UNIVERSIDADE FEDERAL DO CEARÁ
Campus Russas

Disciplina: Fundamentos de Banco de Dados

ACESSO A BANCO DE DADOS COM JDBC

Professora: Marília S. Mendes
E-mail: marilia.mendes@ufc.br

Introdução

- Aplicação + Banco de Dados:
  - Funcionalidade primordial em qualquer sistema.

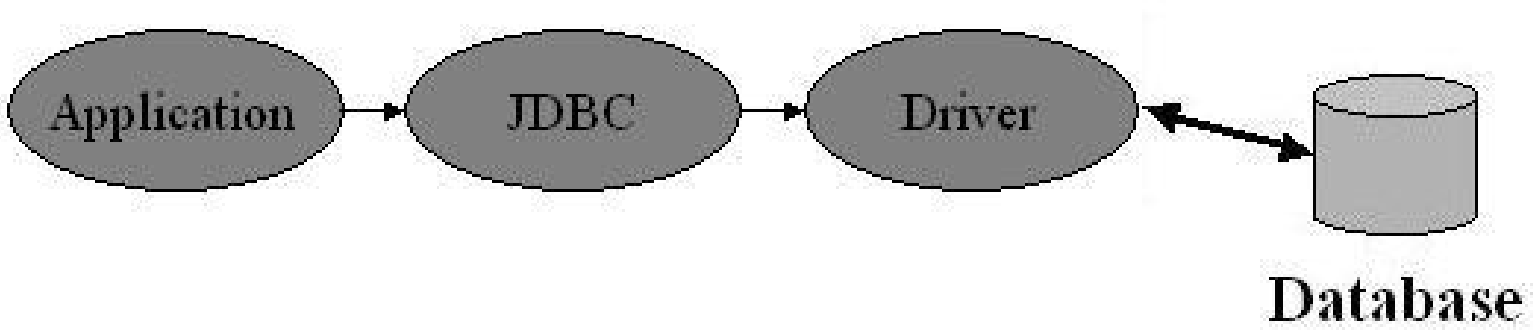
- A Linguagem Java possui uma *Application Programming Interface* (API) que possibilita o acesso a Banco de Dados;
  - *Java DataBase Connectivity* (JDBC)

JDBC

- O que é a JDBC?
  - Consiste em uma biblioteca;
  - Implementada em Java;
  - Disponibiliza classes e interfaces para acessar qualquer banco de dados;

- Importante: Para cada banco de dados existe uma implementação JDBC.
  - *Drivers*.

JDBC



1) A aplicação chama a API JDBC.

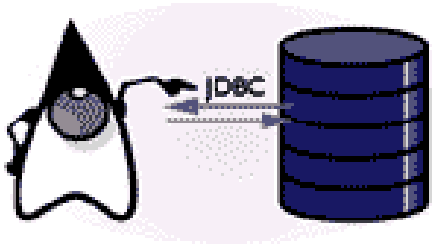
2)A API carrega o *driver* que "entende" o SGBD.

3)A aplicação pode se conectar e enviar requisições ao SGBD.

- Pacote principal: java.sql

O que JDBC faz ?

- De maneira simplificada JDBC faz seis coisas:
  - 1.Estabelece conexão com o banco de dados.
  - 2.Executa consultas.
  - 3.Recebe o conjunto de Resultados das Consultas.
  - 4.Executa *stored procedures*.
  - 5.Obtém informações sobre o banco de dados, tabelas, visões e *stored procedures*.
  - 6.Executa transações.



Pacote java.sql

5 passos básicos:

1. Registrar o driver na aplicação;
2. Conectar no SGBD;
3. Executar comandos SQL;
4. Processar o resultado recebido;
5. Fechar a conexão.

- Principais classes do pacote java.sql:
  - **DriverManager**: responsável por criar uma conexão com o banco de dados;
  - **Connection**: classe responsável por manter uma conexão aberta com o banco de dados;
  - **Statement**: gerencia e executa instruções SQL;
  - **ResultSet**: responsável por receber e apresentar os dados obtidos do banco de dados.

Referência: http://java.sun.com/javase/6/docs/api/java/sql/package-summary.html
Pacote java.sql

JDBC – Passos Básicos

- Registro do *driver*:
  - O *driver* é registrado automaticamente quando a classe é carregada na aplicação.
  - Carrega o *driver* em tempo de execução.
    - Class.forName("org.postgresql.Driver"); // POSTGRESQL
    - Class.forName("com.mysql.jdbc.Driver"); // MYSQL
    - Class.forName("oracle.jdbc.OracleDriver"); //ORACLE

JDBC – Passos Básicos

- Conexão com o SGBD:
  - Após o registro do *driver*, é necessário fornecer informações ao *DriverManager* para realizar a conexão com o Banco de Dados:
    - Connection con = DriverManager.getConnection(url, login, senha)

Connection

- Representa a conexão com o banco de dados;

- Métodos desta classe frequentemente utilizados (SUN, 2007):
  - *commit()*, executa todas as alterações feitas com o banco de dados pela atual transação.
  - *rollback()*, desfaz qualquer alteração feita com o banco de dados pela atual transação.
  - *close()*, libera o recurso que estava sendo utilizado pelo objeto

JDBC – Passos Básicos

- Conexão com o SGBD:
  - url: URL de conexão JDBC
    - jdbc:postgresql://localhost: 5432/cursodb
    - jdbc:mysql://localhost:3306/cursodb
    - jdbc:oracle:thin:@localhost:1521:XE
  - Login: usuário com direitos de acesso ao banco de dados;
  - senha: senha para autenticação.
  - Sintaxe geral de urls: "jdbc:<subprotocolo>://<servidor>:<porta>/<banco_de_dados>"

DriverManager.getConnection(**url**, **login**, **senha**);

JDBC – Passos Básicos

- Conexão com o SGBD:
  - // Carregando o driver oracle em tempo de execução da aplicação
  - Class.forName("oracle.jdbc.OracleDriver");

  - // Estabelecendo a conexão com o Banco de Dados
  - Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE", "system", "senha");

JDBC – Passos Básicos

- Execução de sentenças SQL
  - Para a execução de sentenças é necessário criar um *Statement* e obter o resultado através de um *ResultSet*;
  - Statement stmt = con.createStatement();
  - ResultSet rs = stmt.executeQuery("select * from empregados e");

- Na maioria dos casos, o resultado é armazenado num ResultSet e pode ser percorrido com métodos definidos nesta classe.

Statement

- Fornece métodos para executar uma instrução SQL;
- Não aceita a passagem de parâmetros;

- principais métodos da classe *Statement* são (SUN, 2007):
  - *executeUpdate()*: executa instruções SQL do tipo: *INSERT*, *UPDATE* e *DELETE*;
  - *executeQuery()*: executa instruções SQL de busca de dados, do tipo: *SELECT*;
  - *close()*: libera o recurso que estava sendo utilizado pelo objeto.

Statement

- // Instanciando o objeto statement (stmt)
  - Statement stmt = conn.createStatement();
- // Executando uma instrução SQL.
  - stmt.executeUpdate("INSERT INTO ALUNO VALUES (1, 'Pedro da Silva')");

ResultSet

- Permite o recebimento e gerenciamento do conjunto de dados resultante de uma consulta SQL;

- Métodos freqüentemente utilizados (SUN, 2007):
  - *next()*: move o cursor para a próxima linha de dados, já que o conjunto de dados retornados pela consulta SQL é armazenado como em uma tabela.
  - *close()*: libera o recurso que estava sendo utilizado pelo objeto.
  - *getString(String columnName)*: recupera o valor da coluna informada como parâmetro, da linha atual do conjunto de dados recebidos pelo objeto ResultSet.

ResultSet

- //Recebendo o conjunto de dados da consulta SQL
  - ResultSet rs = stmt.executeQuery("SELECT id, nome FROM ALUNO");
- // Se houver resultados, posiciona-se o cursor na próxima linha de dados
  - **while** (rs.next()) {
  - // Recuperando os dados retornados pela consulta SQL
  - **int** id = rs.getInt("id");
  - String nome = rs.getString("nome");
  - }

Exemplos

EXEMPLO-JDBC

- **exemplo-jdbc**
  - Exemplo de aplicação JDBC simples
- **Download**
  - Através do botão "Download ZIP" ou através de um comando git clone https://github.com/regispires/exemplo-dao-jdbc.git O 'git clone' pode ser realizado diretamente através do Eclipse:
  - Mudar para a perspectiva para "Git Repository Exploring".
  - Clicar no botão Clone Git repository.
  - Colar a URI do repositório em Location -> URI.

No link do título ou no SIGAA

EXEMPLO-JDBC

- **Importar o projeto para o Eclipse**
  - Faça: File -> Import -> General -> Existing Projects into Workspace
  - Clique em 'Next >'
  - Selecione o diretório raiz (Root directory) do projeto baixado
  - Clique em "Finish"
- **Criar o esquema relacional no PostgreSQL**
  - create database trabalho1; create table cliente (id int primary key, nome varchar(50), idade int);

EXEMPLO-DAO-JDBC

- **exemplo-dao-jdbc**
  - Exemplo de aplicação JDBC simples usando padrão de projeto DAO.
- **Download**
  - Através do botão "Download ZIP" ou através de um comando "git clone https://github.com/regispires/exemplo-dao-jdbc.git"
    O 'git clone' pode ser realizado diretamente através do Eclipse:
  - Mudar para a perspectiva para "Git Repository Exploring".
  - Clicar no botão Clone Git repository.
  - Colar a URI do repositório em Location -> URI.

No link do título ou no SIGAA

EXEMPLO-DAO-JDBC

- **Importar o projeto para o Eclipse**
  - O Eclipse JEE (Eclipse IDE for Java EE Developers) versão Kepler ou superior possui suporte nativo ao Maven.
  - Faça: File -> Import -> Maven -> Existing Maven Projects
  - Clique em 'Next >'
  - Selecione o diretório raiz (Root directory) do projeto baixado
  - Clique em "Finish"
- **Criar o esquema relacional no PostgreSQL**
  - create database contatos; create table clientes (id serial primary key, cpf varchar(11), nome varchar(50), fone varchar(11), renda decimal(10,2));

Bibliografia Utilizada nesta aula

- ELMASRI, R.; NAVATHE, S. B. Sistemas de banco de dados. 6 ed. Pearson/Addison-Wesley, 2011. ISBN: 9788579360855
- SUN. Disponível em: http://java.sun.com/javase/6/docs/api/java/sql/package-summary.html

- Exemplos de: Prof. Regis
  - https://github.com/regispires/exemplo-jdbc
  - https://github.com/regispires/exemplo-dao-jdbc