# JDBC

## CSCI 201
## Principles of Software Development

Jeffrey Miller, Ph.D.
*jeffrey.miller@usc.edu*

USC Viterbi
School of Engineering

# Outline

- JDBC
- Prepared Statements
- Program

# Executing SQL

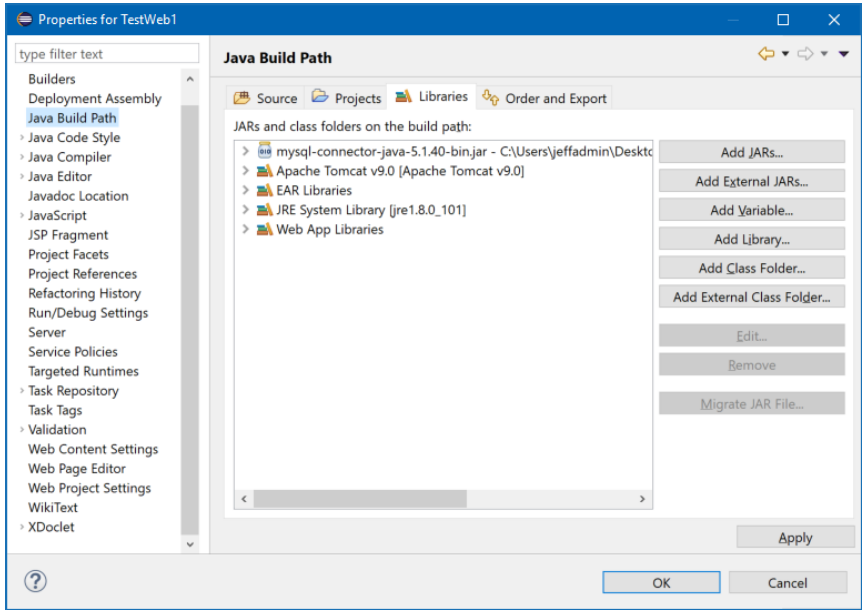
- There are a few ways to execute SQL statements, which typically depend on the DBMS
  - MySQL has different visualization tools, such as MySQL Workbench, and command line options
- We often want to access databases from our code
  - This can be to insert, update, delete, or query the data
- Java Database Connectivity (JDBC) drivers allow us to embed SQL in our Java code and execute those statements on a database programmatically

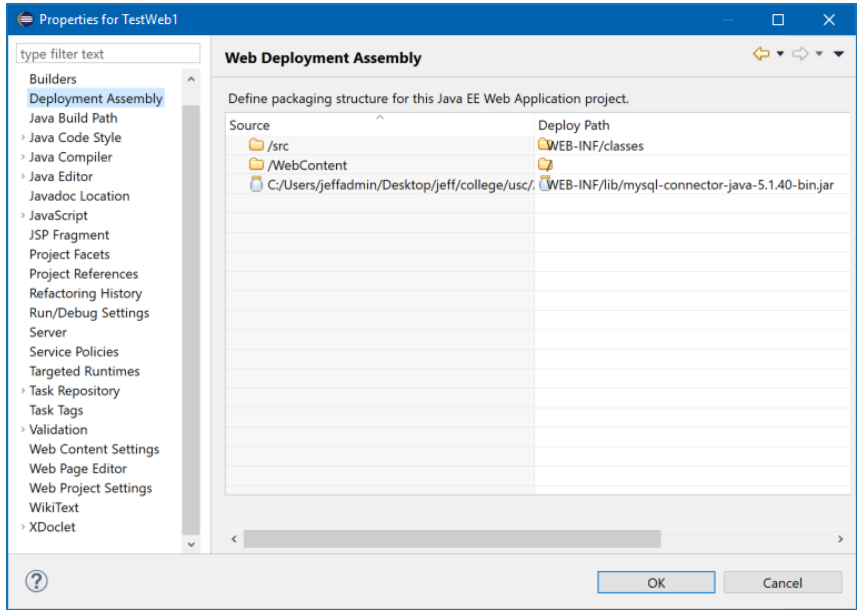
# JDBC Drivers

- JDBC Drivers are typically provided by the DBMS organization that allows you to call into the API and execute embedded SQL
- Java has an API that consists of interfaces and abstract classes in the JDK
- The JDBC Driver we use for a specific DBMS just needs to provide implementations for those interfaces and classes
  - This allows us to change drivers without having to change our program
  - Note that this probably does not allow us to change DBMS providers because the SQL would be different (though the specific Java code calling into the API would be the same)

# JDBC Driver Configuration

- The JDBC Driver will come as a JAR file, so we need to add that file into the `classpath` when running our program
  - We don't need it in the `classpath` when compiling though…why?
  - We also need to add it to the deployment if writing a web application
- In Eclipse, go to Project->Properties and add the JAR file in two locations



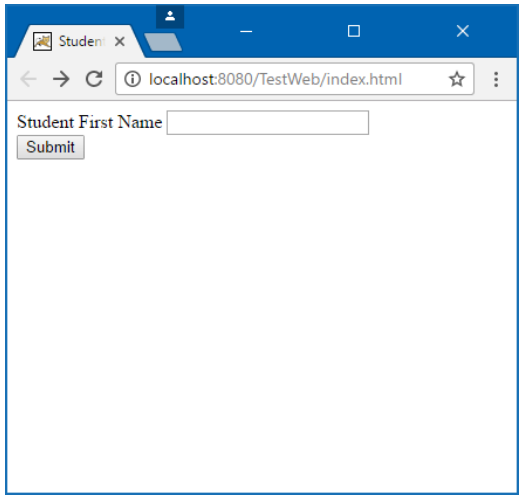
# JDBC Standalone Code

```
1  import java.sql.*; // to save space
2  public class JDBCTest {
3    public static void main (String[] args) {
4      Connection conn = null;
5      Statement st = null;
6      ResultSet rs = null;
7      try {
8        Class.forName("com.mysql.jdbc.Driver");
9        conn = DriverManager.getConnection("jdbc:mysql://localhost/StudentGrades?user=root&password=root&useSSL=false");
10       st = conn.createStatement();
11       String name = "Sheldon";
12       rs = st.executeQuery("SELECT * from Student where fname='" + name + "'");
13       while (rs.next()) {
14         String fname = rs.getString("fname");
15         String lname = rs.getString("lname");
16         int studentID = rs.getInt("studentID");
17         System.out.println ("fname = " + fname);
18         System.out.println ("lname = " + lname);
19         System.out.println ("studentID = " + studentID);
20       }
21     } catch (SQLException sqle) {
22       System.out.println (sqle.getMessage());
23     } catch (ClassNotFoundException cnfe) {
24       System.out.println (cnfe.getMessage());
25     } finally {
```

```
26       try {
27         if (rs != null) {
28           rs.close();
29         }
30         if (st != null) {
31           st.close();
32         }
36         if (conn != null) {
37           conn.close();
38         }
39       } catch (SQLException sqle) {
40         System.out.println(sqle.getMessage());
41       }
42     }
43   }
44 }
```

# JDBC Web Code – index.html

```
1  <html>
2    <head>
3      <title>Student Query Form</title>
4    </head>
5    <body>
6      <form name="studentForm" method="POST" action="queryStudent">
7        Student First Name <input name="fname" /><br />
8        <input type="submit" name="submit" value="Submit" />
9      </form>
10   </body>
11 </html>
```



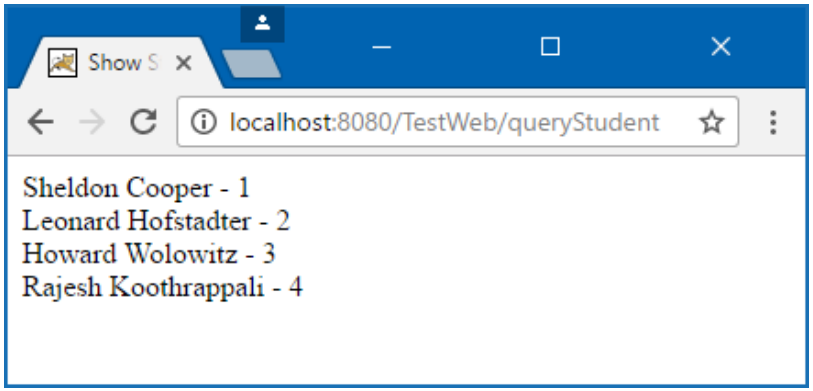
# JDBC Web Code – servlet

```
1  // NOTE: import statements omitted for space
2  @WebServlet("/queryStudent")
3  public class QueryStudent extends HttpServlet {
4    public static final long serialVersionUID = 1;
5    protected void service(HttpServletRequest request, HttpServletResponse response)
6      throws ServletException, IOException {
7      String firstname = request.getParameter("fname");
8      Connection conn = null;
9      Statement st = null;
10     ResultSet rs = null;
11     try {
12       Class.forName("com.mysql.jdbc.Driver");
13       conn = DriverManager.getConnection("jdbc:mysql://localhost/StudentGrades?user=root&password=root&useSSL=false");
14       st = conn.createStatement();
15       if (firstname != null && firstname.length() > 0) {
16         rs = st.executeQuery("SELECT * from Student where fname='" + firstname + "'");
17       }
18       else {
19         rs = st.executeQuery("SELECT * from Student");
20       }
21       ArrayList<String> fnames = new ArrayList<String>();
22       ArrayList<String> lnames = new ArrayList<String>();
23       ArrayList<Integer> studentIDs = new ArrayList<Integer>();
24       while (rs.next()) {
25         String fname = rs.getString("fname");
26         String lname = rs.getString("lname");
27         int studentID = rs.getInt("studentID");
28         fnames.add(fname);
29         lnames.add(lname);
30         studentIDs.add(studentID);
31       }
32       request.setAttribute("fnames", fnames);
33       request.setAttribute("lnames", lnames);
34       request.setAttribute("studentIDs", studentIDs);
35     } catch (SQLException sqle) {
36       System.out.println (sqle.getMessage());
37     } catch (ClassNotFoundException cnfe) {
38       System.out.println (cnfe.getMessage());
39     } finally {
40       try {
41         if (rs != null) { rs.close(); }
42         if (st != null) { st.close(); }
43         if (conn != null) { conn.close(); }
44       } catch (SQLException sqle) {
45         System.out.println(sqle.getMessage());
46       }
47     }
48     String nextJSP = "/displayStudent.jsp";
49     RequestDispatcher dispatcher =
         getServletContext().getRequestDispatcher(nextJSP);
50     dispatcher.forward(request,response);
51   }
52 }
```

USC Viterbi School of Engineering

# JDBC Web Code – displayStudent.jsp

```
1  <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2      pageEncoding="ISO-8859-1" import="java.util.ArrayList" %>
3  <html>
4    <head>
5      <title>Show Student Information</title>
6    </head>
7    <body>
8  <%
9      ArrayList<String> fnames = (ArrayList<String>)request.getAttribute("fnames");
10     ArrayList<String> lnames = (ArrayList<String>)request.getAttribute("lnames");
11     ArrayList<Integer> studentIDs = (ArrayList<Integer>)request.getAttribute("studentIDs");
12     for (int i=0; i < fnames.size(); i++) {
13 %>
14       <%= fnames.get(i) %> <%= lnames.get(i) %> - <%= studentIDs.get(i) %><br />
15 <%
16     }
17 %>
18   </body>
19 </html>
```

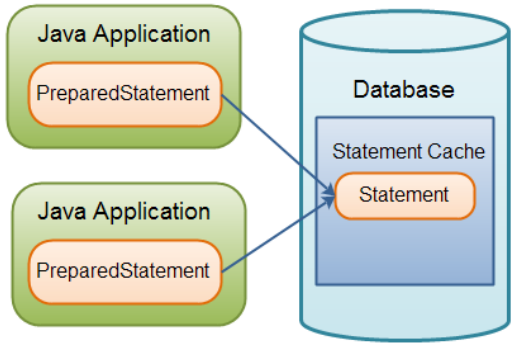


# Outline

- JDBC
- Prepared Statements
- Program

# Prepared Statements

- The `Statement` interface seen in the previous slides is used to execute static SQL statements that contain no parameters
- The `PreparedStatement` interface, extending the `Statement` interface, is used to execute a precompiled SQL statement that may or may not contain parameters
  - `PreparedStatement`s are efficient for repeated executions since they are precompiled
- We will not have to append `String`s or other variable types together in a `PreparedStatement`



# JDBC Prepared Statement Code

```
1  import java.sql.*; // to save space
2  public class JDBCTest {
3    public static void main (String[] args) {
4      Connection conn = null;
5      PreparedStatement ps = null;
6      ResultSet rs = null;
7      try {
8        Class.forName("com.mysql.jdbc.Driver");
9        conn = DriverManager.getConnection("jdbc:mysql://localhost/StudentGrades?user=root&password=root&useSSL=false");
10       String name = "Sheldon";
11       ps = conn.prepareStatement("SELECT * FROM Student WHERE fname=?");
12       ps.setString(1, name); // set first variable in prepared statement
13       rs = ps.executeQuery();
14       while (rs.next()) {
15         String fname = rs.getString("fname");
16         String lname = rs.getString("lname");
17         int studentID = rs.getInt("studentID");
18         System.out.println ("fname = " + fname);
19         System.out.println ("lname = " + lname);
20         System.out.println ("studentID = " + studentID);
21       }
22     } catch (SQLException sqle) {
23       System.out.println (sqle.getMessage());
24     } catch (ClassNotFoundException cnfe) {
25       System.out.println (cnfe.getMessage());
26     } finally {
```

```
27       try {
28         if (rs != null) {
29           rs.close();
30         }
31         if (ps != null) {
32           ps.close();
33         }
34         if (conn != null) {
35           conn.close();
36         }
37       } catch (SQLException sqle) {
38         System.out.println(sqle.getMessage());
39       }
40     }
41   }
42 }
```

# Prepared Statements Benefits

- When should you use a `PreparedStatement`?
  - When using input from a user in a SQL statement (to prohibit SQL injection attacks)
  - When running the same statement more than once (since the statement is precompiled in the DBMS)
- When should you use a `Statement`?
  - When the SQL statement is only being used once AND does not use any input from the user

# Outline

- JDBC
- Prepared Statements
- Program

# SQL Queries

- Create a web page to display the following queries as a formatted table
  - Print out the name and grade for each student in CSCI 103
  - Print out the class and grades for Sheldon Cooper
  - Print out the name, grade, and class for all students in all of the classes
  - Change Leonard's grade in EE 101 to a C+
  - Add Amy Farrah Fowler into the Student table
  - Give Amy grades of A for CSCI 103

SQL

# SQL Queries Solution

- Solutions
  - Print out the name and grade for each student in CSCI 103
    - `SELECT s.fname, s.lname, g.letterGrade FROM student s, grade g, class c WHERE s.studentID=g.studentID AND g.classID=c.classID AND c.prefix='CSCI' AND c.num=103;`
  - Print out the class and grades for Sheldon Cooper
    - `SELECT c.prefix, c.num, g.letterGrade FROM student s, grade g, class c WHERE c.classID=g.classID AND s.studentID=g.studentID AND s.fname='Sheldon' AND s.lname='Cooper';`
  - Print out the name, grade, and class for all students in all of the classes
    - `SELECT s.fname, s.lname, c.prefix, c.num, g.letterGrade FROM student s, grade g, class c WHERE c.classID=g.classID AND s.studentID=g.studentID ORDER BY s.fname, s.lname, c.prefix, c.num;`
  - Change Leonard's grade in EE 101 to a C+
    - `UPDATE grade g, student s, class c SET g.letterGrade='C+' WHERE s.studentID=g.studentID AND s.fname='Leonard' AND s.lname='Hofstadter' AND g.classID=c.classID AND c.prefix='EE' AND c.num='101';`
  - Add Amy Farrah Fowler into the Student table
    - `INSERT INTO student (fname, lname) VALUES ('Amy', 'Farrah Fowler');`
  - Give Amy a grade of A for CSCI 103
    - `INSERT INTO grade (classID, studentID, letterGrade) VALUES ((SELECT classID FROM class WHERE prefix='CSCI' AND num=103), (SELECT studentID FROM student WHERE fname='Amy' AND lname='Farrah Fowler'), 'A');`