Java Database Connectivity (JDBC)

In this lecture we will learn about Java Database Connectivity (JDBC). JDBC is an application programming interface (API) of Java programming language, which defines how a client may access a database. It is a Java-based data access technology used for Java database connectivity. We will cover the following:

1. How to work with MySQL:
2. Accessing MySQL
3. Creating Tables
4. Inserting Data
5. Querying Data
6. How to work with MySQL with Java.

# Introduction

Using cookies to store information is good for a small system. If you are dealing with a lot of customers you need to create a database for that. Moreover the cookies are saved in the client computer. If the client deletes the cookies then your system has no way to find out whether a client is a new or old customer.

To have a better solution we include a database manager named MySQL.

For this lecture you need to install the database manager MySQL and a compressed Java file (has extension jar). I wish I could tell you to use the school server right away to create a database and some tables for the database that we could use in our examples. We could also run our source program by the server. But the problem is:

1. You have to use java compiler (javac.exe) with extra compiler directives that goes with the instruction (javac.exe) to create a \*.class file. This means we would be back to the old ways of running a java program: the time when we did not have an IDE (such as eclipse).

There is a bigger problem with this approach and it is the time that is requires to trouble shoot our servlet program bugs (not syntax errors that are easy to debug). We need to insert System.out.println(…) in different places of the program to find out where the bug is. We also have to run the program so many times to come up with a logically correct server program.

Therefore it is better to have all the tools in our computer (if you still wish to use school server, you can transfer out \*.class file to the school server and continue).

## JDBC

JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database. JDBC works with Java on a variety of platforms, such as Windows, Mac OS, and the various versions of UNIX.

I In general, to process any SQL statement with JDBC, we follow these steps:

1. Establishing a connection n
2. Create a statement
3. Execute the query
4. Process the ResultSet object
5. Close the connection

First, establish a connection with the data source you want to use. A data source can be a Relational/Database Management System (R/DBMS), a legacy file system, or some other source of data with a corresponding JDBC driver. This connection is represented by a Connection object.

A Statement is an interface that represents a SQL statement. You execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set. You need a Connection object to create a Statement object.   
  
There are three different kinds of statements:

* Statement: Used to implement simple SQL statements with no parameters.
* PreparedStatement: (Extends Statement.) Used for precompiling SQL statements that might contain input parameters.
* CallableStatement: (Extends PreparedStatement.) Used to execute stored procedures that may contain both input and output parameters.

To execute a query, call an execute method from Statement such as the following:

* execute: Returns true if the first object that the query returns is a ResultSet object. Use this method if the query could return one or more ResultSet objects. Retrieve the ResultSet objects returned from the query by repeatedly calling Statement.getResultSet.
* executeQuery: Returns one ResultSet object.
* executeUpdate: Returns an integer representing the number of rows affected by the SQL statement. Use this method if you are using INSERT, DELETE, or UPDATE SQL statements.

You access the data in a ResultSet object through a cursor. Note that this cursor is not a database cursor. This cursor is a pointer that points to one row of data in the ResultSet object. Initially, the cursor is positioned before the first row. You call various methods defined in the ResultSetobject to move the cursor.

# 1. Working with MySQL

Before anything we need to install mysql.

## 1.1 Installing MySQL

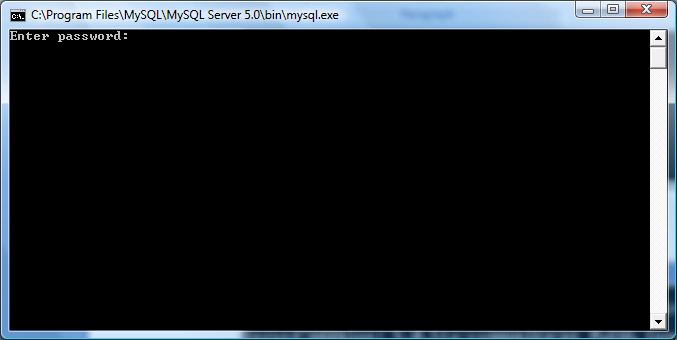
If you have MySQL installed on your PC, ignore this section.

Download mysql installer from the [MySQL Community Server](https://dev.mysql.com/downloads/mysql/) and install latest version of MySQL.

## 1.2 Testing MySQL

The installation wizard makes a folder on your Start 🡪 All Programs named: MySQL.

* 1. Click on *Start 🡪 All programs* and point to *MySQL* to see *MySQL Server 5.7*. Choose *MySQL Command Line Client* to see:



* 1. Enter your password to see (both username and password – root) the following:

Command prompt after log to MySQL.

Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g. 
Your MySQL connection id is 3
Server version: 5.5.11 MySQL Community Server <GPL>
Copyright <c> 2000, 2010, 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>

Note: While installing you may not be prompted for username and password. Both will be set to root. If you are required to set those, please set ‘root’ for both.

* 1. On the dos prompt enter:

SELECT VERSION(), CURRENT\_DATE();

You should see:

Output of select statment on Command prompt.

mysql> SELECT VERSION<>, CURRENT_DATE<>;

Table as outlined:

1. VERSION<> 
a. 5.5.11

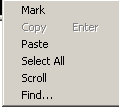
2. CURRENT_DATE<>
a. 2011-04-17

1 row in set <0.00 sec>

Note: You can copy/paste this and the following instructions rather than typing by:

a) Copy the instruction.

b) In the DOS window right-click to see:



c) Click on: Paste.

* 1. Enter *exit* or *quit* to terminate the session.

## 1.3 Creating a database

To learn more about MySQL look at the [MySQL Tutorial](https://www.mysqltutorial.org/) website. We would like to create a table that will hold user name and the password for a number of users. Therefore we need a table with two columns. Do not forget to add a semicolon (;) at the end of each command.

* 1. Bring MySQL as described in section 1.2 and enter:

create database myFirstDataBase;

You should see

Output of create database statment on Command prompt.

mysql> crate database myFirstDataBase; Query OK, 1 row affected <0.00 sec>

* 1. Enter:

show databases;

Output of show database statment on Command prompt.

mysql> show databases; 

Table as outlined: 

1. Database
a. information_schema
b. myfirstdatabase
c. mysql
d. performance_schema
e. test
5 rows in set <0.00 sec>

* 1. To use the new database enter:

use myFirstDataBase;

Output of SQL statment on Command prompt.

mysql> use myFirstDataBase;
Database changed

* 1. We now create a table with two columns for the user name and password. Enter:

create table myFirstTable (name char(20), password char(20));

To see:

Output of create table statment on Command prompt.

mysql> create table myFirstTable <name char<20>, password char<20>>; Query OK, 0 rows affected <0.17 sec>

* 1. To see the table enter:

show tables;

To see:

Output of show table statment on Command prompt.

mysql> show tables;

1. Tables_in_myfirstdatabase
a. myfirsttable
1 row in set <0.00 sec>

The second row shows the name of the table.

* 1. Let us look at the structure of the table:

describe myFirstTable;

Output of describe statment on Command prompt.

mysql> describe myFirstTable;

A table with six columns as outlined:

1. Field
a. name 
b. password
2. Type
a. char<20>
b. char <20>
3. Null
a. YES
b. YES
4. Key
5. Default. 
a. NULL
b. NULL
6. Extra

* 1. Let us enter names and passwords for two customers:

insert into myFirstTable values ('John Doe', 'John');

To see:

Output of insert statment on Command prompt.

mysql> insert into myFirstTable values <'John Doe', 'John'>; Query OK, 1 row affected <0.05 sec>

Repeat for:

insert into myFirstTable values ('Jane Doe', 'Jane');

* 1. Let us look at the content of table:

select \* from myFirstTable;

Output of select statment on Command prompt.

mysql> select * from myFirstTable;

A table with two columns as outlined: 

1. name
a. John Doe
b. Jane Doe
2. password
a. John
b. Jane

2 rows in set (0.00 sec)

* 1. May be we like to see one row:

select \* from myFirstTable where name = 'John Doe';

Output of select statement on Command prompt.

mysql> select * from myFirstTable where name = 'John Doe';

A table with two columns as outlined: 

1. name
a. John Doe
2. password
a. John

1 row in set (0.00 sec)

* 1. Finally we like to delete one row of the lable:

delete from myFirstTable where password = 'Jane';

Output of delete statment on Command prompt.

mysql> delete from myFirstTable where password = 'Jane'; Query OK, 1 row affected (0.01 sec)

* 1. Check to make sure this row is deleted:

select \* from myFirstTable;

Output of select statment on Command prompt.

mysql> select * from myFirstTable;

1. name
a. John Doe
2. password
a. John
1 row in set (0.00 sec)

* 1. Finally we delete our database:

drop database myfirstdatabase;

Output of drop statment on Command prompt.

mysql> drop database myfirstdatabase; Query OK, 1 row affected (0.06 sec)

Note: We also can delete one table from the database:

drop table myfirsttable;

* 1. Let us end this session and move to Java by entering:

exit

# 2. Working with database from a Java program:

To work with MySQL within java we need to connect Java to MySQL via eclipse

## 2.1 Downloading the MySQL JDBC Driver

To download the JDBC Driver or Connector, go to the [MySQL Community Downloads](https://dev.mysql.com/downloads/connector/j/5.1.html) page. The platform independent zip file will work.

1. Download the file: **mysql-connector-java-5.1.15-bin.jar** (or a higher version)
2. Copy this file into two folders:

**C:\Program Files\Java\jre7\lib\ext**

And:

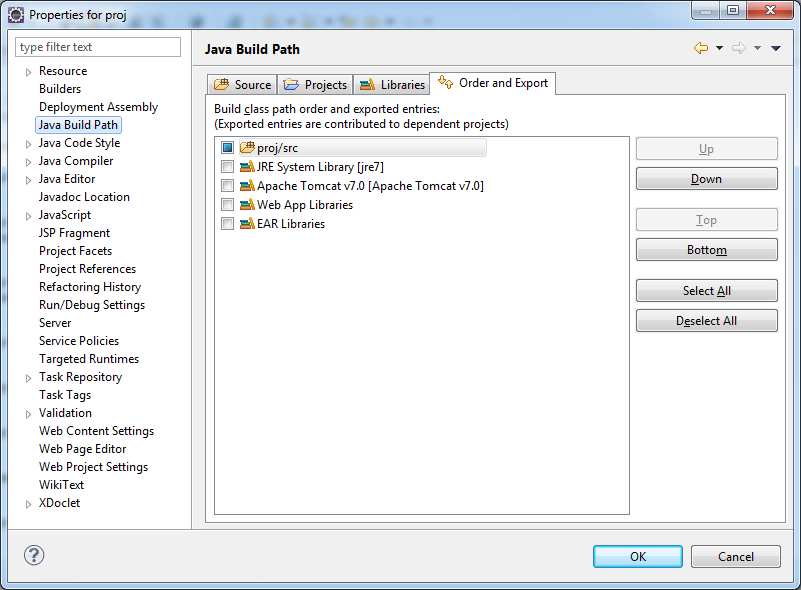
**C:\Program Files\Apache Software Foundation\Tomcat 7.0\lib**

## 2.2 Writing a java program in eclipse

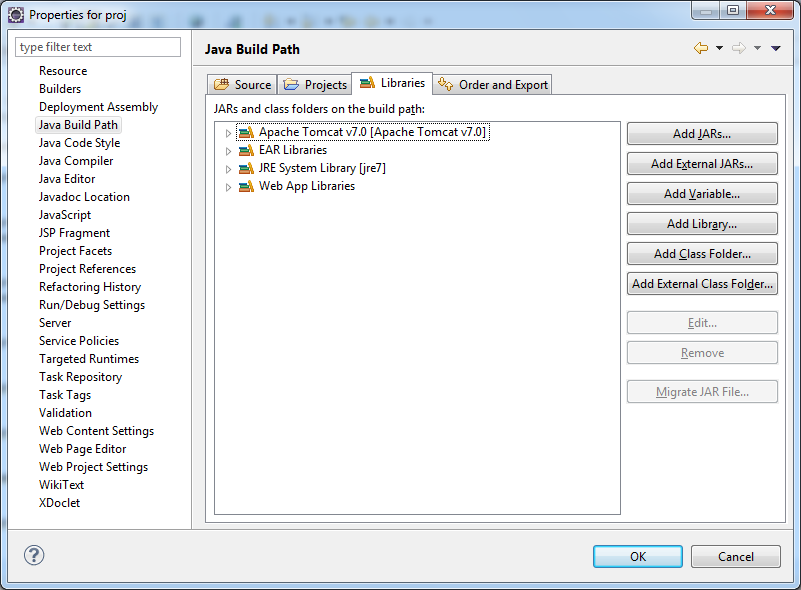
In this section we look at four examples. The first two are just Java program and the last two are Java servlets.

### Example 1

1. Make an empty folder and a java project named *Project* for eclipse. Make a class called: Main. On the menu bar click on: **Project 🡪 Properties** and click on Java Build Path:



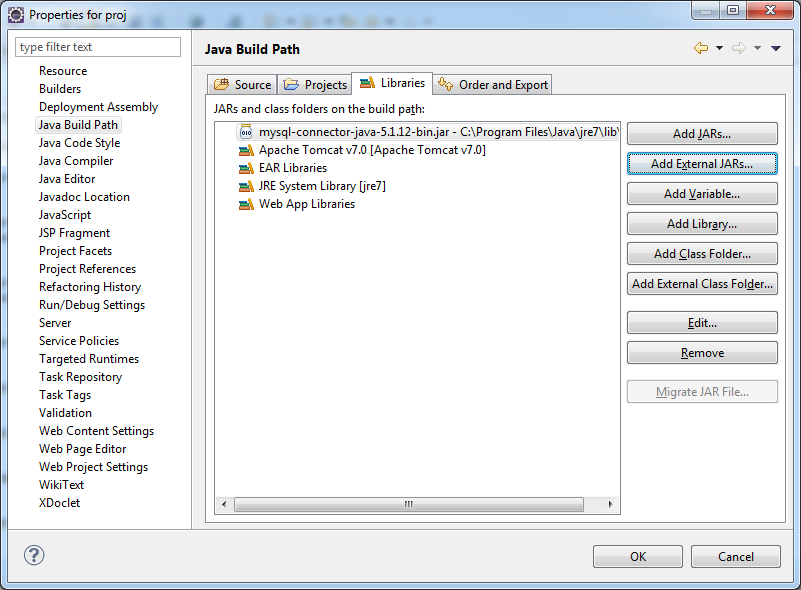
1. Select the tab: Libraries:



1. Click on Add External JARs… and find the file **mysql-connector-java-5.1.15-bin.jar** which is in the folder:

**C:\Program Files\Java\jre6\lib\ext**

Note: I assume you have already copied (moved) this file to the above folder.



1. Click OK.
2. Copy the following class in place of your Main class:

Note: I assume your password is: **root**.

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**try** {

Class.*forName*("com.mysql.jdbc.Driver").newInstance();

Connection conn = DriverManager.*getConnection*(

"jdbc:mysql://localhost/test?user=root&password=root");

} **catch** (Exception ex) {

System.*out*.println("LocalizedMessage: " +

ex.getLocalizedMessage());

System.*out*.println("Message: " + ex.getMessage());

System.*out*.println("Cause: " + ex.getCause());

System.*out*.println("Class: " + ex.getClass());

System.*out*.println("StackTrace: " + ex.getStackTrace());

System.*exit*(0);

}

System.*out*.println("Program terminated with no error.");

}

}

**The output should be**:

Program terminated with no error.

**Note:** If you give the wrong:

User name:

**user=rootA**

Or: Password:

**password=aaaaaaaaaaA**

You get:

LocalizedMessage: Access denied for user 'rootA'@'localhost' (using password: YES)

Message: Access denied for user 'rootA'@'localhost' (using password: YES)

Cause: null

Class: class java.sql.SQLException

StackTrace: [Ljava.lang.StackTraceElement;@19c26f5

If you give the wrong database:

**testA**

You get:

LocalizedMessage: Unknown database 'testa'

Message: Unknown database 'testa'

Cause: null

Class: class com.mysql.jdbc.exceptions.MySQLSyntaxErrorException

StackTrace: [Ljava.lang.StackTraceElement;@19c26f5

Note: The data base **test** is already in MySql when you install.

#### Description

There are two statements new to you:

Class.*forName*("com.mysql.jdbc.Driver").newInstance();

Connection conn = DriverManager.*getConnection*(

"jdbc:mysql://localhost/test?user=root&password=root");

The first statement loads the JDBC driver. The second instruction establishes a connection to the database named *test*. Note that the database *test* is already in MySQL and we do not have to make it.

The URL is: jdbc:mysql://localhost/

The name of the database is: test

The user is: root

User’s password is: root

### Example 2

Let us make a table with one row:

**import** java.sql.\*;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

**try** {

Class.*forName*("com.mysql.jdbc.Driver").newInstance();

Connection con = DriverManager.*getConnection*(

"jdbc:mysql://localhost/test?user=root&password=root");

Statement stmt = con.createStatement();

**try**{

String SQL = "DROP TABLE simple";

stmt.execute(SQL);

}**catch**(Exception e){

}

stmt.execute("CREATE TABLE IF NOT EXISTS simple( name char(30))");

stmt.execute("Insert into simple values('John Doe')");

stmt.execute("Insert into simple values('Jane Doe')");

ResultSet rs = stmt.executeQuery("Select \* from simple");

**while** ( rs.next())

System.*out*.println(rs.getString(1));

} **catch** (Exception ex) {

System.*out*.println("Message: " + ex.getMessage());

System.*exit*(0);

}

System.*out*.println("Program terminated with no error.");

}

}

The output of this program is:

John Doe

Jane Doe

Program terminated with no error.

#### Description

The instruction:

Statement stmt = con.createStatement();

The variable stmt refers to an object of type of the class Statement. We send a sql statement via this object to mysql.

The following try-catch block sends the instruction DROP TABLE simple to mysql to delete a table names *simple* by executing the instruction: stmt.execute(SQL). We include instruction this instruction in a try-catch block:

**try**{

String SQL = "DROP TABLE simple";

stmt.execute(SQL);

}**catch**(Exception e){

}

If the table named simple exist, it will be deleted. But when the table does not exist the sql instruction DROP TABLE simple caused the system to throw an exception. If we do not include the statement stmt.execute(SQL) in a try-catch block this exception is caught by JVM and the program terminates. With this trick we prevent this from happening.

The instruction:

stmt.execute("CREATE TABLE IF NOT EXISTS simple( name char(30))");

If the table does not exist creates a table named *simple* with one column named *name*. In each row of this column we are allowed to write a string of at most 30 characters.

The instructions:

stmt.execute("Insert into simple values('John Doe')");

stmt.execute("Insert into simple values('Jane Doe')");

Insert the names *John Doe* and *Jane Doe* in the table. Therefore the table has two rows.

To retrieve the table we need an object of the type of class ResultSet. It is this object that holds the content of the table. This is the job of the following instruction:

ResultSet rs = stmt.executeQuery("Select \* from simple");

The while-loop:

**while** ( rs.next())

System.*out*.println(rs.getString(1));

Iterates twice. On the first iteration it prints *John Doe* and on the second iteration it prints *Jane Doe*.

The instruction:

rs.getString(1)

Retrieves the first column (we only have one column). On the first iteration it refers to the first row and on the second iteration it refers to the second row.

### Example 3

Let us make an html and a servlet to do the job of the previous example. For the sake of simplicity we enter names from the keyboard.

Follow steps 1-6 of example 1 to add the JDBC driver.

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** java.sql.\*;

**import** java.util.Scanner;

@WebServlet("/Main")

**public** **class** Main **extends** javax.servlet.http.HttpServlet **implements**

javax.servlet.Servlet {

**static** **final** **long** *serialVersionUID* = 1L;

**public** Main() {

**super**();

}

**protected** **void** doGet(HttpServletRequest request,

HttpServletResponse response)**throws** ServletException, IOException {

Scanner key = **new** Scanner(System.*in*);

System.*out*.print("Enter a name: ");

String name = key.nextLine();

**try** {

Class.*forName*("com.mysql.jdbc.Driver").newInstance();

Connection con = DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/test?user=root&password=root");

Statement stmt = con.createStatement();

**try**{

//Sometimes you may decide to remove the table.

//In that case uncomment the following line.

//stmt.execute("DROP TABLE simple");

stmt.execute("CREATE TABLE IF NOT EXISTS simple( name char(30))");

}**catch**(Exception e){

}

//Note that the variable is not part of the string.

String s = "Insert into simple values(\'" + name + "\')";

**if**(name != **null**)

stmt.execute(s);

ResultSet rs = stmt.executeQuery("Select \* from simple");

**while** ( rs.next())

System.*out*.println(rs.getString(1));

} **catch** (Exception ex) {

System.*out*.println(ex);

System.*exit*(0);

}

System.*out*.println("Program terminated with no error.");

}

**protected** **void** doPost(HttpServletRequest request,

HttpServletResponse response) **throws** ServletException, IOException{

// **TODO** Auto-generated method stub

}

}

Run the program few time. I run it as follows:

1. Uncomment //stmt.execute("DROP TABLE simple"); and run it to make sure the table is removed. I entered John Doe and the output was:

Enter a name: John Doe

John Doe

Program terminated with no error.

1. Comment the above instruction and run it. I entered Jane Doe. The output was:

Enter a name: Jane Doe

John Doe

Jane Doe

Program terminated with no error.

Note: If eclipse does not prompt for “Enter a name:” just close eclipse and re-load it.

1. I run the program again and entered my name Mike Smith. The output was:

Enter a name: Mike Smith

John Doe

Jane Doe

Mike Smith

Program terminated with no error.

**Description**: This is similar to the previous example. I replaced the nested try-catch block by the following:

**try**{

//Sometimes you may decide to remove the table.

//In this case uncomment the following line.

//stmt.execute("DROP TABLE simple");

stmt.execute("CREATE TABLE IF NOT EXISTS simple( name char(30))");

}**catch**(Exception e){

}

With this change if the table is already created the flow of the execution jumps in to the catch block and it executes the rest of the statement. On the other hand if the table does not exist the try-catch block creates the table.

Note: The most messages I get from a student asking why my project is not working is about a variable (like: name below). A variable in this kind of string must not be in single or double quotations.

//Note that the variable is not part of the string.

String s = "Insert into simple values(\'" + **name** + "\')";

Suppose the value of the variable: name is: John Doe. The value of s for the above statement is:

"Insert into simple values('John Doe')"

If you make mistake and write something like:

String s = "Insert into simple values(\'" +**"name"**+ "\')";

The value of s is:

"Insert into simple values('name')"

Therefore keep a close eye while writing your program.

As a final example we write a complete client-server program.

### Example 4

Similar to the program in example 3 but this time a client sends a name.

A. We make the following html document for the client side:

<!-- Main.html -->

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Storing the name of a client</title>

</head>

<body>

<form action="http://localhost:8080/Project/Main">

<center>

<br><br>

Enter Your name:

<input type="text" name="name"><br><br><br>

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;

<input type="submit">

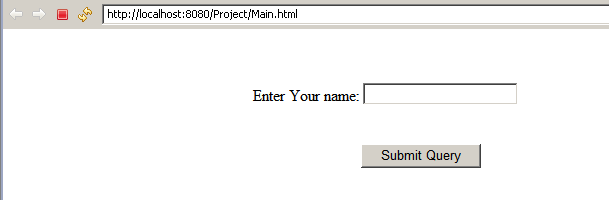
</center>

</form>

</body>

</html>

This html document displays:



B. We make changes to the program in example 3 to have:

**import** java.io.IOException;

**import** javax.servlet.ServletException;

**import** javax.servlet.annotation.WebServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** java.sql.\*;

**import** java.util.Scanner;

@WebServlet("/Main")

**public** **class** Main **extends** javax.servlet.

http.HttpServlet **implements** javax.servlet.Servlet {

**static** **final** **long** *serialVersionUID* = 1L;

**public** Main () {

**super**();

}

**protected** **void** doGet(HttpServletRequest request,

HttpServletResponse response)**throws** ServletException, IOException {

String name = request.getParameter("name");

**try** {

Class.*forName*("com.mysql.jdbc.Driver").newInstance();

Connection con = DriverManager.*getConnection*(

"jdbc:mysql://localhost:3306/test?user=root&password=root");

Statement stmt = con.createStatement();

**try**{

//Sometimes you may decide to remove the table.

//In this case uncomment the following line.

//stmt.execute("DROP TABLE simple");

stmt.execute("CREATE TABLE IF NOT EXISTS simple( name char(30))");

}**catch**(Exception e){

}

//Note that the variable is not part of the string.

String s = "Insert into simple values(\'" + name + "\')";

**if**(name != **null**)

stmt.execute(s);

ResultSet rs = stmt.executeQuery("Select \* from simple");

**while** ( rs.next())

System.*out*.println(rs.getString(1));

} **catch** (Exception ex) {

System.*out*.println(ex);

System.*exit*(0);

}

System.*out*.println("Program terminated with no error.");

}

**protected** **void** doPost(HttpServletRequest request,

HttpServletResponse response)**throws** ServletException, IOException {

// **TODO** Auto-generated method stub

}

}

#### Description

In this program we removed:

**import** java.util.Scanner;

Scanner key = **new** Scanner(System.*in*);

System.out.print("Enter a name: ")

String name = key.nextLine();

And added the following instruction:

String name = request.getParameter("name");

Therefore the servlet gets the name from the input box of the html document.