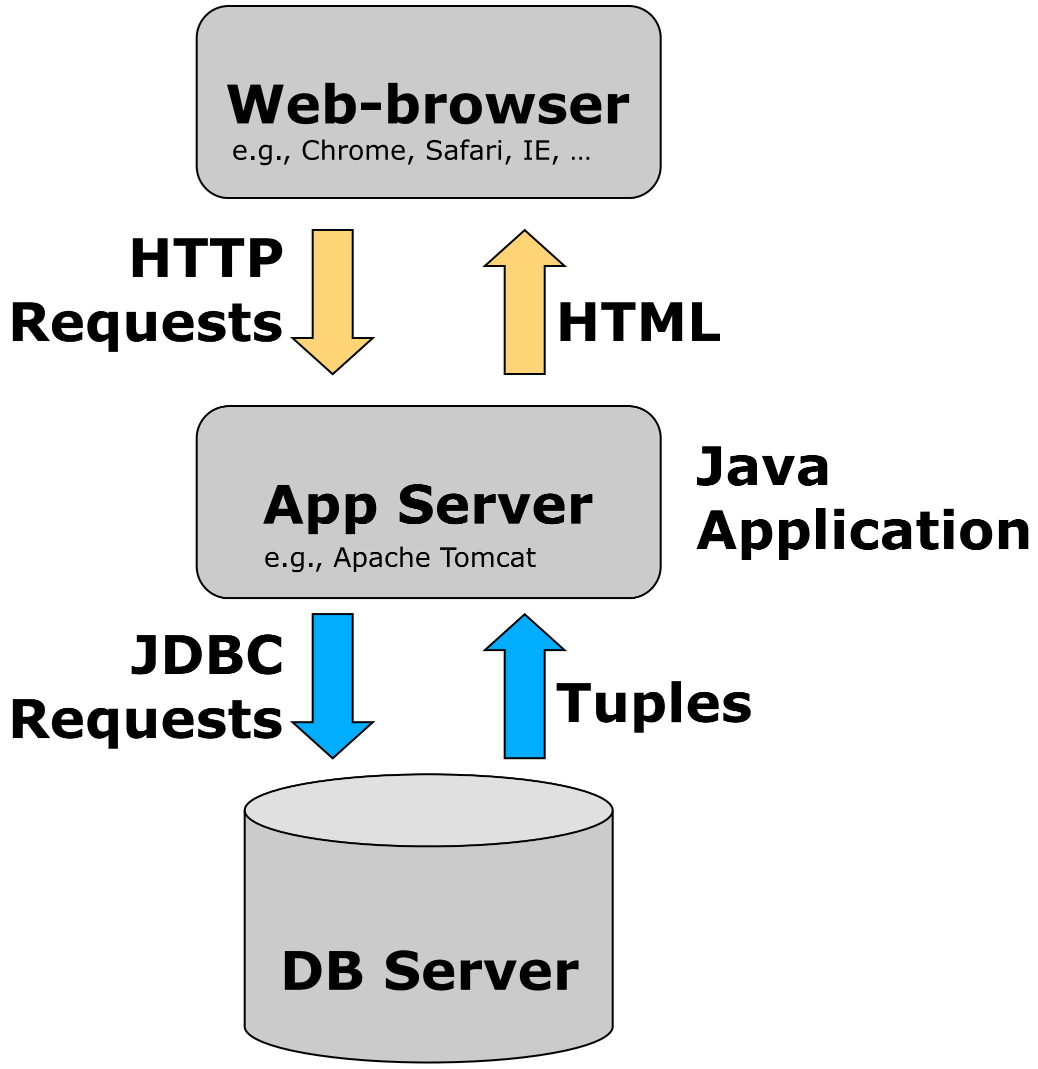
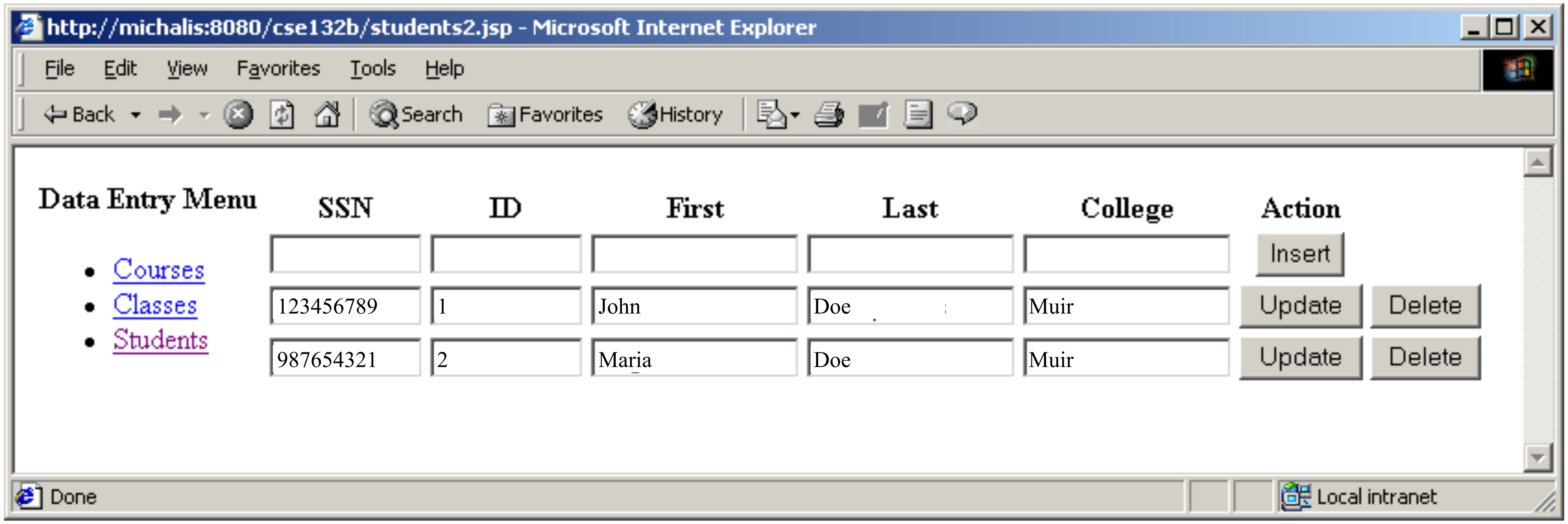
JDBC

Three-Tier Architecture



Example Data Entry Forms



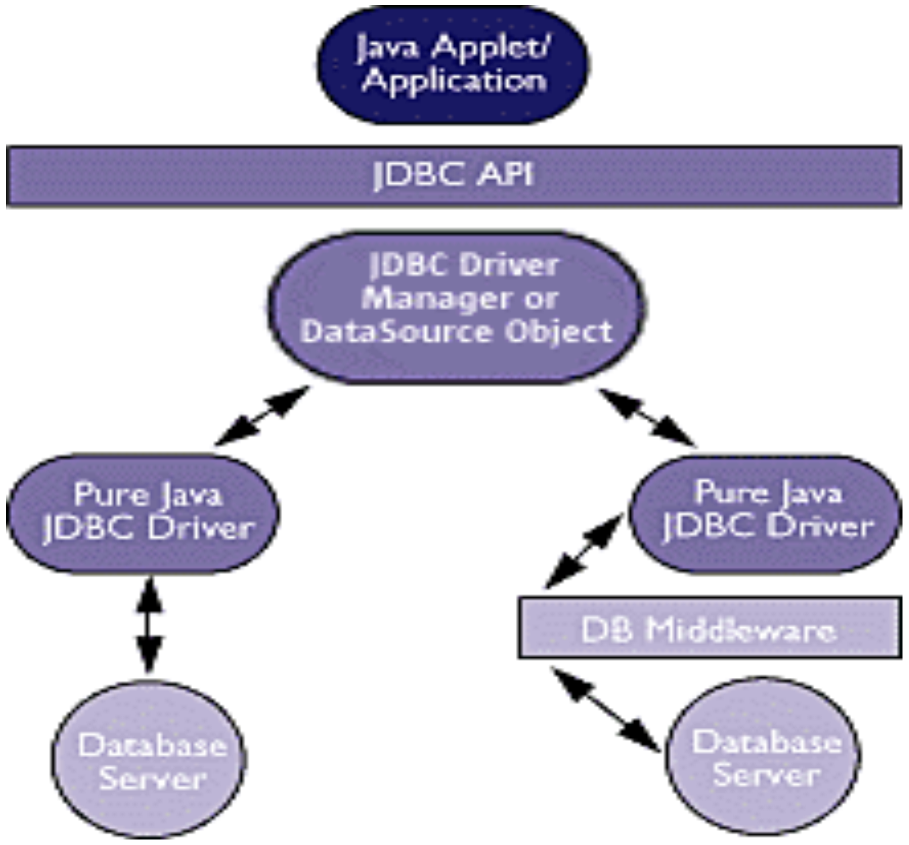
http://michalis:8080/cse132b/students2.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
Local intranet

Data Entry Menu

- Courses
- Classes
- Students

SSN	ID	First	Last	College	Action	
					Insert	
123456789	1	John	Doe	Muir	Update	Delete
987654321	2	Maria	Doe	Muir	Update	Delete

Java Database Connectivity (JDBC)



JDBC Example

```
 // Import JDBC
import java.sql.*;


class JdbcTest {
public static void main (String args []) throws SQLException,
                                               ClassNotFoundException {
 // Load PostgreSQL driver
Class.forName(“org.postgresql.Driver”);
// Connect to the local database
Connection conn =  DriverManager.getConnection
         (“jdbc:postgresql://hostname:port/dbname”,
          “username”, “password”);
```

JDBC Example

```
// Execute query asking for student names
Statement stmt = conn.createStatement ();
ResultSet rset = stmt.executeQuery ("SELECT name FROM
    Student");
// Print the name out (name is the 2nd attribute of Student)
while (rset.next ())
     System.out.println (rset.getString (2));

// Close the result set, statement, and the connection
rset.close();
stmt.close();
conn.close();
```

PreparedStatement Object

If you want to execute a Statement object many times, it will normally reduce execution time to use a PreparedStatement object instead

```
// Create PreparedStatement
PreparedStatement updateStud =
   conn.prepareStatement( "UPDATE Student SET name = ?
   WHERE lastname LIKE ?");
```

Can contain parameters

```
// Instantiate parameters and execute the PreparedStatement
updateStud.setString(1, “John”);
updateStud.setString(2, “Smith”);

updateStud.executeUpdate();
```

PreparedStatement Object

The following two code fragments accomplish the same thing:

- **Code Fragment 1:**

  ```
  String updateString = "UPDATE COFFEES SET SALES = 75
  " + "WHERE COF_NAME LIKE 'Colombian'";
  stmt.executeUpdate(updateString);
  ```

- **Code Fragment 2:**

  ```
  PreparedStatement updateSales =
  con.prepareStatement( "UPDATE COFFEES SET SALES
  = ? WHERE COF_NAME LIKE ? ");
  updateSales.setInt(1, 75);
  updateSales.setString(2, "Colombian");
  updateSales.executeUpdate():
  ```

Retrieving values from a ResultSet

Retrieves the value of the designated column in the current row of this ResultSet object as an int in Java.

- int getInt(int columnIndex)
- int getInt(String columnName)

Retrieves the value of the designated column in the current row of this ResultSet object as a string in Java.

- String getString(int columnIndex)
- String getString(String columnName)

Using Transactions

When a connection is created, it is in auto-commit mode. This means that each individual SQL statement is treated as a transaction and will be automatically committed right after it is executed. To create transactions, do the following:

```
conn.setAutoCommit(false);
....
transaction
...
con.commit();
con.setAutoCommit(true);
```

Using Transactions Example

```
con.setAutoCommit(false);
PreparedStatement updateSales = con.prepareStatement( "UPDATE
    COFFEES SET SALES = ? WHERE COF_NAME LIKE ?");
updateSales.setInt(1, 50);
updateSales.setString(2, "Colombian");
updateSales.executeUpdate();
PreparedStatement updateTotal = con.prepareStatement( "UPDATE
    COFFEES SET TOTAL = TOTAL + ? WHERE COF_NAME LIKE ?");
    updateTotal.setInt(1, 50);
updateTotal.setString(2, "Colombian");
updateTotal.executeUpdate();
con.commit();

con.setAutoCommit(true);
```

Catching Exceptions

JDBC lets you see the warnings and exceptions generated by your DBMS and by the Java compiler. To see exceptions, you can have a catch block print them out. For example, the following two catch blocks from the sample code print out a message explaining the exception:

```
try {
// Code that could generate an exception goes here.
// If an exception is generated, the catch block below
// will print out information about it.
} catch(SQLException ex) {
System.err.println("SQLException: " + ex.getMessage());
}
```