sqlite-jdbc

xerial

Clone ▾ Fork Compare

Overview Source Commits Branches Pull requests Issues **60** Wiki Downloads **7**

HTTPS ▾ https://bitbucket.org/xerial/sqlite

5 Branches · 8 Tags · 38 Forks · 150 Watchers

Owner	Taro L. Saito
Access level	Public
Type	Mercurial
Language	Java
Last updated	2014-01-22
Created	2011-11-11
Size	107.8 MB (download)

SQLite JDBC Driver

SQLite JDBC, developed by Taro L. Saito, is a library for accessing and creating SQLite database files in Java.

Our SQLiteJDBC library requires no configuration since native libraries for major OSs, including Windows, Mac OS X, Linux etc., are assembled into a single JAR (Java Archive) file. The usage is quite simple; download our sqlite-jdbc library, then append the library (JAR file) to your class path.

See the sample code.

What is different from Zentus's SQLite JDBC?

The current sqlite-jdbc implementation is based on the code of Zentus's SQLite JDBC driver (missing link). We have improved it in two ways:

- Support major operating systems by embedding native libraries of SQLite, compiled for each of them.
- Remove manual configurations

In the original version, in order to use the native version of sqlite-jdbc, users had to set a path to the native codes (dll, jnilib, so files, etc.) through the command-line arguments, e.g., `-Djava.library.path=(path to the dll, jnilib, etc.)`, or `-Dorg.sqlite.lib.path`, etc. This process was error-prone and bothersome to tell every user to set these variables. Our SQLiteJDBC library completely does away these inconveniences.

Another difference is that we are keeping this SQLiteJDBC library up-to-date to the newest version of SQLite engine, because we are one of the hottest users of this library. For example, SQLite JDBC is a core component of UTGB (University of Tokyo Genome Browser) Toolkit, which is our utility to create personalized genome browsers.

Public Discussion Forum

- Xerial Public Discussion Group
- Post bug reports or feqture requests to Issue Tracker

Usage

SQLite JDBC is a library for accessing SQLite databases through the JDBC API. For the general usage of JDBC, see JDBC Tutorial or Oracle JDBC Documentation.

1. Download sqlite-jdbc-(VERSION).jar from the download page (or by using Maven) then append this jar file into your classpath.
2. Load the JDBC driver `org.sqlite.JDBC` from your code. (see the example below)

More usage examples are available at https://bitbucket.org/xerial/sqlite-jdbc/wiki/Usage

Usage Example (Assuming `sqlite-jdbc-(VERSION).jar` is placed in the current directory)

```
> javac Sample.java
> java -classpath ".;sqlite-jdbc-(VERSION).jar" Sample   # in Windows
or
> java -classpath ".:sqlite-jdbc-(VERSION).jar" Sample   # in Mac or Linux
name = leo
id = 1
name = yui
id = 2
```

Sample.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Sample
{
  public static void main(String[] args) throws ClassNotFoundException
  {
    // load the sqlite-JDBC driver using the current class loader
    Class.forName("org.sqlite.JDBC");

    Connection connection = null;
    try
    {
      // create a database connection
      connection = DriverManager.getConnection("jdbc:sqlite:sample.db");
      Statement statement = connection.createStatement();
      statement.setQueryTimeout(30);  // set timeout to 30 sec.

      statement.executeUpdate("drop table if exists person");
      statement.executeUpdate("create table person (id integer, name string)");
      statement.executeUpdate("insert into person values(1, 'leo')");
      statement.executeUpdate("insert into person values(2, 'yui')");
      ResultSet rs = statement.executeQuery("select * from person");
      while(rs.next())
      {
        // read the result set
        System.out.println("name = " + rs.getString("name"));
        System.out.println("id = " + rs.getInt("id"));
      }
    }
    catch(SQLException e)
    {
      // if the error message is "out of memory",
      // it probably means no database file is found
      System.err.println(e.getMessage());
    }
    finally
    {
      try
      {
        if(connection != null)
          connection.close();
      }
      catch(SQLException e)
      {
        // connection close failed.
        System.err.println(e);
      }
    }
  }
}
```

How to Specify Database Files

Here is an example to select a file `C:\work\mydatabase.db` (in Windows)

```
Connection connection = DriverManager.getConnection("jdbc:sqlite:C:/work/mydatabase.db");
```

A UNIX (Linux, Mac OS X, etc) file `/home/leo/work/mydatabase.db`

```
Connection connection = DriverManager.getConnection("jdbc:sqlite:/home/leo/work/mydatabase.db");
```

How to Use Memory Databases

SQLite supports on-memory database management, which does not create any database files. To use a memory database in your Java code, get the database connection as follows:

```
Connection connection = DriverManager.getConnection("jdbc:sqlite::memory:");
```

News

- 2014 January 5th: sqlite-jdbc4-3.8.2-SNAPSHOT Introduced JDBC4 version of driver. (Requires at least Java 6).
  - Source code is on branch feature/jdbc4
- 2013 August 27th: sqlite-jdbc-3.8.0 snapshot version is available
- 2013 August 19th: sqlite-jdbc-3.7.15-M1
- 2013 March 24th : sqlite-jdbc-3.7.15-SNAPSHOT-2
- 2013 January 22nd: The repositories and documentations were moved to the bitbucket.
- 2012 December 15th: sqlite-jdbc-3.7.15-SNAPSHOT
  - Removed pure-java.
- 2010 August 27th: sqlite-jdbc-3.7.2 released
- 2010 April 3rd: beta release of sqlite-jdbc-3.6.23.1-SNAPSHOT
  - Added online backup/restore functions. Syntax: `backup to (file name)`, `restore from (file name)`.

- 2009 December 10th: sqlite-jdbc-3.6.20.1 release.
    - Read-only connection, recursive trigger, foreign key validation support etc. using SQLiteConfig class.

      ```
      SQLiteConfig config = new SQLiteConfig();
      // config.setReadOnly(true);
      config.setSharedCache(true);
      config.recursiveTriggers(true);
      // ... other configuration can be set via SQLiteConfig object
      Connection conn = DriverManager.getConnection("jdbc:sqlite:sample.db", config.toPropert
      ```
- 2009 November 12th: sqlite-jdbc-3.6.19 released.
    - added 64-bit OS support: 64-bit native SQLite binaries for Windows (x86_64), Mac (x86_64) and Linux (adm64) are available.
- 2009 August 19th: sqlite-jdbc-3.6.17.1 released.
- 2009 July 2nd: sqlite-jdbc-3.6.16 release.
- 2009 June 4th: sqlite-jdbc-3.6.14.2 released.
- 2009 May 19th: sqlite-jdbc-3.6.14.1 released.
    - This version supports "jdbc:sqlite::resource:" syntax to access read-only DB files contained in JAR archives, or external resources specified via URL, local files address etc. (see also the http://groups.google.com/group/xerial/browse_thread/thread/39acb38f99eb2469/fc6afceabeaa0f76?lnk=gst&q=resource#fc6afceabeaa0f76 detailes
- 2009 February 18th: sqlite-jdbc-3.6.11 released.
    - Fixed a bug in `PrepStmt`, which does not clear the batch contents after `executeBatch()`. Discussion.
- 2009 January 19th: sqlite-jdbc-3.6.10 released. This version is compatible with sqlite version 3.6.10. http://www.sqlite.org/releaselog/3_6_10.html
    - Added `READ_UNCOMMITTED` mode support for better query performance: (see also http://www.sqlite.org/sharedcache.html)

      ```
      // READ_UNCOMMITTED mode works only in shared_cache mode.
       Properties prop = new Properties();
       prop.setProperty("shared_cache", "true");
       Connection conn = DriverManager.getConnection("jdbc:sqlite:", prop);
       conn.setTransactionIsolation(Conn.TRANSACTION_READ_UNCOMMITTED);
      ```
- 2008 December 17th: sqlite-jdbc-3.6.7 released.
    - Related information: http://www.sqlite.org/releaselog/3_6_7.html
- 2008 December 1st: sqlite-jdbc-3.6.6.2 released,
    - Fixed a bug incorporated in the version 3.6.6 http://www.sqlite.org/releaselog/3_6_6_2.html
- 2008 November 20th: sqlite-jdbc-3.6.6 release.
    - Related information sqlite-3.6.6 changes: http://www.sqlite.org/releaselog/3_6_6.html
- 2008 November 11th: sqlite-jdbc-3.6.4.1. A bug fix release
    - Pure-java version didn't work correctly. Fixed in both 3.6.4.1 and 3.6.4. If you have already downloaded 3.6.4, please obtain the latest one on the download page.
- 2008 October 16th: sqlite-jdbc-3.6.4 released.
    - Changes from SQLite 3.6.3: http://www.sqlite.org/releaselog/3_6_4.html
    - `R*-Tree` index and `UPDATE/DELTE` syntax with `LIMIT` clause are available from this build.
- 2008 October 14th: sqlite-jdbc-3.6.3 released. Compatible with SQLite 3.6.3.
- 2008 September 18th: sqlite-jdbc-3.6.2 released. Compatible with SQLite 3.6.2 and contains pure-java and native versions.
- 2008 July 17th: sqlite-jdbc-3.6.0 released. Compatible with SQLite 3.6.0, and includes both pure-java and native versions.
- 2008 July 3rd: [sqlite-jdbc-3.5.9-universal] (http://www.xerial.org/maven/repository/artifact/org/xerial/sqlite-jdbc/3.5.9-universal) released. This version contains both native and pure-java SQLite libraries, so it probably works in any OS environment.
- 2008 May 29th: Current development revision (sqlite-jdbc-3.5.9-1) can be compiled with JDK 6. No need to use JDK 1.5 for compiling SQLiteJDBC.
- 2008 May 20th: sqlite-jdbc-3.5.9 released.
- 2008 May 20th: sqlite-jdbc-3.5.8 released (corresponding to SQLite 3.5.8 and sqlite-jdbc-v047). From this release, Windows, Mac OS X, Linux (i386, amd64) and Solaris (SunOS, sparcv9) libraries are bundled into one jar file.
- 2008 May 1st: sqlite-jdbc is now in the maven central repository! How to use SQLiteJDBC with Maven2
- 2008 Mar. 18th: sqlite-jdbc-3.5.7 released.
    - This version corresponds to SQLite 3.5.7.
- 2008 Mar. 10th: sqlite-jdbc-v042 released.
    - Corresponding to SQLite 3.5.6, which integrates FTS3 (full text search).
- 2008 Jan. 31st: sqlite-jdbc-v038.4 released.
    - SQLiteJDBCLoder.initialize() is no longer requried.
- 2008 Jan. 11th: The Jar files for Windows, Mac OS X and Linux are packed into a single Jar file! So, no longer need to use an OS-specific jar file.
- 2007 Dec. 31th: Upgraded to sqlitejdbc v038

- 2007 Dec. 31th: Upgraded to sqlitejdbc-v038

Download

Download the latest version of SQLiteJDBC from the downloads page.

Beta Release

The early releases (beta) of sqlite-jdbc with some advanced features are available from here

- The old releases are still available from here, but the site might be closed in future.

Supported Operating Systems

Since sqlite-jdbc-3.6.19, the natively compiled SQLite engines will be used for the following operating systems:

- Windows XP, Vista (Windows, x86 architecture, x86_64)
- Mac OS X 10.4 (Tiger), 10.5(Leopard), 10.6 SnowLeopard (for i386, x86_64, Intel CPU machines)
- Linux i386 (Intel), amd64 (64-bit X86 Intel processor)

In the other OSs not listed above, the pure-java SQLite is used. (Applies to versions before 3.7.15)

If you want to use the native library for your OS, [build the source from scratch.

How does SQLiteJDBC work?

Our SQLite JDBC driver package (i.e., `sqlite-jdbc-(VERSION).jar`) contains three types of native SQLite libraries (`sqlite-jdbc.dll`, `sqlite-jdbc.jnilib`, `sqlite-jdbc.so`), each of them is compiled for Windows, Mac OS and Linux. An appropriate native library file is automatically extracted into your OS's temporary folder, when your program loads `org.sqlite.JDBC` driver.

Dependency Tests

- Windows XP (32-bit)
- dependency check

  ```
  DUMPBIN /DEPENDENTS sqlitejdbc.dll
  ```

  KERNEL32.dll msvcrt.dll
- Mac OS X (10.4.10 Tiger ~ 10.5 Leopard)
- dependency check

  ```
  otool -L libsqlitejdbc.jnilib
  libsqlitejdbc.jnilib: build/Darwin-i386/libsqlitejdbc.jnilib (compatibility version 0.0.0,
  current version 0.0.0) /usr/lib/libSystem.B.dylib (compatibility version 1.0.0,
  current version 88.3.9)
  ```
- Linux (glibc-2.5.12)
- Dependency check

  ```
  ldd libsqlitejdbc.so
  linux-gate.so.1 => (0x00b45000) libc.so.6 => /lib/i686/nosegneg/libc.so.6
  (0x002dd000) /lib/ld-linux.so.2 (0x47969000)
  ```

Source Codes

- Mercurial Repository: http://bitbucket.org/xerial/sqlite-jdbc

License

This program follows the Apache License version 2.0 (http://www.apache.org/licenses/) That means:

It allows you to:

- freely download and use this software, in whole or in part, for personal, company internal, or commercial purposes;
- use this software in packages or distributions that you create.

It forbids you to:

- redistribute any piece of our originated software without proper attribution;
- use any marks owned by us in any way that might state or imply that we xerial.org endorse your distribution;
- use any marks owned by us in any way that might state or imply that you created this

software in question.

It requires you to:

- include a copy of the license in any redistribution you may make that includes this software;
- provide clear attribution to us, xerial.org for any distributions that include this software

It does not require you to:

- include the source of this software itself, or of any modifications you may have made to it, in any redistribution you may assemble that includes it;
- submit changes that you make to the software back to this software (though such feedback is encouraged).

See License FAQ http://www.apache.org/foundation/licence-FAQ.html for more details.

Using SQLiteJDBC with Maven2

If you are familiar with Maven2, add the following XML fragments into your pom.xml file. With those settings, your Maven will automatically download our SQLiteJDBC library into your local Maven repository, since our sqlite-jdbc libraries are synchronized with the Maven's central repository.

```
<dependencies>
    <dependency>
      <groupId>org.xerial</groupId>
      <artifactId>sqlite-jdbc</artifactId>
      <version>3.7.2</version>
    </dependency>
</dependencies>
```

To use snapshot/pre-release versions, add the following repository to your Maven settings: *Pre-release repository: https://oss.sonatype.org/content/repositories/releases* Snapshot repository: https://oss.sonatype.org/content/repositories/snapshots

Using SQLiteJDBC with Tomcat6 Web Server

Do not include sqlite-jdbc-(version).jar in WEB-INF/lib folder of your web application package, since multiple web applications hosted by the same Tomcat server cannot load the sqlite-jdbc native library more than once. That is the specification of JNI (Java Native Interface). You will observe `UnsatisfiedLinkError` exception with the message "no SQLite library found".

Work-around of this problem is to put `sqlite-jdbc-(version).jar` file into `(TOMCAT_HOME)/lib` direcotry, in which multiple web applications can share the same native library file (.dll, .jnilib, .so) extracted from this sqlite-jdbc jar file.

If you are using Maven for your web application, set the dependency scope as 'provided', and manually put the SQLite JDBC jar file into (TOMCAT_HOME)/lib folder.

```
<dependency>
    <groupId>org.xerial</groupId>
    <artifactId>sqlite-jdbc</artifactId>
    <version>3.7.2</version>
    <scope>provided</scope>
</dependency>
```

Recent activity

aditsu commented on issue #127 in xerial/sqlite-jdbc
yesterday

nevvermind commented on issue #127 in xerial/sqlite-jdbc
5 days ago

fabian kessler reported issue #130 to xerial/sqlite-jdbc
2014-03-11

> Connect throws exception instead of returning null

Steve Kim reported issue #129 to xerial/sqlite-jdbc
2014-03-11

> SQLite sleeps for whole second (1000 ms) ticks during backoff due to databa…

dolanp commented on issue #127 in xerial/sqlite-jdbc
2014-03-07

Dongcai Shen commented on issue #127 in xerial/sqlite-jdbc
2014-03-05

nguillaumin commented on issue #127 in xerial/sqlite-jdbc
2014-03-05

nguillaumin commented on issue #127 in xerial/sqlite-jdbc

2014-03-05

Linuxhippy commented on issue #126 in xerial/sqlite-jdbc
2014-03-01

Alain O'Dea commented on issue #80 in xerial/sqlite-jdbc
2014-02-21