A Brief Introduction to JDBC

by Faisal Khan.

Overview

This article provides a brief introduction to JDBC. Explains what is JDBC and how it can be used to access RDBMS. Provides a brief overview of JDBC architecture.

What is JDBC?

JDBC stands for "Java DataBase Connectivity". It is an API (Application Programming Interface) which consists of a set of Java classes, interfaces and exceptions and a specification to which both JDBC driver vendors and JDBC developers (like you) adhere when developing applications.

JDBC is a very popular data access standard. RDBMS (Relational Database Management Systems) or third-party vendors develop drivers which adhere to the JDBC specification. Other developers use these drivers to develop applications which access those databases e.g. you'll use ConnectorJ JDBC driver to access MySQL database. Since the drivers adhered to JDBC specification, the JDBC application developers can replace one driver for their application with another better one without having to rewrite their application. If they had used some proprietary API provided by some RDBMS vendor, they will not have been able to change the driver and/or database without having to rewrite the complete application.

Who develops JDBC Specification?

SUN prepares and maintains the JDBC specification. Since JDBC is just a specification (suggestions for writing and using JDBC drivers), third-party vendors develop JDBC drivers adhering to this specification. JDBC developers then use these drivers to access data sources.

Why use JDBC?

JDBC is there only to help you (a Java developer) develop data access applications without having to learn and use proprietary APIs provided by different RDBMS vendors. You just have to learn JDBC and then you can be

sure that you'll be able to develop data access applications which can access different RDBMS using different JDBC drivers.

JDBC Architecture

We'll divide it into 2 parts:

- JDBC API (java.sql & javax.sql packages)
- JDBC Driver Types

JDBC API

The JDBC API is available in the java.sql and javax.sql packages. Following are important JDBC classes, interfaces and exceptions in the java.sql package:

- **DriverManager** - Loads JDBC drivers in memory. Can also be used to open connections to a data source.
- **Connection** - Represents a connection with a data source. Is also used for creating Statement, PreparedStatement and CallableStatement objects.
- **Statement** - Represents a **static SQL statement**. Can be used to retrieve ResultSet object/s.
- **PreparedStatement** - Higher performance alternative to Statement object, represents a **precompiled SQL statement.**
- **CallableStatement** - Represents a stored procedure. Can be used to **execute stored procedures in a RDBMS** which supports them.
- **ResultSet** - Represents a database result set generated by using a SELECT SQL statement.
- **SQLException** - An exception class which encapsulates database base access errors.

javax.sql is part of J2SE 1.4 and J2EE 1.3. It adds following features to JDBC in addition to the ones provided by java.sql package:

- **DataSource** - Abstracts a data source. This object can be used in place of DriverManager to efficiently obtain data source connections (possibly using hidden connection pooling).
- Provides built-in connection pooling.

- **XADataSource**, **XAConnection** - Allows/supports distributed transactions.
- **RowSet** - It extends ResultSet interface to add support for disconnected result sets.

JDBC Driver Types

There are 4 types of JDBC drivers. Commonest and most efficient of which are type 4 drivers. Here is the description of each of them:

- **JDBC Type 1 Driver** - They are JDBC-ODBC Bridge drivers. They delegate the work of data access to ODBC API. They are the slowest of all. SUN provides a JDBC/ODBC driver implementation.
- **JDBC Type 2 Driver** - They mainly use native API for data access and provide Java wrapper classes to be able to be invoked using JDBC drivers.
- **JDBC Type 3 Driver** - They are written in 100% Java and use vendor independent Net-protocol to access a vendor independent remote listener. This listener in turn maps the vendor independent calls to vender dependent ones. This extra step adds complexity and decreases the data access efficiency.
- **JDBC Type 4 Driver** - They are also written in 100% Java and are the most efficient among all driver types.

SUN encourages to develop and use type 4 drivers in your applications.

Summary

In this article we learned what is JDBC, why is JDBC useful and why to develop applications using JDBC.

In next set of articles we will learn how to setup a database and a JDBC driver.