Java Database Connectivity (JDBC)

PDBM 15.3.4

Dr. Chris Mayfield

Department of Computer Science
James Madison University

Mar 24, 2022

Introduction to JDBC

JDBC = Java Database Connectivity

1. Connect to the database → `java.sql.Connection`
2. Send queries and updates → `java.sql.Statement`
3. Retrieve/process results → `java.sql.ResultSet`

```
import java.sql.*
```

PostgreSQL's JDBC driver

- Download jar file from https://jdbc.postgresql.org/
- See https://jdbc.postgresql.org/documentation/head/

Load the driver

Initialization during application startup:

```
Class.forName("org.postgresql.Driver");
```

`ClassNotFoundException` if driver not available

- Make sure the jar is in your class path

Don't need to do this before every connection!

- Usually part of application startup code

Connect to the DB

```
Connection db = DriverManager.getConnection(
                        url, username, password);
```

URL format is specific to the DBMS

- `jdbc:postgresql:database`
- `jdbc:postgresql://host/database`
- `jdbc:postgresql://host:port/database`

Internally, uses same library as psql and pgAdmin

Execute a statement

```
String sql = "SELECT * FROM mytab WHERE foo = 500";
Statement st = db.createStatement();
ResultSet rs = st.executeQuery(sql);
while (rs.next()) {
    System.out.print("Column 1 returned ");
    System.out.println(rs.getString(1));
}
rs.close();
st.close();
```

`ResultSet` can also do getInt(*i*), getFloat(*i*), ...

- Note that column indexes start at 1!

For non-queries, use `rs.executeUpdate(sql)`

Better yet, a prepared statement

```
int foovalue = 500;
String sql = "SELECT * FROM mytab WHERE foo = ?";
PreparedStatement st = db.prepareStatement(sql);
st.setInt(1, foovalue);
ResultSet rs = st.executeQuery();
while (rs.next()) {
    System.out.print("Column 1 returned ");
    System.out.println(rs.getString(1));
}
rs.close();
st.close();
```

The ‘?’ syntax provides additional type safety

- String arguments are automatically escaped
- Helps prevent SQL injection attacks https://xkcd.com/327/

Details about statements

Use a single `Statement` instance as many times as you want

- However, only one `ResultSet` can exist per `Statement` or `PreparedStatement` at a given time
- If you need to run a query while processing a `ResultSet`, simply create and use another `Statement`

If you are using threads, and several are using the database, you must use a separate Statement for each thread.

When you are done using the `Statement` or `PreparedStatement` you should close it.

Details about result sets

Before reading any values, you must call `next()`

- Returns `true` if there is a result
- More importantly, prepares the row for processing

You should close a `ResultSet` once you have finished

- If you make another query with the RS's `Statement` ...
- ...then the `ResultSet` instance is closed automatically

Now you try it!

- Create a Java application that outputs movie titles