRIVER = "com.mysql.jdbc.Driver";

    public static final String JAVA_IO_TMPDIR = "java.io.tmpdir";

    public static void main(String[] args) throws Exception {
        File ourAppDir = new File(System.getProperty(JAVA_IO_TMPDIR));
        File databaseDir = new File(ourAppDir, "mysql-mxj");
        int port = Integer.parseInt(System.getProperty("c-mxj_test_port",
                "3336"));
        String userName = "alice";
        String password = "q93uti0opwhkd";
```

```
        MysqldResource mysqldResource = startDatabase(databaseDir, port,
                userName, password);

        Class.forName(DRIVER);
        Connection conn = null;
        try {
            String dbName = "our_test_app";
            String url = "jdbc:mysql://localhost:" + port + "/" + dbName //
                    + "?" + "createDatabaseIfNotExist=true"//
            ;
            conn = DriverManager.getConnection(url, userName, password);
            String sql = "SELECT VERSION()";
            String queryForString = new QueryUtil(conn).queryForString(sql);

            System.out.println("------------------------");
            System.out.println(sql);
            System.out.println("------------------------");
            System.out.println(queryForString);
            System.out.println("------------------------");
            System.out.flush();
            Thread.sleep(100); // wait for System.out to finish flush
        } finally {
            try {
                if (conn != null) {
                    conn.close();
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
            try {
                mysqldResource.shutdown();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    public static MysqldResource startDatabase(File databaseDir, int port,
            String userName, String password) {
        MysqldResource mysqldResource = new MysqldResource(databaseDir);

        Map database_options = new HashMap();
        database_options.put(MysqldResourceI.PORT, Integer.toString(port));
        database_options.put(MysqldResourceI.INITIALIZE_USER, "true");
        database_options.put(MysqldResourceI.INITIALIZE_USER_NAME, userName);
        database_options.put(MysqldResourceI.INITIALIZE_PASSWORD, password);

        mysqldResource.start("test-mysqld-thread", database_options);

        if (!mysqldResource.isRunning()) {
            throw new RuntimeException("MySQL did not start.");
        }

        System.out.println("MySQL is running.");

        return mysqldResource;
    }
}
```

4.3 Setting Server Options

There are many options you might set for a MySQL database. These options may be specified as part of the JDBC connection string simply by prefixing each server option with `server.`. The following example sets two driver parameters and two server parameters:

```
```

```
        String url = "jdbc:mysql://" + hostColonPort + "/"
                + "?"
                + "cacheServerConfiguration=true"
                + "&"
                + "useLocalSessionState=true"
                + "&"
                + "server.basedir=/opt/myapp/db"
                + "&"
                + "server.datadir=/mnt/bigdisk/myapp/data";
```

Starting with Connector/MXJ 5.0.6, you can use the `initialize-user` property to a connection string. If set to true, the default anonymous and root users will be removed and the user/password combination from the connection URL will be used to create a new user. For example:

```
        String url = "jdbc:mysql:mxj://localhost:" + port
        + "/alice_db"
        + "?server.datadir=" + dataDir.getPath()
        + "&server.initialize-user=true"
        + "&createDatabaseIfNotExist=true"
        ;
```

Chapter 5 Connector/MXJ Reference

Table of Contents

5.1 MysqldResource Constructors 15
5.2 MysqldResource Methods 15

The following sections include detailed information on the different API interfaces to Connector/MXJ.

5.1 MysqldResource Constructors

The `MysqldResource` class supports three different constructor forms:

- `public MysqldResource(File baseDir, File dataDir, String mysqlVersionString, PrintStream out, PrintStream err)`

  Enables you to set the base directory, data directory, select a server by its version string, standard out and standard error.

- `public MysqldResource(File baseDir, File dataDir, String mysqlVersionString)`

  Enables you to set the base directory, data directory and select a server by its version string. Output for standard out and standard err are directed to System.out and System.err.

- `public MysqldResource(File baseDir, File dataDir)`

  Enables you to set the base directory and data directory. The default MySQL version is selected, and output for standard out and standard err are directed to System.out and System.err.

- `public MysqldResource(File baseDir);`

  Enables the setting of the "basedir" to deploy the MySQL files to. Output for standard out and standard err are directed to System.out and System.err.

- `public MysqldResource();`

  The basedir is defaulted to a subdirectory of the java.io.tempdir. Output for standard out and standard err are directed to System.out and System.err;

5.2 MysqldResource Methods

MysqldResource API includes the following methods:

- `void start(String threadName, Map mysqldArgs);`

  Deploys and starts MySQL. The "threadName" string is used to name the thread which actually performs the execution of the MySQL command line. The map is the set of arguments and their values to be passed to the command line.

- `void shutdown();`

  Shuts down the MySQL instance managed by the MysqldResource object.

- `Map getServerOptions();`

Returns a map of all the options and their current (or default, if not running) options available for the MySQL database.

- `boolean isRunning();`

  Returns true if the MySQL database is running.

- `boolean isReadyForConnections();`

  Returns true once the database reports that is ready for connections.

- `void setKillDelay(int millis);`

  The default “Kill Delay” is 30 seconds. This represents the amount of time to wait between the initial request to shutdown and issuing a “force kill” if the database has not shutdown by itself.

- `void addCompletionListenser(Runnable listener);`

  Enables applications to be notified when the server process completes. Each “listener” will be fired off in its own thread.

- `String getVersion();`

  Returns the version of MySQL.

- `void setVersion(int MajorVersion, int minorVersion, int patchLevel);`

  The standard distribution comes with only one version of MySQL packaged. However, it is possible to package multiple versions, and specify which version to use.

Chapter 6 Connector/MXJ Notes and Tips

Table of Contents

6.1 Creating your own Connector/MXJ Package 17
6.2 Deploying Connector/MXJ with a pre-configured database 19
6.3 Running within a JMX Agent (custom) 19
6.4 Deployment in a Standard JMX Agent Environment (JBoss) 20

This section contains notes and tips on using the Connector/MXJ component within your applications.

6.1 Creating your own Connector/MXJ Package

To create a custom Connector/MXJ package that includes a specific `mysqld` version or platform, extract and rebuild the `mysql-connector-mxj.jar` (Connector/MXJ v5.0.3 or earlier) or `mysql-connector-mxj-gpl-[ver]-db-files.jar` (Connector/MXJ v5.0.4 or later) file.

First, create a new directory into which you can extract the current `connector-mxj.jar`:

```
shell> mkdir custom-mxj
shell> cd custom-mxj
shell> jar -xf connector-mxj.jar
shell> ls
5-0-22/
ConnectorMXJObjectTestExample.class
ConnectorMXJUrlTestExample.class
META-INF/
TestDb.class
com/
kill.exe
```

If you are using Connector/MXJ v5.0.4 or later, unpack the `connector-mxj-db-files.jar`:

```
shell> mkdir custom-mxj
shell> cd custom-mxj
shell> jar -xf connector-mxj-db-files.jar
shell> ls
5-0-51a/
META-INF/
connector-mxj.properties
```

The MySQL version directory, `5-0-22` or `5-0-51a` in the preceding examples, contains all of the files used to create an instance of MySQL when Connector/MXJ is executed. All of the files in this directory are required for each version of MySQL to embed. Note as well the format of the version number, which uses hyphens instead of periods to separate the version number components.

Within the version specific directory are the platform specific directories, and archives of the `data` and `share` directory required by MySQL for the various platforms. For example, here is the listing for the default Connector/MXJ package:

```
shell>> ls
Linux-i386/
META-INF/
Mac_OS_X-ppc/
SunOS-sparc/
Win-x86/
com/
data_dir.jar
```

```
share_dir.jar
win_share_dir.jar
```

Platform specific directories are listed by their OS and platform - for example the `mysqld` for Mac OS X PowerPC is located within the `Mac_OS_X-ppc` directory. You can delete directories from this location that you do not require, and add new directories for additional platforms that you intend to support.

To add a platform specific `mysqld`, create a new directory with the corresponding name for your operating system/platform. For example, you could add a directory for Mac OS X/Intel using the directory `Mac_OS_X-i386`.

On Unix systems, you can determine the platform using `uname`:

```
shell> uname -p
i386
```

In Connector/MXJ v5.0.9 and later, an additional `platform-map.properties` file is used to associate a specific platform and operating system combination with the directory in which the `mysqld` for that combination is located. The determined operating system and platform are on the left, and the directory name where the appropriate mysqld is located is on the right. You can see a sample of the file below:

```
Linux-i386=Linux-i386
Linux-x86=Linux-i386
Linux-i686=Linux-i386
Linux-x86_64=Linux-i386
Linux-ia64=Linux-i386

#Linux-ppc=Linux-ppc
#Linux-ppc64=Linux-ppc

Mac_OS_X-i386=Mac_OS_X-i386
Mac_OS_X-ppc=Mac_OS_X-ppc
Rhapsody-PowerPC=Mac_OS_X-ppc
#Mac_OS-PowerPC=
#macos-PowerPC=
#MacOS-PowerPC=

SunOS-sparc=SunOS-sparc
Solaris-sparc=SunOS-sparc
SunOS-x86=SunOS-x86
Solaris-x86=SunOS-x86

FreeBSD-x86=FreeBSD-x86

Windows_Vista-x86=Win-x86
Windows_2003-x86=Win-x86
Windows_XP-x86=Win-x86
Windows_2000-x86=Win-x86
Windows_NT-x86=Win-x86
Windows_NT_(unknown)-x86=Win-x86
```

Now download or compile `mysqld` for the MySQL version and platform you want to include in your custom `connector-mxj.jar` package into the new directory.

Create a file called `version.txt` in the OS/platform directory you have just created that contains the version string/path of the mysqld binary. For example:

```
mysql-5.0.22-osx10.3-i386/bin/mysqld
```

You can now recreate the `connector-mxj.jar` file with the added `mysqld`:

```
shell> cd custom-mxj
shell> jar -cf ../connector-mxj.jar *
```

For Connector/MXJ v5.0.4 and later, repackage to the `connector-mxj-db-files.jar`:

```
shell> cd custom-mxj
shell> jar -cf ../mysql-connector-mxj-gpl-[ver]-db-files.jar *
```

Test this package using the steps outlined in Section 3.3, “Connector/MXJ Quick Start Guide”.

> **Note**
>
> Because the `mysql-connector-mxj-gpl-[ver]-db-files.jar` file is separate from the main Connector/MXJ classes you can distribute different `mysql-connector-mxj-gpl-[ver]-db-files.jar` files to different hosts or for different projects without having to create a completely new main `mysql-connector-mxj-gpl-[ver].jar` file for each one.

6.2 Deploying Connector/MXJ with a pre-configured database

To include a pre-configured/populated database within your Connector/MXJ JAR file you must create a custom `data_dir.jar` file, as included within the main `connector-mxj.jar` (Connector/MXJ 5.0.3 or earlier) or `mysql-connector-mxj-gpl-[ver]-db-files.jar` (Connector/MXJ 5.0.4 or later) file:

1. First extract the `connector-mxj.jar` or `mysql-connector-gpl-[ver]-db-files.jar` file, as outlined in the previous section (see Section 6.1, “Creating your own Connector/MXJ Package”).
2. First, create your database and populate the database with the information you require in an existing instance of MySQL - including Connector/MXJ instances. Data file formats are compatible across platforms.
3. Shutdown the instance of MySQL.
4. Create a JAR file of the data directory and databases that you intend to include your Connector/MXJ package. Include the `mysql` database, which includes user authentication information, in addition to the specific databases to include. For example, to create a JAR of the `mysql` and `mxjtest` databases:

   ```
   shell> jar -cf ../data_dir.jar mysql mxjtest
   ```

5. For Connector/MXJ 5.0.3 or earlier, copy the `data_dir.jar` file into the extracted `connector-mxj.jar` directory, and then create an archive for `connector-mxj.jar`.

   For Connector/MXJ 5.0.4 or later, copy the `data_dir.jar` file into the extracted `mysql-connector-mxj-gpl-[ver]-db-files.jar` directory, and then create an archive for `mysql-connector-mxj-db-gpl-[ver]--files.jar`.

Note that if you are create databases using the InnoDB engine, you must include the `ibdata.*` and `ib_logfile*` files within the `data_dir.jar` archive.

6.3 Running within a JMX Agent (custom)

As a JMX MBean, MySQL Connector/MXJ requires a JMX v1.2 compliant MBean container, such as JBoss version 4. The MBean will uses the standard JMX management APIs to present (and allow the setting of) parameters which are appropriate for that platform.

If you are not using the SUN Reference implementation of the JMX libraries, skip this section. Or, if you are deploying to JBoss, you can also skip to the next section.

We want to see the MysqldDynamicMBean in action inside of a JMX agent. In the `com.mysql.management.jmx.sunri` package is a custom JMX agent with two MBeans:

1. The MysqldDynamicMBean, and
2. A com.sun.jdmk.comm.HtmlAdaptorServer, which provides a web interface for manipulating the beans inside of a JMX agent.

When this very simple agent is started, it will allow a MySQL database to be started and stopped with a web browser.

1. Complete the testing of the platform as above.
   - Current JDK, JUnit, Connector/J, MySQL Connector/MXJ
   - This section *requires* the SUN reference implementation of JMX
   - `PATH`, `JAVA_HOME`, `ANT_HOME`, `CLASSPATH`
2. If not building from source, skip to next step

   Rebuild with the "sunri.present"

   ```
   ant -Dsunri.present=true dist
   re-run tests:
   java junit.textui.TestRunner com.mysql.management.AllTestsSuite
   ```
3. Launch the test agent from the command line:

   ```
   java com.mysql.management.jmx.sunri.MysqldTestAgentSunHtmlAdaptor &
   ```
4. From a browser:

   ```
   http://localhost:9092/
   ```
5. Under MysqldAgent,

   ```
   select "name=mysqld"
   ```
6. Observe the MBean View
7. Scroll to the bottom of the screen press the startMysqld button
8. Click `Back to MBean View`
9. Scroll to the bottom of the screen press stopMysqld button
10. Kill the java process running the Test Agent (jmx server)

6.4 Deployment in a Standard JMX Agent Environment (JBoss)

Once there is confidence that the MBean will function on the platform, deploying the MBean inside of a standard JMX Agent is the next step. Included are instructions for deploying to JBoss.

1. Ensure a current version of java development kit (v1.4.x), see above.
   - Ensure `JAVA_HOME` is set (JBoss requires `JAVA_HOME`)

- Ensure `JAVA_HOME/bin` is in the `PATH` (You will NOT need to set your CLASSPATH, nor will you need any of the jars used in the previous tests).

2. Ensure a current version of JBoss (v4.0RC1 or better)

```
http://www.jboss.org/index.html
select "Downloads"
select "jboss-4.0.zip"
pick a mirror
unzip ~/dload/jboss-4.0.zip
create a JBOSS_HOME environment variable set to the unzipped directory
unix only:
cd $JBOSS_HOME/bin
chmod +x *.sh
```

3. Deploy (copy) the `connector-mxj.jar` to `$JBOSS_HOME/server/default/lib`.
4. Deploy (copy) `mysql-connector-java-3.1.4-beta-bin.jar` to `$JBOSS_HOME/server/default/lib`.
5. Create a `mxjtest.war` directory in `$JBOSS_HOME/server/default/deploy`.
6. Deploy (copy) `index.jsp` to `$JBOSS_HOME/server/default/deploy/mxjtest.war`.
7. Create a `mysqld-service.xml` file in `$JBOSS_HOME/server/default/deploy`.

```
<?xml version="1.0" encoding="UTF-8"?>
 <server>
  <mbean code="com.mysql.management.jmx.jboss.JBossMysqldDynamicMBean"
     name="mysql:type=service,name=mysqld">
  <attribute name="datadir">/tmp/xxx_data_xxx</attribute>
  <attribute name="autostart">true</attribute>
  </mbean>
 </server>
```

8. Start jboss:

   - On unix: `$JBOSS_HOME/bin/run.sh`
   - On windows: `%JBOSS_HOME%\bin\run.bat`

   Be ready: JBoss sends a lot of output to the screen.

9. When JBoss seems to have stopped sending output to the screen, open a web browser to: `http://localhost:8080/jmx-console`
10. Scroll down to the bottom of the page in the `mysql` section, select the bulleted `mysqld` link.
11. Observe the JMX MBean View page. MySQL should already be running.
12. (If "autostart=true" was set, you may skip this step.) Scroll to the bottom of the screen. You may press the Invoke button to stop (or start) MySQL observe `Operation completed successfully without a return value.` Click `Back to MBean View`
13. To confirm MySQL is running, open a web browser to `http://localhost:8080/mxjtest/` and you should see that the following query:

```
SELECT 1
```

returned with the following result:

```
1
```

14. Guided by the `$JBOSS_HOME/server/default/deploy/mxjtest.war/index.jsp` you will be able to use MySQL in your Web Application. There is a `test` database and a `root` user (no password) ready to experiment with. Try creating a table, inserting some rows, and doing some selects.

15. Shut down MySQL. MySQL will be stopped automatically when JBoss is stopped, or: from the browser, scroll down to the bottom of the MBean View press the stop service Invoke button to halt the service. Observe `Operation completed successfully without a return value.` Using `ps` or `task manager` see that MySQL is no longer running

As of 1.0.6-beta version is the ability to have the MBean start the MySQL database upon start up. Also, we've taken advantage of the JBoss life-cycle extension methods so that the database will gracefully shut down when JBoss is shutdown.

Chapter 7 Connector/MXJ Samples

Table of Contents

7.1 Using Connector/MXJ with JSP 23

This section contains some sample applications using Connector/MXJ.

7.1 Using Connector/MXJ with JSP

The following web application was provided by Pavan Venkatesh as an example of a JSP-based application using Connector/MXJ to provide contact data. The example consists of two components, `insertdata.jsp` and `response.jsp`. The `insertdata.jsp` provides a form into which you can enter information to be stored within the MySQL database provided by Connector/MXJ. The `response.jsp` file is called when you submit the form and then provides a table of the data.

Because the data is stored through Connector/MXJ the instantiation of the MySQL database is handled dynamically on the fly, making the script lightweight and self-contained.

Example 7.1 `insertdata.jsp`

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<%@ page import="java.io.*"%>
<%@ page import="java.sql.*"%>

<HTML>
<HEAD>
<TITLE>insert data</TITLE>
<script language="Javascript">

function validFields(){

if (document.form2.ID.value == "" || document.form2.ID.value == null){
alert ( "Please enter ID / Field cannot be left blank" );
return false;
}

if (document.form2.names.value == "" || document.form2.names.value == null){
alert ( "Please enter Name / Field cannot be left blank" );
return false;
}
if (document.form2.place.value == ""|| document.form2.place.value == null){
alert ( "Please enter Place / Field cannot be left blank" );
return false;
}
if (document.form2.phone.value == ""|| document.form2.phone.value == null){
alert ( "Please enter Phone number / Field cannot be left blank" );
return false;
}
return true;
}
</script>
</HEAD>

<BODY bgcolor="#ffffcc">

<H1>Welcome to MySQL world</H1>
```

```
<H2>MySQL with MXJ Connection</H2>

<P>Insert data in existing "contactdetails2009" table under
"Directory" database</P>

<FORM action="response.jsp" method="get" name="form2">

<TABLE style="background-color: #ECE5B6;" WIDTH="30%">

 <TR>
  <TH width="50%">
  <center>ID</center>
  </TH>
  <TD width="50%"><INPUT TYPE="text" NAME="ID"></TD>
 </TR>

 <TR>
  <TH width="50%">
  <center>Name</center>
  </TH>
  <TD width="50%"><INPUT TYPE="text" NAME="names"></TD>
 </TR>

 <TR>
  <TH width="50%">
  <center>City</center>
  </TH>
  <TD width="50%"><INPUT TYPE="text" NAME="place"></TD>
 </TR>

 <TR>
  <TH width="50%">
  <center>Phone</center>
  </TH>
  <TD width="50%"><INPUT TYPE="text" NAME="phone"></TD>
 </TR>

 <TR>
  <TH></TH>
  <TD width="50%"><INPUT TYPE="submit" VALUE="Insert"
   OnClick="return validFields();"></TD>
 </TR>

</TABLE>
</FORM>

</BODY>
```

Example 7.2 `response.jsp`

```
<%@ page import="java.sql.*"%>
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<%@ page import="java.io.*"%>
<%@ page import="java.sql.*"%>
<%@ page import="java.sql.Connection"%>
<%@ page import="java.sql.DriverManager"%>
<%@ page language="java"%>
<%@ page import=" com.mysql.management.util.QueryUtil"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```
   "http://www.w3.org/TR/html4/loose.dtd">

<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
<TITLE>JSP Page</TITLE>
</HEAD>

<BODY bgcolor="#ffffcc">

<P><INPUT type=button value="Back" onCLick="history.back()"></P>

<H1>Welcome to Directory Database</H1>
<%
        String ID = request.getParameter("ID");
        String name = request.getParameter("names");
        String city = request.getParameter("place");
        String phone = request.getParameter("phone");

        File ourAppDir1 = new File("/tmp/src");
        File databaseDir1 = new File(ourAppDir1,"database");
        int port = 4336;
        String dbName = "Directory";

      try
                {
        String url = "jdbc:mysql:mxj://localhost:" + port + "/" + dbName //
                + "?" + "server.basedir=" + databaseDir1 //
                + "&" + "createDatabaseIfNotExist=true"//
                + "&" + "server.initialize-user=true" //
                ;

    Connection connection = null;
    int updateQuery=0;

 Class.forName("com.mysql.jdbc.Driver").newInstance();

       String userName = "alice";
        String password = "q93uti0opwhkd";
            connection = DriverManager.getConnection(url, userName, password);
              PreparedStatement pstatement = null;

             String sql = "SELECT VERSION()";
            String queryForString = new QueryUtil(connection).queryForString(sql);

            String command = "INSERT INTO contactdetails2009 (ID, name,city,phone) VALUES (?,?,?,?)";

             pstatement = connection.prepareStatement(command);
             pstatement.setString(1, ID);
             pstatement.setString(2, name);
    pstatement.setString(3, city);
    pstatement.setString(4, phone);
                        updateQuery =  pstatement.executeUpdate();

            ResultSet resultset = pstatement.executeQuery("select * from contactdetails2009");


              if (updateQuery != 0) { %>

<P>MySQL Version------------&gt; <%         out.println(queryForString);          %>
</P>

<TABLE style="background-color: #ECE5B6;" WIDTH="50%">

 <TR>
  <TH style="text-align: center;">Data successfully inserted into
  MySQL database using MXJ connection</TH>
```

```
 </TR>
 <TR>
  <th style="text-align: center;">Storage Engine used is MyISAM</th>
 </TR>
 <TR>
  <TD>
  <H2>ContactDetails2009 Table</H2>
  <TABLE cellpadding="10" border="1" style="background-color: #ffffcc;">
   <TR>
    <TH>ID</TH>
    <TH>name</TH>
    <TH>city</TH>
    <TH>phone</TH>
   </TR>

   <%
            while(resultset.next()){
        %>

   <TR>
    <TD><%= resultset.getString(1) %></TD>

    <TD><%= resultset.getString(2) %></TD>

    <TD><%= resultset.getString(3) %></TD>

    <TD><%= resultset.getString(4) %></TD>
   </TR>

   <%
           } %>

   <%

               resultset.close();
              pstatement.close();
              connection.close();
              }
           }

       catch (Exception e) {

    %>


   <TR>
    <TD>ERROR------------&gt; <%
                out.println("ID or Row already exist");
           }
        %>
    </TD>
   </TR>
  </TABLE>
  </TD>
 </TR>
</TABLE>

</BODY>

</HTML>
```

Chapter 8 Connector/MXJ Support

There are a wide variety of options available for obtaining support for using Connector/MXJ. Contact the Connector/MXJ community for help before reporting a potential bug or problem. See http://dev.mysql.com/doc/connector-mxj/en/connector-mxj-support-community.html

Chapter 9 MySQL Connector/MXJ Release Notes

Table of Contents

9.1 Changes in MySQL Connector/MXJ 5.0.12 (2011-07-07) 29
9.2 Changes in MySQL Connector/MXJ 5.0.11 (2009-11-24) 29
9.3 Changes in MySQL Connector/MXJ 5.0.10 (Not released) 30
9.4 Changes in MySQL Connector/MXJ 5.0.9 (2008-08-19) 30
9.5 Changes in MySQL Connector/MXJ 5.0.8 (2007-08-06) 30
9.6 Changes in MySQL Connector/MXJ 5.0.7 (2007-05-27) 31
9.7 Changes in MySQL Connector/MXJ 5.0.6 (2007-05-04) 31
9.8 Changes in MySQL Connector/MXJ 5.0.5 (2007-03-14) 32
9.9 Changes in MySQL Connector/MXJ 5.0.4 (2007-01-28) 33
9.10 Changes in MySQL Connector/MXJ 5.0.3 (2006-06-24) 34
9.11 Changes in MySQL Connector/MXJ 5.0.2 (2006-06-15) 34
9.12 Changes in MySQL Connector/MXJ 5.0.1 (Not released) 35
9.13 Changes in MySQL Connector/MXJ 5.0.0 (2005-12-09) 35

9.1 Changes in MySQL Connector/MXJ 5.0.12 (2011-07-07)

Functionality Added or Changed

- **Incompatible Change:** Connector/MXJ now contains MySQL 5.5.9, not MySQL 5.1.
- These platform mappings were added:
  - Windows_Vista-amd64=Win-x86
  - Windows_2003-amd64=Win-x86
  - Windows_2000-amd64=Win-x86
  - FreeBSD-i386=FreeBSD-x86

9.2 Changes in MySQL Connector/MXJ 5.0.11 (2009-11-24)

Functionality Added or Changed

- The default timeout for the kill delay within the embedded test suite has been increased from 10 to 30 seconds.
- The embedded Aspect/J class has been removed.
- The contents of the directory used for bootstrapping the MySQL databases is now configurable by using the `windows-share-dir-jar` property. You should supply the name of a jar containing the files you want to use.
- The embedded MySQL binaries have been updated to MySQL 5.1.40 for GPL releases and MySQL 5.1.40 for Commercial releases.

Bugs Fixed

- On startup Connector/MXJ generated an exception on Windows 7:

```
Exception in thread “Thread-3” java.util.MissingResourceException?: Resource
‘5-0-51a/Windows_7-x86/mysqld-nt.exe’ not found
at com.mysql.management.util.Streams.getResourceAsStream(Streams.java:133)
at com.mysql.management.util.Streams.getResourceAsStream(Streams.java:107)
at com.mysql.management.util.Streams$1.inner(Streams.java:149) at
com.mysql.management.util.Exceptions$VoidBlock?.exec(Exceptions.java:128)
at com.mysql.management.util.Streams.createFileFromResource(Streams.java:162)
at com.mysql.management.MysqldResource?.makeMysqld(MysqldResource?.java:533)
at com.mysql.management.MysqldResource?.deployFiles(MysqldResource?.java:518)
at com.mysql.management.MysqldResource?.exec(MysqldResource?.java:495)
at com.mysql.management.MysqldResource?.start(MysqldResource?.java:216)
at com.mysql.management.MysqldResource?.start(MysqldResource?.java:166)
```

The default `platform-map.properties` file, which maps platforms to the supplied binary bundles, has been updated with additional platforms, including Windows 7, Windows Server 2008, `amd64` and `sparcv9` for Solaris, and Mac OS X 64-bit. (Bug #48298)

9.3 Changes in MySQL Connector/MXJ 5.0.10 (Not released)

This was an internal only release.

Functionality Added or Changed

- The MySQL binary for Windows targets has been updated to be configurable through the `windows-mysqld-command` property. This is to handle the move in MySQL 5.1.33 from `mysqld-nt.exe` to `mysqld.exe`. The default value is `mysqld.exe`.
- The embedded MySQL has been updated to the MySQL 5.1 series. The embedded MySQL binaries have been updated to MySQL 5.1.33 for GPL releases and MySQL 5.1.34 for Commercial releases.

9.4 Changes in MySQL Connector/MXJ 5.0.9 (2008-08-19)

Functionality Added or Changed

- The timeout for kill operations in the embedded test suite has been set to a default of 10 seconds.
- The embedded MySQL binaries have been updated to MySQL 5.0.51a for GPL releases and MySQL 5.0.54 for Commercial releases.
- The utility used to kill MySQL on Windows (`kill.exe`) has been configured to be loaded from the `kill.exe` property, instead of being hard-coded. The corresponding timeout, `KILL_DELAY` has also been moved to the properties file and defaults to 5 minutes.
- The port used in the `ConnectorMXJUrlTestExample` and `ConnectorMXJObjectTestExample` port is no longer hard coded. Instead, the code uses the `x-mxj_test_port` property a default value of `3336`

9.5 Changes in MySQL Connector/MXJ 5.0.8 (2007-08-06)

Functionality Added or Changed

- The embedded MySQL binaries have been updated to MySQL 5.0.45 for GPL releases and MySQL 5.0.46 for Commercial releases.
- The embedded documentation has been updated so that it now points to the main MySQL documentation pages in the MySQL reference manual.

9.6 Changes in MySQL Connector/MXJ 5.0.7 (2007-05-27)

Functionality Added or Changed

- The embedded MySQL binaries have been updated to MySQL 5.0.41 for GPL releases and MySQL 5.0.42 for Commercial releases.
- The `PatchedStandardSocketFactory` class has been removed, because it fixed an issue in Connector/J that was corrected in Connector/J 5.0.6.
- The `ConnectorMXJUrlTestExample` and `ConnectorMXJObjectTestExammple` have been updated to include an example of initializing the user/password and creating an initial database. The `InitializePasswordExample` example class has now been removed.
- Updated the jar file name to be consistent with the Connector/J jar file name. Files are now formatted as `mysql-connector-mxj-`*`mxj-version`*.

Bugs Fixed

- Added a null-check to deal with class loaders where `getClassLoader()` returns null.

9.7 Changes in MySQL Connector/MXJ 5.0.6 (2007-05-04)

Functionality Added or Changed

- Removed references to `File.deleteOnExit`, a warning is printed instead.
- Fixed where `versionString.trim()` was ignored.
- `portFile` now contains a new-line to be consistent with `pidFile`.
- Errors reading `portFile` are now reported using `stacktrace(err)`, previously `System.err` was used.
- Added `main(String[])` to `com/mysql/management/AllTestsSuite.java`.
- Removed obsolete `PatchedStandardSocketFactory` java file.
- Clarified code in `DefaultsMap.entrySet()`.
- Removed obsolete field `SimpleMysqldDynamicMBean.lastInvocation`.
- Added new connection property `initialize-user` which, if set to `true` will remove the default, un-passworded anonymous and root users, and create the user/password from the connection url.
- `ConnectorMXJUrlTestExample` and `ConnectorMXJObjectTestExample` now demonstrate the initialization of user/password and creating the initial database (rather than using "test").
- Added new tests for initial-user & expanded some existing tests.
- Added `InitializeUser` and `QueryUtil` classes to support new feature.
- Added copyright notices to some classes which were missing them.
- Updated commercial license files.
- Updated internal jar file names to include version information and be more consistent with Connector/J jar naming. For example, `connector-mxj.jar` is now `mysql-connector-mxj-${mxj-version}.jar`.

Bugs Fixed

- Removed obsolete `InitializePasswordExample`
- Refactored duplication from tests and examples to `QueryUtil`.
- Removed `PatchedStandardSocketFactory` (fixed in Connector/J 5.0.6).
- Added robustness around reading portfile.
- Added null-check to deal with C/MXJ being loaded by the bootstrap classloaders with JVMs for which `getClassLoader()` returns null.
- Removed `use-default-architecture` property replaced.
- Updated `build.xml` in preparation for next beta build.
- Clarified the startup max wait numbers.
- Moved `platform-map.properties` into `db-files.jar`.
- Delete `portFile` on shutdown.
- Swapped out commercial binaries for v5.0.40.
- Added os.name-os.arch to resource directory mapping properties file.
- Fixed port file to always be written to datadir.
- Changed tests to shutdown mysqld prior to deleting files.

9.8 Changes in MySQL Connector/MXJ 5.0.5 (2007-03-14)

Bugs Fixed

- Removed 5.0.22 binaries from the repository.
- Swapped out gpl binaries for v5.0.37.
- Ensured 5.1.14 compatibility.
- Added Patched `StandardSocketFactory` from Connector/J 5-0 HEAD.
- Replaced windows `kill.exe` resource with re-written version specific to mysqld.
- Added 5.1.14 binaries to repository.
- Added `getDataDir()` to interface `MysqldResourceI`.
- Removed 5.1.14 binaries from the repository.
- Added 5.1.15 binaries to the repository.
- Clarified the immutability of `baseDir`, `dataDir`, `pidFile`, `portFile`.
- `SIGHUP` is replaced with `MySQLShutdown<PID>` event.
- Clarified the synchronization of `MysqldResource` methods.

- Changed `SimpleMysqldDynamicMBean` to create `MysqldResource` on demand to enable setting of `datadir`. (Rubinger bug groundwork).
- Updated `build.xml` in prep for next release.
- Added testcase to `com.mysql.management.jmx.AcceptanceTest` which demonstrates that `dataDir` is a mutable MBean property.
- In testing so far mysqld reliably shuts down cleanly much faster.
- Changed `connector-mxj.properties` default mysql version to 5.0.37.
- Replaced `Boolean.parseBoolean` with JDK 1.4 compliant `valueOf`.
- New windows `kill.exe` fixes a bug where mysqld was being force terminated. Issue reported by bruno haleblian and others, see: MySQL Forums.
- Now incorporates Reggie Bernett's `SafeTerminateProcess` and only calls the unsafe TerminateProcess as a final last resort.
- `build.xml`: `usage` now slightly more verbose; some reformatting.
- Changed protected constructor of `SimpleMysqldDynamicMBean` from taking a `MysqldResource` to taking a `MysqldFactory`, to lay groundwork for addressing BUG discovered by Andrew Rubinger. See: MySQL Forums (Actual testing with JBoss, and filing a bug, is still required.)
- Found and removed dynamic linking in mysql_kill; updated solution.
- Swapped out commercial binaries for v5.0.36.
- Reformatting: Added newlines some files which did not end in them.
- Moved `MysqldFactory` to main package.

9.9 Changes in MySQL Connector/MXJ 5.0.4 (2007-01-28)

Bugs Fixed

- Fixed test to be tolerant of `/tmp` being a symlink to `/foo/tmp`.
- Moved default version string out of java class into a text editable properties file (`connector-mxj.properties`) in the resources directory.
- Minor test robustness improvements.
- Moved mysqld binary resourced into separate jar file NOTICE: `CLASSPATH` will now need to `connector-mxj-db-files.jar`.
- Swapped out commercial binaries for v5.0.32.
- Swapped out gpl binaries for v5.0.27.
- Introduced property for Linux & WinXX to default to 32bit versions.
- Only populate the options map from the help text if specifically requested or in the MBean case.
- Allow multiple calls to start server from URL connection on non-3306 port. (Bug #24004)
- Updated `build.xml` to build to handle with different gpl and commercial mysqld version numbers.

9.10 Changes in MySQL Connector/MXJ 5.0.3 (2006-06-24)

Bugs Fixed

- Swapped out the mysqld binaries for MySQL v5.0.22.
- Removed "TeeOutputStream" - no longer needed.
- Removed unused imports, formatted code, made minor edits to tests.

9.11 Changes in MySQL Connector/MXJ 5.0.2 (2006-06-15)

Bugs Fixed

- ServerLauncherSocketFactory.shutdown API change: now takes 2 File parameters (basedir, datadir)
- ServerLauncherSocketFactory now treats URL parameters in the form of `&server.foo=null` as `serverOptionMap.put("foo", null)`
- ServerLauncherSocketFactory.shutdown(port) no longer throws, only reports to System.err
- extracted splitLines(String) to Str utility class
- ServerLauncherSocketFactory.shutdown now works across JVMs.
- moved PID file into datadir
- MysqldResource now tied to dataDir as well as basedir (API CHANGE)
- swapped out the mysqld binaries for MySQL v5.0.19
- Extended timeout for help string parsing, to avoid cases where the help text was getting prematurely flushed, and thus truncated.
- added ability to specify "mysql-version" as an url parameter
- socket is now "mysql.sock" in datadir
- ServerLauncherSocketFactory.shutdown API change: now takes File parameter (basedir) instead of port.
- Made tests more robust be deleting the /tmp/test-c.mxj directory before running tests.
- swapped out the mysqld binaries for MySQL v5.0.18
- insulated users from problems with "." in basedir
- help parsing test reflects current help options
- altered to be "basedir" rather than "port" oriented.
- reformatted code
- Added trace level logging with Aspect/J. `CLASSPATH` will now need to include `lib/aspectjrt.jar`
- Swapped out the mysqld binaries for MySQL v5.0.21
- extracted array and list printing to ListToString utility class

- "platform" directories replace spaces with underscores
- Replaced string parsing with JDBC connection attempt for determining if a mysqld is "ready for connections" `CLASSPATH` will now need to include Connector/J jar.

9.12 Changes in MySQL Connector/MXJ 5.0.1 (Not released)

This was an internal only release.

Version 5.0.1 has no changelog entries.

9.13 Changes in MySQL Connector/MXJ 5.0.0 (2005-12-09)

Bugs Fixed

- Minor refactorings for type casting and exception handling.
- Ditched "ClassUtil" (merged with Str).
- Swapped out the mysqld binaries for MySQL v5.0.16.
- Altered examples and tests to use new Connector/J 5.0 URL syntax for launching Connector/MXJ ("jdbc:mysql:mxj://")
- Minor test tweaks
- Reorganized utils into a single "Utils" collaborator.
- Removed HelpOptionsParser's need to reference a MysqldResource.