

# JDBC MySQL Database Connectivity

## JDBC MySQL Database Connectivity

Instructor: Rick Palmer, SCWCD

[rick@online-ettraining.com](mailto:rick@online-ettraining.com)

# Topics Covered

- ❑ JDBC API and Driver Overview
- ❑ Connecting to a MySQL Database
- ❑ Creating JDBC Statements
- ❑ Processing JDBC Resultsets

# Java Database Connectivity

- JDBC is a Java API for
  - ⇒ Connecting to databases
  - ⇒ Executing SQL statements
  - ⇒ Processing the results
- Provides cross-vendor connectivity and database access.

# JDBC Driver

- ❑ Connects Java applications to SQL databases.
- ❑ Makes a network socket connection from the application to the database.
  - ⇒ Allows connections to databases anywhere on the Internet using a network-based url (example shown with a database named **accounts** located on the same system as the Java application):

```
jdbc:mysql://localhost/accounts
```

# JDBC Driver Installation

- ❑ Download the MySQL Connector/J 3.1 driver from <http://dev.mysql.com/downloads/connector/j/3.1.html>
- ❑ Open the downloaded zip file and extract mysql-connector-java-3.1.14-bin.jar to a folder on your system.
- ❑ Configure as an External Jar file in the Project property settings in Eclipse (Java Build Path → Libraries → Add External JARs).
  - ⇒ If not using Eclipse, add the jar file to the CLASSPATH

# Main JDBC Interfaces

## ❑ DriverManager

⇒ Manages JDBC drivers

⇒ Connects to the database using a static `getConnection` method.

## ❑ Connection

⇒ A connection (session) with a particular database.

⇒ SQL statements are executed and results returned within the context of a Connection.

## ❑ Statement

⇒ Executes a SQL statement and returns the results

## ❑ ResultSet

⇒ Contains the database results from executing a SQL query.

# Connecting to a Database

```
import java.sql.*;

private Connection getConnection() {

    Connection con = null;

    try {
        // Load the JDBC driver (jar file must exist on CLASSPATH)
        Class.forName("com.mysql.jdbc.Driver");

        // Build the database URL
        String strDatabaseURL = "jdbc:mysql://localhost/accounts";

        // Connect to the database (use your own user and password)
        con = DriverManager.getConnection(strDatabaseURL,
                                         "root", "biffle");
    }
    catch (Exception e) {e.printStackTrace();}

    return con;
}
```

# Querying for a ResultSet

```
import java.sql.*;
```

---

```
// Build the SQL query
```

```
String strSQL = "SELECT * FROM vehicles";
```

```
try {
```

```
    // Get a database Connection
```

```
    ... (code from previous slide)
```

```
    // Create a Statement and execute the SQL query
```

```
    Statement stmt = con.createStatement();
```

```
    ResultSet rstVehicles = stmt.executeQuery(strSQL);
```

```
}
```

```
catch (SQLException se) { se.printStackTrace(); }
```

# Processing a ResultSet

```
try {  
    // Get Connection, create Statement, and execute SQL query  
    ...  
    ResultSet rstVehicles = stmt.executeQuery(strSQL);  
  
    // Loop through the ResultSet  
    while( rstVehicles.next() ) {  
  
        // Get the data field values of this record  
        strVehicleID = rstVehicles.getString("VIN");  
        iYear = rstVehicles.getInt("Year");  
  
        // Create output from the data values...  
    }  
}  
catch (SQLException se) {se.printStackTrace(); }
```

# Updating Records in a Table

```
import java.sql.*;
```

---

```
// Build the SQL command
```

```
String strSQL =
```

```
    "UPDATE vehicles SET LastServiceDate = '11/01/2003'  
    WHERE VIN = '8YTR754' ";
```

```
try {
```

```
    // Get Connection
```

```
    ...
```

```
    // Create a Statement and execute the SQL
```

```
    Statement stmt = con.createStatement();
```

```
    int iAffectedRows = stmt.executeUpdate(strSQL);
```

```
}
```

```
catch (SQLException se) { se.printStackTrace(); }
```

# Cleaning Up SQL Resources

- ❑ Close the Connection when finished with it, which also closes any associated Statements and ResultSets.

```
try {  
    // Get Connection, execute SQL update, etc...  
}  
catch (SQLException se) { se.printStackTrace(); }  
finally {  
    // Close Connection even if error occurred above.  
    if (con != null) {  
        try { con.close(); } catch (SQLException se) { }  
    }  
}
```

# MySQL/JDBC "Gotchas"

- ❑ MySQL does not have a native Boolean datatype.

```
// Convert boolean to 0 for false, or 1 for true.
```

```
private int getMySQLBoolean(boolean b) {  
    int iMySQLBoolean = 1;  
    if (b == false) { iMySQLBoolean = 0; }  
    return iMySQLBoolean;  
}
```

- ❑ Single quotes in SQL statement must be replaced with two single quotes, so it can be correctly parsed by the JDBC driver.

```
"Larry's Gift Shop" → "Larry''s Gift Shop"
```