Objectives
• Review: Databases
• JDBC
• Static Mock-up Demo

Review: Databases
• What do databases do for us?
• What are tables made up of?
• What language do we use to query and update relational databases?
• What is the syntax for the SELECT statement?

Project Overview

JDBC: Java Database Connectivity
• Database-independent connectivity
  ➢ JDBC converts generalized JDBC calls into vendor-specific SQL calls
• Classes in java.sql.* and javax.sql.* packages

Using JDBC in a Java Program
• Load the database driver
• Obtain a connection
• Create and execute statements (SQL queries)
• Use result sets (tables) to navigate through the results
• Close the connection

java.sql.DriverManager
• Provides a common access layer for different database drivers
• Requires that each driver used by the application be registered before use
• Load the database driver by its name using ClassLoader:

```java
Class.forName("org.postgresql.Driver");
```
JDBC: Creating a Connection

- After loading the DB driver, create the connection
  (see API for all ways)

```java
String url = "jdbc:postgresql://hopper:5432/cs297";
Connection con = DriverManager.getConnection(url, username, password);
```

- Close connection when done
  > Release resources
  ```java
  con.close();
  ```

Where should these code fragments go in a Servlet?

JDBC: Executing Queries: Statements

- `executeQuery(String query)`
  > Returns a ResultSet
  > Iterate through ResultSet, row-by-row, getting results from each column
  ```java
  ResultSet rs = stmt.executeQuery("SELECT * FROM table");
  ```

- `executeUpdate(String query)` to update table
  > Returns an integer representing the number of affected rows
  ```java
  Statement stmt = con.createStatement();
  ```

Iterating Through ResultSets

- Example:
  ```java
  ResultSet rs = stmt.executeQuery("SELECT * FROM majors");
  while (rs.next()) {
      String name = rs.getString("name");
      String dept = rs.getString(2); // column 2
      System.out.println(name + "	" + dept);
  }
  ```

  Can access column values by name or which column (count starts at 1, left to right)

Useful ResultSet Methods

- Number of rows in the result:
  ```java
  rs.last();
  int numberOfRows = rs.getRow();
  ```

  Information about the table, such as number, types, and properties of columns:
  ```java
  ResultSetMetaData getMetaData();
  ```

JDBC: Prepared Statements

- `con.prepareStatement(String template)`
  > Compile SQL statement "templates"

  ```java
  updateSales = con.prepareStatement("INSERT INTO Majors (name, department) VALUES (?, ?)");
  ```

  Set parameters
  ```java
  updateSales.setInt(1, 100);
  updateSales.setString(2, "French Roast");
  ```

  Columns start at 1

JDBC

- API Documentation: `java.sql.*`
  > Statements, Connections, ResultSets, etc. are all Interfaces
    > Driver/Library implements interfaces for its database
  > Limitations:
    > Java doesn't compile the SQL statements
    > Exact syntax depends on DB
    > Compile, run, verify queries outside of Java for your database
    > Then copy and use in Java code
Using PostgreSQL on Command-Line

- In a terminal, `ssh` into `hopper`
  - `ssh hopper`
- Run the PostgreSQL client: `psql`, connecting to the appropriate database
  - `psql cs297`
- At the prompt, type in SQL statements, ending in `;`

Examples Using JDBC

Transactions in JDBC

- By default, a connection is in `auto-commit` mode
  - Each statement is a transaction
  - Automatically committed as soon as executed

Transactions in JDBC

- You can turn off auto-commit and execute multiple statements as a transaction
  - Database can keep handling others’ reads
  - Others won’t see updates until you commit

```java
con.setAutoCommit(false);
// execute SQL statements...
con.commit(); // commit those statements
con.setAutoCommit(true);
```

- Can call `rollback` to abort updates

Storing Passwords

- Use `md5` function on passwords
  - `md5('password')`
- Compare user’s input password md5’d with password in database
  - `SELECT COUNT(id) FROM Users WHERE username=? AND password=md5(?)`;
- Example: username and password = 'test'

Connection Pool

- Want to reuse DB connections
  - Reduce overhead of creating and closing connections to database
- Could write our own connection pool class
  - Many examples online
- Apache wrote one that we’ll use
  - `http://commons.apache.org/dbcp/`
  - Need to add its jars to our lib directory
Using the Connection Pool

- Create a `DataSource` object in the `ServletContext`
  - All the servlets can see the `ServletContext`
  - Shared resource, given name, value
- Create a `DBConnectionServlet` class
  - `init` method gets the `DataSource` object from the `ServletContext`
  - When need a connection, call `getConnection` on `DataSource` object
  - Servlets that need the DB extend `DBConnectionServlet`
- Examples in Logic project later

Midterm Info

- Midterm - next Wednesday
- Covers
  - HTML, CSS
  - WWW in general - distributed ideas
  - Usability
  - Servlets, JSPs, Application organization
  - Synchronization, Version Control
  - SQL, JDBC

TODO

- Lab 7: JDBC
  - Due Friday
- Study for midterm
- In a bit: Static HTML Mock-up Demo

Issues

- Welcome Page?
  - Text from Professor Gregory?
  - Documentation links?
- Logout link: Student, Professors
  - From Logout: return to Welcome Page, with message “You have logged out”
- Adding answers: needs the symbol table
- Deleting: before deleting, make sure that’s really what the user wants