الجامعة اللبنانية
Lebanese University

Graphical Interface and Application(I3305)

Chapter 5: Java JDBC

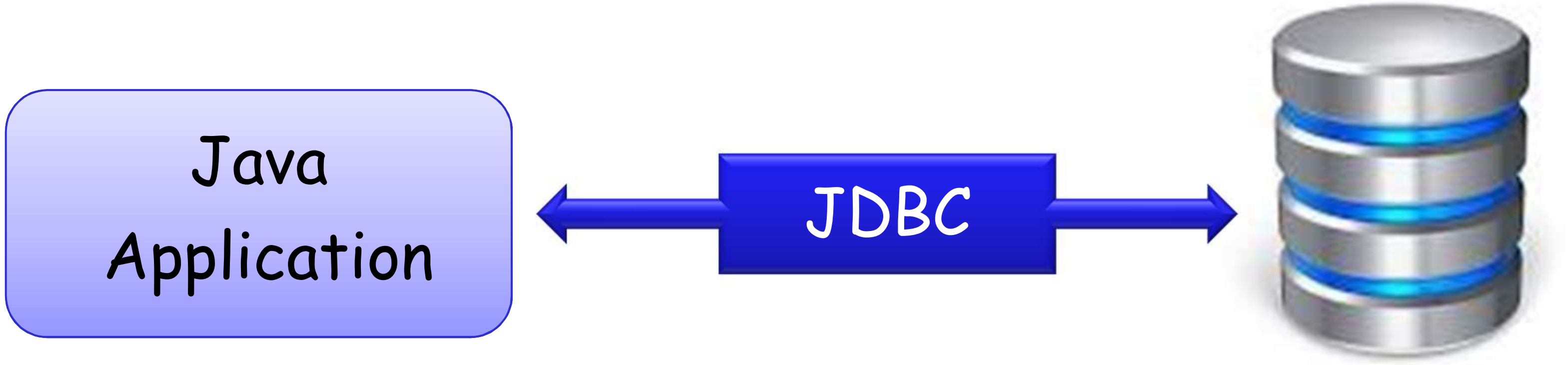
Lebanese University
Faculty of Science 1 - Department of Computer Science
Abed EL Safadi

Outline

- What is JDBC?
- Architecture
- Development Process

What is JDBC?

JDBC stands for Java DataBase Connectivity, which is a standard Java API for database independent connectivity between the Java programming language, and a wide range of databases.



- Local
- Network
- Cloud

What is JDBC?

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

JDBC Architecture

In general, JDBC Architecture consists of **two layers**:

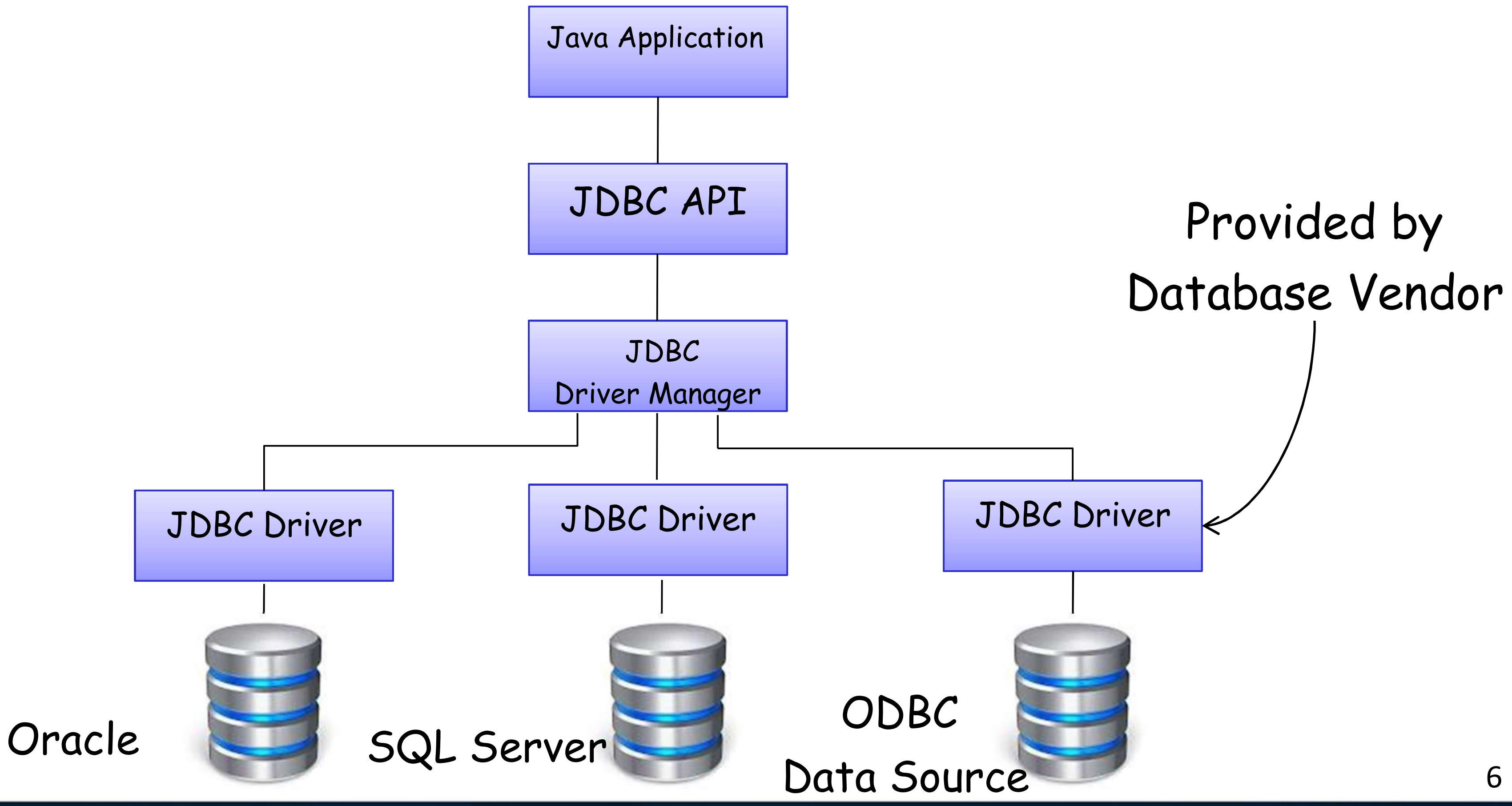
1. **JDBC API:** This provides the application-to-JDBC Manager connection.
2. **JDBC Driver API:** This supports the JDBC Manager-to-Driver Connection.

The JDBC API uses a driver manager and database-specific drivers to provide transparent connectivity to heterogeneous databases.

The JDBC driver manager ensures that the correct driver is used to access each data source.

JDBC Architecture

Following is the architectural diagram, which shows the location of the driver manager with respect to the JDBC drivers and the Java application:



Common JDBC Components

The JDBC API provides the following interfaces and classes:

DriverManager: This class manages a list of database drivers.

Driver: This interface handles the communications with the database server.

Connection: This interface with all methods for contacting a database.

Statement: You use objects created from this interface to submit the SQL statements to the database.

ResultSet: These objects hold data retrieved from a database after you execute an SQL query using Statement objects

SQLException: This class handles any errors that occur in a database application.

JDBC API

JDBC API is defined in the following packages

- Java.sql and javax.sql

Key classes

-Java.sql.DriverManager
-Java.sql.Connection
-Java.sql.Statement
-Java.sql.ResultSet

Development Process

1. Get a connection to database
2. Create a Statement Object
3. Execute SQL query
4. Process Result Set

Development Process

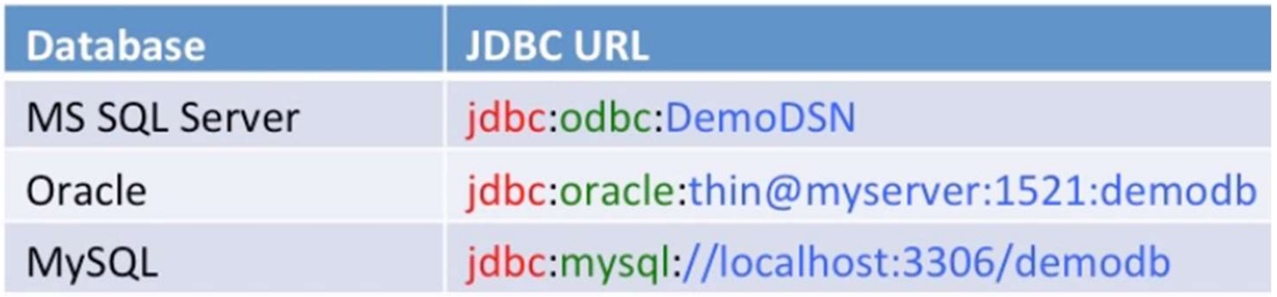
Step 1 Get a connection to database

- In order to connect to database
  - Need a connection string in form of JDBC URL

- Basic syntax

  -jdbc:<driver protocol>:<driver connection details>

- Examples

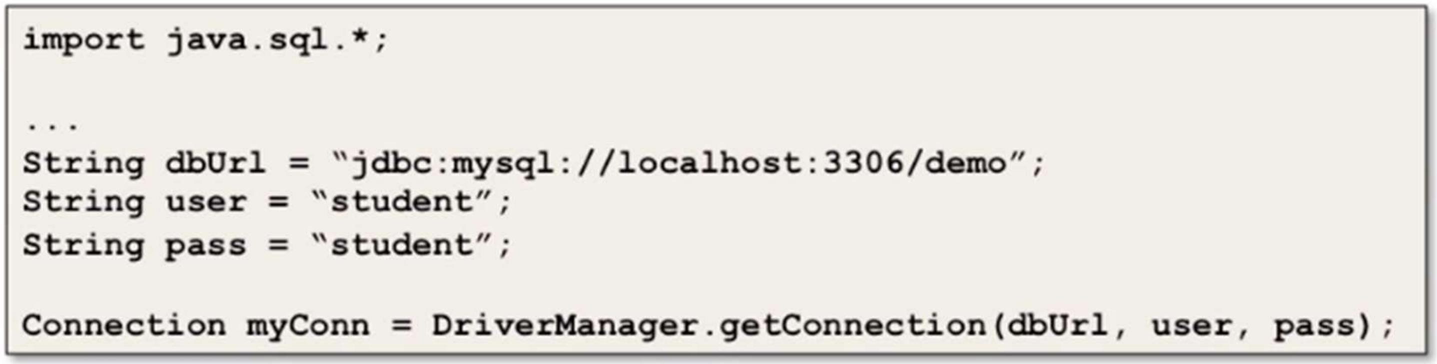


Database	JDBC URL
MS SQL Server	jdbc:odbc:DemoDSN
Oracle	jdbc:oracle:thin@myserver:1521:demodb
MySQL	jdbc:mysql://localhost:3306/demodb

Development Process

Step 1 Get a connection to database-Continued

- Code snippet for connecting to Mysql



```
import java.sql.*;

...
String dbUrl = "jdbc:mysql://localhost:3306/demo";
String user = "student";
String pass = "student";

Connection myConn = DriverManager.getConnection(dbUrl, user, pass);
```

Failure to connect will throw an exception:

-java.sql.SQLException: bad url or credentials

-java.lang.ClassNotFoundException: JDBC driver not in classpath

Development Process

Step 2 Create a Statement object

- The Statement object is based on connection
  -it will be used later to execute SQL query

```
import java.sql.*;

...
String dbUrl = "jdbc:mysql://localhost:3306/demo";
String user = "student";
String pass = "student";

Connection myConn = DriverManager.getConnection(dbUrl, user, pass);

Statement myStmt = myConn.createStatement();
```

Development Process

Step 3 Execute SQL Query

- Pass in your SQL Query

```
import java.sql.*;

...
String dbUrl = "jdbc:mysql://localhost:3306/demo";
String user = "student";
String pass = "student";

Connection myConn = DriverManager.getConnection(dbUrl, user, pass);

Statement myStmt = myConn.createStatement();

ResultSet myRs = myStmt.executeQuery("select * from employees");
```

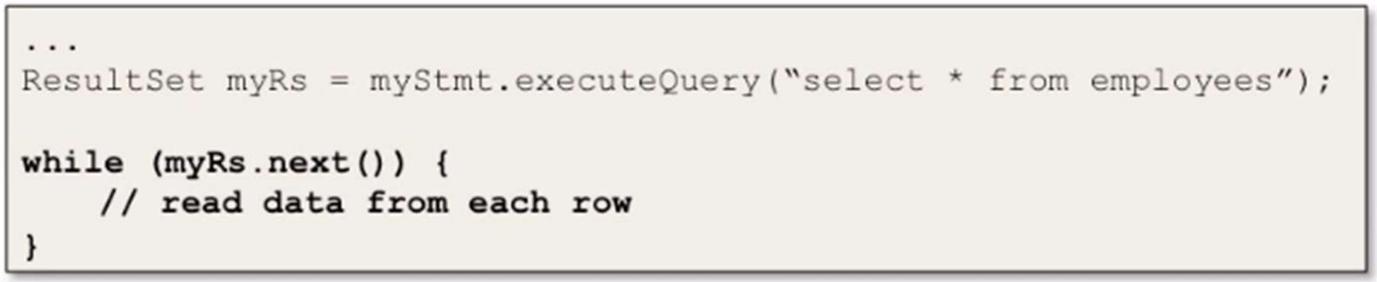
Development Process

Step4 Process the result Set

- Result Set is initially placed before first row
- Method: ResultSet.next()

  -moves forward one row

  -Returns true if there are more rows to process
- Looping throught a result set



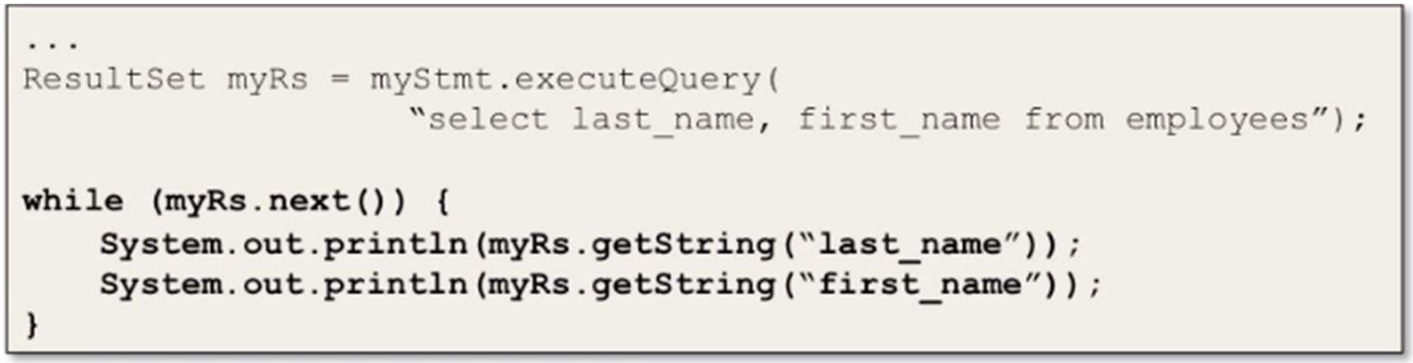
```
...
ResultSet myRs = myStmt.executeQuery("select * from employees");

while (myRs.next()) {
    // read data from each row
}
```

Development Process

Step4 Process the result Set (Continued)

- Collection of methods for reading data

  -getXXX(columnName)



```
...
ResultSet myRs = myStmt.executeQuery(
                    "select last_name, first_name from employees");

while (myRs.next()) {
    System.out.println(myRs.getString("last_name"));
    System.out.println(myRs.getString("first_name"));
}
```