Apache Derby 入門

2005 年 11 月 25 日(木)

花井 志生
http://www.ruimo.com

1.概要

Derbyは、Pure Javaのリレーショナルデータベース管理システム(RDBMS)です。元々1996年にCloudscape社でJBMSとして開発されていたもので、その後Infomix、IBMと買収された後、Apache Software Foundationにオープンソースソフトウェアとして寄贈されました。小型軽量(約2MB)でありながらも、JDBC3.0やXAトランザクションにも対応した本格的なRDBMSです。

Derbyにはサーバーモードとエンベデッドモードがあり、サーバーモードを使用すれば、商用RDBMSの置き換えとして使用する事もできないわけではありませんが、それよりもエンベデッドモードで使用して、商用RDBMSが苦手とする以下のような分野に使用するのが良いでしょう。

- DAO(Data Access Object)を単体テストしたい場合に使用する。
  RDBMSのインストール作業や、DB、テーブルの作成などがJavaから簡単に行なえ、テストが終ったら、簡単に削除できる。RDBMSを常駐させる必要もない。

- リッチクライアントなどで、ローカルに使用できる軽量RDBMSとして使用する。
  クライアント側で一時的なデータの格納、設定情報の保管などにデータベースを使用したいケースがあります。そのために商用のRDBMSを使用すると、大量のクライアントにインストールしなければなりませんし、クライアントが故障した際の復旧も面倒です。

本文書では、Derbyをエンベデッドモードで使用するための基本的な使い方について解説します。

1.1.注意

Derbyを単体テストで使用する場合、特にSQLでサポートされる機能の違い、方言やストアドプロシージャなどに注意してください。特定のRDBMS製品の固有機能を使用している場合には、Derbyでは動作しないかもしれません。こういうケースではHibernateなどの方言を吸収できるORMツールも検討してみてください。

Derbyをエンベデッドモードで使用する場合、Derbyは使用するアプリケーションと同じJVM上で動作します。ネットワークなどを介して複数ユーザー、アプリケーションから同時にアクセスすることは出来ません。

2.使用するツール

本文書では以下のツールを利用します。

名称	バージョン	概要
Derby	10.1.2.1	Pure Java RDBMS
J2SE	5.0_05	Java 実行環境
Apache Ant	1.6.5	ビルドツール
JUnit	3.8.1	単体テストツール

Derby 以外のツールについては有名なものばかりですので、特にここでは導入方法について解説しません。適宜書籍等を参照してください。

3.導入と設定

3.1.導入

Derby は http://db.apache.org/derby/index.html で配布されています。Web ブラウザでアクセスし、左側のメニューから Downloads をクリックしてください。今回は 10.1.2.1 を使用します。リンクをクリックしてください。db-derby-10.1.2.1-bin.zip をクリックしてダウンロードします。後は好きな場所に unzip 等のツールで展開するだけです。

3.2.設定

3.2.1.ライブラリ

次のライブラリを CLASSPATH に指定してください。これらは lib ディレクトリに存在します。

derby.jar
derbyLocale_ja_JP.jar

derby.jar はエンベデッドドライバの本体です。derbyLocale_ja_JP.jar はメッセージの日本語化のために必要です。日本語メッセージが不要であれば指定しなくても構いません。

設定は以上です。他の RDBMS と異なり、特にインストール作業などは不要です。なお 1.4 よりも前の 1.3 ベースの J2SE を使用する場合には、JDBC 2.0 Extensions が必要です。これは、http://java.sun.com/products/jdbc/download.html の JDBC 2.1 Core API and JDBC 2.0 Optional Package API に含まれています。

4.サンプル

データベースの作成、アクセス、シャットダウンの手順を順番に見ていきます。

4.1.ファイル構成

```
sample
|   build.xml
|
├──java
|   └──test
|          Test.java
|
└──lib
        derby.jar
        derbyLocale_ja_JP.jar
```

Test.java がサンプルプログラムです。lib ディレクトリの下に、derby.jar と derbyLocale_ja.JP.jar を予めコピーしておいてください。このファイルは derby のパッケージを展開したディレクトリの lib の下にあります。

Ant のビルドファイルが用意してあるので、ant を実行すれば、コンパイルと実行が行われます。この時、デフォルトでは c:\data にデータベースが作成されます。

```
$ ant
Buildfile: build.xml

debugBuild:
    [mkdir] Created dir: C:¥cygwin¥home¥shanai¥usr¥doc¥icm¥derby¥sample¥build
    [javac] Compiling 1 source file to C:¥cygwin¥home¥shanai¥usr¥doc¥icm¥derby¥s
ample¥build

run:
     [java] Create database... Done!
     [java] Create table... Done!
     [java] みかん    80
     [java] りんご    100
     [java] ばなな    150
     [java] Shutdown database... Done!

BUILD SUCCESSFUL
Total time: 6 seconds
```

このサンプルはデータベースとテーブルを作成して、3 つの行を挿入し、それらを照会するプログラムです。それでは中を順番に見ていきましょう。

4.2.全体構成

サンプルのメインロジックは、run()メソッドです。

```
    void run() {
        try {
            File dbpath = new File(DB_PATH);                    // (1)
```

```
            if (! dbpath.exists()) {
                createDB();
                createTable();
            }
            insertRow();                                        // (2)
            queryRow();
        }
        catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
        catch (SQLException ex) {
            ex.printStackTrace();
        }
        finally {
            try {
                closeConnection();
            }
            catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
```

最初にデータベースを格納するディレクトリが存在する事を確認しています(1)。存在しなければ、createDB()と createTable()を呼び出してデータベースとテーブルを生成しています。その後 insertRow()と queryRow()で行の挿入、照会を行い(2)、finally 節でデータベースコネクションのクローズを行っています。それでは個々のメソッドの中を見ていきましょう。

4.3.データベースの作成

データベースの作成は createDB()メソッドで行われています。

```
    void createDB() throws ClassNotFoundException, SQLException {
        System.err.print("Create database...");
        Class.forName(DRIVER_CLASS_NAME);                       // (1)
        conn = DriverManager.getConnection(DB_URL + ";create=true");     // (2)
        System.err.println(" Done!");
    }
```

通常の JDBC での手順と同様、Class.forName()でドライバを指定し(1)、DriverManager.getConnection()でコネクションを取得します(2)。この時に指定する URL ですが、

```
    static final String DB_PATH = "c:/data";
    static final String DB_URL = "jdbc:derby:" + DB_PATH + "/test";
```

jdbc:derby:の後に、この例のようにデータベースを置くパスを指定します。この例では絶対パス(c:/data/test)を指定していますが、相対パスを指定した場合にはカレントディレクトリからの相対パスになります。この他にもシステムプロパティで指定したり、クラスパスの中から指定する事も可能です。詳細は derby の開発者ガイド(docs/pdf/derbydev.pdf)を参照してください。また、(2)のように";create=true"と指定することで、データベースが存在しなければ自動的に生成する事ができます。このオプションが無い場合は、データベースが存在しなければ SQLException がスローされます。

このように、データベースシステムのインストール作業無しに、何も無いところに簡単にデータベースが作成できることが分かります。

4.4.テーブルの作成

SQLによるテーブルの作成は、他のDBMSと同じです。

```
void createTable() throws ClassNotFoundException, SQLException {
    Statement stmt = getConnection().createStatement();
    System.err.print("Create table...");
    stmt.executeUpdate("create table item(itemid varchar(10) not null, " +
                       "itemname varchar(64) not null, " +
                       "unitprice bigint, " +
                       "primary key (itemid))");
    System.err.println(" Done!");
    stmt.close();
}
```

create tableを使ってテーブルの生成をします。ここではitemid, itemname, unitpriceを列に持つテーブルを生成し、itemidをキーとして設定しています。

4.5.データの挿入

SQLによるデータの挿入も、特に他のDBMSと変わるところはありません。

```
void insertRow() throws ClassNotFoundException, SQLException {
    Statement stmt = null;
    PreparedStatement pstmt = null;

    try {
        stmt = getConnection().createStatement();
        stmt.executeUpdate("delete from item");
        stmt.close();
        pstmt = getConnection().prepareStatement("insert into item values (?, ?, ?)");
        pstmt.setString(1, "1");
        pstmt.setString(2, "りんご");
        pstmt.setInt(3, 100);
        pstmt.execute();

        pstmt.setString(1, "2");
        pstmt.setString(2, "ばなな");
        pstmt.setInt(3, 150);
        pstmt.execute();

        pstmt.setString(1, "3");
        pstmt.setString(2, "みかん");
        pstmt.setInt(3, 80);
        pstmt.execute();
        pstmt.close();
    }
    catch (SQLException ex) {
        throw ex;
    }
    finally {
        try {
```

```
            if (stmt != null) stmt.close();
        }
        catch (SQLException ex) {
            ex.printStackTrace();
        }

        try {
            if (pstmt != null) pstmt.close();
        }
        catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

この例では、最初に delete で全レコードを削除してから、3 つのレコードを insert を使って挿入しています。

4.6.レコードの照会

レコードの照会も、他の DBMS と変わりはありません。

```
void queryRow() throws ClassNotFoundException, SQLException {
    Statement stmt = null;
    ResultSet rs = null;
    try {
        stmt = getConnection().createStatement();
        rs = stmt.executeQuery("select itemname, unitprice from item order by unitprice");
        while (rs.next()) {
            System.out.println(rs.getString(1) + "    " + rs.getInt(2));
        }
    }
    finally {
        try {
            if (rs != null) rs.close();
        }
        catch (SQLException ex) {
            ex.printStackTrace();
        }

        try {
            if (stmt != null) stmt.close();
        }
        catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

商品名と単価を、単価の昇順に表示しています。特に説明を要するところはないでしょう。

4.7.データベースのシャットダウン

エンベデッドモードの場合、データベースとアプリケーションが一体となっているので、明示的にデータベースシステムを終了させる必要があります。

```
static void shutdownDB() throws SQLException {
    try {
        System.err.print("Shutdown database...");
        DriverManager.getConnection(DB_URL + ";shutdown=true");
    }
    catch (SQLException ex) {
        if (! ex.getSQLState().equals("08006")) throw ex;
    }
    System.err.println(" Done!");
}
```

シャットダウンは、コネクション取得メソッドを流用しています。パラメータに shutdown=true を付けることでシャットダウンされます。データベースのシャットダウンにあたっては、正常時も異常時も、必ず例外が送出されるという奇妙な仕様になっています。シャットダウンの際には SQLState が、08006 になっているので、それをチェックするのが良いでしょう。

スタンドアロンアプリケーションで使用する場合には、このシャットダウン処理が、アプリケーションの異常終了時にも確実に呼び出されるように、Runtime.addShutdownHook()でフックに登録すると良いでしょう。

```
Runtime.getRuntime().addShutdownHook(new Thread() {
    @Override public void run() {
        try {
            shutdownDB();
        }
        catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
});
```

これにより、正常終了時だけでなく、Ctrl+C で中断した場合や、RuntimeException などで終了した場合にも、確実にデータベースのシャットダウンが行われます。

もしもサーバーアプリケーションなどのアプリケーションコンテナ内で使用する場合には、その環境での終了処理に上記処理を記述してください。例えばサーブレットなら、destroy()メソッドが該当します。

5.まとめ

本文書では、Derby のエンベデッドモードにおける使用方法を簡単に解説しました。Derby を使えば、面倒なデータベースシステムの導入作業無しに、簡単にデータベースアプリケーションを配備できることを示すことができたと思います。

このような特性は、DAO を単体テストする際や、クライアントアプリケーションで使用する場合に大きなメリットとなるでしょう。