Week 10 Research Assignment

2023-04-25-be-mountain

Cody Schliebe

Java Database Connectivity (JDBC)

Java Database Connectivity, or JDBC, is a Java API (application programming interface) developed by Oracle that allows Java applications to connect to, and interact with, SQL databases. While MySQL is a powerful database management system, it exists to manage a database and not much else. Using an API such as JDBC allows your application to use the database for endless purposes. If you have a database full of migratory information for certain animals, it would be difficult to craft a query that would get you the information you need to try to predict where an animal may go next, or how the migratory patterns of several different species may affect each other. But if you build an application in Java, program in the options you want, and connect it to your database using JDBC, then that information (and the inferences that come with it) can be a button click away.

I've been using SQL (Microsoft's SQL Server) for years to pull information from a massive database. It's slow, cumbersome, and it's very easy to make mistakes that can give you wrong and misleading data. When we started building the query strings for our Project application, I got a little giddy. Here was a way that you could hard-code a query, and using variables! [1]

Primary JDBC Classes

The JDBC API has four main classes it uses in its interface with a SQL database. The first, java.sql.DriverManager, loads driver for the JDBC and manages the connection to the SQL database. The second, java.sql.Connection, establishes the connection to the database. The third, java.sql.Statement, manages the building and executing of an SQL statement within the connection. The fourth, java.sql.ResultSet, gathers the results from the statement executed by Statement, and allows access to them. [2]

a) To attach or connect to a database, you'd use the java.sql.Connection class. To do so, you'd use something like the following code:

```
String url = "jdbc:mysql://HOST:PORT/SCHEMA?user=USER&password=PASSWORD&userSSL=false"

Connection con = DriverManager.getConnection(url);
```

b) To create a statement to perform one of the CRUD operations on a database, you would use the java.sql.Statement class, like so:

```
String stmt = "SELECT * FROM projects WHERE project_id = ?";

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

c) To execute the query created in part b and gather the resulting output, you'd use java.sql.ResultSet, like in this example:

```
ResultSet rs = state.executeQuery(stmt);
```

d) To process (and display) the retrieved data from part c, you can loop through the ResultSet, as so:

```
while ( rs.next() ) {
        System.out.println(rs);
}
```

e) Finally, to close out the connection to the database established in part a, you would once again use the java.sql.Connection class, as shown here:

```
con.Close();
```

1 https://www.geeksforgeeks.org/itroduction-to-jdbc/

2 https://www.zaielacademic.net/web_databases_jsp_mysql/jdbc_classes.htm