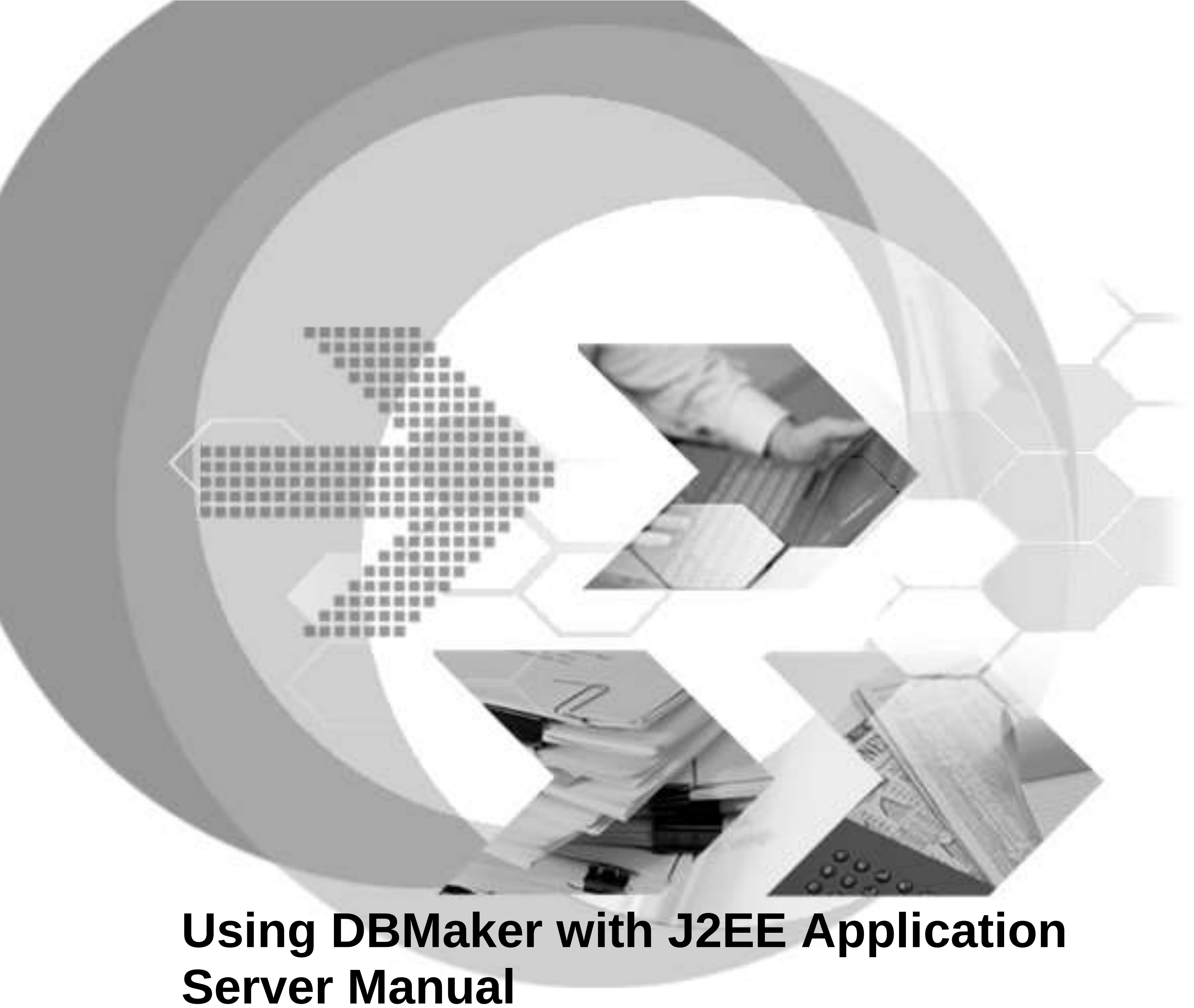


Using DBMaker with J2EE Application Server Manual

Version: 01.01

Document No: 54/DBM54-T12182016-12-DBJT
Author: DBMaker Support Team, SYSCOM Computer Engineering CO.
Publication Date: Feb 6, 2017

凌羣電腦
THE SYSCOM GROUP

Table of Content

1. Introduction

DBMaker is a powerful and flexible SQL Database Management System (DBMS) that supports an interactive Structured Query Language (SQL), a Microsoft Open Database Connectivity (ODBC) compatible interface, and Embedded SQL for C (ESQL/C). DBMaker also supports a Java Database Connectivity compliant interface and DBMaker COBOL interface (DCI). The unique open architecture and native ODBC interface give the user freedom to build custom applications using a wide variety of programming tools, or query a database using existing ODBC, JDBC, or DCI compliant applications.

From basic architectures of JDBC and AP Server,take Tomcat as the sample AP Server, we introduce Architectures that AP Server accesses DBMaker via JDBC on one machine and different machines. If you understand these two kinds architectures that DBMaker used, you will make your AP Server access DBMaker easily and do not make too much settings.

1.1 Additional Resources

DBMaster provides a complete set of DBMS manuals in addition to this one. For more detailed information on a particular subject, consult one of the books listed below:

- For an introduction to DBMaster's capabilities and functions, refer to the DBMaster *Tutorial*.
- For more information on designing, administering, and maintaining a DBMaster database, refers to the *Database Administrator's Guide*.
- For more information on the SQL language used in dmSQL, refer to the *SQL Command and Function Reference* manual.
- For more information on the UNLOAD/LOAD programming, refer to the *SQL Command and Function Reference* manual – dmSQL commands.
- For more information on stored procedure, refer to the *Stored Procedure User's Guide*.
- Document Conventions

This book uses a standard set of typographical conventions for clarity and ease of use. The NOTE, Procedure, Example, and Command Line conventions also have a second setting used with indentation.

CONVENTION	DESCRIPTION
Italics	Italics indicate placeholders for information that must be supplied, such as user and table names. The word in italics should not be typed, but is replaced by the actual name. Italics also introduce new words, and are occasionally used for emphasis in text.
Boldface	Boldface indicates filenames, database names, table names, column names, user names, and other database schema objects. It is also used to emphasize menu commands in procedural steps.
KEYWORDS	All keywords used by the SQL language appear in uppercase when used in normal paragraph text.
SMALL CAPS	Small capital letters indicate keys on the keyboard. A plus sign (+) between two key names indicates to hold down the first key while pressing the second. A comma (,) between two key names indicates to release the first key before pressing the second key.
NOTE	Contains important information.
Procedure	Indicates that procedural steps or sequential items will follow. Many tasks are described using this format to provide a logical sequence of steps for the user to follow
Example	Examples are given to clarify descriptions, and commonly include text, as it will appear on the screen. Other forms of this convention include Prototype and Syntax.
`Command Line`	Indicates text, as it should appear on a text-delimited screen. This format is commonly used to show input and output for dmSQL commands or the content in the dmconfig.ini file

Table 1-1Document Conventions

1.2 Technical Support

DBMaster provides thirty days of complimentary email and phone support during the evaluation period. When software is registered an additional thirty days of support will be included. Thus extend the total support period for software to sixty days. However, DBMaster will continue to provide email support for any bugs reported after the complimentary support or registered support has expired (free of charges).

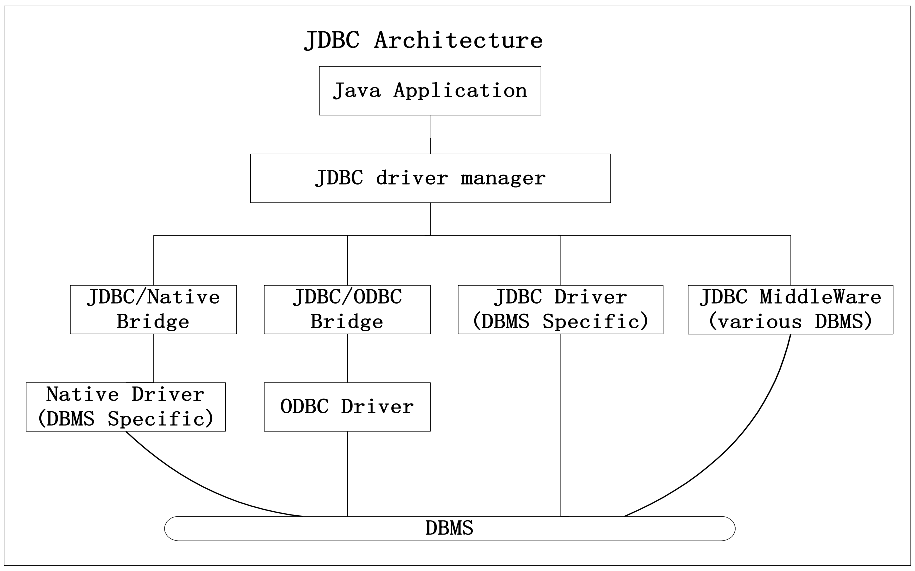
Additional support is available beyond the sixty days for most products and may be purchased for twenty percent of the retail price of the product. Please contact sales@casemaker.com for more details and prices.

DBMaster support contact information for your area (by snail mail, phone, or email) can be located at: www.casemaker.com/support. It is recommended that the current database of FAQ's be searched before contacting DBMaster support staff.

Please have the following information available when phoning support for a troubleshooting enquiry or include the information with a snail mail or email enquiry:

- Product name and version number
- Registration number
- Registered customer name and address
- Supplier/distributor where product was purchased
- Platform and computer system configuration
- Specific action(s) performed before error(s) occurred
- Error message and number, if any
- Any additional information deemed pertinent

2. About JDBC Architecture



2.1 What's JDBC

JDBC is a short for Java Data Base Connectivity and does't derive from Microsoft ODBC (Open Data Base Connectivity) specification. JDBC is Java API for connecting programs written in Java to the data in relational database, but ODBC is C interface.

JDBC and ODBC base on X or open SQL Command Layer Interface (CLI). Generally, JDBC application has two layers:One is application- layer,developers can use API to access databse or get results via SQL. Another one is driver-layer, it processes communications of specific drivers.

JDBC provides a standard API for tool or database developers and makes it possible to write database applications using a pure Java API. The standard defined by Sun Microsystems, allowing individual providers to implement and extend the standard with their own JDBC drivers. And JDBC still consists of a set of classes and interfaces written in the Java programming language.

Each JDBC application (or applet) has one JDBC driver at least and only for one kind of DBMS. But these drivers don't need connect to database directly.

2.2 Types for JDBC Driver

2.2.1 JDBC-ODBC BRIDGE

As usual, it only can run on Microsoft Windows Systems. Because ODBC is a very popular standard interface for DBMS,there have been a lot of available ODBC drivers can exchange data with database.

Generally speaking,don’t suggest using this method,but developer can do some enterprise development without great trouble.

2.2.2 JDBC NATIVE BRIDGE

JDBC-Native provides a JDBC interface on local database drivers. Drivers for JDBC change standard call for JDBC into local call for database API.

2.2.3 JDBC-NETWORK BRIDGE

JDBC-Network doesn’t need database drivers of client, it accesses a database via network-server layer.

This is a network protocol and whole Java technology performance driver.

This kind of drviers can run on different platforms,and don’t need installation and management on client. Therefore it’s very fit for Internet application.

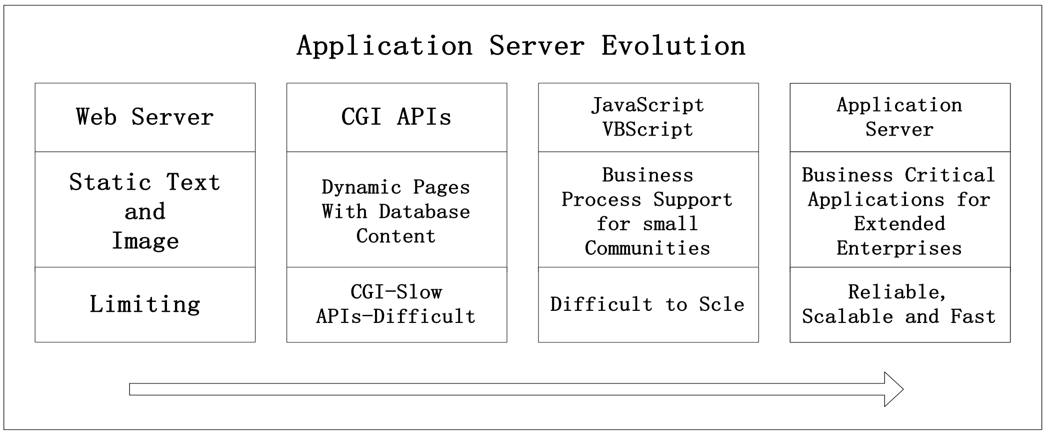
2.2.4 PURE JAVA JDBC DRIVER

Pure Java Drivers run on client and access database directly. Therefore, run this mode will need a two-tier architecture. Via EJB including codes for accessing database, and EJB provides his clients with service which isn’t related to database, so that you can use this kind of drivers in a n-tier architecture.

This is local protocol and whole Java technology performance driver, and provides Java application with mechanism for running JDBC call.

3. About AP Server

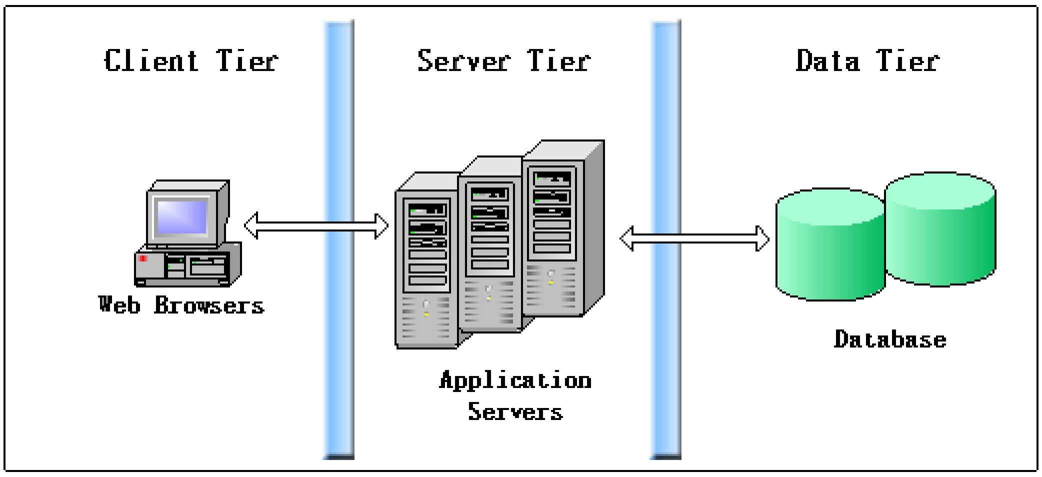
3.1 Application Server Evolution



Application Server Evolution

Web Server	CGI APIs	JavaScript VBScript	Application Server
Static Text and Image	Dynamic Pages With Database Content	Business Process Support for small Communities	Business Critical Applications for Extended Enterprises
Limiting	CGI-Slow APIs-Difficult	Difficult to Scle	Reliable, Scalable and Fast

3.2 A typical three tiers model



3.3 Types of AP Server

3.3.1 Pure-Play Web Application Server

It is independent with developing tools, supports some standard component models such as CORBA and EJB, and is in charge of extension and implement of connection for different databases.

3.3.2 DEVELOP-AND-DEPLOY SERVER

It integrates develop and deploy environment with application server tightly, and allows developers to develop application system and deploy the system rapidly.

3.3.3 APPLICATION SERVERS FROM CLIENT/SERVER VENDORS

3.4 What AP Sever does

- Security Services
- Satate and Session Management
- Load Banlancing and Fail-Over
- Business and Processing Logic
- Rich-Client Access to Server Componets
- HTML Generation
- Data Access
- Transaction Management
- Connection Pooling
- Thread Pooling and Instance Pooling

4. About Tomcat

The Tomcat server is a Java-based Web Application container that was created to run Servlet and Java Server Page web applications. It has become the reference implementation for both the Servlet and JSP specifications.

4.1 Tomcat Architecture

A web browser makes a request to server. And Tomcat receives this request through its built-in web server on port 8080 (standalone), or from a web server connector, a piece of native code attached to the web server forwarding requests to the servlet engine (embedded). If a static resource, such as a file, is requested, the DefaultServlet will open the file, read it, and send it back to the client. If a servlet is requested, the InvokerServlet will execute the servlet.

The Tomcat architecture has made it very easy: to focus on writing the servlet pieces needed to perform work, you don't need to be concerned about loading and unloading a servlet,handling the HTTP protocol, performing chaining, invoking filters, and so on.

4.2 How to configure Tomcat

After installation, the first job is to set some environment variables for Tomcat. One is CATALINA_HOME, another is CATALINA_BASE, and one value for them is path that Tomcat is installed in. Then modify the environment values CLASSPATH, and append '%CATALINA_HOME%\commond\lib\servlet.jar'.

Startup Tomcat, it means that you've setup Tomcat successfully if you will see welcome page via a web browser.

5. How Tomcat work with DB

After install and configure Tomcat, you can use JDBC to access DBMaker.

There is an important file for Linux and Windows: dmjdbc30.jar. There are some classes in this Jar file, you can access DBMaker via functions provided by these classes. For Linux, there still are some important .so files.

5.1 Set environment values

In Linux, firstly make sure that JDK is installed on your machine and have already set some environment variables about JDK such as PATH and CLASSPATH. The next, set environment variable CLASSPATH "/DBMaker_Home/lib/so" into your shell initialization file, "/DBMaker_Home" stands for dbmaker home directory. For sh you should add some lines in .profile, for bash you should add in .bashrc, these lines are as below:

```
LD_LIBRARY_PATH=/DBMAKER_HOME/lib/so:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
```

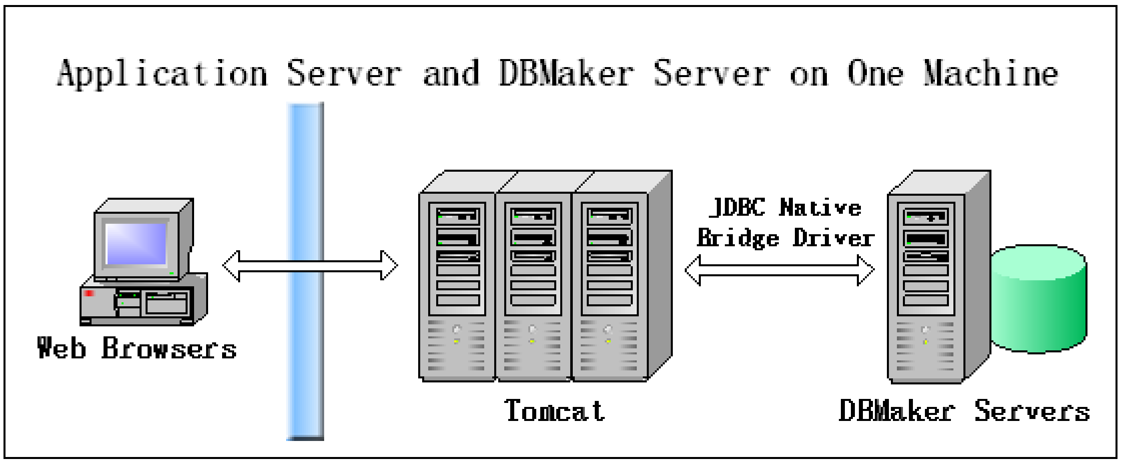
In Windows, you can set java.library.path as below in startup.bat

```
SET PATH=C:\DBMaker\5.4\bin;%PATH%
```

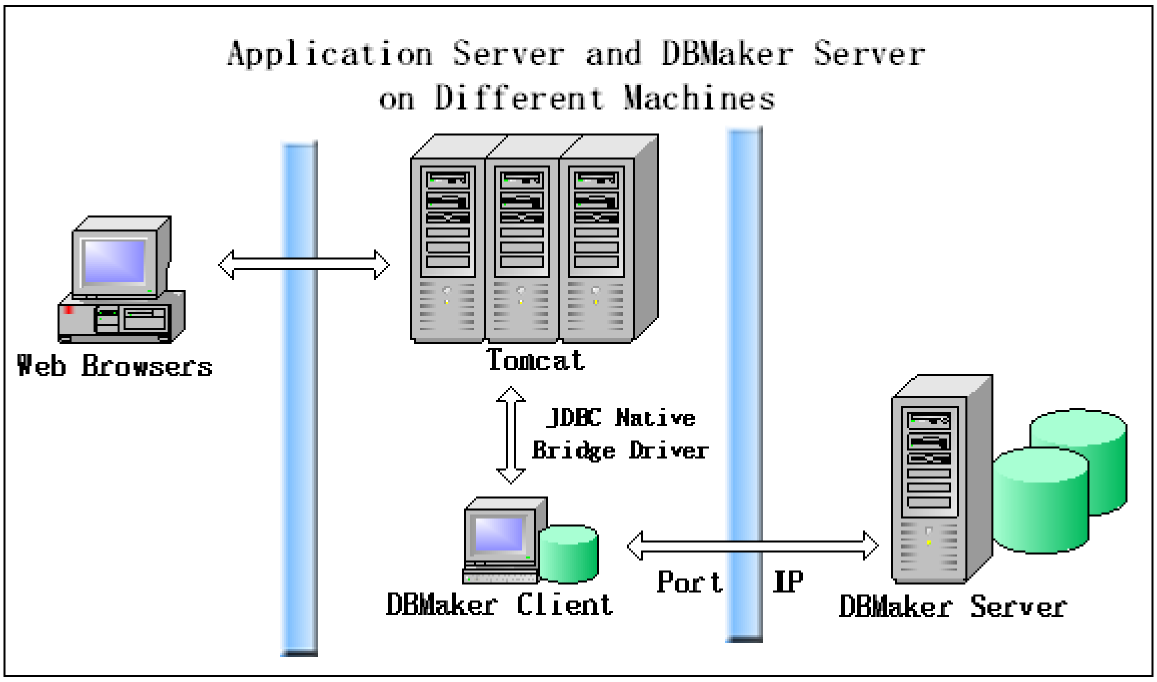
5.2 DB setting in dmconfig.ini

Generally speaking, use the function getConnection of DriverManager via parameter and the paramter must be database URL. This URL stands for database which you want to connect, and database connection program.

If application server and DBMaker server on the same machine, you should not make any special settings for your program or DBMaker server.See the following figure


Application Server and DBMaker Server on One Machine
Web Browsers
JDBC Native Bridge Driver
Tomcat
DBMaker Servers

Because we only support JDBC Native Bridge, if your application server and DBMaker server are installed on different machines, you cannot connect DBMaker via URL. How to connect, look at the following figure


Application Server and DBMaker Server on Different Machines
Web Browsers
Tomcat
JDBC Native Bridge Driver
DBMaker Client
Port
IP
DBMaker Server

Before connect DBMaker server, you should install a DBMaker client on application server at first. After installation, make some special settings in dmconfig.ini on DBMaker server and client.

On DBMaker Client-end, you should append some lines as below:

```
[DBName]
DB_SvAdr=xxx.xxx.xxx.xxx
DB_PtNum=xxxx
```

DBName is database name which you want to connect, DB_SvAdr is IP address of DBMaker server,DB_PtNum is port for the communications of DBMaker server and client.

On DBMaker server-end, you should modify or append some lines as below:

```
[DBName]
DB_SvAdr=xxx.xxx.xxx.xxx
DB_PtNum=xxxx
```

Make sure that DB_SvAdr is DBMaker server's IP address not local address 127.0.0.1, and DB_PtNum is the same with DBMaker client's.

Application server only knows DBMaker client via JDBC-Native Bridge Driver and doesn't know anything about DBMaker server. DBMaker client links DBMaker server with port and IP, so application server can exchange data with DBMaker server indirectly.

5.3 Connect DBMaker via JDBC-Native Bridge

While you access DBMaker via JDBC-Native Bridge Driver, firstly you should register this driver. How to register JDBC-Native Bridge, look at the following example:

```
   Class.forName("dbmaker.sql.jdbcOdbcDriver").newInstance();
```

Sencondly connect database, it's a very important step, not only for connecting DBMaker,but also for understanding the architecture of DBMaker very well. The syntax to connect to DBMaker is as below

```
DriverManger.getConnection ("jdbc:dbmaker:DSN",User,Password);
```

DSN is Data Source Name,User is login account,and Password is login Password.

Using those two steps, you should be able to retrieve and update your database.

6. Connect DB via DataSource

6.1 Install and configure Tomcat

Install Tomcat and configure Tomcat as part 3, and set environments as part 4.

6.2 Copy dmjdbc30.jar to lib directory

Copy dmjdbc30.jar to the Tomcat install directly, e.g. D:\apache-tomcat-8.0.39\lib.

6.3 Add following setting in conf\context.xml

```
<Resource name="jdbc/dbsample5"
auth="Container"
type="javax.sql.DataSource"
username="sysadm"
password=""
driverClassName="dbmaker.sql.JdbcOdbcDriver"
url="jdbc:dbmaker:dbsample5"
validationQuery="select 1"
/>
```

6.4 Start DB and Create tables for testing

Start dbsample5 and Create table/Insert data with below SQL statements.

```
dmSQL> create table test(id int, fname char(20),lname char(20),title varchar(50));
dmSQL> insert into test values(100,'my','test','mytest');
```

6.5 Testing with http://localhost/testDatasource.jsp

```
<%@ page contentType="text/html;charset=GBK"
import="javax.naming.,javax.sql.DataSource,java.sql."
%>
<%try{
InitialContext ctx = new InitialContext();
DataSource ds = (DataSource)ctx.lookup("java:comp/env/jdbc/dbsample5");
Connection conn = ds.getConnection();
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select * from test");
while(rs.next()){out.println(rs.getString("ID") + ". " +rs.getString("FNAME") + "," +
rs.getString("LNAME")+ " - " + rs.getString("TITLE"));
}
}catch(Exception ex){
out.println(ex.getMessage());}
%>
```

7. Setting and samples for Hibernate

7.1 Install Hibernate

Download and unzip hibernate package.

For example:

D:\hibernate-release-5.2.5.Final

7.2 Create Java Project

Create Java Project, and some java samples as following.

7.2.1 CREATE USERS TABLE IN DATABASE

Connect to DB and create a table for running hibernate sample.

```
dmSQL> create table users (id int, username char(30), password char(30));
```

7.2.2 USERS.JAVA

Add some get and set methods for mapping table Users (id,username,password)

```
package dbmaker.test;
public class Users {
   private Integer id;
   private String username;
   private String password;
   public Integer getId() {
      return id;
   }
   public void setId(Integer id) {
      this.id = id;
   }
   public String getUsername() {
      return username;
   }
   public void setUsername(String username) {
      this.username = username;
   }
   public String getPassword() {
      return password;
   }
   public void setPassword(String password) {
      this.password = password;
   }
}
```

7.2.3 USERS.HBM.XML

Add Users.hbm.xml for Users.java.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
 "http://hibernate.sourceforge.NET/hibernate-mapping-3.0.dtd">
 <hibernate-mapping>
   <class name="dbmaker.test.Users" table="Users">
      <id name="id">
         <generator class="assigned" />
      </id>
      <property name="username" />
      <property name="password" />
   </class>
</hibernate-mapping>
```

7.2.4 HIBERNATE.CFG.XML

Add hibernate.cfg.xml for URL/connection string and so on.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
       "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
       "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
     <property name="show sql">True</property>
     <property
name="hibernate.connection.driver class">dbmaker.sql.JdbcOdbcDriver</property>
     <property name="hibernate.connection.url">jdbc:dbmaker:DBSAMPLE5</property>
     <property name="hibernate.connection.username">sysadm</property>
       <property name="hibernate.dialect">org.hibernate.dialect.DBMakerDialect</property>
     <mapping resource="dbmaker/test/Users.hbm.xml" />
  </session-factory>
</hibernate-configuration>
```

7.2.5 TEST.JAVA

Add main function in Test.java for testing DB with Hibernate.

```
import org.hibernate.*;
import org.hibernate.cfg.Configuration;
public class Test {
  public static void main(String[] args) {
      try {
           org.hibernate.SessionFactory sf = new Configuration().configure()
                    .buildSessionFactory();
           Session session = sf.openSession();
           Transaction tx = session.beginTransaction();
           Users user = new Users();
           user.setId(1);
           user.setUsername("name1");
```

```
            user.setPassword("pass1");
            session.save(user);
            tx.commit();
            session.close();
            sf.close();
        } catch (HibernateException e) {
            e.printStackTrace();
        }
    }
}
```

7.3 Add library for Java Project

You must add some required library jar for building and running this test case.

7.3.1 REQUIRED LIBRARY IN HIBERNATE

Add all jar library from D:\hibernate-release-5.2.5.Final\lib\required in CLASSPATH.

7.3.2 DB JAVA DRIVER

Add dmjdbc30.jar from C:\DBMaker\5.4\bin in CLASSPATH.

7.4 Prepare Dialect for DBMaker

Create directory org\hibernate\dialect\ and copy DBMakerDialect.java in it, otherwise, you will get the ERROR:

-- *Unable to resolve name [DBMakerDialect] as strategy [org.hibernate.dialect.Dialect]*

Note: You can get DBMakerDialect.java from DB Support Team.

7.5 Build and run Test.java

After building, if run it successfully, you can check the inserted records as following:

```
dmSQL> select * from users;

   ID            USERNAME                PASSWORD
=========== ============================== ==============================
          1 name1                          pass1

1 rows selected
```