Java Server Pages (JSP) and Database Connection

Dr. Awais Majeed

Outline

Database connection methods

- Java Database Connectivity (JDBC) API is provided by Java to access different types of database engines.
- A JDBC API needs a database driver to interact with a particular database system.
- A driver is an implementation of the JDBC interface that is suited to a specific database.
- A JDBC driver makes it possible to do three things:
    - Establish a connection with a data source
    - Run queries and other SQl statements on the data source
    - Retrieve result sets

JDBC Driver Types

JDBC driver types are classified into 4 categories:

- Type 1: JDBC-ODBC Bridge driver: provides JDBC access through drivers that conform to the Open Database Connectivity (ODBC) standard. It is useful when a pure Java based driver is not available.
- Type2: Native-API Partly Java Driver: converts JDBC calls into calls on native methods in the database API for Oracle, Sybase, etc. It depends on some operating system specific binary code be loaded on each client machine, thus portability is an issue.
- Type3: JDBC-Net Protocol Pure Java Driver: translates JDBC calls into a DBMS-independant net protocol, which is then translated to a DBMS protocol by a custom middleware server.
- Type4: Native-protocol pure Java driver: converts JDBC calls directly into the network protocol used by a DBMS. It improves the performance of the system.

Mapping SQL data to Java

Java Types	SQL Type
String	CHAR, VARCHAR, or LONGVARCHAR
boolean	BIT
byte	TINYINT
short	SMALLINT
int	INTEGER
float	REAL
double	DOUBLE
java.sql.Date	DATE
java.sql.Time	TIME

JSP and MS Access

- Use of MS Access database is suitable during development, prototyping phase.
- Not suitable when Data intensive applications have to be developed or commercial scale development.
- Native Java driver is not available for MS access, therefore JDBC-ODBC bridge driver is used to access the database.

Loading JDBC-ODBC Driver

- import java.sql.*;
- First we need to setup the DriverManager so that it knows what sort of database we want to talk to.

```
try{
Class.forName(”sun.jdbc.odbc.jdbcOdbcDriver”); //loads
the driver
}
catch(Exception e){
out.println(”Error:  ”  + e);
}
```

Connect to database

- Once drivers have been loaded, we have to identify our database.
- MS Access database is a file based system.
- 2 ways to access the database;
  - Use a Data Source Name (DSN)
  - Define the database file and access mechanism by using Java objects
- We will follow the second approach
-

```
String DBUrl="E:\\Development\\eCatalogue.mdb";
String catDB="jdbc:odbc:Driver={Microsoft
Access Driver (*.mdb)};DBQ=" + DBUrl +
";DriverID=22;READONLY=false";
try {
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //loads
the driver

Connection con=DriverManager.getConnection(catDB);}
```

Executing SQL Statement

- To execute a statement, we need Statement object created from the connection created above.

```
Statement st=con.createStatement();
```

- Next thing is to create an SQL statement using a simple string.

```
String sql=”insert into catalogue (ItemName,
ItemDescription, Price, Quantity) Values
('Product1','Electronics',6000,20)”;
st.execute(sql);
```

- Statement object may through execptions so it must be wihin the try-catch block
- Close the statement object by using close() method when a statement has been executed.

ResultSet and retrieving records

- Retrieved records as a result of a select query can be stored in a Java object ResultSet
- ResultSet is a collection of records, with methods available to move through a particular record set.

```
ResultSet rs;
sql="select * from Catalogue";
st.execute(sql);
rs=st.getResultSet();
if(rs!=null)
while (rs.next()){
out.print(”Product ID: ” + rs.getString(1) + ”<br>” );
out.print(”Product Name:  ”  + rs.getString(2) +
”<br>” );
out.print(”Price:  ”  + rs.getString(3) + ”<br>”);
}
st.close();
con.close();
```