INSTITUTO MAUÁ DE TECNOLOGIA

MAUÁ

Linguagens I

Java + Banco de Dados

Profº. Tiago Sanches da Silva

Persistência através de sockets

É possível conectar-se com qualquer base de dados através da abertura de um socket TCP com o servidor que o hospeda, por exemplo um Oracle ou MySQL e nos comunicarmos com ele através de seu protocolo proprietário.

Porém conhecer o protocolo proprietário complexo em profundidade é difícil, e trabalhar com ele é muito trabalhoso.

Conexão em Java

Conectar-se a um banco de dados com Java é feito de maneira elegante.

Para **evitar** que cada banco tenha a sua própria API e conjunto de classes e métodos, temos um único conjunto de **interfaces** muito bem definidas que devem ser implementadas.

Esse conjunto de interfaces fica dentro do pacote **java.sql** e nos referiremos a ela como **JDBC**.

Java DataBase Connectivity

Conexão em Java

Interfaces java.sql:

https://docs.oracle.com/javase/8/docs/api/java/sql/package-summary.html

JDBC API:

https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/

O que é o JDBC?

Pode-se dizer que é uma **API** (Interface de Programação de Aplicativos) que **reúne conjuntos de classes e interfaces** escritas na linguagem Java na qual possibilita se conectar através de um **driver específico** do banco de dados desejado.

Com esse driver pode-se executar instruções SQL de qualquer tipo de banco de dados relacional.

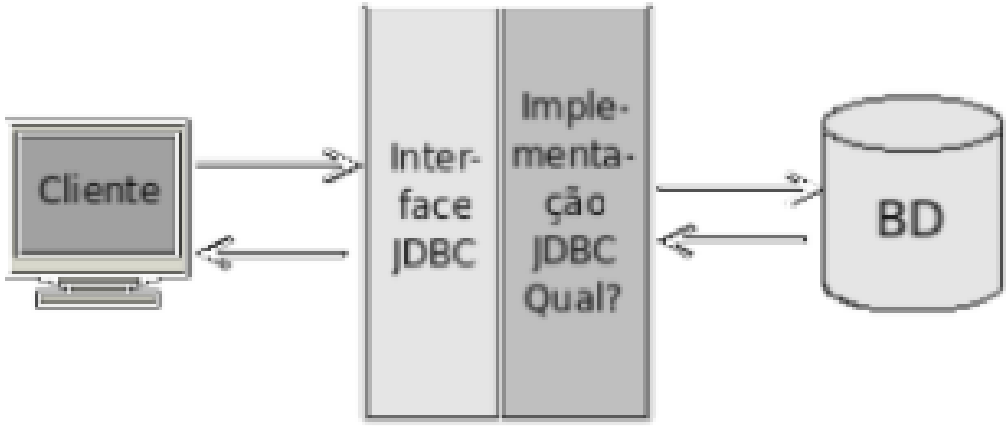
Para fazer a comunicação entre a **aplicação** e o Banco de Dados é necessário possuir um **driver para a conexão desejada**. Geralmente, as empresas de Banco de Dados oferecem o driver de conexão que seguem a especificação **JDBC**.

MySQL: https://dev.mysql.com/downloads/connector/j/

Driver?

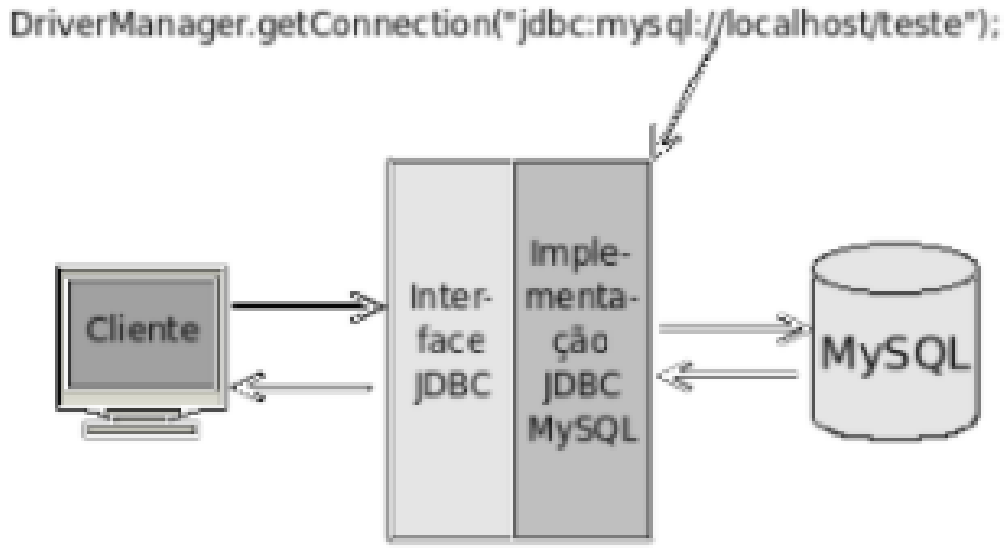
Caso queiramos trabalhar com o **MySQL,** precisamos de classes concretas que implementem essas interfaces do pacote **java.sql**.

Esse conjunto de classes concretas é quem fará a ponte entre o código cliente que usa a **API JDBC** e o **banco de dados**. São essas classes que sabem se **comunicar através do protocolo proprietário** do banco de dados. Esse conjunto de classes recebe o nome de **driver**.



Driver?

Todos os principais bancos de dados do mercado possuem **drivers JDBC** para que você possa utilizá-los com Java.



Atenção

Importe do java.sql

Existe um ponto de atenção na importação das classes ou interfaces relacionadas ao pacote a ser usado no momento do desenvolvimento.

A correta a importação do pacote referente à classe Connection pertencente ao pacote **java.sql**.

Esse é um fator a ser observado com cautela, pois isso é considerado um dos erros mais comuns justamente pelo fato do desenvolvedor pensar muitas vezes em usar o pacote **com.mysql.jdbc** sendo que está utilizando o **driver JDBC** do banco **MySQL**.

Pacote java.sql

Pacote java.sql

Esse pacote oferece a biblioteca Java o acesso e processamento de dados em uma banco de dados. As classes e interfaces mais importantes são:

Classe	Interface
DriverManager	Driver
	Connection
	Statement
	ResultSet
	PreparedStatement

DriverManager

Para abrir uma conexão com um banco de dados, precisamos utilizar sempre um driver. A classe **DriverManager** é a responsável por se comunicar com todos os drivers que você deixou disponível.

Para isso, invocamos o método estático **getConnection** com uma **String** que indica a qual banco desejamos nos conectar.

Essa **String** - chamada de **String de conexão JDBC** - que utilizaremos para acessar o **MySQL** tem sempre a seguinte forma:

jdbc:mysql://ip/nome_do_database

DriverManager - Exemplo

jdbc:mysql://ip/nome_do_database

```
DriverManager.getConnection("jdbc:mysql://localhost/alunosteste", "root", "XXXXXXXX");
```

Ao tentar executar essa linha sem ter carregado o pacote com o driver correto receberemos uma exception.

```
java.sql.SQLException: No suitable driver found for
```

A conexão não pôde ser aberta por que?

Carregar o pacote do driver para o projeto

O que precisamos fazer é adicionar o driver do **MySQL** ao **classpath**, o arquivo **.jar** contendo a implementação **JDBC** do **MySQL** (**mysql connector**) precisa ser colocado em um lugar visível pelo seu projeto ou adicionado à variável de ambiente **CLASSPATH**.

Mas eu vi em algum tutorial...

E o Class.forName?

Até a versão 3 do JDBC, antes de chamar o `DriverManager.getConnection()` era necessário registrar o driver JDBC que iria ser utilizado através do método `Class.forName("com.mysql.jdbc.Driver")`, no caso do MySQL, que carregava essa classe, e essa se comunicava com o `DriverManager`.

A partir do JDBC 4, que está presente no Java 6, esse passo não é mais necessário. Mas lembre-se: caso você utilize JDBC em um projeto com Java 5 ou anterior, será preciso fazer o registro do Driver JDBC, carregando a sua classe, que vai se registrar no `DriverManager`.

Interface Connection

Representa uma conexão ao banco de dados. Nessa interface são apresentados os métodos mais utilizados.

Caso o **DriverManager** consiga realizar a conexão com o banco de dados ele retorna uma instancia de um objeto **Connection**.

Com ele você conseguirá executar **queries**.

Prática + Discussão

Ok. Show me the code!

Perguntas?

Referências

- DevMedia (Thiago Vinícius)
- Oracle
- Caelum