



Database Access with JDBC

Customized J2EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, Struts, JSF/MyFaces, Hibernate, Ajax, GWT, Java 5, Java 6, etc. Ruby/Rails coming soon. Developed and taught by well-known author and developer. At public venues or onsite at *your* location.



For live JSF training, please see training courses at <http://courses.coreservlets.com/>.



Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
 - Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, GWT, custom courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
 - Spring, Hibernate, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Overview

- Overview of JDBC technology
- JDBC drivers
- Seven basic steps in using JDBC
- Using JNDI with JDBC
- Retrieving data from a ResultSet
- Using prepared and callable statements
- Handling SQL exceptions
- Submitting multiple statements as a transaction

5

J2EE training: <http://courses.coreservlets.com>

JDBC Introduction

- **JDBC provides a standard library for accessing relational databases**
 - API standardizes
 - Way to establish connection to database
 - Approach to initiating queries
 - Method to create stored (parameterized) queries
 - The data structure of query result (table)
 - Determining the number of columns
 - Looking up metadata, etc.
 - API does not standardize SQL syntax
 - JDBC is not embedded SQL
 - JDBC classes are in the java.sql package
- **Note: JDBC is not officially an acronym; unofficially, “Java DataBase Connectivity”**

6

J2EE training: <http://courses.coreservlets.com>

On-line Resources

- **Sun's JDBC Site**
 - <http://java.sun.com/products/jdbc/>
- **JDBC Tutorial**
 - <http://java.sun.com/docs/books/tutorial/jdbc/>
- **List of Available JDBC Drivers**
 - <http://industry.java.sun.com/products/jdbc/drivers/>
- **API for java.sql**
 - <http://java.sun.com/j2se/1.4/docs/api/java/sql/package-summary.html>

7

J2EE training: <http://courses.coreservlets.com>

Oracle On-line Resources

- **JDBC Road Map**
 - <http://technet.oracle.com/tech/java/jroadmap/index2.htm?Info&jdbc/listing.htm>
- **SQLJ & JDBC Basic Samples**
 - http://technet.oracle.com/tech/java/sqlj_jdbc/index2.htm?Code&files/basic/basic.htm
- **JDBC Drivers**
 - http://technet.oracle.com/software/tech/java/sqlj_jdbc/htdocs/listing.htm
 - Requires free registration
- **Certification**
 - <http://technet.oracle.com/training/certification/>

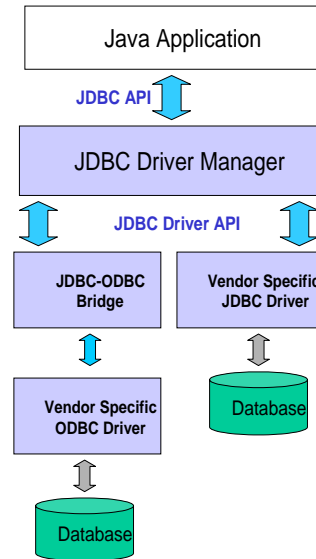
8

J2EE training: <http://courses.coreservlets.com>

JDBC Drivers

- **JDBC consists of two parts:**

- JDBC API, a purely Java-based API
- JDBC Driver Manager, which communicates with vendor-specific drivers that perform the real communication with the database.
 - Point: translation to vendor format is performed on the client
 - No changes needed to server
 - Driver (translator) needed on client



JDBC Data Types

JDBC Type	Java Type
BIT	boolean
TINYINT	byte
SMALLINT	short
INTEGER	int
BIGINT	long
REAL	float
FLOAT	double
DOUBLE	
BINARY	byte[]
VARBINARY	
LONGVARBINARY	
CHAR	String
VARCHAR	
LONGVARCHAR	

JDBC Type	Java Type
NUMERIC	BigDecimal
DECIMAL	
DATE	java.sql.Date
TIME	java.sql.Timestamp
TIMESTAMP	
CLOB	Clob*
BLOB	Blob*
ARRAY	Array*
DISTINCT	mapping of underlying type
STRUCT	Struct*
REF	Ref*
JAVA_OBJECT	underlying Java class

*SQL3 data type supported in JDBC 2.0

Seven Basic Steps in Using JDBC

1. Load the driver
2. Define the Connection URL
3. Establish the Connection
4. Create a Statement object
5. Execute a query
6. Process the results
7. Close the connection

11

J2EE training: <http://courses.coreservlets.com>

JDBC: Details of Process

1. Load the driver

```
try {  
    Class.forName("connect.microsoft.MicrosoftDriver");  
    Class.forName("oracle.jdbc.driver.OracleDriver");  
} catch (ClassNotFoundException cnfe) {  
    System.out.println("Error loading driver: " cnfe);  
}
```

2. Define the Connection URL

```
String host = "dbhost.yourcompany.com";  
String dbName = "someName";  
int port = 1234;  
String oracleURL = "jdbc:oracle:thin:@" + host +  
    ":" + port + ":" + dbName;  
String sybaseURL = "jdbc:sybase:Tds:" + host +  
    ":" + port + ":" +  
    "?SERVICENAME=" + dbName;
```

12

J2EE training: <http://courses.coreservlets.com>

JDBC: Details of Process (Continued)

3. Establish the Connection

```
String username = "jay_debese";
String password = "secret";
Connection connection =
    DriverManager.getConnection(oracleURL,
                                username,
                                password);
```

- Optionally, look up information about the database

```
DatabaseMetaData dbMetaData =
    connection.getMetaData();
String productName =
    dbMetaData.getDatabaseProductName();
System.out.println("Database: " + productName);
String productVersion =
    dbMetaData.getDatabaseProductVersion();
```

13

J2EE training: <http://courses.coreservlets.com>

JDBC: Details of Process (Continued)

4. Create a Statement

```
Statement statement =
    connection.createStatement();
```

5. Execute a Query

```
String query =
    "SELECT col1, col2, col3 FROM sometable";
ResultSet resultSet =
    statement.executeQuery(query);
```

- To modify the database, use `executeUpdate`, supplying a string that uses UPDATE, INSERT, or DELETE
- Use `setQueryTimeout` to specify a maximum delay to wait for results

14

J2EE training: <http://courses.coreservlets.com>

JDBC: Details of Process (Continued)

6. Process the Result

```
while(resultSet.next()) {  
    System.out.println(resultSet.getString(1) + "  
    " +  
    resultSet.getString(2) + "  
    " +  
    resultSet.getString(3));  
}
```

- First column has index 1, not 0
- ResultSet provides various getXxx methods that take a colu index *or column name* and returns the data
- You can also access result meta data (column names, etc.)

7. Close the Connection

```
connection.close();
```

- Since opening a connection is expensive, postpone this step if additional database operations are needed.

15

J2EE training: <http://courses.coreservlets.com>

The Microsoft Access Northwind Database

- Database that comes preinstalled with Microsoft Office

- Or download from

<http://office.microsoft.com/downloads/2000/Nwind2K.aspx>

The screenshot shows the Microsoft Access interface for the Northwind Database. The 'Employees' table is selected in the 'Objects' pane on the left. The main window displays the 'Employees : Table' view, which is a table with the following data:

	Last Name	First Name	Title	Title Of	Birth Date
▶	Davolio	Nancy	Sales Representative	Ms.	08-Dec-1968
+	Fuller	Andrew	Vice President, Sales	Dr.	19-Feb-1952
+	Leverling	Janet	Sales Representative	Ms.	30-Aug-1963
+	Peacock	Margaret	Sales Representative	Mrs.	19-Sep-1958
+	Buchanan	Steven	Sales Manager	Mr.	04-Mar-1955
+	Suyama	Michael	Sales Representative	Mr.	02-Jul-1963
+	King	Robert	Sales Representative	Mr.	29-May-1960
+	Callahan	Laura	Inside Sales Coordinator	Ms.	09-Jan-1958
+	Dodsworth	Anne	Sales Representative	Ms.	02-Jul-1969
*					

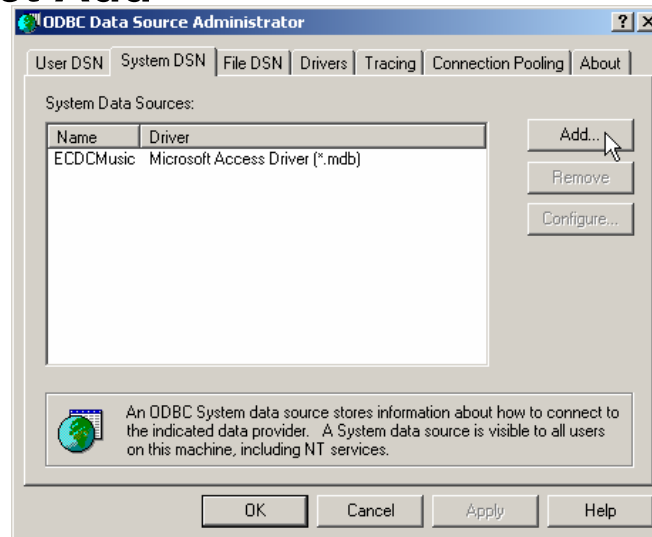
Record: 1 of 9

16

J2EE training: <http://courses.coreservlets.com>

Using Microsoft Access via ODBC

- Click Start, Control Panel, Administrative Tools, Data Sources, System DSN, and select Add

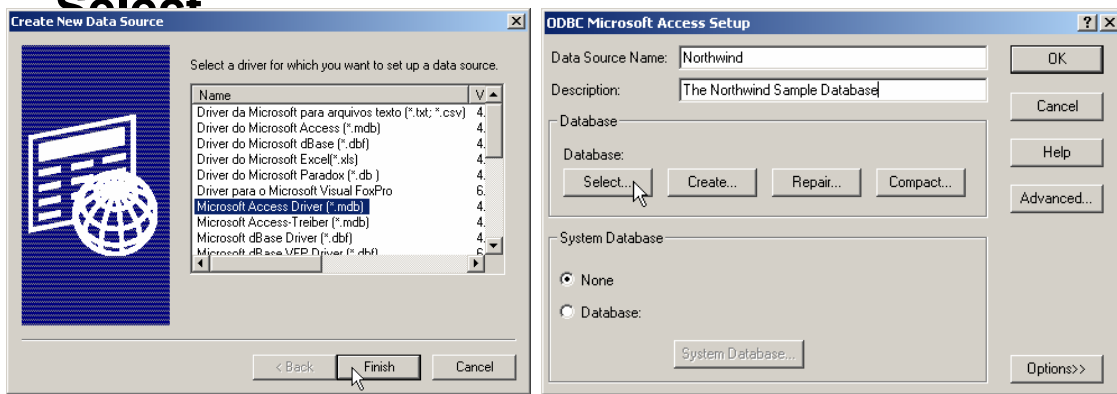


17

J2EE training: <http://courses.coreservlets.com>

Using Microsoft Access via ODBC (Continued)

- Select Microsoft Access Driver, Finish, type a name under Data Source Name, and hit Select

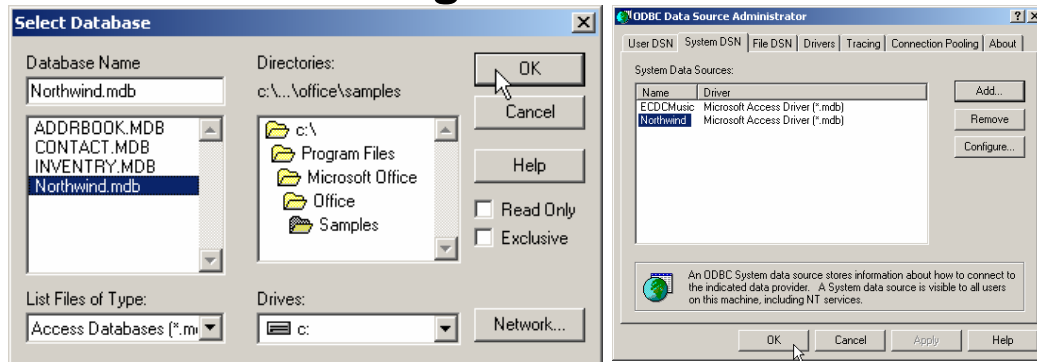


18

J2EE training: <http://courses.coreservlets.com>

Using Microsoft Access via ODBC (Continued)

- Navigate to the Samples directory of MS Office, select Northwind.mdb, hit OK, then hit OK in following two windows



19

J2EE training: <http://courses.coreservlets.com>

Using Microsoft Access via ODBC (Continued)

- Use `sun.jdbc.odbc.JdbcOdbcDriver` as the class name of the JDBC driver.
 - `Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`
- Use `"jdbc:odbc:Northwind"` as the database address, and use empty strings for the username and password.
 - Connection connection =
`DriverManager.getConnection("jdbc:odbc:Northwind", "", "");`

20

J2EE training: <http://courses.coreservlets.com>

Simple Standalone Northwind Test

```
package coreservlets;

import java.sql.*;

public class NorthwindTest {
    public static void main(String[] args) {
        String driver =
            "sun.jdbc.odbc.JdbcOdbcDriver";
        String url = "jdbc:odbc:Northwind";
        String username = "";
        String password = "";
        showEmployeeTable(driver, url,
            username, password);
    }
}
```

21

J2EE training: <http://courses.coreservlets.com>

Simple Standalone Northwind Test (Continued)

```
    public static void showEmployeeTable(String driver,
                                         String url,
                                         String username,
                                         String password)
    {
        try {
            // Load database driver if not already loaded.
            Class.forName(driver);
            // Establish network connection to database.
            Connection connection =
                DriverManager.getConnection(url,
                                         username,
                                         password);
            System.out.println("Employees\n" +
                               "=====");
            Statement statement =
                connection.createStatement();
            String query =
                "SELECT firstname, lastname FROM employees";
            // Send query to database and store results.
```

22

J2EE training: <http://courses.coreservlets.com>

Simple Standalone Northwind Test (Continued)

```
// Print results.
while(resultSet.next()) {
    // First name
    System.out.print(resultSet.getString(1) + " ");
    // Last name
    System.out.println(resultSet.getString(2));
}
} catch(ClassNotFoundException cnfe) {
    System.err.println("Error loading driver: " +
cnfe);
} catch(SQLException sqle) {
    System.err.println("Error connecting: " + sqle);
}
}
```

23

J2EE training: <http://courses.coreservlets.com>

Simple Standalone Northwind Test: Results

```
Prompt> java
    coreservlets.NorthwindTest
```

```
Employees
=====
Nancy Davolio
Andrew Fuller
Janet Leverling
Margaret Peacock
Steven Buchanan
Michael Suyama
Robert King
Laura Callahan
Anne Dodsworth
```

24

J2EE training: <http://courses.coreservlets.com>

Using MetaData

- **System-wide data**
 - connection.getMetaData().getDatabaseProductName()
 - connection.getMetaData().getDatabaseProductVersion()
- **Table-specific data**
 - resultSet.getMetaData().getColumnCount()
 - When using the result, remember that the index starts at 1, not 0
 - resultSet.getMetaData().getColumnName()

25

J2EE training: <http://courses.coreservlets.com>

Using MetaData: Example

```
public class NorthwindServlet extends HttpServlet {
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        ... out.println(docType + ...);
        String driver = "sun.jdbc.odbc.JdbcOdbcDriver";
        String url = "jdbc:odbc:Northwind";
        String username = "";
        String password = "";
        String tableName =
request.getParameter("tableName");
        if ((tableName == null) || (tableName.equals(""))
        {
            tableName = "employees";
        }
        showTable(driver, url, username, password,
            tableName, out);
        out.println("</CENTER></BODY></HTML>");
    }
}
```

26

J2EE training: <http://courses.coreservlets.com>

Using MetaData: Example (Continued)

```
private void showTable(String driver,
                      String url,
                      String username,
                      String password,
                      String tableName,
                      PrintWriter out) {
    try {
        Class.forName(driver);
        Connection connection =
            DriverManager.getConnection(url, username,
password);
        DatabaseMetaData dbMetaData =
connection.getMetaData();
        out.println("<UL>");
        String productName =
            dbMetaData.getDatabaseProductName();
        out.println(" <LI><B>Database:</B> " +
            productName);
        String productVersion =
            dbMetaData.getDatabaseProductVersion();
        out.println(" <LI><B>Version:</B> " +
```

27

J2EE training: <http://courses.coreservlets.com>

Using MetaData: Example (Continued)

```
Statement statement = connection.createStatement();
String query =
    "SELECT * FROM " + tableName;
ResultSet resultSet = statement.executeQuery(query);
out.println("<TABLE BORDER=1>");
ResultSetMetaData resultsMetaData =
    resultSet.getMetaData();
int columnCount = resultsMetaData.getColumnCount();
out.println("<TR>");
for(int i=1; i<columnCount+1; i++) {
    out.print("<TH>" +
resultsMetaData.getColumnName(i));
}
out.println();
while(resultSet.next()) {
    out.println("<TR>");
    for(int i=1; i<columnCount+1; i++) {
        out.print("<TD>" + resultSet.getString(i));
    }
    out.println();
}
}
```

28

J2EE training: <http://courses.coreservlets.com>

Using MetaData: Results

DB Results - Microsoft Internet Explorer

Address <http://localhost/servlet/coreservlets.NorthwindServlet?tableName=customers>

Database Results

- Database: ACCESS
- Version: 04.00.0000

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	Berlin	null	12209	Germany	030-0074321	030-0076545
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	México				(5) 555-5555	(5) 555-5555
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México					
AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	London					
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	Luleå					
BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	Mannheim					
BLONP	Blondel père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	Strasbourg					
BOLID	Bólido Comidas preparadas	Martin Sommer	Owner	C/ Araquil, 67	Madrid					
BONAP	Bon app'	Laurence Lebihan	Owner	12, rue des Bouchers	Marseille					

DB Results - Microsoft Internet Explorer

Address <http://localhost/servlet/coreservlets.NorthwindServlet?tableName=shippers>

Database Results

- Database: ACCESS
- Version: 04.00.0000

ShipperID	CompanyName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

JNDI: Alternative Approach for Obtaining a Connection

- **Idea**
 - Use abstract name to get connection from a data source
- **Advantages**
 - Lets you change data source without changing code
 - Available in multiple Web apps
- **Disadvantage**
 - Requires server-specific registration of data source
- **Code for steps 1-3 replaced by:**

```
Context context = new InitialContext();
DataSource dataSource = (DataSource)context.lookup
    ("java:comp/env/jdbc/dbName");
Connection connection = dataSource.getConnection();
```

Defining a DataSource in Tomcat (tomcat_dir/conf/server.xml)

```
<DefaultContext reloadable="true">
  <Resource name="jdbc/Northwind" auth="Container"
    type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/Northwind">
    <parameter>
      <name>factory</name>
      <value>
        org.apache.commons.dbcp.BasicDataSourceFactory
      </value>
    </parameter>
    <parameter><name>driverClassName</name>
      <value>sun.jdbc.odbc.JdbcOdbcDriver</value>
    </parameter>
    <parameter><name>url</name>
      <value>jdbc:odbc:Northwind</value></parameter>
    <parameter><name>username</name>
      <value></value></parameter>
    <parameter><name>password</name>
      <value></value></parameter>
```

31

<http://courses.coreservlets.com>

Using Statement

• Overview

- Through the Statement object, SQL statements are sent to the database.
- Three types of statement objects are available:
 - **Statement**
 - For executing a **simple SQL statement**
 - **PreparedStatement**
 - For executing a **precompiled SQL statement** passing in parameters
 - **CallableStatement**
 - For executing a database **stored procedure**

32

J2EE training: <http://courses.coreservlets.com>

Useful Statement Methods

- **executeQuery**

- Executes the SQL query and returns the data in a table (ResultSet)
- The resulting table may be empty but never null

```
ResultSet results =  
    statement.executeQuery("SELECT a, b FROM  
table");
```

- **executeUpdate**

- Used to execute for INSERT, UPDATE, or DELETE SQL statements
- The return is the number of rows that were affected in the database
- Supports Data Definition Language (DDL) statements CREATE TABLE, DROP TABLE and ALTER TABLE

```
int rows =  
    statement.executeUpdate("DELETE FROM  
EMPLOYEES" +
```

33

J2EE training: <http://courses.coreservlets.com>

Useful Statement Methods (Continued)

- **execute**

- Generic method for executing stored procedures and prepared statements
- Rarely used (for multiple return result sets)
- The statement execution may or may not return a ResultSet (use statement.getResultSet). If the return value is true, two or more result sets were produced

- **getMaxRows/setMaxRows**

- Determines the maximum number of rows a ResultSet may contain
- Unless explicitly set, the number of rows is unlimited (return value of 0)

- **getQueryTimeout/setQueryTimeout**

- Specifies the amount of a time a driver will wait for a STATEMENT to complete before throwing a SQLException

34

J2EE training: <http://courses.coreservlets.com>

Prepared Statements (Precompiled Queries)

- **Idea**
 - If you are going to execute similar SQL statements multiple times, using “prepared” (parameterized) statements can be more efficient
 - Create a statement in standard form that is sent to the database for compilation before actually being used
 - Each time you use it, you simply replace some of the marked parameters using the `setXxx` methods
- **As PreparedStatement inherits from Statement the corresponding execute methods have no parameters**
 - `execute()`
 - `executeQuery()`

35

J2EE training: <http://courses.coreservlets.com>

Prepared Statement, Example

```
Connection connection =
    DriverManager.getConnection(url, user,
    password);
PreparedStatement statement =
    connection.prepareStatement("UPDATE
    employees "+
                                "SET salary = ?
    " +
                                "WHERE id = ?");
int[] newSalaries = getSalaries();
int[] employeeIDs = getIDs();
for(int i=0; i<employeeIDs.length; i++) {
    statement.setInt(1, newSalaries[i]);
    statement.setInt(2, employeeIDs[i]);
}
```

36

J2EE training: <http://courses.coreservlets.com>

Useful Prepared Statement Methods

- **setXxx**
 - Sets the indicated parameter (?) in the SQL statement to the value
- **clearParameters**
 - Clears all set parameter values in the statement
- **Handling Servlet Data**
 - Query data obtained from a user through an HTML form may have SQL or special characters that may require escape sequences
 - To handle the special characters, pass the string to the PreparedStatement **setString** method which will automatically escape the string as necessary

37

J2EE training: <http://courses.coreservlets.com>

Transactions

- **Idea**
 - By default, after each SQL statement is executed the changes are **automatically committed** to the database
 - Turn auto-commit off to group two or more statements together into a transaction
 - `connection.setAutoCommit(false)`
 - Call **commit** to permanently record the changes to the database after executing a group of statements
 - Call **rollback** if an error occurs

38

J2EE training: <http://courses.coreservlets.com>

Transactions: Example

```
Connection connection =
    DriverManager.getConnection(url, username, passwd);
connection.setAutoCommit(false);
try {
    statement.executeUpdate(...);
    statement.executeUpdate(...);

    connection.commit();
} catch (Exception e) {
    try {
        connection.rollback();
    } catch (SQLException sqle) {
        // report problem
    }
} finally {
    try {
        connection.close();
    } catch (SQLException sqle) { }
}
```

39

J2EE training: <http://courses.coreservlets.com>

Useful Connection Methods (for Transactions)

- **getAutoCommit/setAutoCommit**
 - By default, a connection is set to auto-commit
 - Retrieves or sets the auto-commit mode
- **commit**
 - Force all changes since the last call to commit to become permanent
 - Any database locks currently held by this `Connection` object are released
- **rollback**
 - Drops all changes since the previous call to commit
 - Releases any database locks held by this `Connection` object

40

J2EE training: <http://courses.coreservlets.com>

More JDBC Options

- **Stored procedures**
- **Changing buffer size**
- **Connection pooling**
- **JSP Standard Tag Library (JSTL) – custom tags to hide JDBC details**

Summary

- **You use the same Java syntax with all databases**
 - Translation to native format is done on the client via a JDBC driver
 - Standardized Java syntax does not equate to standardized SQL syntax
- **Steps in using JDBC**
 1. Load the driver
 2. Define the Connection URL
 3. Establish the Connection
 4. Create a Statement object
 5. Execute a query
 6. Process the results



Questions?

Customized J2EE Training: <http://courses.coreservlets.com/>

Servlets, JSP, Struts, JSF/MyFaces, Hibernate, Ajax, GWT, Java 5, Java 6, etc. Ruby/Rails coming soon.
Developed and taught by well-known author and developer. At public venues or onsite at *your* location.