Java Database Connectivity Using MySQL: A Tutorial

Article · December 2017

CITATIONS
0

READS
15,462

1 author:

Richard Johnson
Missouri State University
18 PUBLICATIONS **297** CITATIONS

SEE PROFILE

VOLUME 7 NUMBER 1 June 2018
ISSN 2304-7712 (Print)
ISSN 2304-7720 (Online)

International Journal of Advanced Engineering and Science

ELITE HALL PUBLISHING HOUSE

ELITE HALL PUBLISHING HOUSE

International Journal of Advanced Engineering and Science

ABOUT JOURNAL

The International Journal of Advanced Engineering and Science (Int. j. adv. eng. sci. / IJAES) was first published in 2012, and is published semi-annually (May and November). IJAES is indexed and abstracted in: **ProQuest, Ulrich's Periodicals Directory, EBSCO Open Access Journals, Scientific Indexing Service, getCITED, ResearchBib, IndexCopernicus, NewJour, Electronic Journals Library, Directory of Research Journals Indexing, Open J-Gate, CiteFactor, JournalSeek, WZB Berlin Social Science Center, GEOMAR Library Ocean Research Information Access**. Since 2013, the IJAES has been included into the ProQuest one of the leading full-text databases around the world.

The International Journal of Advanced Engineering and Science is an open access peer-reviewed international journal for scientists and engineers involved in research to publish high quality and refereed papers. Papers reporting original research or extended versions of already published conference/journal papers are all welcome. Papers for publication are selected through peer review to ensure originality, relevance, and readability.

ISSN 2304-7712

International Journal of Advanced Engineering and Science

Publisher: Elite Hall Publishing House

Editor in Chief:

Editorial Board:

Web: http://ijaes.elitehall.com/
ISSN 2304-7712 (Print)
ISSN 2304-7720 (Online)

JAVA DATABASE CONNECTIVITY USING MYSQL: A TUTORIAL

Richard A. Johnson
Professor of Management and Information Technology
Missouri State University
901 S. National Ave.
Springfield, MO 65897
richardjohnson@missouristate.edu
Office: 417 836-6685
Fax: 417 836-6907

BIOGRAPHY

Dr. Richard A. Johnson was a systems analyst and engineer for several Fortune 500 companies before earning his doctorate in Computer Information Systems and Quantitative Analysis at the University of Arkansas (1998). For the past twenty years he has been teaching Java programming, database management systems, networking, cybersecurity, and management information systems at Missouri State University in Springfield, MO. He has authored several computer programming texts as well as articles in *The Data Base for Advances in Information Systems*, *IEEE Transactions*, and *Communications of the ACM*.

ABSTRACT

Java is the most popular programming language world-wide, used in everything from web servers to Blu-Ray to Android. Additionally, relational database processing is a mission-critical activity for all types of organizations. However, educators and students often struggle to link computer programs with databases in an efficient and rewarding manner. This tutorial presents a straightforward framework for connecting a Java program with MySQL, a popular open-source relational database management system. Three popular tools will be used: Command Prompt, TextPad, and Eclipse. A very rudimentary knowledge of Java is assumed, but no knowledge of these tools or SQL is required for this tutorial. Upon completion of this tutorial the student will understand how to use these tools to connect a simple Java program to a MySQL database. Then the student will be equipped to continue a study of SQL and Java graphical user interfaces toward the development of sophisticated Java database applications.
KEYWORDS: database, Java, programming, SQL, MySQL, TextPad, Eclipse

INTRODUCTION

The purpose of this tutorial is to present a straightforward approach for connecting a Java program to a relational database using the MySQL relational database management system. The Java language was selected because it is the most popular programming language in both the classroom and workplace ("TIOBE Software", 2017). For example, a keyword search of "Java" on the job search site indeed.com returns about 60,000 positions while the same search for "Python" returns 50,000, "C++" 30,000 and "C#" 26,000. Likewise, database programming is a vital skill owing to the overwhelming prevalence and

ISSN 2304-7712

criticality of databases in all facets of business, science, government, and education.

Upon surveying several popular Java programming texts (Liang, 2015; Savitch & Mock, 2016; Deitel & Deitel, 2018; Horstman, 2015; Gaddis, 2016; Farrell, 2016), one finds that early chapters usually include these basic topics: data types, variables, conditions, loops, methods, arrays, and object-orientation. Later chapters delve into more challenging topics such as inheritance, polymorphism, exception handling, string processing, file input/output, and graphical user interfaces (GUI's). Some ambitious Java texts may include data structures, collections, recursion, generics, multithreading, graphics, and database programming (the subject of this tutorial).

It is evident that this is a formidable list of potential topics for a two-semester sequence in Java programming. Some topics are often ignored due to complexity and/or lack of time—regrettably database programming is often one of the overlooked topics. Some leading texts don't even include database programming, or perhaps relegate it to an appendix or a companion web site. This is unfortunate for at least three reasons: (1) database programming is vital given the criticality of database processing in organizations; (2) database programming has a strong connection with the topic of GUI's, which is usually taught in Java courses; and (3) students receive a great deal of satisfaction and motivation by getting GUI's and databases to work together in Java applications, thus encouraging further study in the field.

One reason instructors often bypass Java database programming is the difficulty that students encounter when installing database software or configuring database software for use with Java programming tools. There are primarily two relational database products discussed in most Java texts: (1) Java DB ("Java DB Technical Documentation", 2017), which is Oracle's Apache Derby ("Apache Derby", 2017) open-source database that is included with the Java JDK distribution; and (2) MySQL ("MySQL", 2017), a popular open-source relational database management system. Another viable option for Java database programming is SQLite ("SQLite.", 2017), an extremely simple open-source database engine. Using SQLite for Java database programming is explained by Johnson (2014), and using Java DB for Java database programming is also explained by Johnson (2016).

The purpose of this tutorial is to provide a brief introduction to Java database programming using the MySQL database management system. The specific steps provided here are for a Windows 10 PC, although the tutorial should provide a basis for application to other platforms. Further in-depth study of Java database programming would consist of (1) learning to execute a wide variety of SQL commands within a Java application that accessed a MySQL database and (2) learning to create a fully functional GUI-based Java database application that interacts with a MySQL database.

INSTALLING THE JAVA JDK

You may already have the Java JDK installed on your computer. If not, or if you need to upgrade to the most recent version, follow these steps ("Java SE Downloads", n.d.):

1. Go to http://www.oracle.com/technetwork/java/javase/downloads/index.html.
2. Click the Java Platform (JDK) download graphic.
3. Click the 'Accept License Agreement' radio button.

ISSN 2304-7712

4. Click the appropriate Java SE Development Kit download link for your platform (Linux, macOS, Windows, or Solaris) and follow the prompts.

The Java JDK allows programmers to create Java applications that can then run using the accompanying JRE (Java Runtime Environment). After installing the Java JDK on a Windows 10 computer, the contents of the folder **C:\Program Files\Java\jdk-9.0.1** (JDK version 9.0.1 at the time of this writing) are shown in Figure 1.

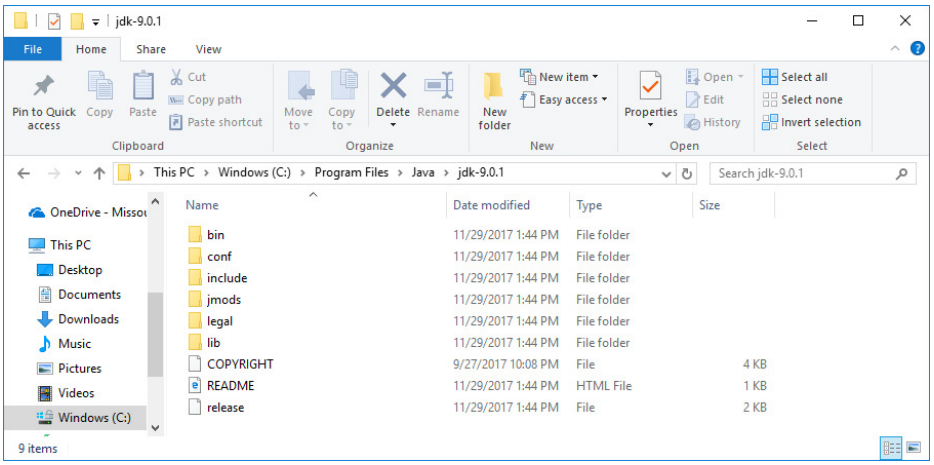

Fig. 1 Java JDK installed on a 64-bit Windows 10 computer

CONFIGURING WINDOWS TO USE JAVA

Once the Java JDK is installed, Windows needs to know where to find the files required to compile and run Java applications. The paths to these files must be stored in operating system environment variables which can be created in Windows 10 using these steps:

1. Press the Windows logo key on the keyboard, type 'control' and click Control Panel from the list of options.
2. Click System and Security, System, and Advanced system settings.
3. In the System Properties dialog, click Environment Variables.
4. In the Environment Variables dialog, click New.
5. In the New System Variable dialog, type **JAVA_HOME** in the Variable name textbox.
6. In the New System Variable dialog, type **C:\Program Files\Java\jdk-9.0.1** (the complete absolute path to the Java JDK folder) in the Variable value textbox—the path on your computer may be different depending on the name and location of the JDK folder. *TIP: Double-click the JDK folder in Windows Explorer, click the address textbox, right-click this address selection, and click Copy. Then paste this path (such as **C:\Program Files\Java\jdk-9.0.1**) in the Variable value textbox—this should help avoid typographical errors.*
7. Click OK to close the New System Variable dialog.
8. While still in the Environment Variables dialog, scroll down in the System variables list until the Path variable is visible. Click Path, then click Edit.
9. In the Edit environment variable dialog box, click New and enter **%JAVA_HOME%\bin**

ISSN 2304-7712

10. This path variable now represents the path to the bin folder that is within the JDK folder, which in turn is within the Java folder. This bin folder contains the javac.exe and java.exe files required to compile and run a Java program, respectively. When finished, click OK to close the Edit environment variable dialog, click OK to close the Environment Variables dialog, and click OK to close the System Properties dialog. You may need to restart the computer for these system changes to take effect.

INSTALLING THE TEXTPAD JAVA EDITOR AND ECLIPSE JAVA IDE

You may already be using tools such as the Java editor TextPad ("TextPad", 2017) or the Java integrated development environment (IDE) Eclipse ("Eclipse", 2017). There are many other popular Java IDE's available, such as NetBeans ("NetBeans IDE", 2017), which can also be used in the context of this tutorial. In case you are not currently using such tools, this section explains how to install both TextPad and Eclipse:

1. To install TextPad, visit textpad.com, click the Download tab, and download the 32-bit or 64-bit version as appropriate (Windows only). Follow the prompts to complete the installation.
2. To download Eclipse, visit eclipse.org, click the Download buttons appropriate for your OS in order to download the executable installation file (such as eclipse-inst-win64.exe). Double-click the .exe file and follow the prompts to install Eclipse.

TESTING INSTALLATIONS WITH A SIMPLE JAVA PROGRAM

In order to test the installations of the Java JDK, you can write a simple Hello.java program and run it using (1) Notepad and Command Prompt, (2) TextPad, and (3) Eclipse. These are the three tools that will be used later to test a simple Java database application.

```
1 public class Hello{
2
3     public static void main( String [] args ){
4           System.out.println( "Hello, world!" );
5     }
6 }
```

Fig. 2 A simple Hello.java program

Create, compile, and run Hello.java using Notepad and Command Prompt

Using the Java code in Figure 2, create the program Hello.java and test the installations using Notepad and Command Prompt:

1. Enter the code in Figure 2 into Notepad and save the file as Hello.java on flash drive E: (your flash drive letter may be different, such as D:, or you can also use hard drive C:). Note that the indentation used in Hello.java is not required, but is conventional for writing Java programs.
2. Run Command Prompt, enter **e:** at the prompt to move to drive E: and enter **javac Hello.java** (note that Java is case sensitive) at the prompt to compile the program.
3. If no errors occur, enter **java Hello** at the prompt to run **Hello.class** (the compiled version of Hello.java). The output, "Hello, world!" should be displayed, as shown in Figure 3.

Note that the OS knows where the Java compiler and interpreter (javac.exe and java.exe, respectively) are

ISSN 2304-7712

located because of the Path variable being set, as described earlier.

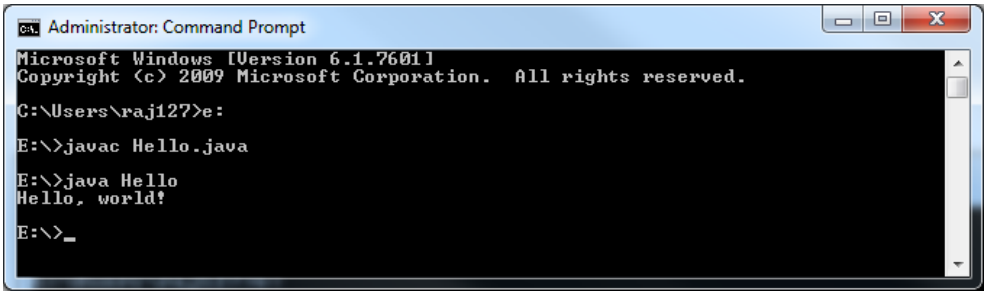

Fig. 3 Compiling and running Hello.java using Command Prompt

Create, compile, and run Hello.java using TextPad

Use the Java code in Figure 2 to create the program Hello.java and test the installations using TextPad:

1. Launch TextPad, enter the code exactly as shown in Figure 2 within the TextPad editor, and save the file as Hello.java on drive E: (or any other drive), as shown in Figure 4.
2. Press Ctrl-1 to compile the program, then Ctrl-2 to run the program. If everything runs smoothly, the output "Hello, world!" will appear in a separate Command Prompt window.

If the Tool Output window in TextPad displays errors, check the Hello.java program for typographical errors, save, re-compile, and re-run. If the commands Ctrl-1 and Ctrl-2 do not function properly in TextPad, you may need to perform the following steps from the TextPad menu to add the Compile and Run tools: Click Configure, Preferences…, Tools, Add, Java SDK Commands, OK. If these tools are still not available, check the modification of the Path variable in Windows as previously discussed.

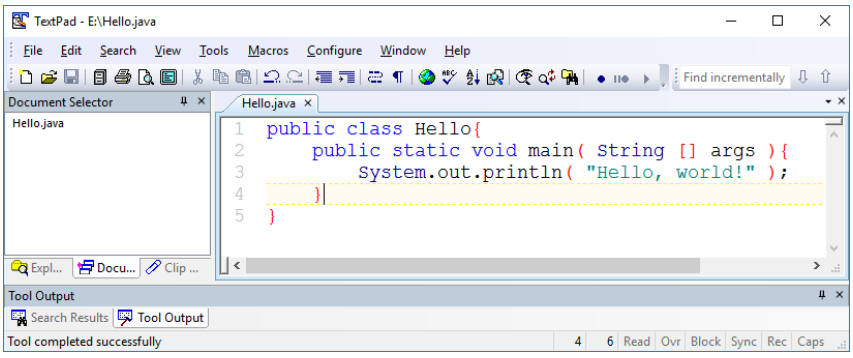

Fig. 4 Hello.java created and compiled in TextPad

Create, compile, and run Hello.java using Eclipse

Use the Java code in Figure 2 to create an Eclipse project that contains the program Hello.java:

1. Create a folder called 'workspace' (or a folder of some other name of your choice, but 'workspace' is the convention) on flash drive E: (or any other drive) to store Eclipse projects.
2. Launch Eclipse.
3. To select the workspace folder (the folder where Eclipse projects are stored), click Browse in the Eclipse Launcher dialog, navigate to the folder 'workspace' previously created on drive E:, click that folder, click OK.

ISSN 2304-7712

4. Click Launch in the Eclipse Launcher dialog.
5. Close the Workbench tab to exit the Welcome screen and view the development window.
6. Click File, New, Project, Java project. Enter a project name (such as "Test"), and click Finish. *NOTE: An Eclipse project is a folder where your Java programs and various supporting files are stored.*
7. Click File, New, Class. Enter 'Hello' as the name of the class (case sensitive), click Finish.
8. Enter the code exactly as shown in Figure 2 within the Eclipse editor and save the file by clicking the Save icon in the toolbar.
9. Click the Run icon in the toolbar (see Figure 5).

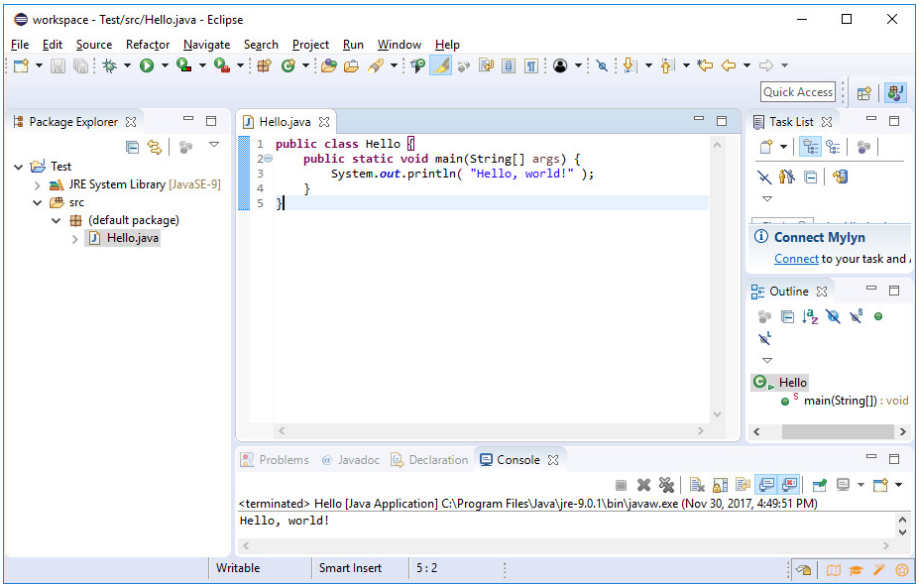

Fig. 5 Project Test containing Hello.java created and run in Eclipse

As one can see, creating and running a Java application using Command Prompt, TextPad, and Eclipse become increasingly complex. However, a Java editor (such as TextPad) provides many features not available in Command Prompt, and a Java IDE (such as Eclipse) provides even more features that aid in professional application development.

INSTALLING THE MYSQL DATABASE MANAGEMENT SYSTEM

In order to create a Java application that uses MySQL, the user must install MySQL on the computer using the following steps:

1. Go to https://www.oracle.com/mysql/index.html and click the Download MySQL Database button.
2. Select the appropriate OS (Windows, Linux, macOS) and OS version (32-bit or 64-bit) from the drop-down lists.
3. Click the appropriate Download button, depending on whether you have an online connection (web) or not.
4. Click the "No thanks, just start my download" link. Click Save, then click Run.
5. Double-click the MSI file that was downloaded (which will be in your Downloads folder).
6. Accept the license terms, click Next.

ISSN 2304-7712

7. Accept the "Developer Default" for Setup Type, click Next.
8. Check the special requirements (usually optional), click Next, then click Execute to install, accepting default options. Some accompanying products may not install depending on your system. Click Next when complete.
9. Click Next for product configuration.
10. Accept default for Type and Networking, click Next, Next.
11. When prompted, supply a root account password of your choice (e.g., 'password'), click Next (obviously, more complex passwords are preferred).
12. Accept the defaults for Windows Service, click Next.
13. Accept the defaults for Plugins and Extensions, click Next, click Execute, click Finish.
14. Click Next on Product Configuration dialog, click Finish.
15. The MySQL Shell (mysqlsh.exe, the command line interface) and MySQL Workbench (MySQLWorkbench.exe, the MySQL GUI) should open automatically.

The MySQL Shell (mysqlsh.exe, the command line interface) and MySQL Workbench (MySQLWorkbench.exe, the graphical user interface) should open automatically, and you can close these applications. For this tutorial, you won't actually use the MySQL database system directly, but will instead access MySQL via a Java program.

DOWNLOADING THE JDBC DATABASE DRIVER FILE

In order for a Java program to interact with MySQL, you need to download and store the appropriate JDBC database driver file, which is a JAR (Java archive) file that contains the Java classes required for database appplications. Follow these instructions to obtain this JAR file:

1. Go to https://dev.mysql.com/downloads/connector/ and click the Connector/J link.
2. Click the Download button for the ZIP archive. Then click 'No thanks, just start my download'.
3. Click Save, then click Open. The folder *mysql-connector-java-5.1.45* (the version number may be different due to updates) will now appear in your Downloads folder. Note that this folder contains the JAR file (*mysql-connector-java-5.1.45-bin.jar*) needed for the Java program to interact with a MySQL database.
4. Copy this folder to your computer in a convenient location, such as C:\lib (a directory called lib on the C: drive, which you can create). You can place this folder virtually anywhere, and you can even just copy the JAR file (*mysql-connector-java-5.1.45-bin.jar*) as a standalone file anywhere. However, in this tutorial, the absolute path to this JAR file is:
   C:\lib\mysql-connector-java-5.1.45\mysql-connector-java-5.1.45-bin.jar

CONFIGURING WINDOWS 10 FOR THE JDBC DRIVER

Now that you have downloaded the JDBC driver (JAR file), you need to configure Windows 10 so that your Java database application can locate it. Follow these instructions to perform the configuration:

1. Run Control Panel (press the Windows key, type 'control', and click Control Panel), click System and Security, System, Advanced system settings, Environment Variables.

ISSN 2304-7712

2. In the System variables list, click New, enter CLASSPATH, click Browse File, navigate to the *mysql-connector-java-5.1.45-bin.jar* file that you just placed in C:\lib\mysql-connector-java-5.1.45, and select the JAR file. Then click Open. (Remember that this JAR file could be located anywhere on your system, but it's best to choose a logical location.)
3. One crucial step: Click at the end of the Variable value (path to the JAR file) in the Edit System Variable dialog. Then type a semi-colon (;) followed by a period (.), which represents the current working directory. This adds the current working directory (wherever your Java database application is located on the C: drive or a flash drive) to CLASSPATH (without this your application will not function). After following these steps, the Edit System Variable dialog will appear as in Figure 6.
4. Click OK, OK, OK to close all dialogs.

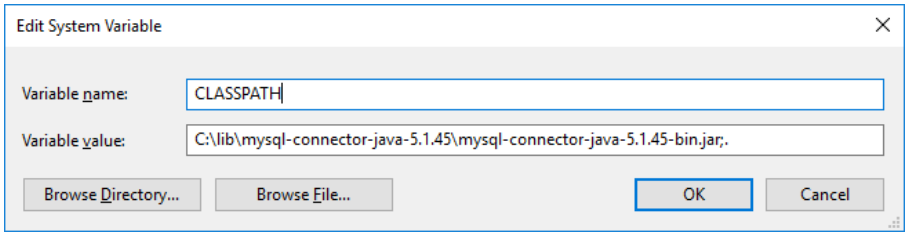

Fig. 6 Adding the CLASSPATH variable for the JDBC driver in Windows 10

CONFIGURING ECLIPSE FOR THE JDBC DRIVER

The previous section discussed how to configure Windows 10 so that a Java database application run from Command Prompt and TextPad can access the Java JDK and the JDBC driver (JAR file). However, Eclipse requires its own configuration process in order to access the JDBC driver, as described below:

1. Run Eclipse, select the workspace folder (in this example, on drive D:), create the new Java project MySQL Test, and create the new Java class MySQLConnection (the details of creating a Java project and class in Eclipse were presented earlier). No Java code has been added to the project yet.
2. Right-click on the project (MySQL Test), select Build Path and select Configure Build Path.
3. In the dialog, click the Libraries tab, click Classpath, and click Add External JARs.
4. Navigate to the desired JAR file (in this example that would be **C:\lib\mysql-connector-java-5.1.45\ mysql-connector-java-5.1.45-bin.jar**, as shown in Figure 7. Note that the location of the JAR file can be anywhere as determined by the user. Then click Open.
5. Click Apply and Close in the Eclipse dialog.

ISSN 2304-7712

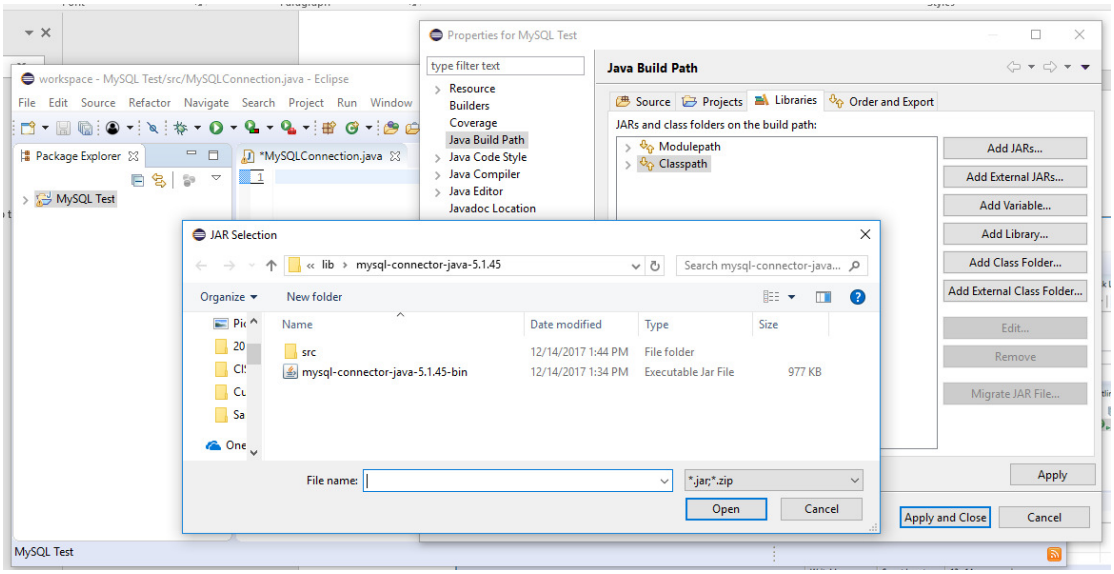

Fig. 7 Configuring the JDBC driver in the MySQL Test project in Eclipse

RUNNING A JAVA DATABASE APPLICATION USING MYSQL

At this point in the tutorial you have installed MySQL, the Java JDK, TextPad, Eclipse, and the JDBC driver, and you have configured Windows to work with these applications. In this section, you will create a very simple Java program to test the connection between the program and a MySQL database file.

Figure 8 presents an extremely simple Java database program that does nothing more than create a MySQL database file and establish a connection to it. Of course, a complete Java database application would involve creating all components of a three-tier design: presentation tier (the GUI—how the user interacts with the application), middle tier (the business logic—what the application does), and data tier (the database—what the application stores) ("Using a Three-Tier Architecture Model.", 2015). Creating a complete Java application that easily connects to Java DB could be covered in additional tutorials or Java texts, but the basic connectivity procedure would be as presented here.

```
1  import java.sql.Connection;
2  import java.sql.DriverManager;
3  import java.sql.SQLException;
4
5  public class MySQLConnection {
6      public static void main( String args[] ) throws SQLException{
7
8          String dbURL = "jdbc:mysql://localhost:3306/test?useSSL=false";
9         Connection conn = DriverManager.getConnection( dbURL,"root","password" );
10        System.out.println( "Database created successfully" );
11     }
12 }
```

Fig. 8 The MySQLConnection.java file that tests the database connection

The code in Figure 8 may appear daunting to the beginning Java student, but taken in segments it's relatively easy to understand.

ISSN 2304-7712

1. Lines 1-3 import three Java classes that are used in the program. Notice that they are all from the `java.sql` package, meaning that they are classes designed to work with relational databases.
2. Line 5 is simply the `class` declaration (the class can be named anything).
3. Line 6 is the `main` method declaration. Because database programs may encounter situations where they incorrectly access the database, the Java language requires that exception objects from class `SQLException` be *thrown* (in the `main` method declaration, Line 6) or *caught* within the program using `try` and `catch` blocks (exception handling is covered in most intermediate Java courses). In this program, any `SQLException` objects are simply thrown (not caught) to be displayed in the output.
4. Line 8 is critical. It creates a string that contains the syntax for creating a MySQL database file. This line contains many parts (separated by :'s). `jdbc` refers to the Java Database Connectivity protocol. `mysql` refers to the type of database software used (subprotocol). `//localhost:3306` is the syntax for connecting to the MySQL database on '`localhost`' (the current computer) using port `3306` (the default port for MySQL). `test` is simply the name of the database file being created (supplied by the user). Thus, `jdbc:mysql://localhost:3306/test` is the full URL for the MySQL database called `test`. The `?` is the syntax used to append special instructions to the URL. In this case, the instruction `useSSL=false` is stating that the SSL (Secure Sockets Layer) protocol (for encryption) will not be used. (Adding encryption here is beyond the scope of this tutorial.) Without this instruction, a long annoying warning message will appear whenever the program is run. In a production environment, it is likely that SSL will indeed be used for computer security purposes. Thus, the string `dbURL` (Line 8) contains the complete specification for how to create the MySQL database within a Java program.
5. Line 9 creates a Java Connection object (called `conn`) that uses `dbURL`, the userid (which was set by default as 'root' during the MySQL installation), and the database password (which was set to 'password' by the user during the MySQL installation).
6. Line 10 (if reached without error) displays a message confirming the program does indeed work.

As you can see, installing MySQL and configuring an OS to use it can be somewhat challenging, which is probably why many Java courses avoid it.

TESTING THE SAMPLE JAVA DATABASE PROGRAM

The Java database program in Figure 8 can now be tested using Command Prompt, TextPad, and Eclipse, as described below:

1. To test the code in Figure 8 using Command Prompt, enter the code into Notepad and save the file as MySQLConnection.java on drive D: (your drive letter may be different, or you can use drive C:). Then run Command Prompt, move to drive D:, enter javac MySQLConnection.java to

ISSN 2304-7712

compile, then enter java MySQLConnection to run. The message, “Database created successfully”, should appear.

2. To test the Java program in Figure 8 using TextPad, launch TextPad, enter the code within the TextPad editor, save the file as MySQLConnection.java (on drive D: or any location), press Ctrl-1 to compile and press Ctrl-2 to run. The confirmation message, “Database created successfully”, should appear in Command Prompt.
3. To test the Java program in Figure 8 using Eclipse, launch Eclipse, select the workspace folder on D: (or elsewhere if you choose), create a new Java project (called **MySQL Test**, for example), create a new class called **MySQLConnection**, enter the code in the Eclipse editor, save the file, then click the Run icon in the Eclipse toolbar. The confirmation message, “Database created successfully”, should appear in the Console window of Eclipse.

If errors appear while trying to compile or run the Java program in Figure 8, first check the typing in MySQLConnection.java. If errors persist, retrace the steps throughout this tutorial to ensure that programs such as Hello.java are indeed working in Command Prompt, TextPad, and Eclipse, that the JDBC driver file has been copied to the proper location, and that the Windows environment variables have been created correctly. If no errors occur, you are ready to learn the SQL language and then create Java GUI’s that interact with a MySQL database.

CONCLUSION

This tutorial first presented steps to install the MySQL database management system, the Java JDK, the TextPad Java editor, the Eclipse Java IDE, and the JDBC driver. After creating and testing a simple **Hello.java** program using Command Prompt, TextPad, and Eclipse, the procedures for configuring Windows to use MySQL with Java were provided. Finally, a simple **MySQLConnection.java** database program was demonstrated to verify that a Java program can indeed interact with a MySQL database. The procedures for running a Java database application using MySQL are more complex than using other database products, such as SQLite and Java DB, but MySQL is a very pervasive, industrial-strength, open-source database management system, and worth the effort. Additional tutorials can provide information on how to execute a wide variety of SQL commands using a Java program and the MySQL database (“Learn JDBC”, 2017).

REFERENCES

"Apache Derby." (2017), [Online] Available: http://db.apache.org/derby/ (Dec. 1, 2017)

Deitel, P. J., & Deitel, H. M. (2018). Java: How to program. Upper Saddle River, NJ: Prentice Hall.

"Eclipse." (2017), [Online] Available: http://eclipse.org/ (Dec. 1, 2017)

Farrell, J. (2016). Java programming. Boston, MA: Course Technology Cengage Learning.

Gaddis, T. (2016). Starting out with Java: From Control Structures Through Objects. Boston: Pearson.

Horstmann, C. S. (2015). Big Java: Early Objects. Hoboken, N.J: Wiley.

"Java DB Technical Documentation." (2015), [Online] Available: http://docs.oracle.com/javadb/index.html (Dec. 1, 2017)

ISSN 2304-7712

"Java SE Downloads." (n.d.), [Online] Available: http://www.oracle.com/technetwork/java/javase/downloads/ (Dec. 14, 2017)

“Learn JDBC” (2017), [Online] Available: http://www.tutorialspoint.com/jdbc/ (Dec. 14, 2017)

Johnson, R. A. (2014) Java database connectivity using SQLite: A tutorial. International Journal of Information, Business and Management, 6(3), 207-217.

Johnson, R. A. (2016) Java Database Connectivity Using Java DB (Apache Derby): A Tutorial. International Journal of Information, Business and Management, 8(3), 295-309.

Liang, Y. D. (2015). Introduction to Java programming. Boston: Pearson.

"MySQL." (2017), [Online] Available: http://www.mysql.com/ (Dec. 1, 2017)

"NetBeans IDE." (2017), [Online] Available: https://netbeans.org/ (Dec. 1, 2017)

Savitch, W. J., & Mock, K. (2016). Absolute Java. Boston: Addison-Wesley.

"SQLite." (2017), [Online] Available: http://www.sqlite.org/ (Dec. 1, 2017)

"TextPad." (2017), [Online] Available: http://textpad.com/ (Dec. 1, 2017)

"TIOBE Software." (2017), [Online] Available: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (Dec. 1, 2017)

"Using a Three-Tier Architecture Model." (2015), [Online] Available: http://msdn.microsoft.com/en-us/library/windows/desktop/ms685068%28v%3Dvs.85%29.aspx (Dec. 30, 2015)

ISSN 2304-7712

International Journal of Advanced Engineering and Science
http://ijaes.elitehall.com/