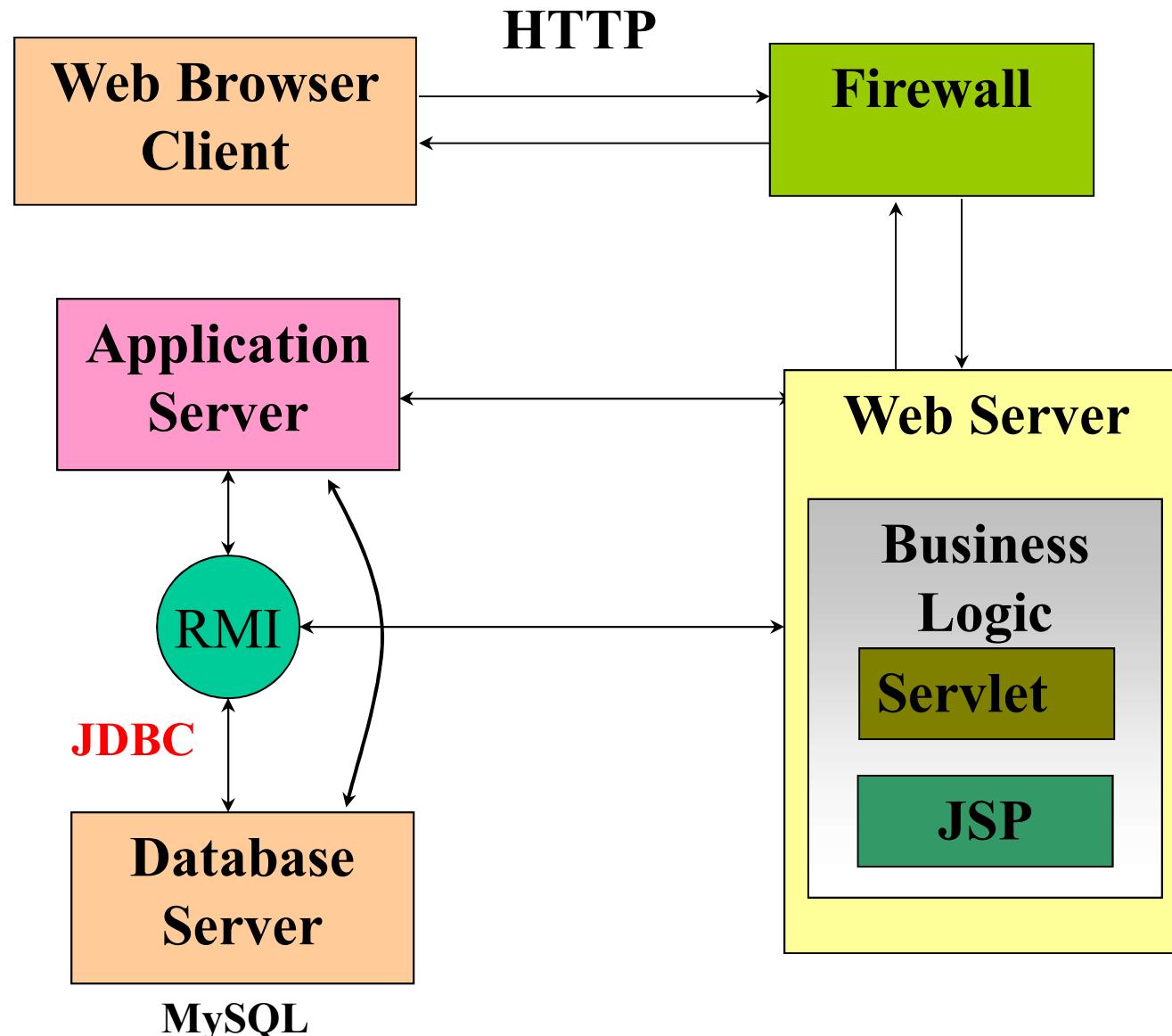


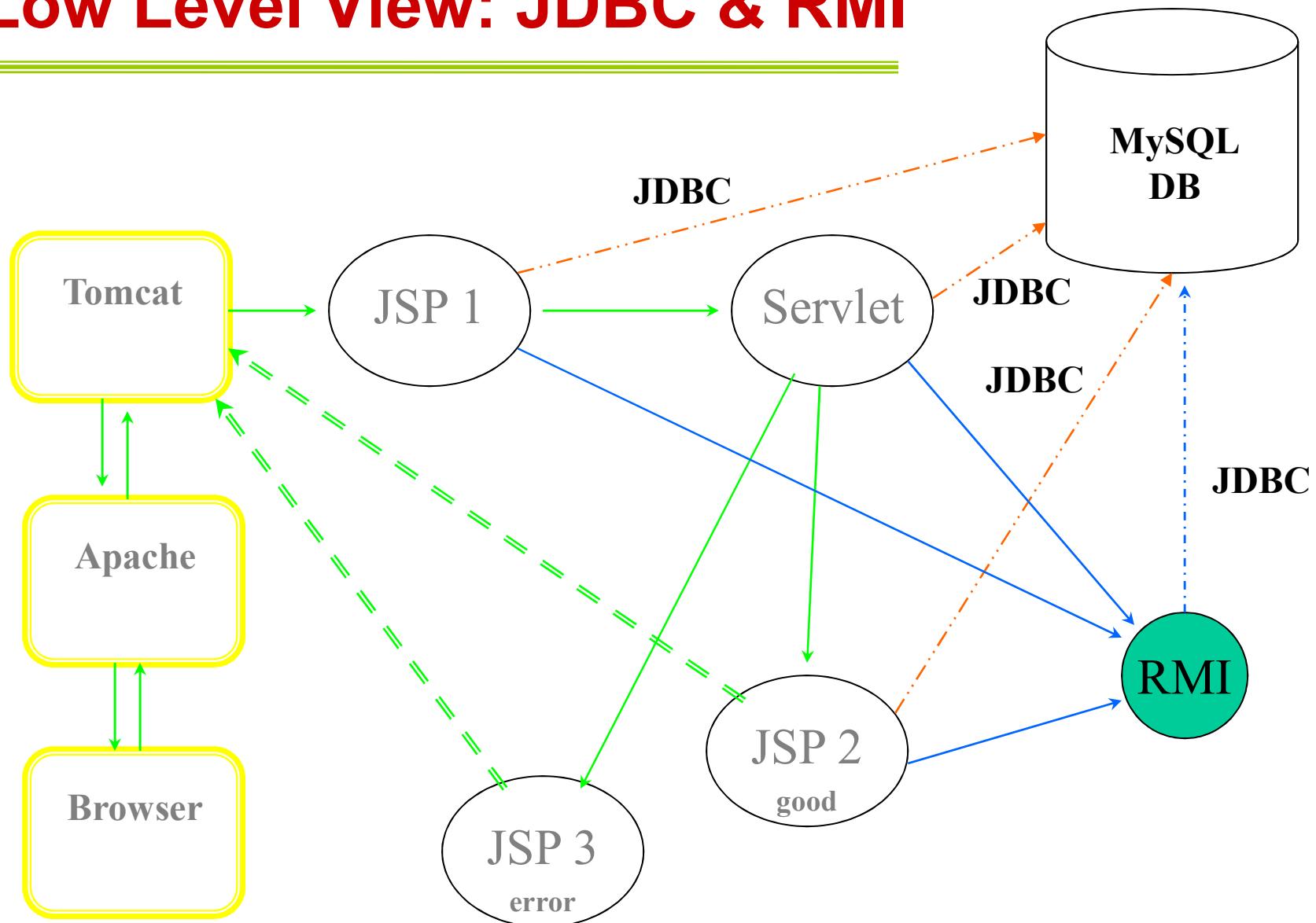
JDBC

Java DataBase Connectivity

JDBC & RMI



Low Level View: JDBC & RMI



JDBC

- The most popular style of database system is the relational database. A language called ***structured query language (SQL)*** is used among relational database systems to make queries.
- Java allows the programmers to use SQL to access information in the relational database systems.

Relational Database Model

- A relational database model is developed by Codd is a logical representation of the data that allows the relationships between the data to be concerned without concerning oneself with the physical implementation of the data structure.

Relational Database Model

- A relational database consists of tables.example:EMPLOYEE table

Number	Name	Department	Salary	Location
23603	JONES. A	413	1600	NEW JERSY
24568	KEWIN.R	413	2000	NEW JERSY
34589	LARSON P	642	1800	LOS ANGELES

Relational Database Model

- The name of the above table is EMPLOYEE and its primary purpose is to illustrate the various attributes of an employee and how they are related to a specific employee.
- Any particular row of a table is called record(or row). The above table has three records.

Relational Database Model

- The employee number field of each record in this table is used as the *primary key*.
Tables in a database normally have primary keys, but primary keys are not required.
- The primary key can be composed of more than one column in the database. Primary key fields in a table cannot have duplicate values

Relational Database Model

- Each column of the table represents a different field.
- We can perform many kinds of operations using the tables. We can find particular fields of a table and combine two or more smaller tables into a one big table.
- The former operation is called *projection* and later is called *join*.

How to use a MySQL database

Objectives

Applied

1. Use MySQL Workbench to start and stop the MySQL server.
2. Use MySQL Workbench to run SQL statements.
3. Use MySQL Workbench to run SQL scripts.
4. Code simple SELECT, INSERT, UPDATE, and DELETE statements and use MySQL Workbench to test them.

Knowledge

1. Distinguish between SQL's Data Definition Language (DDL) and Data Manipulation Language (DML).
2. Describe the capabilities of a SELECT statement.
3. Describe the capabilities of INSERT, UPDATE, and DELETE statements.
4. Describe what a SQL script does.

Go to MySQL download tab:
1. Download the Community server
2. Download Connector/J for Java

The world's most popular open source database

MySQL.com **Downloads** Documentation Developer Zone

Enterprise **Community** Yum Repository APT Repository SUSE Repository Windows Archives

Download MySQL Community Server

MySQL Community Edition is a freely downloadable version of the world's most popular open source database that is supported by an active community of open source developers and enthusiasts.

MySQL Cluster Community Edition is available as a separate download. The reason for this change is so that MySQL Cluster can provide more frequent updates and support using the latest sources of MySQL Cluster Carrier Grade Edition.

Important Platform Support Updates

Online Documentation: Looking for previous GA versions?

- [Installation Instructions, Documentation and Change History](#) for the MySQL 5.7 Generally Available (GA) Release
- [Installation Instructions, Documentation and Change History](#) for the MySQL 5.6 Generally Available (GA) Release
- [Installation Instructions, Documentation and Change History](#) for the MySQL 5.5 Generally Available (GA) Release

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

You need to download
MySQL Community Server

The world's most popular open source database

MySQL.com Downloads Documentation Developer Zone

Enterprise Community Yum Repository APT Repository SUSE Repository Windows Archives

Download MySQL Community Server

MySQL Community Edition is a freely downloadable version of the world's most popular open source database that is supported by an active community of open source developers and enthusiasts.

MySQL Cluster Community Edition is available as a separate download. The reason for this change is so that MySQL Cluster can provide more frequent updates and support using the latest sources of MySQL Cluster Carrier Grade Edition.

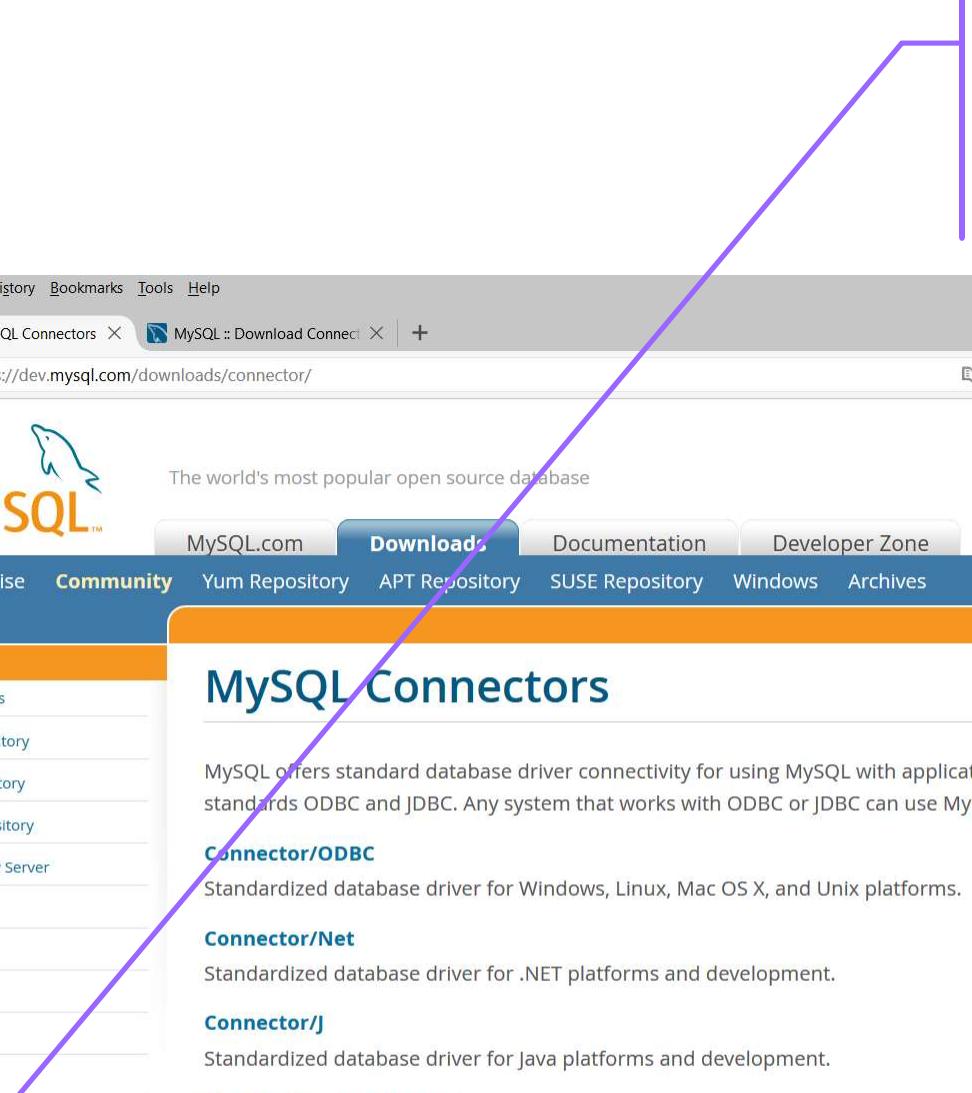
Important Platform Support Updates

Online Documentation: Looking for previous GA versions?

- [Installation Instructions, Documentation and Change History](#) for the MySQL 5.7 Generally Available (GA) Release
- [Installation Instructions, Documentation and Change History](#) for the MySQL 5.6 Generally Available (GA) Release
- [Installation Instructions, Documentation and Change History](#) for the MySQL 5.5 Generally Available (GA) Release

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

Click MySQL Connectors



The world's most popular open source database

MySQL.com **Downloads** Documentation Developer Zone

Enterprise Community Yum Repository APT Repository SUSE Repository Windows Archives

MySQL Connectors

MySQL offers standard database driver connectivity for using MySQL with applications and tools that are compatible with industry standards ODBC and JDBC. Any system that works with ODBC or JDBC can use MySQL.

Connector/ODBC
Standardized database driver for Windows, Linux, Mac OS X, and Unix platforms.

Connector/Net
Standardized database driver for .NET platforms and development.

Connector/J
Standardized database driver for Java platforms and development.

New! Connector/Node.js
Standardized database driver for Node.js platforms and development.

Connector/Python
Standardized database driver for Python platforms and development.

Connector/C++
Standardized database driver for C++ development.

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

Select Connector/J



The world's most popular open source database

MySQL.com **Downloads** Documentation Developer Zone

Enterprise Community Yum Repository APT Repository SUSE Repository Windows Archives

MySQL Connectors

MySQL offers standard database driver connectivity for using MySQL with applications and tools that are compatible with industry standards ODBC and JDBC. Any system that works with ODBC or JDBC can use MySQL.

Connector/ODBC
Standardized database driver for Windows, Linux, Mac OS X, and Unix platforms.

Connector/Net
Standardized database driver for .NET platforms and development.

Connector/J
Standardized database driver for Java platforms and development.

New! Connector/Node.js
Standardized database driver for Node.js platforms and development.

Connector/Python
Standardized database driver for Python platforms and development.

Connector/C++
Standardized database driver for C++ development.

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

MySQL :: MySQL Connectors | MySQL :: Download Connectors

You need to download this
and save in tomcat/lib
directory

File Edit View History Bookmarks Tools Help

MySQL :: Download Connector X MySQL :: Download Connector X +

https://dev.mysql.com/downloads/connector/j/

Enterprise Community Yum Repository APT Repository SUSE Repository Windows Archives

Download Connector/J

MySQL Connector/J is the official JDBC driver for MySQL.

Online Documentation:

- MySQL Connector/J Installation Instructions
- Documentation
- MySQL Connector/J X DevAPI Reference (requires Connector/J 6.0)
- Change History

Please report any bugs or inconsistencies you observe to our Bugs Database.

Thank you for your support!

MySQL open source software is provided under the [GPL License](#).
OEMs, ISVs and VARs can purchase commercial licenses.

Generally Available (GA) Releases **Development Releases**

Connector/J 5.1.42

Select Operating System:

Looking for previous GA versions?

Platform Independent (Architecture Independent), Compressed TAR Archive	5.1.42	3.8M	Download
(mysql-connector-java-5.1.42.tar.gz)			MD5: bc23a03d813af3f7ac44b8e7a5cb0d54 Signature

Platform Independent (Architecture Independent), ZIP Archive	5.1.42	4.1M	Download
(mysql-connector-java-5.1.42.zip)			MD5: 691dab78be9921dfacdacc5ebc427182 Signature

! We suggest that you use the MD5 checksums and GnuPG signatures to verify the integrity of the packages you download.

STRIKE TO

You need to download this
and save in tomcat/lib
directory

File Edit View Bookmarks Tools Help

Index of /maven2/javax/s...

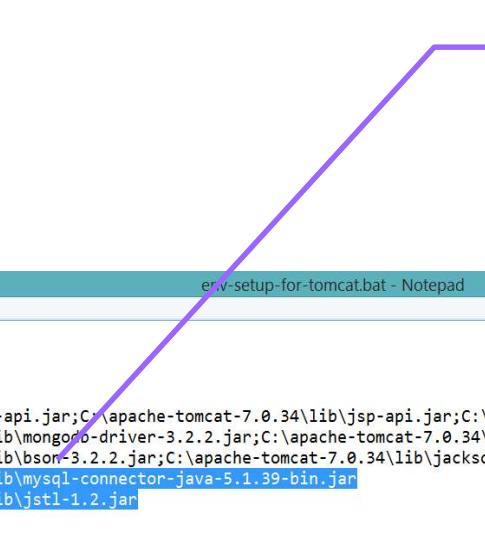
central.maven.org/maven2/javax/servlet/jstl/1.2/

Index of /maven2/javax/servlet/jstl/1.2/

..

jstl-1.2-sources.jar	10-May-2006 22:32	487044
jstl-1.2-sources.jar.md5	27-Nov-2012 15:40	32
jstl-1.2-sources.jar.sha1	18-Jan-2008 03:06	63
jstl-1.2.jar	23-Jun-2011 17:20	414240
jstl-1.2.jar.md5	27-Nov-2012 15:40	32
jstl-1.2.jar.sha1	23-Jun-2011 17:20	55
jstl-1.2.pom	23-Jun-2011 17:20	356
jstl-1.2.pom.md5	27-Nov-2012 15:40	32
jstl-1.2.pom.sha1	23-Jun-2011 17:20	55

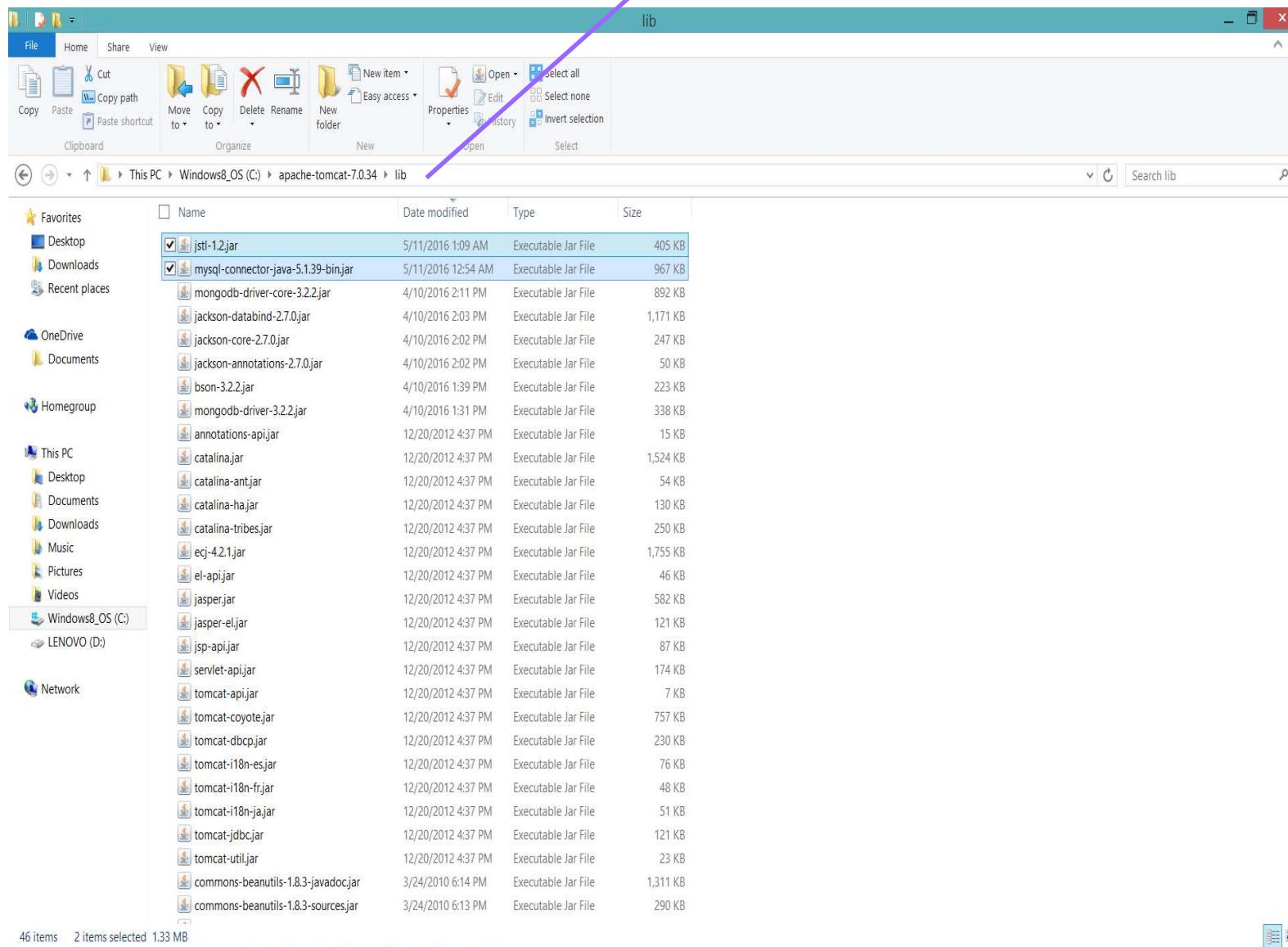
Add these two lines to
env-setup-for-tomcat
batch file



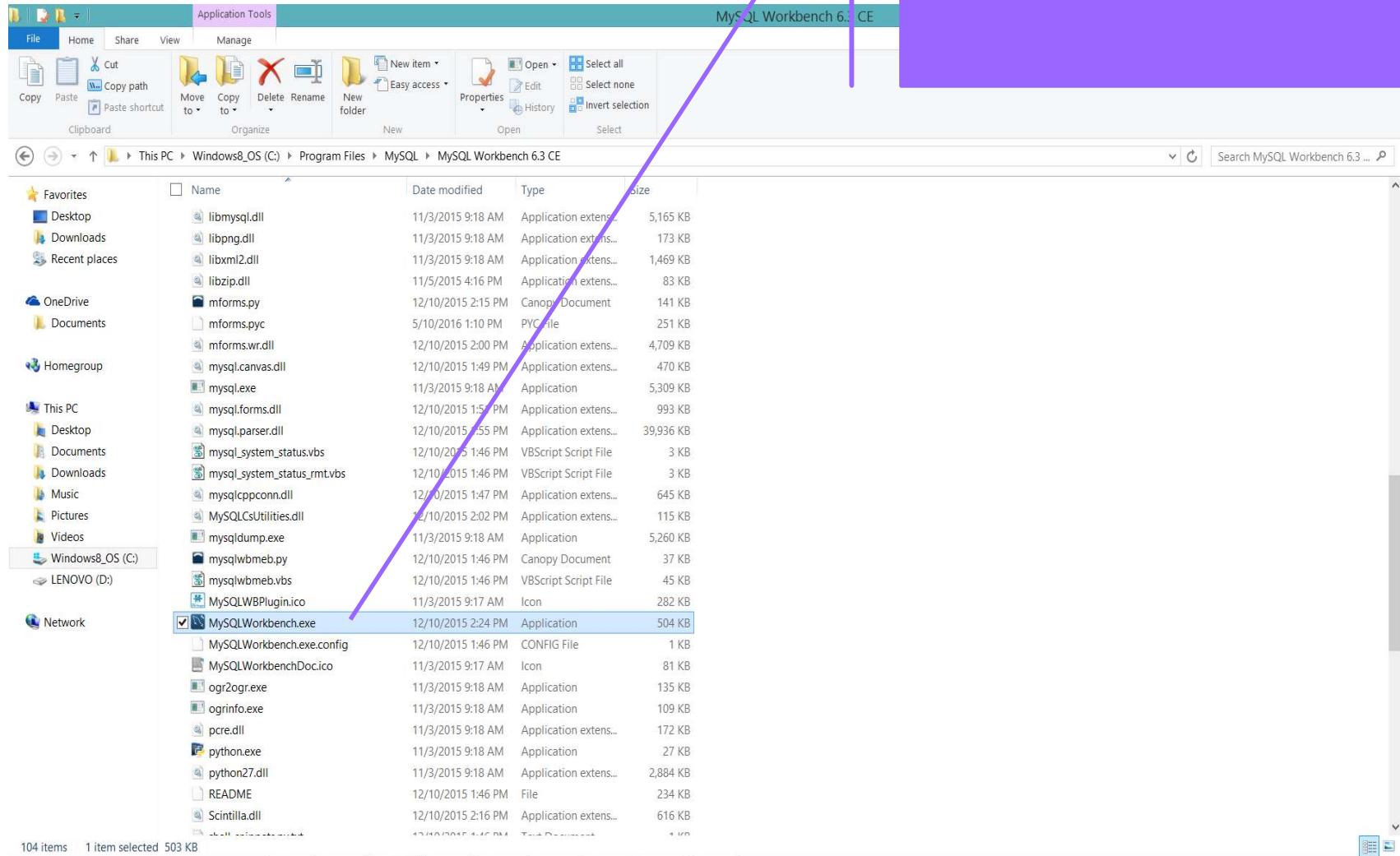
```
env-setup-for-tomcat.bat - Notepad
File Edit Format View Help
set JAVA_HOME=C:\jdk1.7
set PATH="C:\jdk1.7\bin";%PATH%
set CLASSPATH=.;C:\apache-tomcat-7.0.34\lib\servlet-api.jar;C:\apache-tomcat-7.0.34\lib\jsp-api.jar;C:\apache-tomcat-7.0.34\lib\el-api.jar;C:\apache-tomcat-7.0.34\lib\comm
set CLASSPATH=%CLASSPATH%;C:\apache-tomcat-7.0.34\lib\mongodb-driver-3.2.2.jar;C:\apache-tomcat-7.0.34\lib\mongodb-driver-core-3.2.2.jar
set CLASSPATH=%CLASSPATH%;C:\apache-tomcat-7.0.34\lib\bson-3.2.2.jar;C:\apache-tomcat-7.0.34\lib\jackson-databind-2.7.0.jar;C:\apache-tomcat-7.0.34\lib\jackson-core-2.7.0;
set CLASSPATH=%CLASSPATH%;C:\apache-tomcat-7.0.34\lib\mysql-connector-java-5.1.39-bin.jar
set CLASSPATH=%CLASSPATH%;C:\apache-tomcat-7.0.34\lib\jstl-1.2.jar

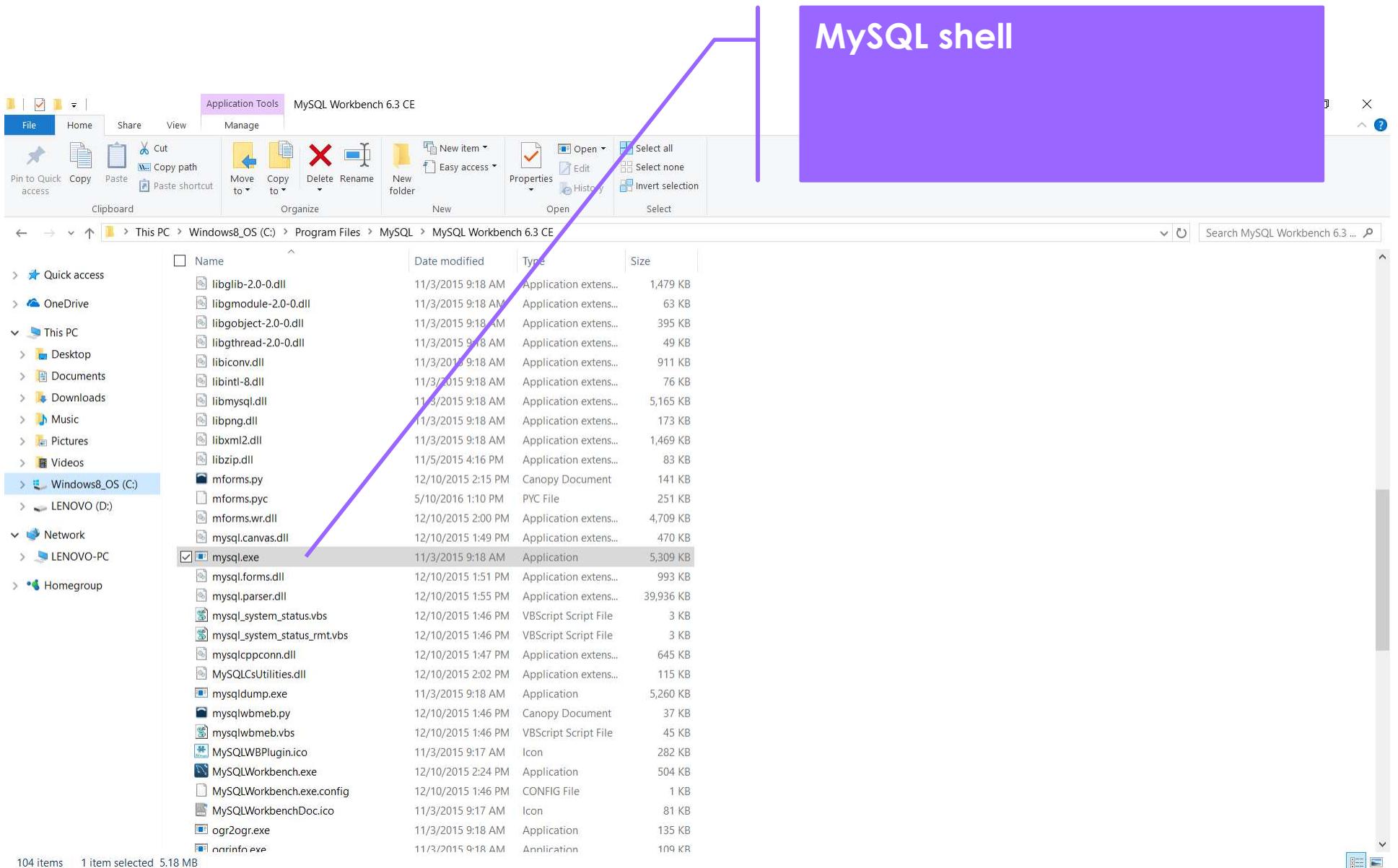
set ANT_HOME=c:\apache-tomcat-7.0.34
set TOMCAT_HOME=C:\apache-tomcat-7.0.34
set CATALINA_HOME=C:\apache-tomcat-7.0.34
```

The jar files in lib



MySQL workbench





MySQL shell

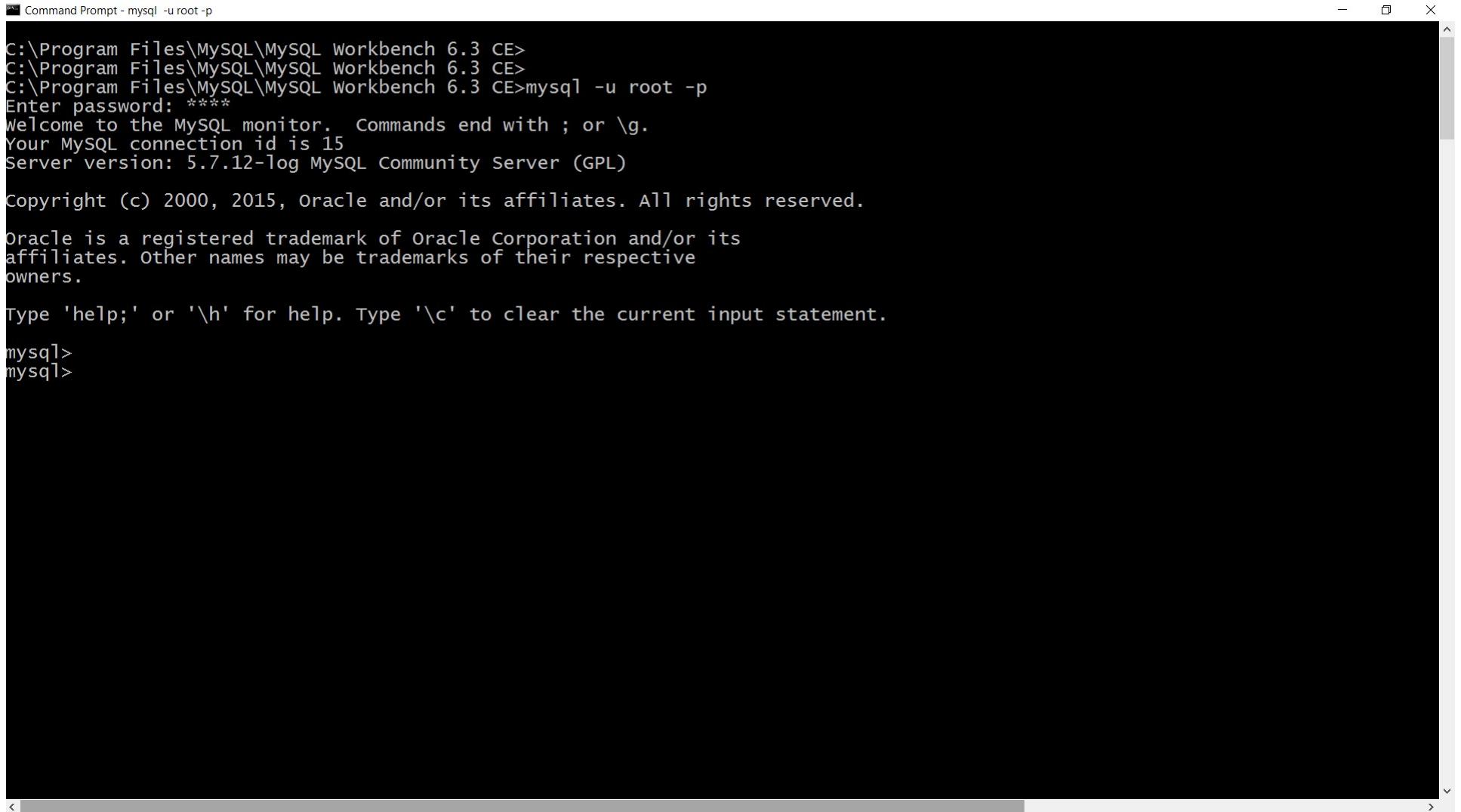
MySQL is...

- **Inexpensive.** Free for most uses and relatively inexpensive for other uses.
- **Fast.** One of the fastest relational databases currently available.
- **Easy to use.** Easy to install and use.
- **Portable.** Runs on most modern operating systems including Windows, OS X, and Linux.

MySQL provides...

- **Support for SQL.** Like any modern database product, MySQL supports SQL.
- **Support for multiple clients.** Supports access from multiple clients from a variety of interfaces and programming languages including Java, PHP, Python, Perl, and C.
- **Connectivity.** Provides access to data via an intranet or the Internet.
- **Security.** Protects access to your data so only authorized users can view the data.
- **Referential integrity.** With MySQL 5.5 and later, InnoDB tables are used by default, which support referential integrity.
- **Transaction processing.** With version 5.5, MySQL uses InnoDB tables by default, which provide support for transaction processing.

A command-line tool – I logged in as root



Command Prompt - mysql -u root -p

```
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.7.12-log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql>
```

A command-line tool – Use murach

```
Command Prompt - mysql -u root -p

C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 15
Server version: 5.7.12-Log MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> use murach
Database changed
mysql>
mysql> select * from user;
+-----+-----+-----+-----+
| UserID | Email           | FirstName | LastName |
+-----+-----+-----+-----+
|     1  | jsmith@gmail.com | John      | Smith    |
|     2  | andi@murach.com | Andrea   | Steelman |
|     3  | joelmurach@yahoo.com | Joel    | Murach   |
|     4  | bobrinkin@gmail.com | Bob     | Rinkin   |
|     5  | TimKameniski@gmail.com | Tim    | Kameniski |
|     9  | xyz@gmail.com | X        | YZ       |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql> █
```

MySQL Workbench

```
Command Prompt

C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>dir mysqlworkbench*
Volume in drive C is Windows8_OS
Volume Serial Number is 12AF-F3E4

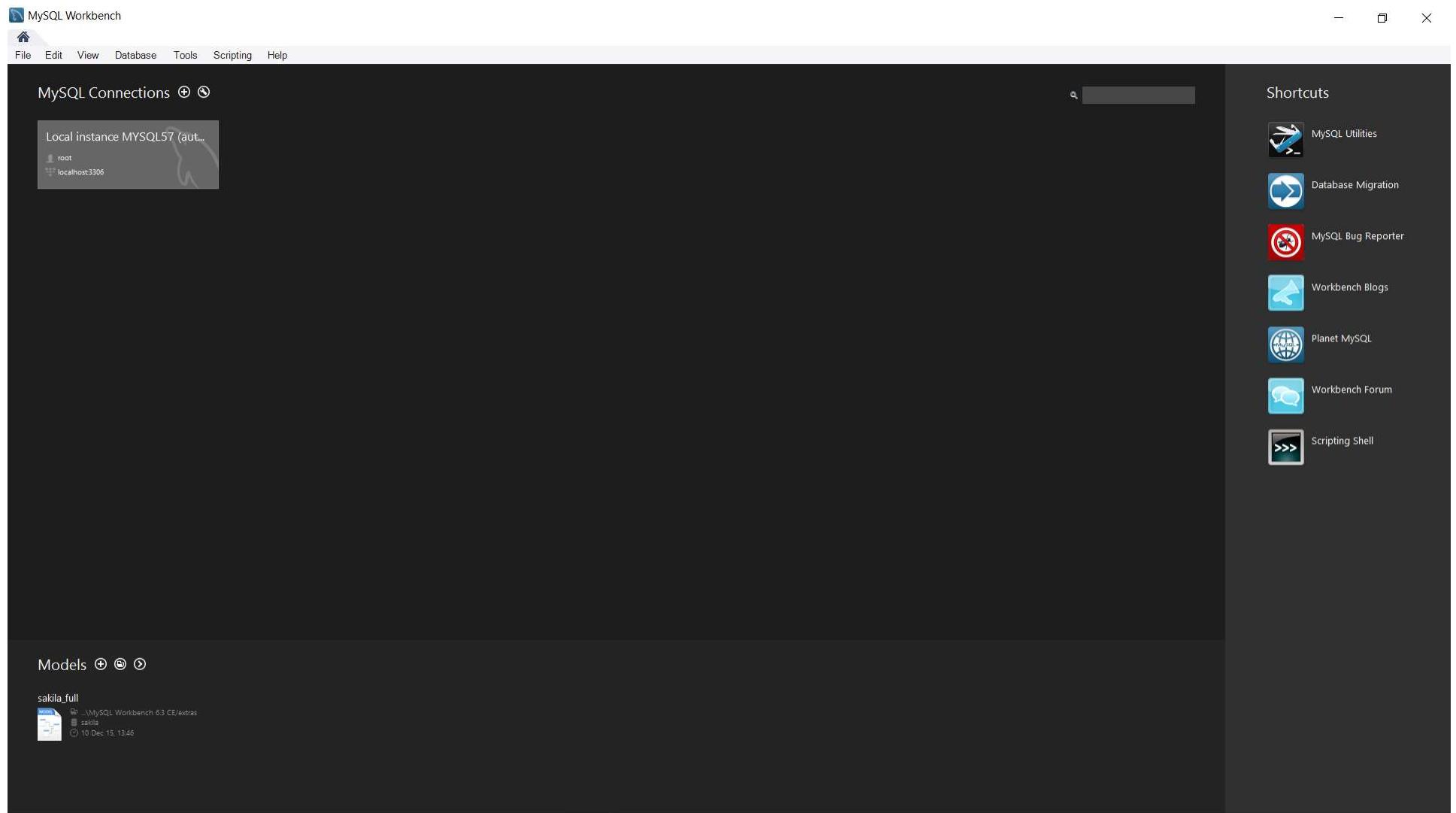
Directory of C:\Program Files\MySQL\MySQL Workbench 6.3 CE

12/10/2015  03:24 PM      515,584 MySQLWorkbench.exe
12/10/2015  02:46 PM        199 MySQLWorkbench.exe.config
11/03/2015  10:17 AM     82,726 MySQLWorkbenchDoc.ico
                   3 File(s)   598,509 bytes
                   0 Dir(s)  170,848,243,712 bytes free

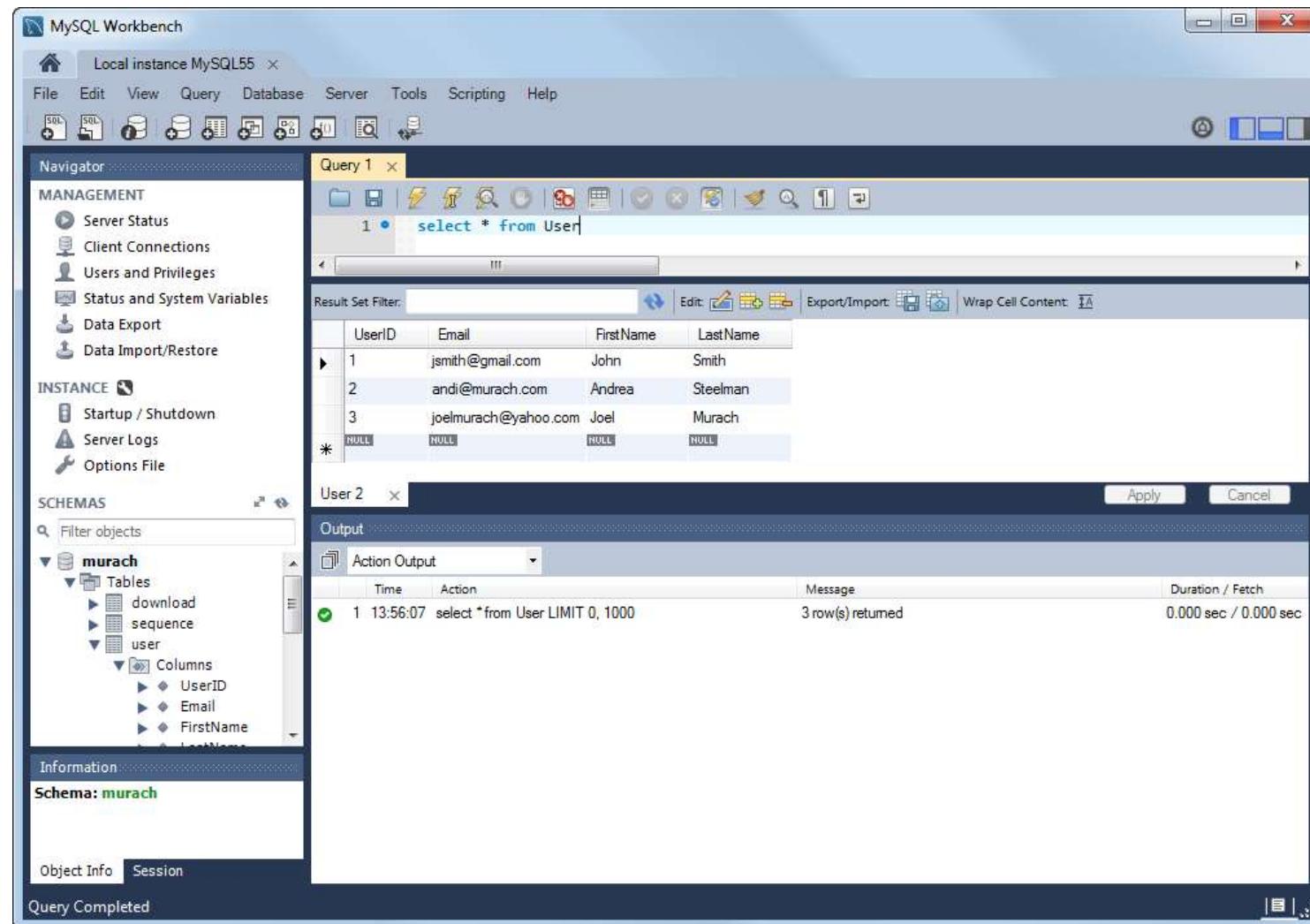
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
C:\Program Files\MySQL\MySQL Workbench 6.3 CE>mysqlworkbench

C:\Program Files\MySQL\MySQL Workbench 6.3 CE>
```

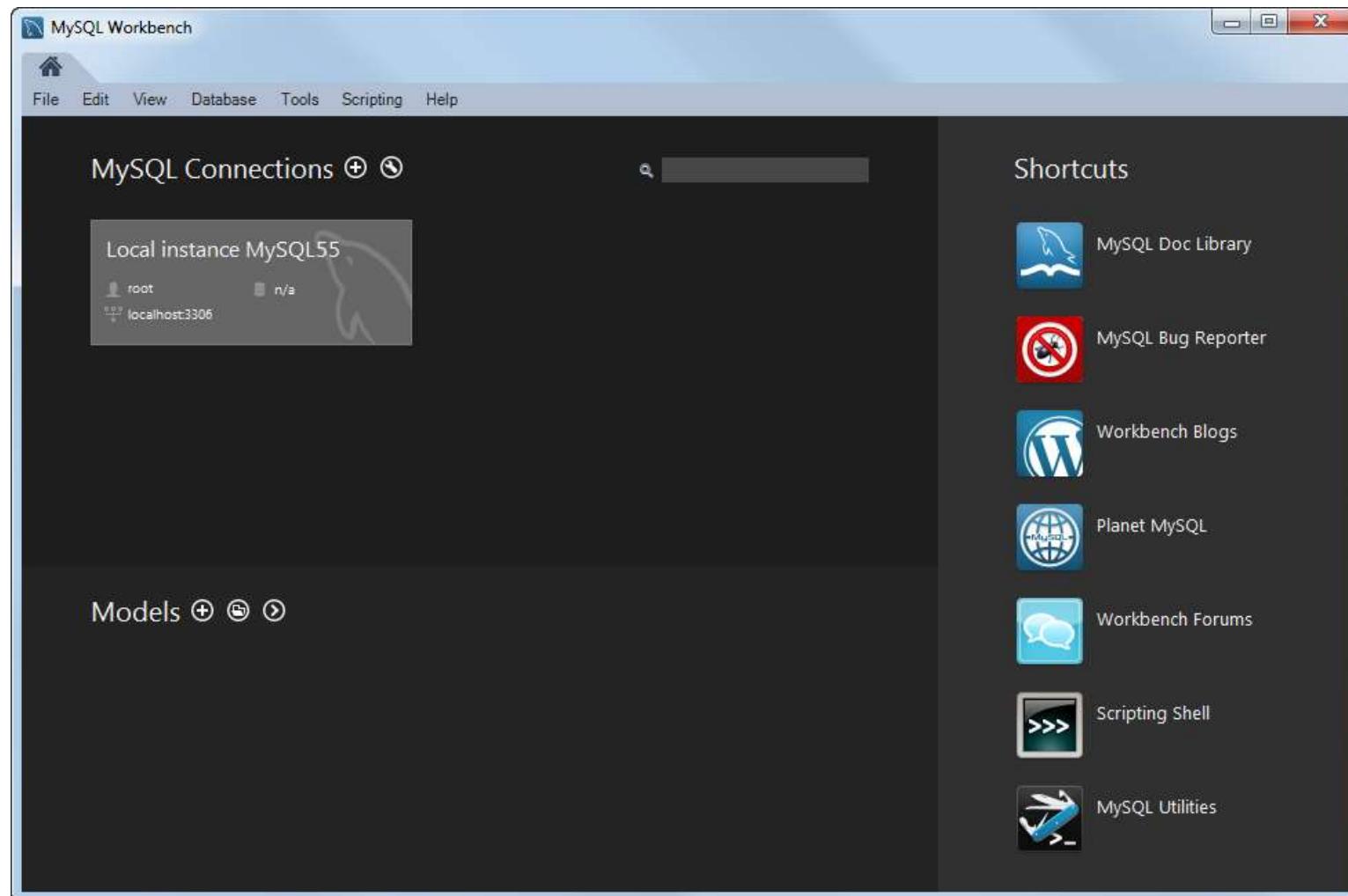
MySQL Workbench



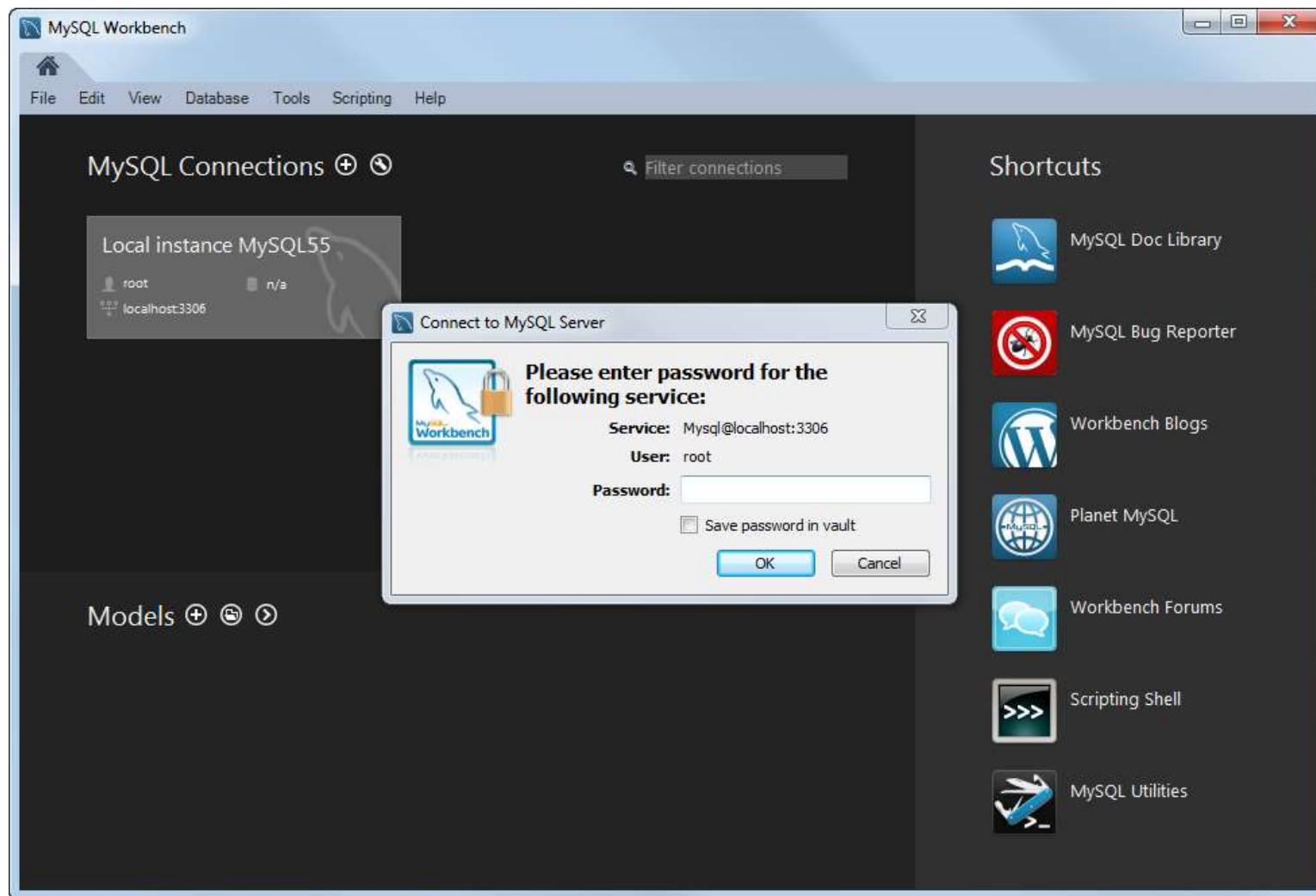
MySQL Workbench



The Home tab of MySQL Workbench



The dialog box for opening database connections



The Startup/Shutdown option

MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator Local instance MySQL... x

MANAGEMENT Local instance MySQL57

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE Local instance MySQL57

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE Local instance MySQL57

- Dashboard
- Performance Reports
- Performance Schema Setup

SCHEMAS Filter objects

No connected

Information No object selected

Object Info Session Refresh Status

Administration - Startup /...

Startup / Shutdown MySQL Server

The database server is stopped. To start the Server, use the "Start Server" button

The database server instance is **stopped** **Start Server**

If you stop the server, you and your applications will not be able to use the Database and all current connections will be closed

Startup Message Log

```
2017-04-30T15:38:21 0 Note Shutting down plugin 'ngram'
2017-04-30T15:38:21 0 Note Shutting down plugin 'partition'
2017-04-30T15:38:21 0 Note Shutting down plugin 'BLACKHOLE'
2017-04-30T15:38:21 0 Note Shutting down plugin 'ARCHIVE'
2017-04-30T15:38:21 0 Note Shutting down plugin 'PERFORMANCE_SCHEMA'
2017-04-30T15:38:21 0 Note Shutting down plugin 'MYISAM'
2017-04-30T15:38:21 0 Note Shutting down plugin 'ISAM'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_VIRTUAL'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_DATAFILES'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_TABLESPACES'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_FOREIGN'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_FIELDS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_COLUMNS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_INDEXES'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_SYS_TABLESTATS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_FT_INDEX_TABLE'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_FT_INDEX_CACHE'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_FT_CONFIG'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_FT_BEING_DELETED'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_FT_DELETED'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_FT_DEFAULT_STOPWORD'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_METRICS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_TEMP_TABLE_INFO'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_BUFFER_POOL_STATS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_BUFFER_PAGE_LRU'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_CMP_PER_INDEX_RESET'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_CMP_PER_INDEX'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_CMPMEM_RESET'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_CMPMEM'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_CMP_RESET'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_LOCK_WAITS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_LOCKS'
2017-04-30T15:38:21 0 Note Shutting down plugin 'INNODB_TRX'
2017-04-30T15:38:21 0 Note Shutting down plugin 'InnoDB'
2017-04-30T15:38:21 0 Note Innodb: FTS optimize thread exiting.
2017-04-30T15:38:21 0 Note Innodb: Starting shutdown...
2017-04-30T15:38:21 0 Note Innodb: Dumping buffer pool(s) to C:\ProgramData\MySQL\MySQL Server 5.7\Data\ib_buffer_pool
2017-04-30T10:38:22 0 Note Innodb: Buffer pool(s) dump completed at 170430 10:38:21
2017-04-30T10:38:22 - MySQL server is currently running
2017-04-30T10:38:22 - Checking server status...
2017-04-30T10:38:22 - Can't connect to MySQL server on '127.0.0.1' (10061) (2003)
2017-04-30T10:38:22 - Assuming server is not running
2017-04-30T10:38:22 - Server is stopped
2017-04-30T10:38:31 - Starting server...
```

Please enter password for the following service:

Service: Mysql@localhost:3306

User: root

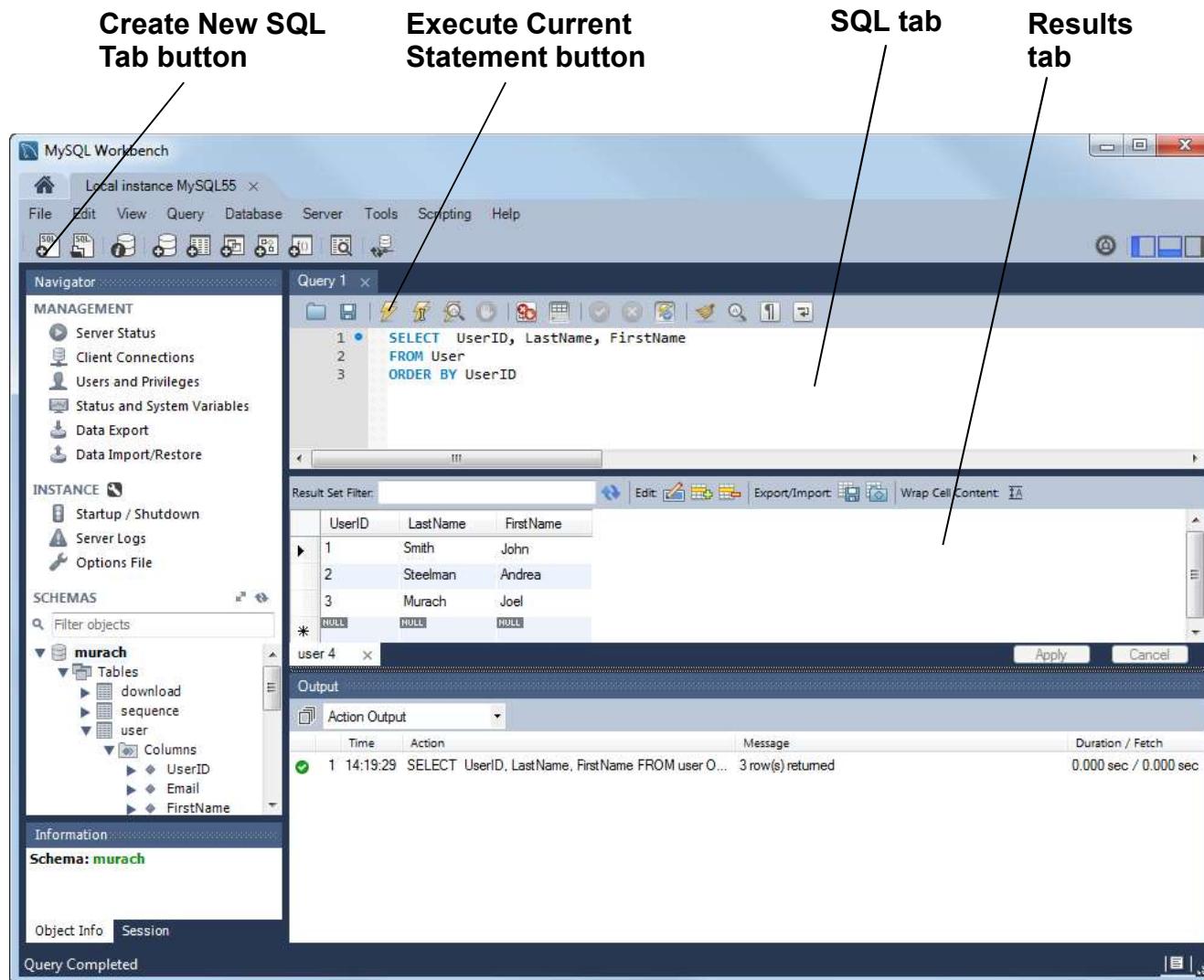
Password: ****

Save password in vault

OK Cancel

Clear Messages Copy to Clipboard

A SELECT statement and its results

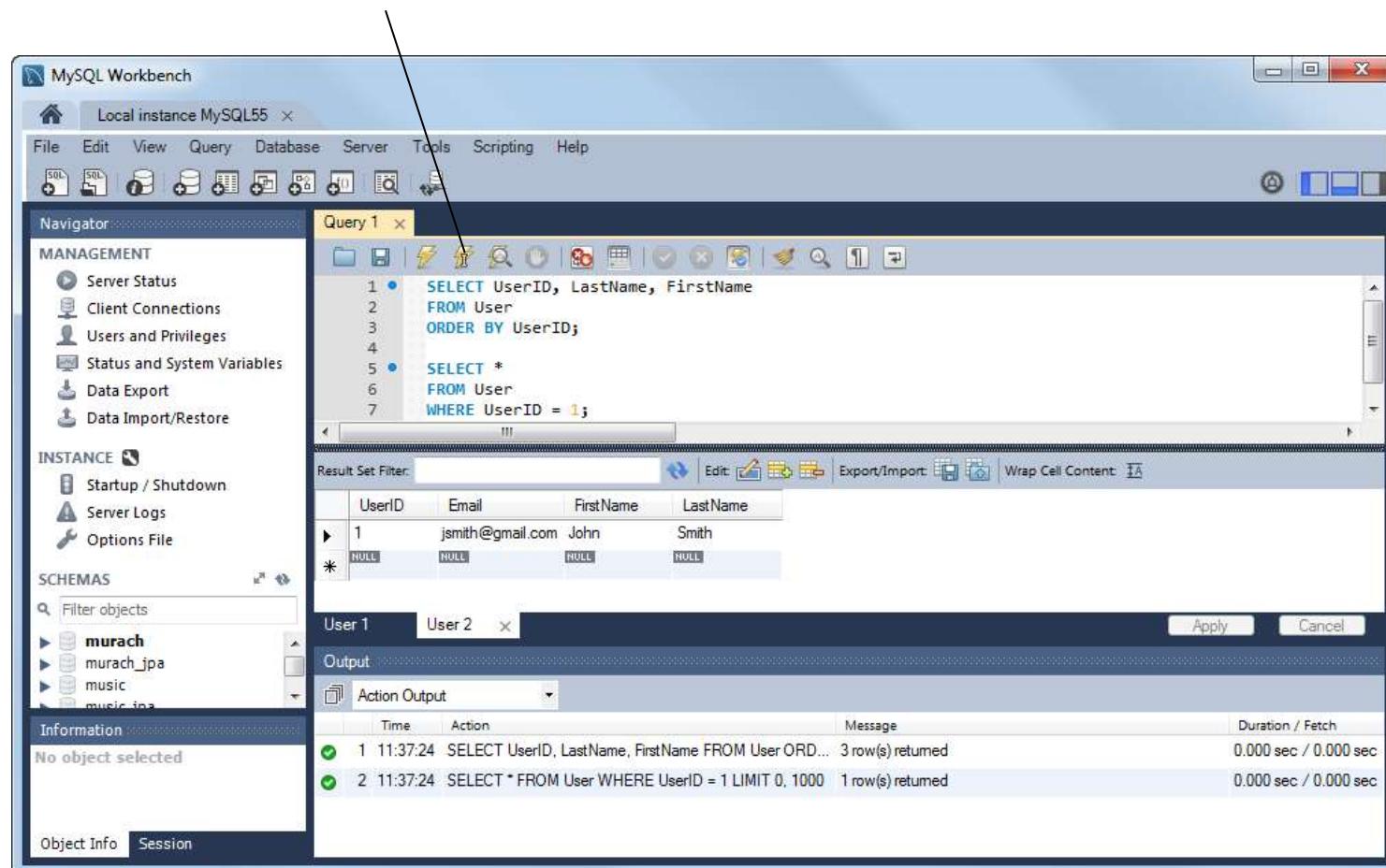


How to enter and execute a SQL statement

- To open a new SQL tab, press Ctrl+T or click the Create New SQL Tab button in the SQL editor toolbar.
- To select the current database, double-click it in the Schemas section of the Navigator window. This displays the selected database in bold.
- To enter a SQL statement, type it into the SQL tab.
- As you enter the text for a statement, the SQL tab applies color to various elements, such as SQL keywords, to make them easy to identify.
- To execute a SQL statement, press Ctrl+Enter, or click the Execute Current Statement button in the SQL editor toolbar. If the statement retrieves data, the data is displayed in a Results tab below the SQL tab.

A SQL script and its results

Execute SQL Script
button



How to enter and execute a SQL script

- When you code a script that contains more than one statement, code a semicolon at the end of each statement.
- To run an entire SQL script, press the Ctrl+Shift+Enter keys or click the Execute SQL Script button that's located just to the left of the Execute Current Statement button in the SQL editor toolbar.
- When you run a SQL script, the results of each statement that returns data are displayed in a separate Results tab.
- To execute one SQL statement within a script, move the insertion point into that statement and press Ctrl+Enter or click the Execute Current Statement button. If the statement retrieves data, the data is displayed in a Results tab.
- To execute two or more statements within a script, select them in the editor and then press Ctrl+Shift+Enter or click the Execute SQL Script button.

After a statement has been executed

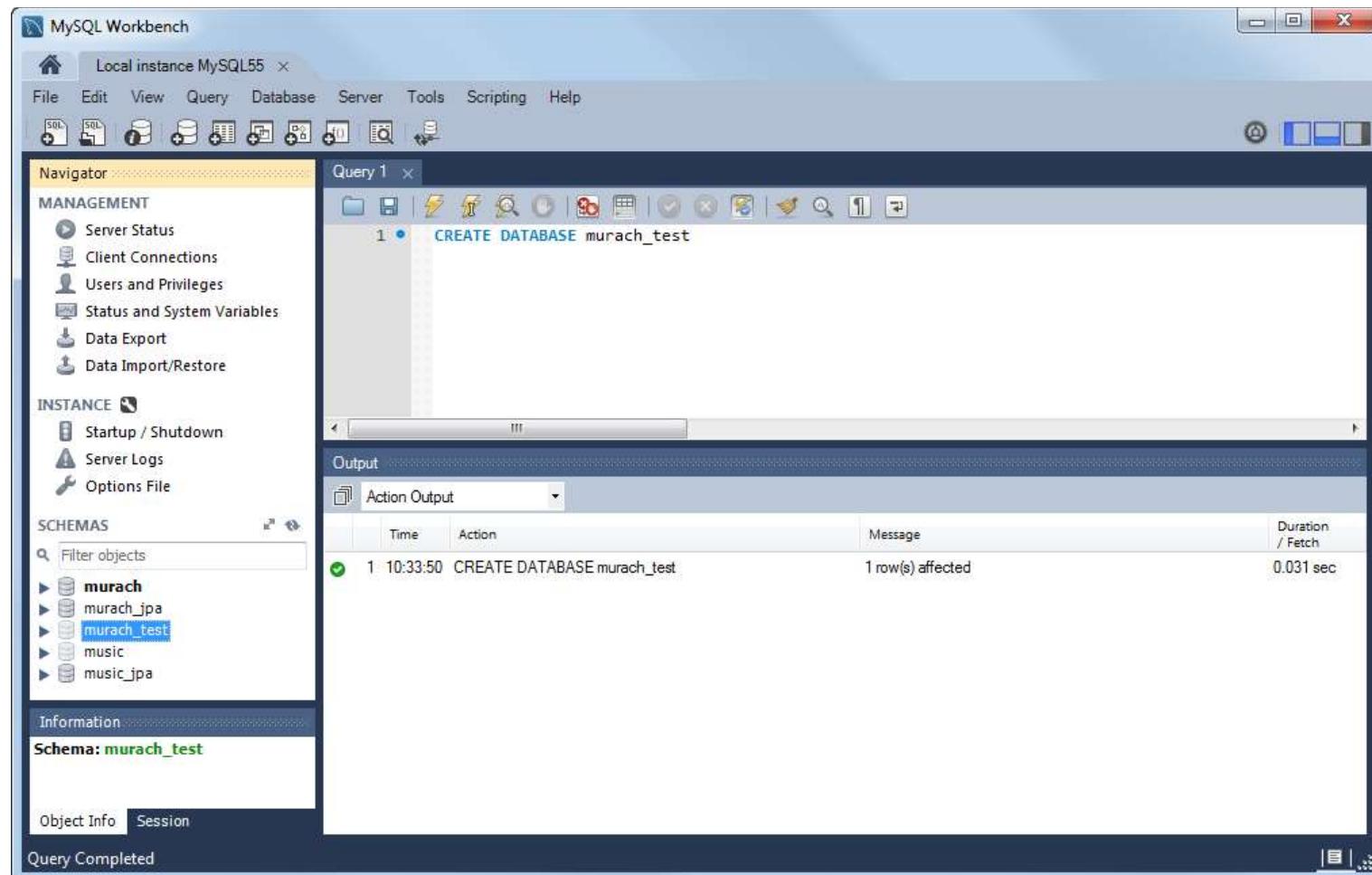
The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** MANAGEMENT (Server Status, Client Connections, Users and Privileges, Status and System Variables, Data Export, Data Import/Restore), INSTANCE (Startup / Shutdown, Server Logs, Options File), PERFORMANCE (Dashboard, Performance Reports, Performance Schema Setup), SCHEMAS (murach, userrole, download, user, userpass, userrole, Views, Stored Procedures, Functions, murach_jpa, music_jpa, util).
- SQL Editor:** SQL File 4* containing the query: `select * from user;`
- Result Grid:** Displays the results of the query:

User ID	Email	First Name	Last Name
1	jsmith@gmail.com	John	Smith
2	andi@murach.com	Andrea	Steelman
3	joelmurach@yahoo.com	Joel	Murach
4	bobrinkin@gmail.com	Bob	Rinkin
5	TimKameniski@gmail.com	Tim	Kameniski
9	xyz@gmail.com	X	YZ

- SQL Additions:** Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
- Right Panel:** Result Grid, Form Editor, Field Types, Query Stats, Execution Plan.
- Bottom:** Object Info, Session, user 1, Apply, Revert, Context Help, Snippets.

After a statement has been executed



How to create a database

```
CREATE DATABASE murach_test
```

How to select a database for use

```
USE murach_test
```

How to drop a database

```
DROP DATABASE murach_test
```

How to create, select, and drop a database

- Use the CREATE DATABASE statement to create a database and the DROP DATABASE statement to delete a database. These are *SQL statements*.
- Use the USE command to select the database that you want to work with. This is a *MySQL command*.
- You can also select a database by double-clicking on it in the Schemas section in MySQL Workbench.
- The selected database will appear in bold in the Schemas section.

How to create a table

```
CREATE TABLE User (
    UserID INT NOT NULL AUTO_INCREMENT,
    Email VARCHAR(50),
    FirstName VARCHAR(50),
    LastName VARCHAR(50),
    PRIMARY KEY (UserID)
)
```

How to drop a table

```
DROP TABLE User
```

How to drop a table only if it exists

```
DROP TABLE IF EXISTS User
```

Tables, rows, columns, etc.

- A *relational database* consists of one or more *tables* that consist of *rows (records)* and *columns (fields)*.
- The *primary key* in a table is the one that uniquely identifies each of the rows in the table.
- A *foreign key* is used to relate the rows in one table to the rows in another table.
- When you create a table, you define each of its columns and you identify its primary key.
- To define a column, you must supply the name and the data type, whether it's automatically generated for new rows, and so on.
- On Unix systems, the table and column names are case-sensitive.

An INSERT statement that inserts multiple rows

```
INSERT INTO User
  (FirstName, LastName, Email)
VALUES
  ('John', 'Smith', 'jsmith@gmail.com'),
  ('Andrea', 'Steelman', 'andi@murach.com'),
  ('Joel', 'Murach', 'joelmurach@yahoo.com')
```

How to insert multiple rows into a table

- The INSERT statement lets you insert one or more rows into one table of a database. When you code it, you need to include data for all columns that aren't defined with default values or aren't automatically generated.
- On a Unix system, table and column names are case-sensitive.

A SELECT statement that gets all columns

Syntax

```
SELECT *
FROM table-1
[WHERE selection-criteria]
[ORDER BY column-1 [ASC|DESC] [, column-2 [ASC|DESC] ...]]
```

A statement that selects all rows and columns

```
SELECT * FROM User
```

Result set

	UserID	Email	First Name	Last Name
▶	1	jsmith@gmail.com	John	Smith
	2	andi@murach.com	Andrea	Steelman
	3	joelmurach@yahoo.com	Joel	Murach
*	NULL	NULL	NULL	NULL

A SELECT statement that gets selected columns

Syntax

```
SELECT column-1 [,column-2] ...
FROM table-1
[WHERE selection-criteria]
[ORDER BY column-1 [ASC|DESC] [,column-2 [ASC|DESC] ...]]
```

A statement that selects two rows and two columns

```
SELECT FirstName, LastName
FROM User
WHERE UserID < 3
ORDER BY LastName ASC
```

Result set

	FirstName	LastName
▶	John	Smith
	Andrea	Steelman

How to select data from a single table

- A SELECT statement is a SQL DML statement that returns a *result set* (or *result table*) that consists of the specified rows and columns.
- To specify the columns, use the SELECT clause.
- To specify the rows, use the WHERE clause.
- To specify the table that the data should be retrieved from, use the FROM clause.
- To specify how the result set should be sorted, use the ORDER BY clause.

A SELECT statement that joins two tables

Syntax

```
SELECT column-1 [,column-2] ...
FROM table-1
    {INNER | LEFT OUTER | RIGHT OUTER} JOIN table-2
        ON table-1. column-1 {=|<|>|<=|>=|<>} table-2.column-2
[WHERE selection-criteria]
[ORDER BY column-1 [ASC|DESC] [,column-2 [ASC|DESC] ...]]
```

A SELECT statement that joins two tables (cont.)

A statement that joins the User and Download tables

```
SELECT Email, DownloadFilename, DownloadDate  
FROM User  
    INNER JOIN Download  
        ON User.UserID = Download.UserID  
WHERE DownloadDate > '2014-01-01'  
ORDER BY Email ASC
```

Result set

	Email	DownloadFilename	DownloadDate
▶	andi@murach.com	jr01_filter.mp3	2014-03-27 13:05:39
	joelmurach@yahoo.com	jr01_so_long.mp3	2014-03-27 13:05:39
	jsmith@gmail.com	jr01_so_long.mp3	2014-02-01 00:00:00
	jsmith@gmail.com	jr01_filter.mp3	2014-03-27 13:05:39

How to select data from multiple tables

- To return a result set that contains data from two tables, join the tables. To do that, use a JOIN clause. Most of the time, you'll want to code an *inner join* so that rows are only included when the key of a row in the first table matches the key of a row in the second table.
- In a *left outer join*, the data for all of the rows in the first table (the one on the left) are included in the table, but only the data for matching rows in the second table are included. In a *right outer join*, the reverse is true.
- An inner join is the default type of join. As a result, it's common to omit the INNER keyword from a SELECT statement for an inner join.

The INSERT statement

Syntax

```
INSERT INTO table-name [(column-list)]
VALUES (value-list)
```

A statement that adds one row to the Download table

```
INSERT INTO Download
  (UserID, DownloadDate, DownloadFilename, ProductCode)
VALUES
  (1, '2014-05-01', 'jr01_so_long.mp3', 'jr01')
```

A statement that uses MySQL's NOW function to get the current date

```
INSERT INTO Download
  (UserID, DownloadDate, DownloadFilename, ProductCode)
VALUES
  (1, NOW(), 'jr01_filter.mp3', 'jr01')
```

The UPDATE statement

Syntax

```
UPDATE table-name  
SET expression-1 [, expression-2] ...  
WHERE selection-criteria
```

**A statement that updates the FirstName column
in one row**

```
UPDATE User  
SET FirstName = 'Jack'  
WHERE Email = 'jsmith@gmail.com'
```

**A statement that updates the ProductPrice column
in selected rows**

```
UPDATE Product  
SET ProductPrice = 36.95  
WHERE ProductPrice = 36.50
```

The DELETE statement

Syntax

```
DELETE FROM table-name  
WHERE selection-criteria
```

**A statement that deletes one row
from the User table**

```
DELETE FROM User WHERE Email = 'jsmith@gmail.com'
```

**A statement that deletes selected rows
from the Downloads table**

```
DELETE FROM Download WHERE DownloadDate < '2014-06-01'
```

How to insert, update, and delete data

- INSERT, UPDATE, and DELETE statements modify the data that's stored in a database, but they don't return a result set. Instead, they return the number of rows that were affected by the query.
- These statements are sometimes referred to as *action queries*.

How to use JDBC to work with a database

Objectives

Applied

1. Develop data access classes that use JDBC to provide all of the methods that your servlets need to work with a database.
2. Develop a utility class that allows you to get a connection from a connection pool.
3. Develop servlets that use the methods of your data classes.

Objectives (continued)

Knowledge

1. Describe how a web application can use the DriverManager, Connection, Statement, PreparedStatement, and ResultSet classes to get data from a database.
2. Explain how prepared statements can improve the performance and security of database operations.
3. Describe the use of a ResultSetMetaData object.
4. Explain how connection pooling can improve the performance of a web application.
5. Describe O/R (object-relational) mapping.

The four types of JDBC database drivers

- | | |
|--------|---|
| Type 1 | A <i>JDBC-ODBC bridge driver</i> converts JDBC calls into ODBC calls that access the DBMS protocol. |
| Type 2 | A <i>native protocol partly Java driver</i> converts JDBC calls into calls in the native DBMS protocol. |
| Type 3 | A <i>net protocol all Java driver</i> converts JDBC calls into a net protocol that's independent of any native DBMS protocol. |
| Type 4 | A <i>native protocol all Java driver</i> converts JDBC calls into a native DBMS protocol. |

How to make a database driver available to an application

- Before you can use a database driver, you must make it available to your application. To do this, add the JAR file for the driver to your application classpath.
- To add the MySQL JDBC driver to your application, add the jar file to class libraries folder.

Database URL syntax

```
jdbc:subprotocolName:databaseURL
```

How to connect to a MySQL database with automatic driver loading

```
try {
    String dbURL = "jdbc:mysql://localhost:3306/murach";
    String username = "root";
    String password = "sesame";
    Connection connection = DriverManager.getConnection(
        dbURL, username, password);
} catch(SQLException e) {
    for (Throwable t : e)
        t.printStackTrace();
}
```

How to connect to an Oracle database with automatic driver loading

```
Connection connection = DriverManager.getConnection(
    "jdbc:oracle:thin@localhost/murach", "scott", "tiger");
```

How to explicitly load a database driver

```
try {
    Class.forName("com.mysql.jdbc.Driver");
} catch(ClassNotFoundException e) {
    e.printStackTrace();
}
```

How to connect to a database

- Before you can get or modify the data in a database, you need to connect to it. To do that, use the `getConnection` method of the `DriverManager` class to return a `Connection` object.
- When you use the `getConnection` method of the `DriverManager` class, you must supply a URL for the database, a username, and a password. This method throws an `SQLException`.
- With JDBC 4.0, the `SQLException` class implements the `Iterable` interface. As a result, use an enhanced for statement to loop through any nested exceptions.
- **With JDBC 4.0, the database driver is loaded automatically. This is known as *automatic driver loading*.**
- Typically, you only need to connect to one database for an application. However, it's possible to load multiple database drivers and establish connections to multiple types of databases.

How to create a result set with 1 row and 1 column

```
Statement statement = connection.createStatement();
ResultSet userIDResult = statement.executeQuery(
    "SELECT UserID FROM User " +
    "WHERE Email = 'jsmith@gmail.com'");
```

How to create a result set with multiple columns and rows

```
Statement statement = connection.createStatement();
ResultSet products = statement.executeQuery(
    "SELECT * FROM Product");
```

How to move the cursor to the first row

```
boolean userIDExists = userIDResult.next();
```

How to loop through a result set

```
while (products.next()) {
    // statements that process each row
}
```

ResultSet methods for forward-only, read-only result sets

Method	Description
<code>next()</code>	Moves the cursor to the next row in the result set.
<code>last()</code>	Moves the cursor to the last row in the result set.
<code>close()</code>	Releases the result set's resources.
<code>getRow()</code>	Returns an int value that identifies the current row of the result set.

How to return a result set and move the cursor through it

- To return a *result set*, use the `createStatement` method of a `Connection` object to create a `Statement` object. Use the `executeQuery` method of the `Statement` object to execute a `SELECT` statement that returns a `ResultSet` object.
- By default, the `createStatement` method creates a forward-only, read-only result set. You can only move the *cursor* through it from the first row to the last and you can't update it. This is appropriate for most web applications.
- When a result set is created, the cursor is positioned before the first row.
- Use the methods of the `ResultSet` object to move the cursor.
- To move the cursor to the next row, call the `next` method. If the row is valid, this method moves the cursor to the next row and returns a true value.

Methods of a ResultSet object that return data

Method	Description
<code>getXXX(int columnIndex)</code>	Returns data from the specified column number.
<code>getXXX(String columnName)</code>	Returns data from the specified column name.

Code that uses indexes to return columns

```
String code = products.getString(1);  
String description = products.getString(2);  
double price = products.getDouble(3);
```

Code that uses names to return columns

```
String code = products.getString("ProductCode");  
String description = products.getString("ProductDescription");  
double price = products.getDouble("ProductPrice");
```

Code that creates a Product object from the products result set

```
Product product = new Product(products.getString(1),  
                               products.getString(2),  
                               products.getDouble(3));
```

The getXXX methods

- The getXXX methods can be used to return all eight primitive types. For example, the getInt method returns the int type and the getLong method returns the long type.
- The getXXX methods can also be used to return strings, dates, and times. For example, the getString method returns any object of the String class, and the getDate, getTime, and getTimestamp methods return objects of the Date, Time, and Timestamp classes of the java.sql package.

How to use the executeUpdate method to...

Add a row

```
String query =  
    "INSERT INTO Product (ProductCode, ProductDescription, ProductPrice) " +  
    "VALUES ('" + product.getCode() + "', " +  
        "'" + product.getDescription() + "', " +  
        "'" + product.getPrice() + "')";  
Statement statement = connection.createStatement();  
int rowCount = statement.executeUpdate(query);
```

Update a row

```
String query = "UPDATE Product SET " +  
    "ProductCode = '" + product.getCode() + "', " +  
    "ProductDescription = '" + product.getDescription() + "', " +  
    "ProductPrice = '" + product.getPrice() + "' " +  
    "WHERE ProductCode = '" + product.getCode() + "'";  
Statement statement = connection.createStatement();  
int rowCount = statement.executeUpdate(query);
```

How to use the executeUpdate method to...(cont.)

Delete a row

```
String query = "DELETE FROM Product " +
               "WHERE ProductCode = '" + productCode + "'";
Statement statement = connection.createStatement();
int rowCount = statement.executeUpdate(query);
```

How to insert, update, and delete data

- The executeUpdate method is an older method that works with most JDBC drivers. Some newer methods require less SQL code, but they may not work properly with all JDBC drivers.
- The executeUpdate method returns an int value that identifies the number of rows affected by the SQL statement.

Warning

- If you build an SQL statement from user input and use a method of the Statement object to execute that SQL statement, you may be susceptible to an SQL injection attack.
- An *SQL injection attack* allows a hacker to bypass authentication or execute SQL statements against your database that can read data, modify data, or delete data.
- To prevent most types of SQL injection attacks, use prepared statements.

How to use a prepared statement

To return a result set

```
String preparedSQL = "SELECT ProductCode, ProductDescription, "
+ "          ProductPrice "
+ "FROM Product WHERE ProductCode = ?";
PreparedStatement ps = connection.prepareStatement(preparedSQL);
ps.setString(1, productCode);
ResultSet product = ps.executeQuery();
```

To modify a row

```
String preparedSQL = "UPDATE Product SET "
+ "      ProductCode = ?, "
+ "      ProductDescription = ?, "
+ "      ProductPrice = ?"
+ "WHERE ProductCode = ?";
PreparedStatement ps = connection.prepareStatement(preparedSQL);
ps.setString(1, product.getCode());
ps.setString(2, product.getDescription());
ps.setDouble(3, product.getPrice());
ps.setString(4, product.getCode());
ps.executeUpdate();
```

How to use a prepared statement (continued)

To insert a row

```
String preparedQuery =
    "INSERT INTO Product "
    + "(ProductCode, ProductDescription, ProductPrice) "
    + "VALUES "
    + "(?, ?, ?)";

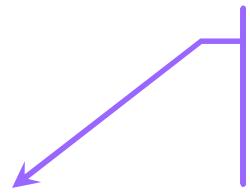
PreparedStatement ps = connection.prepareStatement(preparedQuery);
ps.setString(1, product.getCode());
ps.setString(2, product.getDescription());
ps.setDouble(3, product.getPrice());
ps.executeUpdate();
```

To delete a row

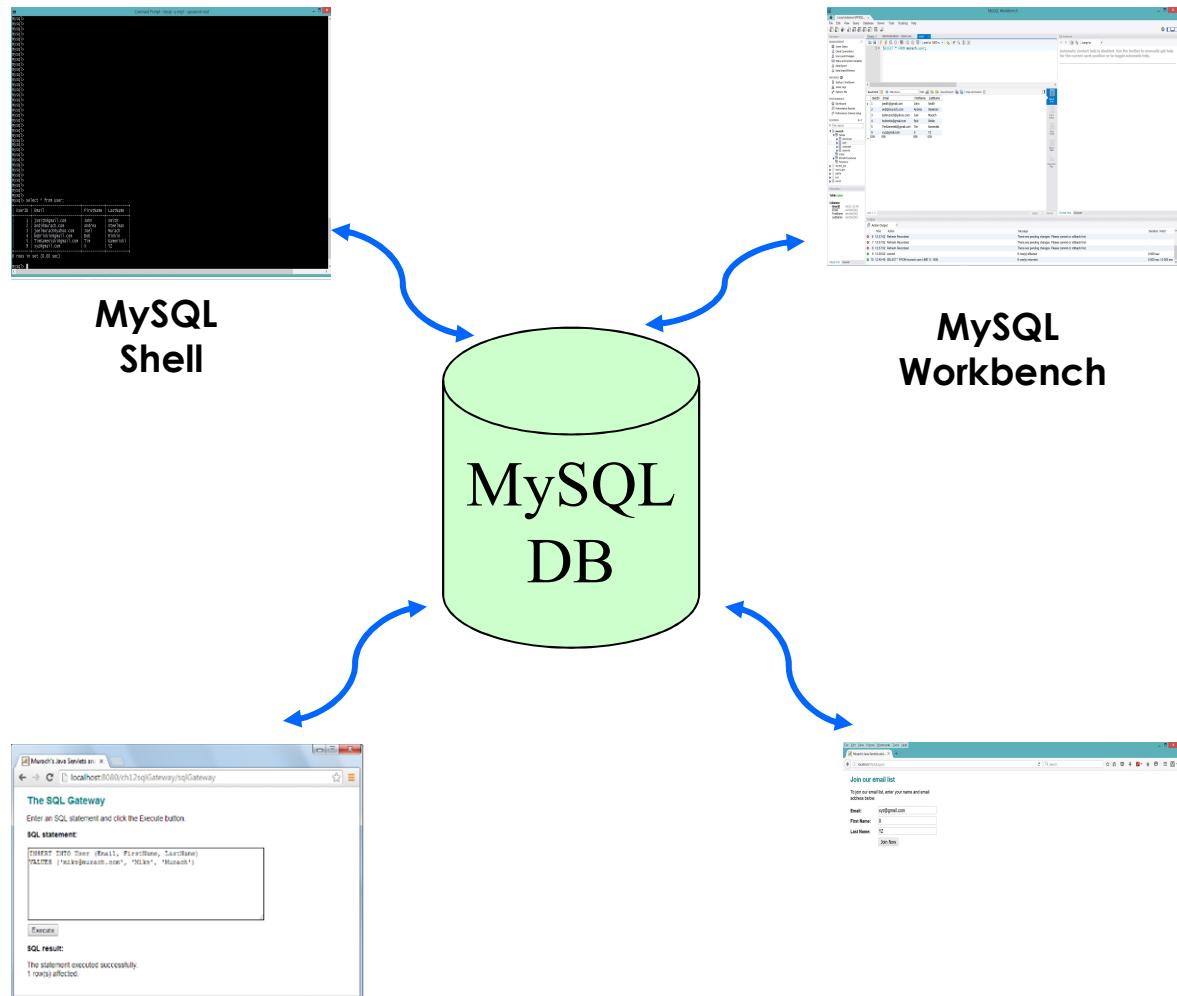
```
String preparedQuery = "DELETE FROM Product "
                      + "WHERE ProductCode = ?";

PreparedStatement ps = connection.prepareStatement(preparedQuery);
ps.setString(1, productCode);
ps.executeUpdate();
```

This is the skyview
for Chicago



This is the skyview
for mySQLapp1 and
mySQLapp2



MySQLapp1

MySQLapp2

The SQL Gateway application after executing an INSERT statement

This is the servlet
used to insert the
row below

A screenshot of a web browser window titled "Murach's Java Servlets and JSP". The address bar shows the URL "localhost:8080/ch12sql/Gateway/sqlGateway". The main content area is titled "The SQL Gateway" and contains the following text:
Enter an SQL statement and click the Execute button.
SQL statement:

```
INSERT INTO User (Email, FirstName, LastName)
VALUES ('mike@murach.com', 'Mike', 'Murach')
```

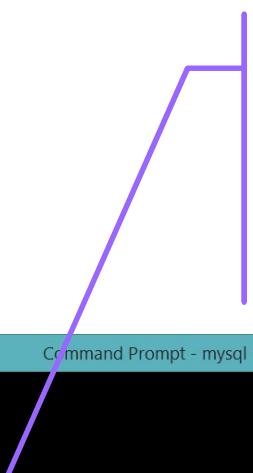

Execute
SQL result:
The statement executed successfully.
1 row(s) affected.

The SQL Gateway application after executing a SELECT statement

A screenshot of a web browser window titled "Murach's Java Servlets and JSP". The address bar shows "localhost:8080/ch12sqlGateway/sqlGateway". The page content is titled "The SQL Gateway" and contains instructions: "Enter an SQL statement and click the Execute button." Below this is a text input field labeled "SQL statement:" containing the query "SELECT * FROM User". An "Execute" button is located below the input field. To the right of the input field, a purple callout box with a diagonal line points to the browser window, containing the text: "This is the servlet used to retrieve what got inserted". Below the input field, under the heading "SQL result:", is a table displaying the retrieved data:

UserID	Email	FirstName	LastName
1	jsmith@gmail.com	John	Smith
2	andi@murach.com	Andrea	Steelman
3	joelmurach@yahoo.com	Joel	Murach
4	mike@murach.com	Mike	Murach

You could also use
MySQL shell to see what's
in the database



```
Command Prompt - mysql -u eng1 -p
c:\MySQL\MySQL Server 5.7\bin>mysql -u eng1 -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 59
Server version: 5.7.12-log MySQL Community Server (GPL)

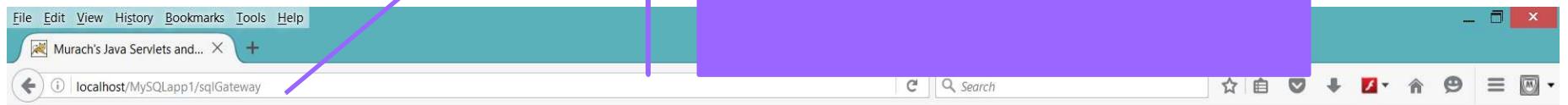
Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

This is the servlet used
to query the table
before inserting a row



The SQL Gateway

Enter an SQL statement and click the Execute button.

SQL statement:

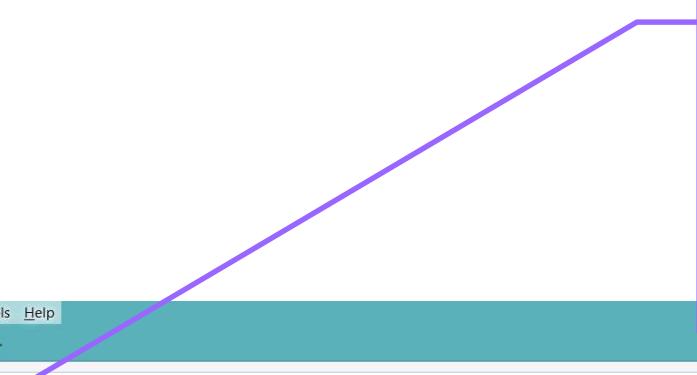
```
select * from User
```

Execute

SQL result:

UserID	Email	FirstName	LastName
1	jsmith@gmail.com	John	Smith
2	andi@murach.com	Andrea	Steelman
3	joelmurach@yahoo.com	Joel	Murach
4	bobrinkin@gmail.com	Bob	Rinkin
5	TimKameniski@gmail.com	Tim	Kameniski

This is the index.jsp
used to call
EmailListServlet to
insert a row into DB



File Edit View History Bookmarks Tools Help

Murach's Java Servlets and... X +

localhost/MySQLapp2/ Search ☆ 📁 🌐 🌐 🌐 🌐 🌐 🌐

Join our email list

To join our email list, enter your name and email address below.

Email:

First Name:

Last Name:



This is the servlet
used to insert the
row below

Thanks for joining our email list

Here is the information that you entered:

Email: xyz@gmail.com

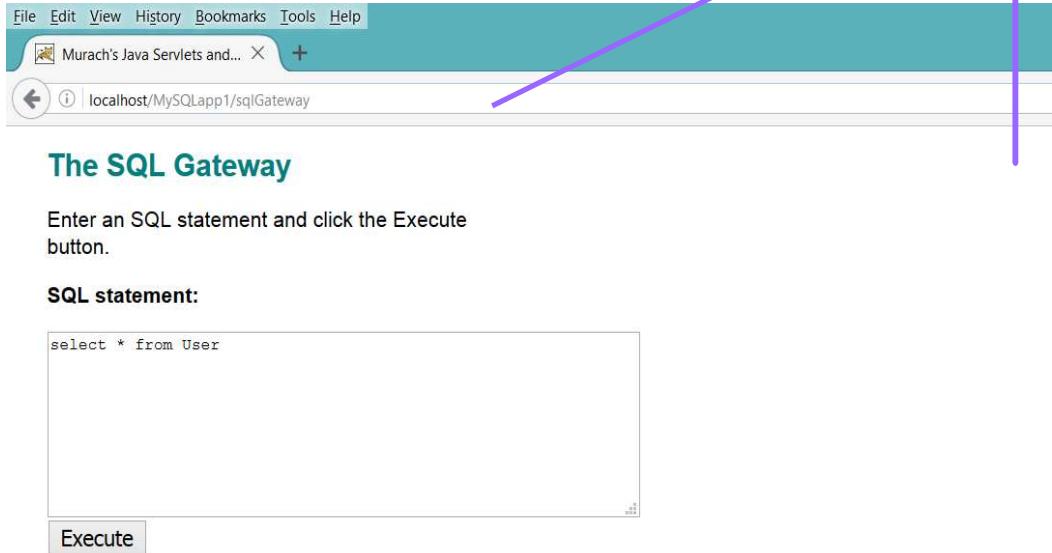
First Name: X

Last Name: YZ

To enter another email address, click on the Back button in your browser or the Return button shown below.

[Return](#)

This is the servlet used
to query the table
after inserting a row



The SQL Gateway

Enter an SQL statement and click the Execute button.

SQL statement:

```
select * from User
```

Execute

SQL result:

UserID	Email	FirstName	LastName
1	jsmith@gmail.com	John	Smith
2	andi@murach.com	Andrea	Steelman
3	joelmurach@yahoo.com	Joel	Murach
4	bobrinkin@gmail.com	Bob	Rinkin
5	TimKameniski@gmail.com	Tim	Kameniski
9	xyz@gmail.com	X	YZ

From MySQL shell
query the table after
inserting the row

```
mysql>
mysql> select * from user;
+-----+-----+-----+-----+
| UserID | Email           | FirstName | LastName |
+-----+-----+-----+-----+
|     1  | jsmith@gmail.com | John      | Smith    |
|     2  | andi@murach.com | Andrea    | Steelman |
|     3  | joelmurach@yahoo.com | Joel      | Murach   |
|     4  | bobrinkin@gmail.com | Bob       | Rinkin   |
|     5  | TimKameniski@gmail.com | Tim      | Kameniski |
|     9  | xyz@gmail.com    | X         | YZ        |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

mysql>
```

From Workbench query the table after inserting the row

The screenshot shows the MySQL Workbench interface. On the left is the Navigator pane with sections for MANAGEMENT, INSTANCE, PERFORMANCE, SCHEMAS, and INFORMATION. Under SCHEMAS, the 'murach' database is selected, showing its tables: download, user, userpass, userrole, views, stored procedures, and functions. The 'user' table is selected in the 'Tables' section.

In the center, a 'Query 1' window displays the result of the query `SELECT * FROM murach.user;`. The results are shown in a 'Result Grid':

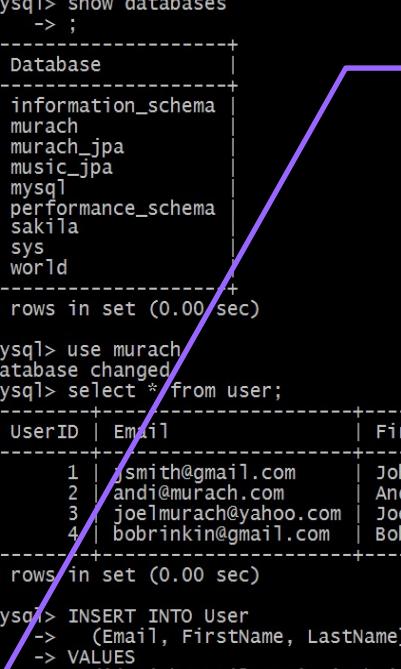
UserID	Email	FirstName	LastName
1	jsmith@gmail.com	John	Smith
2	andi@murach.com	Andrea	Steelman
3	joelmurach@yahoo.com	Joel	Murach
4	bobrinkin@gmail.com	Bob	Rinkin
5	TimKameniski@gmail.com	Tim	Kameniski
9	xyz@gmail.com	X	YZ
*	NULL	NULL	NULL

A purple arrow points from the text 'From Workbench query the table after inserting the row' to the 'user' table in the Navigator pane.

On the right, an 'Output' pane shows the following log entries:

Action	Message	Duration / Fetch
Refresh Recordset	There are pending changes. Please commit or rollback first.	
Refresh Recordset	There are pending changes. Please commit or rollback first.	
Refresh Recordset	There are pending changes. Please commit or rollback first.	
commit	0 row(s) affected	0.000 sec
SELECT * FROM murach.user LIMIT 0, 1000	6 row(s) returned	0.000 sec / 0.000 sec

Verify the users, roles, and privileges



```
c:\MySQL\MySQL Server 5.7\bin>mysql -u eng1 -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 58
Server version: 5.7.12-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
    -> ;
+-----+
| Database
+-----+
| information_schema
| murach
| murach_jpa
| music_jpa
| mysql
| performance_schema
| sakila
| sys
| world
+-----+
9 rows in set (0.00 sec)

mysql> use murach
Database changed
mysql> INSERT INTO User
    -> (Email, FirstName, LastName)
    -> VALUES
    -> ('TimKameniski@gmail.com', 'Tim', 'Kameniski')
Query OK, 1 row affected (0.01 sec)

mysql>
```

```
c:\MySQL\MySQL Server 5.7\bin>mysql -u eng2 --password=root
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 57
Server version: 5.7.12-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases
    -> ;
+-----+
| Database
+-----+
| information_schema
| murach
| murach_jpa
| music_jpa
| mysql
| performance_schema
| sakila
| sys
| world
+-----+
9 rows in set (0.00 sec)

mysql> use murach
Database changed
mysql> select * from user;
+-----+
| UserID | Email           | FirstName | LastName |
+-----+
| 1     | jsmith@gmail.com | John      | Smith    |
| 2     | andi@murach.com  | Andrea   | Steelman |
| 3     | joelmurach@yahoo.com | Joel    | Murach   |
| 4     | bobrinkin@gmail.com | Bob     | Rinkin   |
+-----+
4 rows in set (0.00 sec)

mysql> INSERT INTO User
    -> (Email, FirstName, LastName)
    -> VALUES
    -> ('jsmith@gmail.com', 'John', 'Smith');
ERROR 1142 (42000): INSERT command denied to user 'eng2'@'localhost' for table 'user'
mysql>
```

If a user doesn't have
the right privileges,
you get ERROR
command denied

Giving eng1 the right privileges

Step 1: Select the user

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MYSQL5... x, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Toolbar:** Standard MySQL Workbench icons.
- Navigator:** Local instance MYSQL57, Users and Privileges.
- User Accounts Table:** Shows a list of users with their host information.

User	From Host
murach_user	localhost
abader	%
eng1	localhost
eng2	localhost
eng3	localhost
mysql.sys	localhost
root	localhost
- Details for account eng1@localhost:** Shows the schema and privileges for the eng1 user.

Schema	Privileges
murach	ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, INDEX, INSERT, RE...
- Buttons:** Revoke All Privileges, Delete Entry, Add Entry..., Unselect All, Select "All", Revert, Apply.
- Information Panel:** Schema: murach.
- Object Info and Session Buttons:** Add Account, Delete, Refresh, Revert, Apply.

Giving eng1 the right privileges

Step 2: Select system privileges

The screenshot shows the MySQL Workbench interface with the 'Users and Privileges' window open. The 'Schema Privileges' tab is selected for the user 'eng1@localhost'. In the 'Schema' column, 'murach' is listed. The 'Privileges' column shows various system privileges granted to this user, including ALTER, CREATE, and EXECUTE. The 'Object Rights' section contains checkboxes for SELECT, INSERT, UPDATE, DELETE, EXECUTE, and SHOW VIEW. The 'DDL Rights' section contains checkboxes for CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, DROP, and TRIGGER. The 'Other Rights' section contains checkboxes for GRANT OPTION, CREATE TEMPORARY TABLES, and LOCK TABLES. Buttons for 'Revoke All Privileges', 'Delete Entry...', and 'Add Entry...' are visible at the bottom.

Local instance MYSQL5... x MySQL Workbench

File Edit View Query Database Server Tools Scripting Help

Navigator: Administration - Users and...

User Accounts Details for account eng1@localhost

User	From Host
murach_user	localhost
abader	%
eng1	localhost
eng2	localhost
eng3	localhost
mysql.sys	localhost
root	localhost

Login Account Limits Administrative Roles Schema Privileges

Schema Privileges

Schema: murach

Object Rights:

- SELECT
- INSERT
- UPDATE
- DELETE
- EXECUTE
- SHOW VIEW

DDL Rights:

- CREATE
- ALTER
- REFERENCES
- INDEX
- CREATE VIEW
- CREATE ROUTINE
- ALTER ROUTINE
- EVENT
- DROP
- TRIGGER

Other Rights:

- GRANT OPTION
- CREATE TEMPORARY TABLES
- LOCK TABLES

Revoke All Privileges Delete Entry... Add Entry...

Unselect All Select "All"

Add Account Delete Refresh Revert Apply

Object Info Session

Giving eng1 the right privileges

Step 3: Select the schema

The screenshot shows the MySQL Workbench interface with the 'Administration - Users and Privileges' tab selected. On the left, the 'SCHEMAS' tree view shows the 'murach' schema expanded, revealing tables like 'download', 'user', 'userpass', and 'userrole'. In the center, the 'User Accounts' table lists users including 'eng1'. The 'Schema Priviliges' tab is active for the 'eng1' account, showing it has full privileges ('ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, INDEX, INSERT, RE...') on the 'murach' schema. The 'Object Rights' section lists various database operations like SELECT, INSERT, UPDATE, DELETE, EXECUTE, and SHOW VIEW, all of which are checked. The 'DDL Rights' section also contains many checked options, and the 'Other Rights' section includes 'GRANT OPTION', 'CREATE TEMPORARY TABLES', and 'LOCK TABLES'. Buttons at the bottom include 'Revert', 'Apply', 'Unselect All', and 'Select "ALL"'. A purple line highlights the 'Schema Priviliges' tab.

Giving eng1 the right privileges

Step 4: Select the privileges

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MYSQL57, Administration - Users an...
- Navigator:** Local instance MYSQL57, Users and Privileges.
- Management:** Server Status, Client Connections, Users and Privileges (selected), Status and System Variables, Data Export, Data Import/Restore.
- Performance:** Dashboard, Performance Reports, Performance Schema Setup.
- Schemas:** murach (selected), Tables (download, user, userpass, userrole), Views, Stored Procedures, Functions (murach_jpa, music_jpa).
- User Accounts:** User Accounts table showing users: murach_user, abader, eng1, eng2, eng3, mysql.sys, root.
- Details for account eng1@localhost:**
 - Schema Privileges:** Schema: murach, Privileges: ALTER, ALTER ROUTINE, CREATE, CREATE ROUTINE, CREATE TEMPORARY TABLES, CREATE VIEW, DELETE, DROP, EVENT, EXECUTE, INDEX, INSERT, RE...
 - Object Rights:** SELECT, INSERT, UPDATE, DELETE, EXECUTE, SHOW VIEW.
 - DDL Rights:** CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, DROP, TRIGGER.
 - Other Rights:** GRANT OPTION, CREATE TEMPORARY TABLES, LOCK TABLES.
- Buttons:** Revolve All Privileges, Delete Entry, Add Entry..., Unselect All, Select "ALL", Revert, Apply.

Giving eng2 the right privileges

The screenshot shows the MySQL Workbench interface for managing users and privileges. The left sidebar contains navigation links for Management, Instance, Performance, Schemas, and Information. The main area is titled 'Administration - Users and Privileges' and shows a list of 'User Accounts'. The 'eng2' account is highlighted with a purple line. The 'Details for account eng2@localhost' pane shows the current privileges: 'Schema' murach and 'Privileges' SELECT. Below this, there are sections for 'Object Rights' (SELECT, INSERT, UPDATE, DELETE, EXECUTE, SHOW VIEW), 'DDL Rights' (CREATE, ALTER, REFERENCES, INDEX, CREATE VIEW, CREATE ROUTINE, ALTER ROUTINE, EVENT, DROP, TRIGGER), and 'Other Rights' (GRANT OPTION, CREATE TEMPORARY TABLES, LOCK TABLES). Buttons at the bottom include 'Add Account', 'Delete', 'Refresh', 'Revert', 'Apply', 'Unselect All', 'Select "All"', 'Revokes All Privileges', 'Delete Entry...', and 'Add Entry...'. The top bar includes tabs for 'Query 1' and 'Administration - Users and Privileges'.

How to give privileges to a user on a schema

The screenshot shows the MySQL Workbench interface. On the left, the Navigator pane displays the database structure, including the 'murach' schema which contains 'Tables', 'Views', 'Stored Procedures', 'Functions', and other databases like 'information_schema', 'mysql', 'music_jpa', 'sakila', 'sys', and 'world'. The main window shows the 'Users and Privileges' administration screen for the 'Local instance MYSQL57'. It lists user accounts with their host and privileges. A modal dialog titled 'New Schema Privilege Definition' is open, prompting the user to select a schema for the 'eng2' user. The 'Selected schema:' radio button is selected, and a dropdown menu shows options: 'information_schema', 'information_schema', 'murach', 'murach_jpa', 'music_jpa', and 'mysql'. The 'murach' schema is highlighted. The background shows the 'Schema Privileges' tab of the main window, where 'eng2' is granted 'SELECT' privilege on the 'murach' schema.

Giving eng1 the right roles

Giving eng1
the right
privileges

MySQL Workbench

Local instance MYSQL57

User Accounts

User	From Host
murach_user	localhost
abader	%
eng1	localhost
eng2	localhost
eng3	localhost
mysql.sys	localhost
root	localhost

Details for account eng1 @localhost

Role Description

- DBA grants the rights to perform all tasks
- MaintenanceAdmin grants rights needed to maintain server
- ProcessAdmin rights needed to assess, monitor, and kill any user process
- UserAdmin grants rights to create users logins and reset passwords
- SecurityAdmin rights to manage logins and grant and revoke server accounts
- MonitorAdmin minimum set of rights needed to monitor server
- DBManager grants full rights on all databases
- DBDesigner rights to create and reverse engineer any database schema
- ReplicationAdmin rights needed to setup and manage replication
- BackupAdmin minimal rights needed to backup any database

Global Privileges

- ALTER
- ALTER ROUTINE
- CREATE
- CREATE ROUTINE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP
- EVENT
- EXECUTE
- FILE
- GRANT OPTION
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN
- SUPER
- TRIGGER
- UPDATE

Add Account | Delete | Refresh | Revert | Apply | Object Info | Session | Revive All Privileges

Giving eng2 the right roles

MySQL Workbench

Query 1 Administration - Users and Privileges

Local instance MySQL57

Users and Privileges

User Accounts

User	From Host
murach_user	localhost
abader	%
eng1	localhost
eng2	localhost
eng3	localhost
mysql.sys	localhost
root	localhost

Details for account eng2@localhost

Login Account Limits Administrative Roles Schema Privileges

Role Description

- DBA grants the rights to perform all tasks
- MaintenanceAdmin grants rights needed to maintain server
- ProcessAdmin rights needed to assess, monitor, and kill any user proce...
- UserAdmin grants rights to create users logins and reset passwords
- SecurityAdmin rights to manage logins and grant and revoke server an...
- MonitorAdmin minimum set of rights needed to monitor server
- DBManager grants full rights on all databases
- DBDesigner rights to create and reverse engineer any database sche...
- ReplicationAdmin rights needed to setup and manage replication
- BackupAdmin minimal rights needed to backup any database
- Custom custom role

Global Privileges

- ALTER
- ALTER ROUTINE
- CREATE
- CREATE ROUTINE
- CREATE TABLESPACE
- CREATE TEMPORARY TABLES
- CREATE USER
- CREATE VIEW
- DELETE
- DROP
- EVENT
- EXECUTE
- FILE
- GRANT OPTION
- INDEX
- INSERT
- LOCK TABLES
- PROCESS
- REFERENCES
- RELOAD
- REPLICATION CLIENT
- REPLICATION SLAVE
- SELECT
- SHOW DATABASES
- SHOW VIEW
- SHUTDOWN
- SUPER
- TRIGGER
- UPDATE

Add Account Delete Refresh

Revert Apply

Object Info Session

MySQLapp1

The screenshot shows a web browser window with the title "Murach's Java Servlets and..." and the URL "localhost/MySQLapp1/". The page content is titled "The SQL Gateway" and contains instructions: "Enter an SQL statement and click the Execute button." Below this, there is a text input field containing the SQL statement "select * from User" and an "Execute" button. The browser interface includes a menu bar with File, Edit, View, History, Bookmarks, Tools, and Help, and a toolbar with various icons.

The SQL Gateway

Enter an SQL statement and click the Execute button.

SQL statement:

```
select * from User
```

SQL result:

MySQLapp1

MySQLapp1 uses JSP
and JSTL

The screenshot shows a web browser window with the Oracle Java Technology Network (OTN) website loaded. The URL in the address bar is www.oracle.com/technetwork/java/index-jsp-135995.html. The page features a red header with the ORACLE logo. The main content area has a white background and includes a sidebar on the left with links for Java SE, Java EE, Java ME, Java DB, Web Tier, Java Card, Java TV, New to Java, Community, and Java Magazine. The main content area contains a section titled "JavaServer Pages Standard Tag Library" which describes JSTL and its alignment with the Unified Expression Language (EL). To the right of the main content are two columns of links under "Java SDKs and Tools" and "Java Resources". The "Java Resources" column includes links for Java APIs, Technical Articles, Demos and Videos, Forums, Java Magazine, Java.net, and Developer Training.

File Edit View History Bookmarks Tools Help

JavaServer Pages Standard 1 X +

www.oracle.com/technetwork/java/index-jsp-135995.html 150% Search

Sign In/Register Help Country Communities I am a... I want to... Search OTN

Products Solutions Downloads Store Support Training Partners About

Oracle Technology Network > Java

Java SE

Java EE

Java ME

Java SE Support

Java SE Advanced & Suite

Java Embedded

Java DB

Web Tier

Java Card

Java TV

New to Java

Community

Java Magazine

JavaServer Pages Standard Tag Library

JavaServer Pages Standard Tag Library (JSTL) encapsulates as simple tags the core functionality common to many Web applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags.

The JSTL 1.2 Maintenance Release aligns with the Unified Expression Language (EL) that is being delivered as part of the JavaServer Pages (JSP) 2.1 specification. Thanks to the Unified EL, JSTL tags, such as the JSTL iteration tags, can now be used with JavaServer Faces components in an intuitive way.

JSTL 1.2 is part of the Java EE 5 platform.

- **JSTL Project** Go to the JSTL project for the latest API and implementation of JSTL 1.2.
- **JSTL 1.2 Maintenance Review Specification Available!** Find out how changes to the JSTL specification help support the alignment of Java-based web-tier technologies.

Learn More

Java SDKs and Tools

- Java SE
- Java EE and Glassfish
- Java ME
- Java Card
- NetBeans IDE
- Java Mission Control

Java Resources

- Java APIs
- Technical Articles
- Demos and Videos
- Forums
- Java Magazine
- Java.net
- Developer Training

MySQLapp1

The SQL Gateway

Enter an SQL statement and click the Execute button.

SQL statement:

```
select * from User
```

Execute

SQL result:

Remember Tomcat has default page names that it defines in the web.xml of the web-app

```
<welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

The code for the JSP

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Murach's Java Servlets and JSP</title>
    <link rel="stylesheet" href="styles/main.css" type="text/css"/>
</head>
<body>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:if test="${sqlStatement == null}">
    <c:set var="sqlStatement" value="select * from User" />
</c:if>

<h1>The SQL Gateway</h1>
<p>Enter an SQL statement and click the Execute button.</p>
```

The code for the JSP (continued)

```
<p><b>SQL statement:</b></p>
<form action="sqlGateway" method="post">
    <textarea name="sqlStatement" cols="60"
rows="8">${sqlStatement}</textarea>
    <input type="submit" value="Execute">
</form>

<p><b>SQL result:</b></p>
${sqlResult}

</body>
</html>
```

The SQLGatewayServlet class

```
package murach.sql;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import java.sql.*;

public class SqlGatewayServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {

        String sqlStatement = request.getParameter("sqlStatement");
        String sqlResult = "";
        try {
            // load the driver
            Class.forName("com.mysql.jdbc.Driver");

```

The SQLGatewayServlet class (continued)

```
// get a connection
    String dbURL = "jdbc:mysql://localhost:3306/murach";
    String username = "murach_user";
    String password = "sesame";
    Connection connection = DriverManager.getConnection(
        dbURL, username, password);

    // create a statement
    Statement statement = connection.createStatement();

    // parse the SQL string
    sqlStatement = sqlStatement.trim();
    if (sqlStatement.length() >= 6) {
        String sqlType = sqlStatement.substring(0, 6);

        if (sqlType.equalsIgnoreCase("select")) {
            // create the HTML for the result set
            ResultSet resultSet
                = statement.executeQuery(sqlStatement);
            sqlResult = SQLUtil.getHtmlTable(resultSet);
            resultSet.close();
        }
    }
}
```

The SQLGatewayServlet class (continued)

```
        } else {
            int i = statement.executeUpdate(sqlStatement);
            if (i == 0) { // a DDL statement
                sqlResult =
                    "<p>The statement executed successfully.</p>";
            } else { // an INSERT, UPDATE, or DELETE statement
                sqlResult =
                    "<p>The statement executed successfully.<br>" +
                    + i + " row(s) affected.</p>";
            }
        }
    }
    statement.close();
    connection.close();
} catch (ClassNotFoundException e) {
    sqlResult = "<p>Error loading the database driver: <br>" +
        + e.getMessage() + "</p>";
} catch (SQLException e) {
    sqlResult = "<p>Error executing the SQL statement: <br>" +
        + e.getMessage() + "</p>";
}
```

The SQLGatewayServlet class (continued)

```
HttpSession session = request.getSession();
session.setAttribute("sqlResult", sqlResult);
session.setAttribute("sqlStatement", sqlStatement);

String url = "/index.jsp";
getServletContext()
    .getRequestDispatcher(url)
    .forward(request, response);
}

}
```

Note

- The web.xml file for this application maps the SQLGatewayServlet class to the /sqlGateway URL.

The SQLUtil class

```
package murach.sql;

import java.sql.*;

public class SQLUtil {

    public static String getHtmlTable(ResultSet results)
        throws SQLException {

        StringBuilder htmlTable = new StringBuilder();
        ResultSetMetaData metaData = results.getMetaData();
        int columnCount = metaData.getColumnCount();

        htmlTable.append("<table>");

    }
}
```

The SQLUtil class (continued)

```
// add header row
htmlTable.append("<tr>");
for (int i = 1; i <= columnCount; i++) {
    htmlTable.append("<th>");
    htmlTable.append(metaData.getColumnName(i));
    htmlTable.append("</th>");
}
htmlTable.append("</tr>");

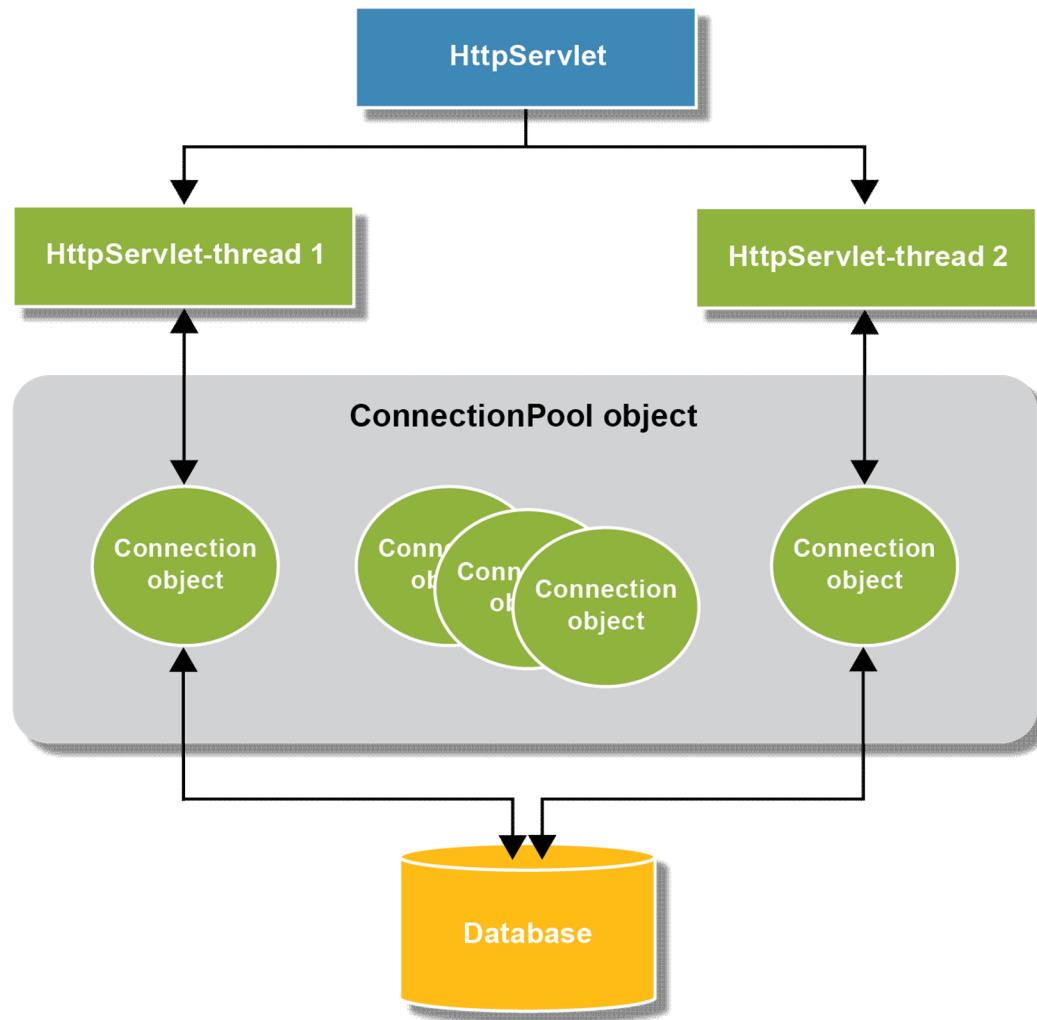
// add all other rows
while (results.next()) {
    htmlTable.append("<tr>");
    for (int i = 1; i <= columnCount; i++) {
        htmlTable.append("<td>");
        htmlTable.append(results.getString(i));
        htmlTable.append("</td>");
    }
    htmlTable.append("</tr>");
}

htmlTable.append("</table>");
return htmlTable.toString();
}
```

The SQLUtil class (continued)

- The getHtmlTable method in this class accepts a ResultSet object and returns a String object that contains the HTML code for the result set so it can be displayed by a browser.
- The getMetaData method of a ResultSet object returns a ResultSetMetaData object.
- The getColumnCount method of a ResultSetMetaData object returns the number of columns in the result set.
- The getColumnName method of a ResultSetMetaData object returns the name of a column in the result set.

How connection pooling works



How connection pooling works (continued)

- When *database connection pooling (DBCP)* is used, a limited number of connections are opened for a database and are shared by the users who connect to the database. This improves the performance of the database operations.
- When one of the threads of a servlet needs to perform a database operation, the thread gets a Connection object from the ConnectionPool object and uses that Connection object to do the operation.
- When a thread finishes using a Connection object, it returns the object to the pool.
- Before you can use the connection pool, you must make it available to your application.

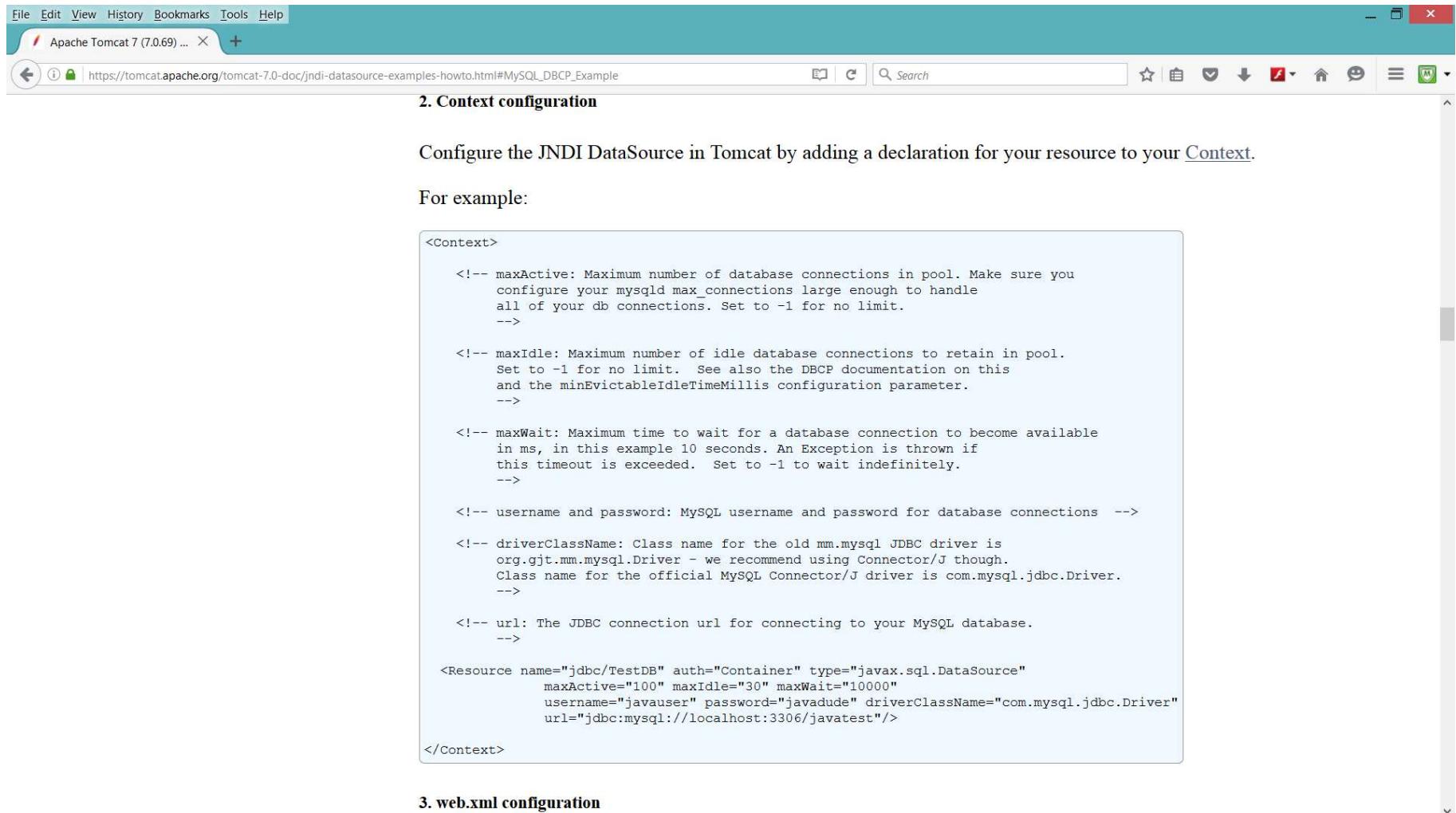
A context.xml file that configures a connection pool

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/ch12email">

    <Resource name="jdbc/murach" auth="Container"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/murach?autoReconnect=true"
        username="murach_user" password="sesame"
        maxActive="100" maxIdle="30" maxWait="10000"
        logAbandoned="true" removeAbandoned="true"
        removeAbandonedTimeout="60" type="javax.sql.DataSource" />

</Context>
```

A context.xml file that configures a connection pool



The screenshot shows a web browser window with the following details:

- Title Bar:** Apache Tomcat 7 (7.0.69) ...
- Address Bar:** https://tomcat.apache.org/tomcat-7.0-doc/jndi-datasource-examples-howto.html#MySQL_DBCP_Example
- Content Area:**
 - Section 2. Context configuration:** Configure the JNDI DataSource in Tomcat by adding a declaration for your resource to your [Context](#).
 - Text:** For example:
 - Code Example:**

```
<Context>
    <!-- maxActive: Maximum number of database connections in pool. Make sure you
        configure your mysqld max_connections large enough to handle
        all of your db connections. Set to -1 for no limit.
    -->

    <!-- maxIdle: Maximum number of idle database connections to retain in pool.
        Set to -1 for no limit. See also the DBCP documentation on this
        and the minEvictableIdleTimeMillis configuration parameter.
    -->

    <!-- maxWait: Maximum time to wait for a database connection to become available
        in ms, in this example 10 seconds. An Exception is thrown if
        this timeout is exceeded. Set to -1 to wait indefinitely.
    -->

    <!-- username and password: MySQL username and password for database connections -->

    <!-- driverClassName: Class name for the old mm.mysql JDBC driver is
        org.gjt.mm.mysql.Driver - we recommend using Connector/J though.
        Class name for the official MySQL Connector/J driver is com.mysql.jdbc.Driver.
    -->

    <!-- url: The JDBC connection url for connecting to your MySQL database.
    -->

    <Resource name="jdbc/TestDB" auth="Container" type="javax.sql.DataSource"
        maxActive="100" maxIdle="30" maxWait="10000"
        username="javauuser" password="javadude" driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/javatest"/>

</Context>
```
 - Section 3. web.xml configuration:**

A context.xml file that configures a connection pool

File Edit View History Bookmarks Tools Help

Application Developer's Guide +

https://tomcat.apache.org/tomcat-7.0-doc/appdev/deployment.html

120% Search

Web Application Deployment Descriptor

As mentioned above, the `/WEB-INF/web.xml` file contains the Web Application Deployment Descriptor for your application. As the filename extension implies, this file is an XML document and defines everything about your application that a server needs to know (except the context path, which is defined by the URL when the application is deployed).

The complete syntax and semantics for the deployment descriptor are defined in the Servlet Specification, version 2.3. Over time, it is expected that the specification will be updated to reflect changes in the deployment descriptor for you. In the meantime, you can refer to the JavaServer Faces Specification for more information. This file includes comments that describe the purpose of each included element.

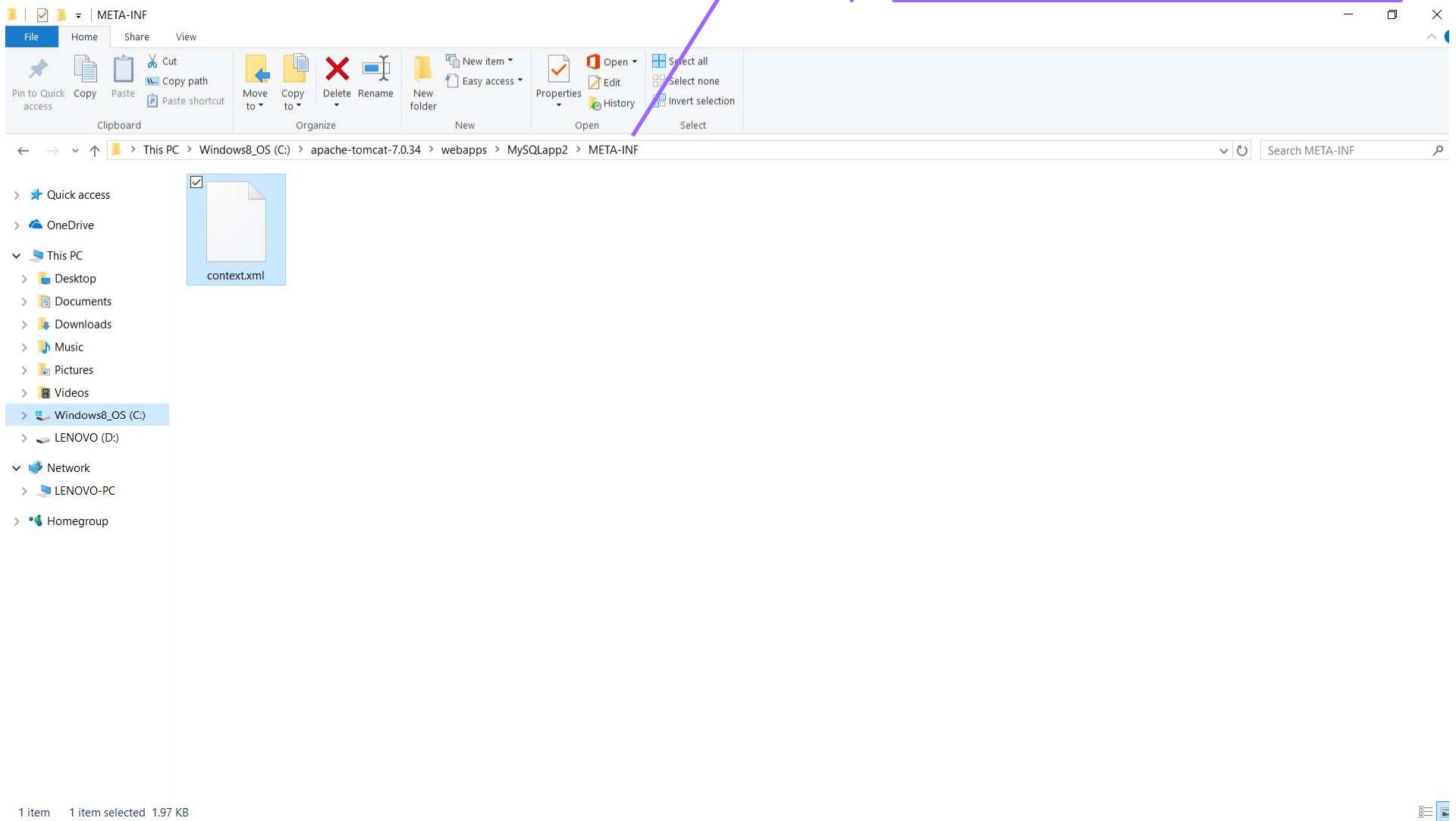
NOTE - The Servlet Specification includes a Document Type Descriptor (DTD) for the web application deployment descriptor, and Tomcat enforces the rules defined here when processing your application's `/WEB-INF/web.xml` file. In particular, you **must** enter your descriptor elements (such as `<filter>`, `<servlet>`, and `<servlet-mapping>`) in the order defined by the DTD (see Section 13.3).

Tomcat Context Descriptor

A `/META-INF/context.xml` file can be used to define Tomcat specific configuration options, such as an access log, data sources, session manager configuration and more. This XML file must contain one Context element, which will be considered as if it was the child of the Host element corresponding to the Host to which the web application is being deployed. The [Tomcat configuration documentation](#) contains information on the Context element.

Deployment With Tomcat

Make a note where context.xml gets stored



A class that defines a connection pool

```
package murach.data;

import java.sql.*;
import javax.sql.DataSource;
import javax.naming.InitialContext;
import javax.naming.NamingException;

public class ConnectionPool {

    private static ConnectionPool pool = null;
    private static DataSource dataSource = null;

    private ConnectionPool() {
        try {
            InitialContext ic = new InitialContext();
            dataSource = (DataSource)
                ic.lookup("java:/comp/env/jdbc/murach");
        } catch (NamingException e) {
            System.out.println(e);
        }
    }
}
```

A class that defines a connection pool (continued)

```
public static synchronized ConnectionPool getInstance() {  
    if (pool == null) {  
        pool = new ConnectionPool();  
    }  
    return pool;  
}  
  
public Connection getConnection() {  
    try {  
        return dataSource.getConnection();  
    } catch (SQLException e) {  
        System.out.println(e);  
        return null;  
    }  
}  
  
public void freeConnection(Connection c) {  
    try {  
        c.close();  
    } catch (SQLException e) {  
        System.out.println(e);  
    }  
}
```

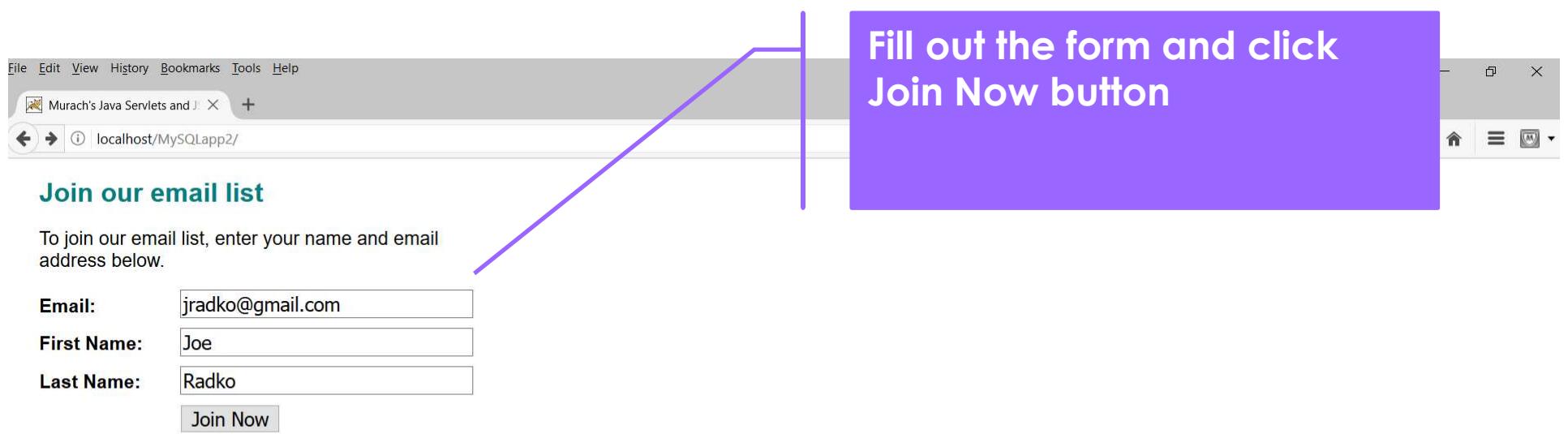
Code that uses the connection pool

```
ConnectionPool pool = ConnectionPool.getInstance();  
Connection connection = pool.getConnection();  
  
// code that uses the connection to work with the database  
  
pool.freeConnection(connection);
```

How to implement and use a connection pool

- With Tomcat 6 and later, you can use the context.xml file to configure connection pooling for an application.
- The ConnectionPool class provides the getConnection and freeConnection methods that make it easy for programmers to get connections and to return connections to the connection pool.

An Email List application that displays an error message if the email already inserted into DB



Fill out the form and click
Join Now button

Join our email list

To join our email list, enter your name and email address below.

Email:

First Name:

Last Name:

Confirmation Email Entered



Thanks for joining our email list

Here is the information that you entered:

Email: jradko@gmail.com

First Name: Joe

Last Name: Radko

To enter another email address, click on the Back button in your browser or the Return button shown below.

[Return](#)

Enter the same Record Again you get the Error message email address already exists

The screenshot shows a web browser window with the title "Murach's Java Servlets and JSP". The URL in the address bar is "localhost/MySQLapp2/emailList". The page content is as follows:

Join our email list

To join our email list, enter your name and email address below.

*This email address already exists.
Please enter another email address.*

Email:

First Name:

Last Name:

A purple line connects the error message text to the "Email" input field. A purple box on the right contains the text: "Try To enter Same email again you get an error message email already exist".

The code for the JSP

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Murach's Java Servlets and JSP</title>
    <link rel="stylesheet" href="styles/main.css" type="text/css"/>
</head>
<body>
    <h1>Join our email list</h1>
    <p>To join our email list, enter your name and
        email address below.</p>
    <p><i>${message}</i></p>
    <form action="emailList" method="post">
        <input type="hidden" name="action" value="add">

        <label class="pad_top">Email:</label>
        <input type="email" name="email" value="${user.email}"
            required><br>
```

The code for the JSP (continued)

```
<label class="pad_top">First Name:</label>
<input type="text" name="firstName" value="${user.firstName}"
       required><br>

<label class="pad_top">Last Name:</label>
<input type="text" name="lastName" value="${user.lastName}"
       required><br>

<label>&nbsp;</label>
<input type="submit" value="Join Now" class="margin_left">
</form>
</body>
</html>
```

The code for the servlet

```
package murach.email;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

import murach.business.User;
import murach.data.UserDB;

public class EmailListServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request,
                          HttpServletResponse response)
        throws ServletException, IOException {
        String url = "/index.html";

        // get current action
        String action = request.getParameter("action");
        if (action == null) {
            action = "join"; // default action
        }
    }
}
```

The code for the servlet (continued)

```
// perform action and set URL to appropriate page
if (action.equals("join")) {
    url = "/index.jsp";      // the "join" page
}
else if (action.equals("add")) {
    // get parameters from the request
    String firstName = request.getParameter("firstName");
    String lastName = request.getParameter("lastName");
    String email = request.getParameter("email");

    // store data in User object
    User user = new User(firstName, lastName, email);
```

The code for the servlet (continued)

```
// validate the parameters
if (UserDB.emailExists(user.getEmail())) {
    message = "This email address already exists.<br>" +
               "Please enter another email address.";
    url = "/index.jsp";
}
else {
    message = "";
    url = "/thanks.jsp";
    UserDB.insert(user);
}
request.setAttribute("user", user);
request.setAttribute("message", message);
}
getServletContext()
    .getRequestDispatcher(url)
    .forward(request, response);
}
```

The UserDB class

```
package murach.data;

import java.sql.*;

import murach.business.User;

public class UserDB {

    public static int insert(User user) {
        ConnectionPool pool = ConnectionPool.getInstance();
        Connection connection = pool.getConnection();
        PreparedStatement ps = null;

        String query
            = "INSERT INTO User (Email, FirstName, LastName) "
            + "VALUES (?, ?, ?)";

        ...
```

The UserDB class (continued)

```
try {
    ps = connection.prepareStatement(query);
    ps.setString(1, user.getEmail());
    ps.setString(2, user.getFirstName());
    ps.setString(3, user.getLastName());
    return ps.executeUpdate();
} catch (SQLException e) {
    System.out.println(e);
    return 0;
} finally {
    DBUtil.closePreparedStatement(ps);
    pool.freeConnection(connection);
}
}
```

The UserDB class (continued)

```
public static int update(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;

    String query = "UPDATE User SET "
                  + "FirstName = ?, "
                  + "LastName = ? "
                  + "WHERE Email = ?";

    try {
        ps = connection.prepareStatement(query);
        ps.setString(1, user.getFirstName());
        ps.setString(2, user.getLastName());
        ps.setString(3, user.getEmail());

        return ps.executeUpdate();
    } catch (SQLException e) {
        System.out.println(e);
        return 0;
    } finally {
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
}
```

The UserDB class (continued)

```
public static int delete(User user) {
    ConnectionPool pool = ConnectionPool.getInstance();
    Connection connection = pool.getConnection();
    PreparedStatement ps = null;

    String query = "DELETE FROM User "
                  + "WHERE Email = ?";
    try {
        ps = connection.prepareStatement(query);
        ps.setString(1, user.getEmail());

        return ps.executeUpdate();
    } catch (SQLException e) {
        System.out.println(e);
        return 0;
    } finally {
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
}
```

The UserDB class (continued)

```
public static boolean emailExists(String email) {  
    ConnectionPool pool = ConnectionPool.getInstance();  
    Connection connection = pool.getConnection();  
    PreparedStatement ps = null;  
    ResultSet rs = null;  
  
    String query = "SELECT Email FROM User "  
        + "WHERE Email = ?";  
    try {  
        ps = connection.prepareStatement(query);  
        ps.setString(1, email);  
        rs = ps.executeQuery();  
        return rs.next();  
    } catch (SQLException e) {  
        System.out.println(e);  
        return false;  
    } finally {  
        DBUtil.closeResultSet(rs);  
        DBUtil.closePreparedStatement(ps);  
        pool.freeConnection(connection);  
    }  
}
```

The UserDB class (continued)

```
public static User selectUser(String email) {  
    ConnectionPool pool = ConnectionPool.getInstance();  
    Connection connection = pool.getConnection();  
    PreparedStatement ps = null;  
    ResultSet rs = null;  
  
    String query = "SELECT * FROM User "  
        + "WHERE Email = ?";  
    try {  
        ps = connection.prepareStatement(query);  
        ps.setString(1, email);  
        rs = ps.executeQuery();  
        User user = null;  
        if (rs.next()) {  
            user = new User();  
            user.setFirstName(rs.getString("FirstName"));  
            user.setLastName(rs.getString("LastName"));  
            user.setEmail(rs.getString("Email"));  
        }  
        return user;  
    }
```

The UserDB class (continued)

```
    } catch (SQLException e) {
        System.out.println(e);
        return null;
    } finally {
        DBUtil.closeResultSet(rs);
        DBUtil.closePreparedStatement(ps);
        pool.freeConnection(connection);
    }
}
```

The DBUtil class

```
package murach.data;

import java.sql.*;

public class DBUtil {

    public static void closeStatement(Statement s) {
        try {
            if (s != null) {
                s.close();
            }
        } catch (SQLException e) {
            System.out.println(e);
        }
    }
}
```

The DBUtil class (continued)

```
public static void closePreparedStatement(Statement ps) {
    try {
        if (ps != null) {
            ps.close();
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
}

public static void closeResultSet(ResultSet rs) {
    try {
        if (rs != null) {
            rs.close();
        }
    } catch (SQLException e) {
        System.out.println(e);
    }
}
```