**O que iremos apresentar?**

Apresentaremos algumas classes, interfaces e métodos do pacote java.sql e que são responsáveis em efetuar o CRUD em Java.

* Classe: java.sql.DriverManager
* Interface: java.sql.Connection
* Interface: java.sql.PreparedStatement
* Interface: java.sql.ResultSet // bônus

**O que é o pacote java.sql?**

Fornece os recursos necessários para acessar diferentes tipos de banco de dados. Trabalha conjuntamente do driver do banco de dados. Possibilita interface necessária para ler e gravar quaisquer fontes de dados em formato tabular. Atualmente está na versão 4.1 (documentação Oracle).

**Qual o funcionamento geral desse pacote?**

Possibilita efetuar uma conexão ao driver de banco de dados através da **classe java.sql.DriverManager**. E enviar comandos SQL (em inglês SQL statements) através das **interfaces** Statement, **PreparedStatement**, CallableStatement, **Connection** e Savepoint. Por fim, é capaz de receber os resultados da consulta SQL através da **interface java.sql.ResultSet**. Além de propiciar um conjunto de interfaces e classes para o mapeamento e tratamento dos dados, inclusive para data e tempo.

**O escopo geral do funcionamento?**

É necessário efetuar a abertura de uma conexão com o driver através de um objeto Connection instanciado pela classe DriverManager. A conexão/connection é utilizada para criar comandos SQL através de uma String e a interface PreparedStatement que é capaz de executar a query. Quando necessário armazenar o resultado é instanciado o objeto através da interface ResultSet. Por fim é necessário efetuar o commit (quando as alterações são definitivas) e fechar a conexão com o banco de dados.

**Class DriverManager**

Fornece os recursos para carregar e conectar com diferentes drivers para diferentes bancos de dados. Atualmente não é necessário o sistema carregar os drivers, pois a classe consegue carregar manualmente. Antigamente era necessário carregar através do método Class.forName(). Para isso é necessário que o driver esteja nas bibliotecas e referenciado corretamente (ver na documentação). Possui principalmente os métodos .getConnection() para efetuar a conexão com o driver e métodos .getDrivers() para apresenta todos os drivers carregados pelo JDBC.

**static Connection DriverManager.getConnection(String url, String user, String password);**

Pode ser utilizado encaminhando diferentes parâmetros, mas em última instancia é necessário encaminhar 3 Strings contendo a URL do banco de dados, o login e a senha. E efetua a conexão com banco de dados e retorna um objeto do tipo Connection.

**Interface Connection**

Uma sessão com uma base de dados. Comandos SQL são gerados, executados e retornam através dessa interface. Possui algumas constantes responsáveis por sinalizar os modos de transação de dados com o banco de dados. Por padrão a interface utiliza o autoCommit, porém pode ser desativado e controlado pelo desenvolvedor (através dos métodos .getAutoCommit(), .commit() e .rollback()). Possui diversos métodos para consulta SQL.

**void connection.close()**

É o método responsável por fechar a conexão com o banco de dados.

**PreparedStatement connection.prepareStatement(String sql)**

É o método responsável para criar um objeto PreparedStatement e que irá encaminhar o comando SQL. Pode ser utilizado de diferentes maneiras, apresentaremos a mais simples que é a que encaminhamos apenas a String com o comando. No entanto as variações permitem retornar diferentes tipos de dados.

**Outros métodos importantes:**

.commit();

.getAutoCommit();

.rollback();

.setSavepoint();

.isClosed();

.isValid();

.setSchema();

**Interface PreparedStatement**

É um objeto que representa um pre comando SQL em uma String, esse comando pode ter valores atribuídos com métodos setters específicos dessa interface. Funciona basicamente assim:

String sql = “SELECT \* FROM pessoa WHERE ID = ?”;

Com métodos setters é possível substituir o ‘?’ por qualquer valor em qualquer tipo necessário.

É esse objeto responsável por executar a query e gerar o objeto contendo os resultados da consulta (ResultSet).

Os principais métodos são .execute(), .executeQuery(), e os métodos setters .set[tipo](), aonde [tipo] pode ser um Array, byte, int, long, String, char, ou objeto.

**boolean PreparedStatement.execute();**

Executa o comando SQL e retorna um boolean que identifica o sucesso da operação.

**ResultSet PreparedStatement.executeQuery();**

Executa o comando SQL e retorna o resultado em um objeto do tipo ResultSet.

**Void PreparedStatement.setString(Int num, String “texto);**

O número indica a posição do ‘?’ no comando SQL. O texto identifica o valor a ser atribuído a esse ‘?’. A prática mais comum seria utilizar os métodos getters dos objetos.

**Outros métodos são**

.clearParameters();

.getParametersMetaData();

**Interface ResultSet**

O objeto ResultSet contém o resultado da consulta SQL, uma tabela. E possui todas as funcionalidades para navegar na tabela, alterar e coletar dados. Se por um lado a interface PreparedStatement funciona como set de atributos a interface ResultSet funciona como get de atributos.

Os métodos de navegação são

.next();

.previous();

.absolute(int row);

.afterLast();

.beforeFirst();

.cancelRowUpdates();

.clearWarnings();

.close();

.deleteRow();

.findColum(String columLabel);

.first();

Os métodos getters funcionam como de praxe, .get[tipo](string nomelabel / int columnIndex);

Exemplo:

ResultSet.getString(get

É necessário compreender:

|  |  |  |
| --- | --- | --- |
| A | B | C |
| 1 | 2 | 3 |

É importante compreender que os métodos de navegação permitem acessar cada unidade das tabelas, é comumente utilizado os métodos .next() e .previous() para ir para frente e para trás. No entanto outros métodos existem. A navegação sempre ocorre uma coluna após a outra uma linha após a outra.

Métodos

Assim funciona com qualquer banco de dados ou tabela