1, 设置环境变量.   
set DERBY\_HOME=E:\ db-derby-10.4.1.3-bin   
set path=%DERBY\_HOME%\bin;%PATH%   
  
2， 需要Derby的jar包添加到classpath环境变量中   
derby.jar - 内嵌模式   
derbyclient.jar -网络模式   
  
  
  
3， 如何在java中访问derby数据库   
  
The connection URL syntax is as follows:   
jdbc:derby:[subsubprotocol:][databaseName][;attribute=value]\*   
  
  
Subsubprotocol, which is not typically specified, determines how Derby looks for a   
database: in a directory, in a class path, or in a jar file. Subsubprotocol is one of the   
following:   
1，directory: The default. Specify this explicitly only to distinguish a database that   
might be ambiguous with one on the class path.   
2，classpath: Databases are treated as read-only databases, and all databaseNames   
must begin with at least a slash, because you specify them "relative" to the   
classpath directory. See Accessing databases from the classpath for details.   
3，jar: Databases are treated as read-only databases. DatabaseNames might require   
a leading slash, because you specify them "relative" to the jar file. See Accessing   
databases from a jar or zip file for details.   
jar requires an additional element immediately before the database name:   
(pathToArchive)   
  
  
PS：   
使用Java代码访问Derby数据库与访问其它数据库的区别如下：   
1） JDBC驱动的不同；   
2） 数据库连接URL的不同；   
3） 在访问内嵌模式数据库时，需要显示关闭数据库。   
  
下面分别实例访问内嵌模式和网络模式Derby数据库的代码   
  
  
1） 访问内嵌模式Derby数据库   
String driver = “org.apache.derby.jdbc.EmbeddedDriver”;   
String url = “jdbc:derby:firstdb;create=true”;   
Connection conn;   
try {   
Class.forName(driver);   
conn = DriverManager.getConnection(url);   
}catch(Exception e) {   
……   
}finally {   
……   
DriverManager.getConnection("jdbc:derby:;shutdown=true");   
}   
建立好连接后，其它的数据操作，如查询、更新数据都和其它数据库一样，这里不详述。有一点需要注意，通过Java应用程序访问内嵌模式Derby数据   
  
库时，应用程序有责任需要在程序结束时关闭Derby数据库，如上面代码finally中的   
DriverManager.getConnection("jdbc:derby:;shutdown=true");   
shutdown参数用于关闭Derby数据库，如果url中指定了数据库命，则只会关闭指定的数据库，而不会关闭整个Derby数据库。数据库关闭成功时，   
  
Derby会抛出一个错误码为XJ015和一个08006的异常表示关闭成功，应用程序可以不处理这两个异常。   
  
2） 访问网络模式Derby数据库   
网络模式和内嵌模式的不同出在于：   
A． 数据库连接URL的不同；   
B． 应用程序退出时无效关闭Derby数据库；   
C． 数据库驱动的不同；   
String driver = “org.apache.derby.jdbc.ClientDriver”;   
String url = “jdbc:derby: //localhost:1527/firstdb;create=true”;   
Connection conn;   
try {   
Class.forName(driver);   
conn = DriverManager.getConnection(url);   
}catch(Exception e) {   
……   
}   
由于网络模式下，Derby数据库做为一个独立运行的数据库，可以被多个应用程序所访问，所以应用程序在运行结束时不应该关闭Derby数据库。   
  
PS：   
e. Edit the DEFINE VARIABLES SECTION of the program so that the   
driver variable contains the name of the Derby client driver class and the   
connectionURL variable contains the hostname and port number of the   
Network Server.   
  
  
Original definitions   
String driver = "org.apache.derby.jdbc.EmbeddedDriver";   
String dbName="jdbcDemoDB";   
String connectionURL = "jdbc:derby:" + dbName +   
";create=true";   
  
New definitions   
String driver = "org.apache.derby.jdbc.ClientDriver";   
...   
String connectionURL = "jdbc:derby://localhost:1527/" +   
dbName + ";create=true";   
  
  
PS：   
BOOT DATABASE SECTION:   
The DriverManager class loads the database using   
the Derby connection URL stored in the variable connectionURL. This URL includes   
the parameter ;create=true so that the database will be created if it does not already   
exist. The primary try and catch block begins here. This construct handles errors for   
the database access code   
  
  
4， derby.properties   
The text file derby.properties contains the definition of properties, or configuration   
parameters that are valid for the entire system.   
derby.properties should be placed in the directory specified by the derby.system.home properties.   
  
Here is a sample excerpt from the derby.properties file:   
# Users definition   
#   
derby.user.sa=derbypass   
derby.user.mary=little7xylamb   
derby.drda.startNetworkServer=true   
  
  
4.1   
derby.system.home   
Function   
Specifies the Derby system directory, which is the directory that contains subdirectories   
holding databases that you create and the text file derby.properties.   
If the system directory that you specify with derby.system.home does not exist at startup,   
Derby creates the directory automatically.   
  
  
Default   
Current directory (the value of the JVM system property user.dir).   
If you do not explicitly set the derby.system.home property when starting Derby, the   
default is the directory in which Derby was started.   
Note: You should always explicitly set the value of derby.system.home.   
Example   
-Dderby.system.home=C:\derby   
Dynamic or static   
This property is static; if you change it while Derby is running, the change does not take   
effect until you reboot.   
  
  
Set in DOS   
java -Dderby.system.home=C:\home\Derby\   
-Dderby.storage.pageSize=8192 JDBCTest   
You start up your application, being sure to set the derby.system.home property   
appropriately:   
java -Dderby.system.home=c:\system\_directory MyApp   
  
Example   
-Dderby.system.home=C:\derby   
  
  
  
Set in window   
The following example sets derby.system.home on Windows.   
Properties p = System.getProperties();   
p.put("derby.system.home", "C:\databases\sample");   
  
  
There should be one derby.properties file per system, not per database.   
  
4.2 derby数据库的位置   
\* The directory derbyDB will be created under the directory that   
\* the system property derby.system.home points to, or the current   
\* directory (user.dir) if derby.system.home is not set.   
  
  
5， 加密   
  
二次加密： jdbc:derby:newDB;create=true;dataEncryption=true;   
encryptionAlgorithm=DES/CBC/NoPadding;encryptionKey=6162636465666768   
  
三次加密：jdbc:derby:encryptedDB;create=true;dataEncryption=true;   
encryptionProvider=com.sun.crypto.provider.SunJCE;   
encryptionAlgorithm=DESede/CBC/NoPadding;   
bootPassword=cLo4u922sc23aPe   
  
  
Encrypting a database when you create it   
If your environment is configured properly, you can create your database as an encrypted   
database (one in which the database is encrypted on disk). To do this, you use the   
dataEncryption=true attribute to turn on encryption and the bootPassword=key   
attribute or the encryptionKey attribute to specify a key for the encryption.   
You can also specify an encryption provider and encryption algorithm other   
than the defaults with the encryptionProvider=providerName and   
encryptionAlgorithm=algorithm attributes.   
  
jdbc:derby:encryptedDB;create=true;dataEncryption=true;   
bootPassword=DBpassword   
  
Creating an encrypted database with an external key   
  
  
Derby supplies or supports the following optional security mechanisms:   
1， User authentication   
Derby verifies user names and passwords before permitting them access to the   
Derby system.   
2， User authorization   
A means of granting specific users permission to read a database or to write to a   
database.   
3， Disk encryption   
A means of encrypting Derby data stored on disk.   
  
4， Validation of Certificate for Signed Jar Files   
In a Java 2 environment, Derby validates certificates for classes loaded from signed   
jar files.   
5， Network encryption and authentication   
Derby network traffic may be encrypted with SSL/TLS. SSL/TLS certificate   
authentication is also supported. See "Network encryption and authentication with   
SSL/TLS" in the Java DB Server and Administration Guide for details.   
The following figure shows some of the Derby security mechanisms at work   
  
  
======================== Refer ============================   
  
Getting Started with Java DB   
Describes how to install and configure Java DB. Includes a self-study tutorial for   
users new to Java DB and a quick-start guide for experienced JDBC users. This   
guide introduces the dblook, ij, and sysinfo tools, and the libraries and scripts   
that are included with Java DB.   
Java DB Developer's Guide   
Describes the functionality and features of Java DB common to all deployments, such   
as Java DB's JDBC and SQL specifics, deploying Java DB applications, security, and   
other advanced features.   
Java DB Reference Manual   
Documents the implementation of the SQL language in Java DB. This guide provides   
reference information about the JDBC and JTA implementations, keywords, system   
tables, properties, and SQLExceptions in Java DB.   
Tuning Java DB   
Explains how to configure and tune Java DB through properties and provides   
reference information on properties. This guide also offers performance tips, an   
in-depth discussion of performance, and information about the Java DB optimizer.   
Java DB Tools and Utilities Guide   
Describes how to use the the Java DB tools such as dblook, ij, andsysinfo, and   
how to use the system procedures to import and export data, and how to store Java   
code in the database.   
Java DB Server and Administration Guide   
Part One of this guide discusses configuring servers, how to program client   
applications, and database administration. In addition, some systems might require   
administrative tasks such as backing up databases. These tasks are independent of   
any server framework but are unique to multi-user or large systems.   
Part Two of this guide discusses administrative issues such as backups and   
debugging deadlocks.   
Java DB API Reference   
An API Reference that is automatically generated for all public Java DB classes.   
No reference is provided for the JDBC API, which is part of the Java 2 Platform,   
Standard Edition. For more information about the classes in the API, see the Java DB   
Reference Manual.   
You can access the Java DB manuals and API Reference from the Java DB   
Documentation page on the Java DB web site. The product documentation is also   
installed with Java DB. The manuals are installed in the docs subdirectory and the API   
Reference is installed in the javadoc subdirectory.   
  
get from javaDB docs...   
  
==================== Refer 2 ==================================   
public void execute() throws InstantiationException, IllegalAccessException, ClassNotFoundException, SQLException {   
  
Statement stmt = null;   
//specify the derby.system.home variable   
Properties p = System.getProperties();   
p.put("derby.system.home",hp.getConfig("JavaDB.home"));   
  
  
try {   
Connection conn = null;   
ArrayList statements = new ArrayList(); // list of Statements, PreparedStatements   
PreparedStatement psInsert = null;   
PreparedStatement psUpdate = null;   
Statement s = null;   
ResultSet rs = null;   
  
String protocol = "jdbc:derby:";   
String driver = "org.apache.derby.jdbc.EmbeddedDriver";   
  
// String protocol = "jdbc:derby://localhost:1527/";   
// String driver = "org.apache.derby.jdbc.ClientDriver";   
  
//String dbName = "DB2EToJavaDB";   
String dbName = DBHelper.getConfig("NewJavaDB.DatabaseName");   
String username = DBHelper.getConfig("NewJavaDB.username");   
String password = DBHelper.getConfig("NewJavaDB.password");   
  
Class.forName(driver).newInstance();   
try {   
conn = DriverManager.getConnection(protocol + dbName   
//, props);   
+ ";user="+username+";create=true;" +   
"dataEncryption=true;" +   
"encryptionAlgorithm=DESede/CBC/NoPadding;" +   
"encryptionProvider=com.ibm.crypto.provider.IBMJCE;"+   
//"encryptionKey=GYIAUE3xcZl9N9zOsWf3Bw==");   
"bootPassword="+password);//GYIAUE3xcZl9N9zOsWf3Bw==   
//"bootPassword=GYIAUE3xcZl9N9zOsWf3Bw==");   
} catch (   
Exception e) {   
// TODO Auto-generated catch block   
logger.printMsg(StartMigrate2.getExceptionTrace(e),"RED");   
}   
stmt = conn.createStatement();