31. Accessing MS-Access with Java

Java

Fall 2009
Instructor: Dr. Masoud Yaghini
Accessing MS-Access with Java

Outline

- JDBC-ODBC driver
- Creating an ODBC Data Source
- Connecting to a Database
- Querying a Database
- Retrieving Metadata
- Updating a Database
- References
JDBC-ODBC driver
To use the JDBC-ODBC driver to access databases in Java, two drivers must be installed on the computer:

- a universal JDBC-ODBC bridge driver
- a vendor-specific ODBC driver
Accessing MS-Access with Java

JDBC-ODBC driver
The JDBC-ODBC driver comes with Java 2 SDK 1.3 or higher.

The JDBC-to-ODBC Bridge allows any Java program to access any ODBC data source.

The driver is class JdbcOdbcDriver in package sun.jdbc.odbc.
ODBC driver

- On the Microsoft Windows platform, most databases support access via Open Database Connectivity (ODBC).
- ODBC is a technology developed by Microsoft to allow generic access to disparate database systems on the Windows platform.
ODBC driver

- By default the ODBC driver is installed on Windows 98, NT, 2000, XP, and Vista.
- If not, install MS Access to get the proper ODBC driver on your system.
- Upon successful installation, you should see the icon Data Sources (ODBC) in the Administrative Tools window under the Control Panel.
Creating an ODBC Data Source
Creating an ODBC Data Source

- From the Windows **Start** button, choose **Setting, Control Panel** to bring up the **Control Panel** dialog box.
- Double-click **Administrative Tools**, and then double-click **Data Sources (ODBC)** to display the **ODBC Data Source Administrator** dialog box, as shown in the Figure.
Accessing MS-Access with Java

Creating an ODBC Data Source

An ODBC User data source stores information about how to connect to the indicated data provider. A User data source is only visible to you, and can only be used on the current machine.
Creating an ODBC Data Source

- Click Add to bring up the Create New Data Source dialog box, as shown in the Figure.
Creating an ODBC Data Source

- Select Microsoft Access Driver (*.mdb, *.accdb) and press Finish to bring the ODBC Microsoft Access Setup dialog window, as shown in the Figure.
Type Books in the Data Source Name field, and type Deitel Books Database in the Description filed.

Click Select to bring up the Select Database dialog window, as shown in the Figure.
Creating an ODBC Data Source

- Select Books.accdb from the appropriative directory.
- Press OK to close the Select Database dialog window.
- Click OK to close the ODBC Microsoft Access Setup window.
- Click OK to close the ODBC Data Source Administrator window.
The JDBC driver for MS Access is `sun.jdbc.odbc.JdbcOdbcDriver` contained in JDK.

The database URL for Access is `jdbc:odbc:dataSource`.

For example, if the ODBC data source is named **Books**, the URL is `jdbc:odbc:Books`. 
Connecting to a Database
Connecting to and Querying a Database

- This section illustrates:
  - Connecting to a database
  - Querying the database
  - Display the results of the query in JTable

- The following discussion presents the key JDBC aspects of the program.
A typical Java program takes the steps outlined below to access the database:

1. **Loading drivers**
   - An appropriate driver must be loaded using the statement shown below before connecting to a database.
     
     ```java
     Class.forName("JDBCDriverClass");
     ```
   - A driver is a concrete class.
   - For **MS-Access** we will use:
     
     ```java
     Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
     ```
Accessing MS-Access with Java

Accessing a database

2. Establishing connections
   - To connect to a database, use the static method `getConnection(databaseURL)` in the `DriverManager` class, as follows:
     ```java
     Connection connection = DriverManager.getConnection(databaseURL);
     ```
   - The URLs for the Access database:
     ```java
     jdbc:odbc:dataSource
     ```
   - Suppose a data source named `Books` has been created for an Access database. The following statement creates a `Connection` object:
     ```java
     Connection connection = DriverManager.getConnection(jdbc:odbc:Books);
     ```
3. Creating statements

- If a `Connection` object can be envisioned as a cable linking your program to a database, an object of `Statement` or its subclass can be viewed as a cart that delivers SQL statements for execution by the database and brings the result back to the program.

- Once a `Connection` object is created, you can create statements for executing SQL statements as follows:
  
  ```java
  Statement statement = connection.createStatement();
  ```
4. Executing statements

- An SQL update statement can be executed using `executeUpdate(String sql)`, Example:
  ```java
  statement.executeUpdate("INSERT INTO authors ( firstName, lastName ) VALUES ( 'Sue', 'Smith' )");
  ```

- An SQL query statement can be executed using `executeQuery(String sql)`. The result of the query is returned in `ResultSet`, Example:
  ```java
  ResultSet resultSet = statement.executeQuery("SELECT authorID, firstName, lastName FROM authors");
  ```
5. **Processing ResultSet**
   - The ResultSet maintains a table whose current row can be retrieved.
   - The initial row position is `null`.
   - You can use the `next` method to move to the next row and the various get methods to retrieve values from a current row.
   - For example, the code given below displays all the results from the preceding SQL query:

```java
while (resultSet.next())
    System.out.println(resultSet.getString(1) + " " + resultSet.getString(2) + " " + resultSet.getString(3));
```
5. Processing ResultSet (cont.)

- Alternatively, you can use `getString("firstName"), getString("mi"), and getString("lastName")` to retrieve the same three column values.
- The first execution of the `next()` method sets the current row to the first row in the result set, and subsequent invocations of the `next()` method set the current row to the second row, third row, and so on, to the last row.
Querying a Database
DisplayAuthors.java performs a simple query on the books database that retrieves the entire authors table and displays the data.

This program retrieves the entire authors table and displays the data in the standard output stream.

The Program:
- DisplayAuthors.java
Querying a Database

- **Lines 3**
  - import the JDBC classes from package `java.sql` used in this program.

- **Line 12**
  - uses `static` method `forName` of class `Class` to load the class for the database driver.
  - This line throws a checked exception of type `java.lang.ClassNotFoundException` if the class loader cannot locate the driver class.
Querying a Database

- Lines 15
  - creates a `Connection` object (package `java.sql`) referenced by `connection`.
  - An object that implements interface `Connection` manages the connection between the Java program and the database.
  - `Connection` objects enable programs to create SQL statements that access databases.
  - The program initializes `Connection` with the result of a call to static method `getConnection` of class `DriverManager` (package `java.sql`), which attempts to connect to the database specified by its URL.
Querying a Database

- Lines 15 (cont.)
  - The URL locates the database (possibly on a network or in the local file system of the computer).
  - If the `DriverManager` cannot connect to the database, method `getConnection` throws a `SQLException` (package `java.sql`).

- Line 18
  - invokes `Connection` method `createStatement` to obtain an object that implements interface `Statement` (package `java.sql`).
  - The program uses the `Statement` object to submit SQL to the database.
Querying a Database

- Lines 21-22
  - use the `Statement` object's `executeQuery` method to submit a query that selects all the author information from table authors.
  - This method returns an object that implements interface `ResultSet` and contains the result of the query.
  - The `ResultSet` methods enable the program to manipulate the query result.
Querying a Database

- **Lines 30-35**
  - display the data in each ResultSet row.
  - Before processing the ResultSet, the program positions the ResultSet cursor to the first row in the ResultSet with method next (line 45).
  - The cursor points to the current row.
  - Method next returns boolean value true if it is able to position to the next row; otherwise the method returns false (end of table).
  - Initially, a ResultSet cursor is positioned before the first row. Attempting to access a ResultSet's contents before positioning the ResultSet cursor to the first row with method next causes a SQLException.
Querying a Database

- Lines 30-35 (cont.)
  - Specifying column number 0 when obtaining values from a ResultSet causes a SQLException.
Retrieving Metadata
Retrieving Metadata

- JDBC provides the `DatabaseMetaData` interface for obtaining database-wide information and the `ResultSetMetaData` interface for obtaining information on the specific `ResultSet`, such as column count and column names.

- The program:
  - `DisplayAuthors2.java`
Retrieving Metadata

- Line 25
  - obtains the metadata for the ResultSet as a ResultSetMetaData (package java.sql) object.
  - The metadata describes the ResultSet's contents.
  - Programs can use metadata programmatically to obtain information about the ResultSet's column names and types.

- Line 26
  - uses ResultSetMetaData method getColumnCount to retrieve the number of columns in the ResultSet.

- Lines 29-31
  - display the column names.
Updating a Database
Updating a Database

- The program:
  - UpdatingDatabase.java
References
Accessing MS-Access with Java

References


The End