SOEN 387 Web-based Enterprise Application Design

Stuart Thiel

Concordia University
Department of Computer & Software Engineering

Fall, 2015

Interacting with Databases

Transaction Scripts

Data Layer Patterns

Row Data Gateway

Outline

Interacting with Databases

Transaction Scripts

Data Layer Patterns

Row Data Gateway

Making a connection

- To create a db connection, handshake with db server
- Can be expensive if server is remote
- Subsequent access is easy, if connection is passed around
- Don't share that connection with another request!!!

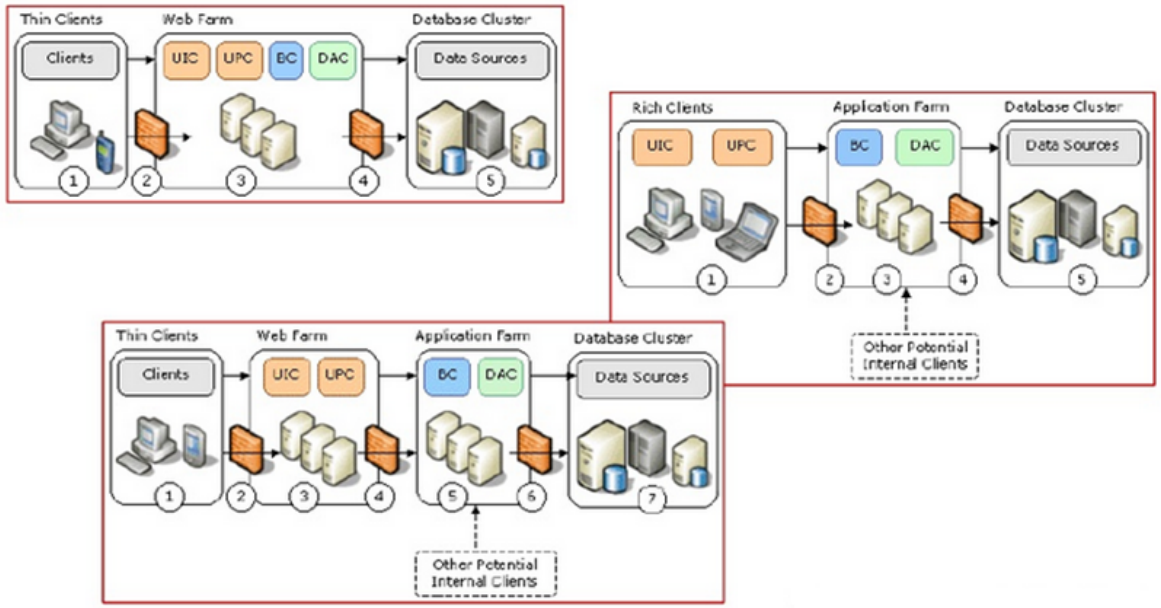
Local Database Servers

- Local database servers eliminate much of the handshaking cost
- This is a common setup, but not always this way

Separate Database Server

- To eliminate those handshaking costs, pool connections
- Just clean them after use
- Can pollute next request's use if not careful
- How to recover hung connections?

Two-Tier Systems



- Layers are usually a logical separation
- Tiers usually imply physical separation
- In this can, we see example of separated data source

Java Database Connections, MySQL

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
            // The newInstance() call is a work around for some
            // broken Java implementations

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

            conn = DriverManager.getConnection(
                        "jdbc:mysql://localhost/test?"
                    + "user=monty&password=greatsqldb");

        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

Java Database Connections, Configuration

- There are additional configurations to be had using the query notation
- Not covered right now, but often indicate utf-8

Making Select Queries

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
                // ...
            // assume you have conn above

                String query = "SELECT * FROM users";
                Statement st = conn.createStatement();
                ResultSet rs = st.executeQuery(query);

        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

Results From Select Queries

```
public class LoadDriver {
    public static void main(String[] args) {
        try {
                // ...
            // assume you have the resultset rs
            while (rs.next()) {
              int id = rs.getInt("id");
              String firstName = rs.getString("first_name");
              String lastName = rs.getString("last_name");
              Date dateCreated = rs.getDate("date_created");
              boolean isAdmin = rs.getBoolean("is_admin");
              int numPoints = rs.getInt("num_points");
            }
            st.close();
        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

Making Database Modification: Insert

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
                // ...
            // assume you have conn above

                String query = " insert into users "
                                + " (first_name, last_name, num_points)"
                        + " values (\"Fred\", \"Durst\", 1)";
                Statement st = conn.createStatement();
                int count = st.executeUpdate(query);

        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

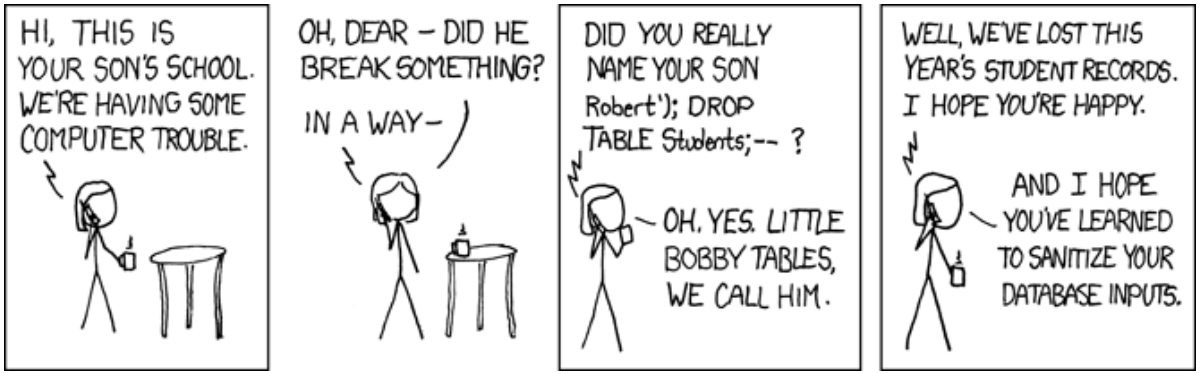
Making Database Modification: Update

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
                // ...
            // assume you have conn above

                String query = "update users set "
                        + "num_points = 2 where first_name = \"Fred\"";
                Statement st = conn.createStatement();
                int count = st.executeUpdate(query);

        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

Adding/Dropping Tables

- You can also delete rows
- You can add/drop tables

Prepared Statements



- Protect your data!

Prepared Statements Code

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

// Notice, do not import com.mysql.jdbc.*
// or you will have problems!

public class LoadDriver {
    public static void main(String[] args) {
        try {
                // ...
            // assume you have conn above

                String query = " insert into users "
                             + "(first_name, last_name, num_points)"
                       + " values (?, ?, ?)";
                PreparedStatement preparedStmt =
                             conn.prepareStatement(query);
                preparedStmt.setString(1, "Fred");
                preparedStmt.setString(2, "Durst");
                preparedStmt.setInt   (3, 1);

                int count = preparedStmt.executeUpdate();

        } catch (Exception ex) {
            // handle the error
        }
    }
}
```

Transaction Scripts

Organizes business logic by procedures where each procedure handles a single request from the presentation.

- Reads any parameters
- Pulls any data
- Manipulates any data
- Writes any data
- Formats response for client

Transaction Scripts

Organizes business logic by procedures where each procedure handles a single request from the presentation.

- Reads any parameters
- Pulls any data
- Manipulates any data
- Writes any data
- Formats response for client
- . . . all in one class

Transaction Scripts

Organizes business logic by procedures where each procedure handles a single request from the presentation.

- Reads any parameters
- Pulls any data
- Manipulates any data
- Writes any data
- Formats response for client
- . . . all in one class
- Is this a good idea?

Example TS


Application / Presentation
HelloWorldServlet
Domain
+ procesGet(req:HttpServletRequest, resp:HttpServletResponse) : void
Technical Services

Example TS Layers

- Tutorial, Phase 1
  - Received request
  - Read parameters
  - Formated result
  - Returned result to client

What is wrong with this

- Cohesion
- Coupling
- Layers?

What is wrong with this

- Cohesion
- Coupling
- Layers?
- What did we do in the tutorial?

How Much To See

- We can also start cleaning up TS at the data-source layer
- What if full names come from a DB?
- Transaction Script can access rows as a service

Row Data Gateway, Refactoring