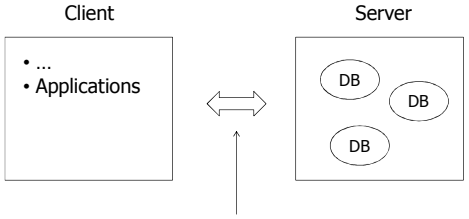
CS320 Web and Internet Programming
Database Access with JDBC

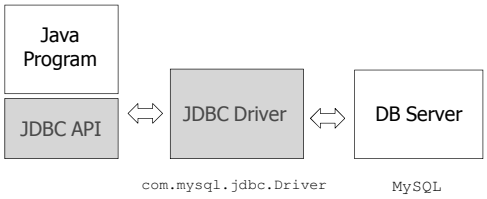
Chengyu Sun
California State University, Los Angeles

Client-Server Architecture of Databases



A *library* that sends SQL and other commands from client to server, and get the results from server to client.

JDBC



About JDBC

- Java DataBase Connectivity
- JDBC API is DBMS independent
- JDBC Driver implements JDBC API for a particular DBMS

Example: HelloJDBC.java

- Load JDBC driver
  - Only need to be done once per application
- Make connections
- Execute queries
- Process results
- Handle exceptions (and close connections)

JDBC Basics …

- import java.sql.*;
- Initialize driver
  - Class.forName("com.mysql.jdbc.Driver")
- Create connection
  - URL
    - jdbc:mysql://[host:port]/[database][?user=cs320stu31&password=abcd]
  - DriverManager.getConnection(URL)
  - DriverManager.getConnection(URL, user, pass)

… JDBC Basics

- Create statement
  - Statement stmt = c.createStatement();
    - stmt.executeQuery(String sql)
    - stmt.executeUpdate(String sql)
- Get result back
  - ResultSet rs

http://docs.oracle.com/javase/tutorial/jdbc/index.html

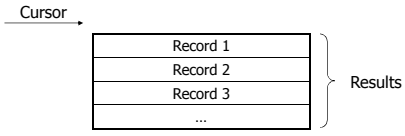
DB Query Results

- In a program, we want to
  - Access each record
  - Access each attribute in a record
  - Access the name of each attribute

select * from items;

name	price	quantity
Milk	3.89	2
Beer	6.99	1

JDBC ResultSet – Row Access



- `next()` – move cursor down one row
  - Cursor starts from *before the 1st record*
  - `true` if the current record is valid
  - `false` if no more records

Common Code for Processing ResultSet

- Process each row
  - `while(rs.next()) {…}`
- Check whether a result set is empty
  - `if(rs.next()) {…}`

JDBC ResultSet – Column Access

- Access the columns of *current row*
- getXxx(String columnName)
  - E.g. getString("user");
- getXxx(int columnIndex)
  - columnIndex starts from 1
  - E.g. getString(1);

http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html

JDBC ResultSet – Access Column Names

ResultSetMetaData meta = rs.getMetaData();

- ResultSetMetaData
  - getColumnName(columnIndex)
    - Column name
  - getColumnLabel(columnIndex)
    - Column title for display or printout

JDBC ResultSet – Size

- No size() method?
- Something about *FetchSize*
  - getFetchSize()
  - setFetchSize(int nrows)

Handle Exceptions

```
catch( SQLException e )
{
   throw new ServletException( e );
}
finally
{
   try
   {
      if( c != null ) c.close();
   }
   catch( SQLException e )
   {
      throw new ServletException( e );
   }
}
```

Example: GuestBook (JDBC) – Display

- Create a `guest_book` table
- Retrieve the entries in a servlet
- Display the entries in a JSP

Example: GuestBook (JDBC) – Add

- Save new guest book entries to the database
  - executeQuery() vs. executeUpdate()
- Potential problems of handing user input
  - Special characters
  - SQL injection attack

Example: SQL Injection Attack

- User input should NOT be trusted
- Regular user input
  - Username: `cysun`
  - Password: `abcd`
- Malicious user input
  - Username: `someuser`
  - Password: `something' or '1`
- *Prevent SQL injection attack?*

Prepared Statements

- Statements with parameters

```
String sql = "insert into items values (? ? ?)";

PreparedStatement pstmt =c.prepareStatement(sql);

pstmt.setString(1, "orange");
pstmt.setBigDecimal(2, 0.59);
pstmt.setInt(3, 4);

pstmt.executeUpdate();
```

Benefits of Prepared Statements

- Special characters are properly handled
- Secure if the SQL statement is constructed from user input
- The SQL statement is more readable
- Better performance (maybe)