UFABC	Universidade Federal do ABC Disciplina: Programação para Web Atividade 2 2Q-2018

Atividade 2: Conectando com o banco de dados

1. Instalação do Banco de Dados (SGBD)

Para os exercícios das aulas, você poderá utilizar um dos bancos de dados (SGBD) abaixo:

- PostGres https://www.postgresql.org/download/
- H2 (Java SQL Database) http://www.h2database.com/

No Respositório do Tidia há um guia para configuração do ambiente de desenvolvimento utilizando o postgres.

2. Criando um banco de dados para a aplicação

Vamos criar um banco de dados para a aplicação do exercício da aula passada (projeto SistemaBancario). Para maiores detalhes, siga as instruções do guia de instalação de acordo com o banco de dados que está utilizando (Tidia → Repositorio).

1. Inicie o programa do banco de dados. Crie um banco de dados chamado **SistemaBancario**.

No Postgres utilize o comando:

```
CREATE DATABASE "SistemaBancario"
  WITH
   OWNER = postgres
   ENCODING = 'UTF8'
   CONNECTION LIMIT = -1;
```

2. Crie uma tabela chamada **ContaCorrente** com as seguintes colunas: **id**, **numero**, **descricao**, **ativa** e **variacao**.

No Postgres utilize:

```
CREATE TABLE ContaCorrente
(
   id serial NOT NULL,
   numero character varying(255) NOT NULL,
   agencia character varying(255) NOT NULL,
   descricao character varying(255) NOT NULL,
   ativa boolean NOT NULL,
   variacao integer NOT NULL,
   PRIMARY KEY (id)
)
```

3. Faça alguns testes utilizando os comandos SQL para inserir, alterar, remover registros no banco de dados. Exemplos:

   Inserindo uma conta corrente:

INSERT INTO ContaCorrente(
numero, agencia, descricao, ativa, variacao)
VALUES ('1222-3', '333', 'Conta Corrente do Banco do Brasil', true, 0);

Cheque a inserção, listando os registros da tabela **ContaCorrente**:
SELECT * FROM ContaCorrente;

De forma semelhante, insira mais alguns registros de contas correntes com diferentes valores e verifique o conteúdo do banco de dados.

Altere o status **ativo** para **falso** do registro cujo id=1:
UPDATE ContaCorrente
SET ativa=false
WHERE id=1;

Altere outras colunas da tabela, por exemplo, **numero**, **agencia**, **varicao** ou **descricao** e liste os registros para conferir as alterações.

4. Teste outros comandos (por exemplo, remoção e outras formas de seleção) e verifique o conteúdo do banco de dados para certificar que o comando foi executado corretamente. Veja nos slides da aula exemplos de consultas SQL para testar.

3. Incluindo o driver JDBC

Para que uma aplicação possa acessar o banco de dados, é preciso que o driver JDBC seja incluído no projeto (para isso basta indicar a localização do driver (arquivo .jar) no *classpath* do projeto, conforme é explicado a seguir). No arquivo **configurar-ambiente.pdf** (no Repositório do Tidia da teoria) há instruções para baixar o driver.
Para adicionar o driver ao projeto, clique direito sobre o projeto e selecione ***Build Path -> Configure Build Path -> Libraries -> Add External JARs*** e selecione o arquivo .jar do driver.
Teste a conexão com o banco de dados, utilize a classe abaixo (crie uma nova classe no pacote **teste** (faça os *imports* necessários – atenção para selecionar o pacote java.sql.classe_a_importar):

```
public class TestaConexao {

   public static void main(String[] args) {
      Connection conexao = null;
      try {
            String url = "jdbc:postgresql://localhost/SistemaBancario";
            conexao = DriverManager.getConnection(url, "postgres", "postgres");
            // para o H2 descomente as linhas abaixo e comente as linhas acima
            // String url = "jdbc:h2:tcp://localhost/~/progweb";
            // conexao =  DriverManager.getConnection(url,"admin","admin");
            System.out.println("Conectou!");
      } catch (SQLException e1) {
            System.out.println("Erro ao abrir a conexao" + e1.getMessage());
      } finally {
            try {
                  conexao.close();
            } catch (SQLException e2) {
                  System.out
                        .println("Erro ao fechar a conexão" + e2.getMessage());
            }
```

```
        }
    }
}
```

4. Conectando a aplicação com o banco de dados

Vamos utilizar o projeto Prograd da aula passada para exemplificar como conectar com o banco de dados usando o JDBC.

1. No projeto **SistemaBancario**, crie um pacote que será responsável pelas conexões com o BD: **`br.com.bb.sistemabancario.jdbc`**
2. No pacote **br.com.bb.sistemabancario.jdbc**, crie uma classe chamada **ConnectionFactory** com o código:

```
public class ConnectionFactory {
    public Connection getConnection(){
        System.out.println("Conectando ao banco de dados");
        try {
            String url = "jdbc:postgresql://localhost/SistemaBancario";
            return DriverManager.getConnection(url, "postgres", "postgres");

            // para o H2: descomente a parte abaixo e comente as duas linhas acima
            // String url = "jdbc:h2:tcp://localhost/~/progweb";
             // return  DriverManager.getConnection(url,"admin","admin");

        } catch (Exception e) {
              throw new RuntimeException(e);
        }
    }
}
```

3. No projeto **SistemaBancario**, crie também um pacote que será responsável por objetos de acessos aos dados (DAO - Data Access Object): **br.com.bb.sistemabancario.dao**
4. No pacote **br.com.bb.sistemabancario.dao**, crie a classe **ContaCorrenteDAO**;
5. Lembre-se de importar **java.sql** (cuidado para não confundir com a classe **com.sql**); Atalho para importar bibliotecas faltantes: **Ctrl+Shift+O**.
6. Coloque a conexão com o banco de dados no construtor do **ContaCorrenteDAO**:

```
public class ContaCorrenteDAO {

    private Connection connection;

    public ContaCorrenteDAO() {

        this.connection = new ConnectionFactory().getConnection();

    }

}
```

7. Implemente a inserção de registro (conta corrente) através do seguinte método:

```
public void insere(ContaCorrente cc) {
      String sql = "insert into contacorrente (numero, agencia, descricao,
ativa, variacao) values (?,?,?,?,?)";

      try { // prepared statement para inserção
            PreparedStatement stmt = connection.prepareStatement(sql);

            // seta valores
```

```
            stmt.setString(1, cc.getNumero());
            stmt.setString(2, cc.getAgencia());
            stmt.setString(3, cc.getDescricao());
            stmt.setBoolean(4, cc.isAtiva());
            stmt.setInt(5, cc.getVariacao());

            // executa
            stmt.execute();

            // fecha statement
            stmt.close();
      } catch (SQLException e) {
            throw new RuntimeException(e);
      }
}
```

8. Modifique a classe principal **CriaContaCorrente**, acrescentando um código para testar o método de inserção:

```
// gravando registro
ContaCorrenteDAO dao = new ContaCorrenteDAO();
dao.insere(cc);
System.out.println("Gravado!");
```

   Após executar o programa, verifique no BD se os dados foram registrados corretamete.

9. Implemente as outras operações: remoção, alteração e listagem de registros na classe **ContaCorrenteDAO** e teste os métodos na classe **CriaContaCorrente**. Veja os exemplos no slides da aula teórica.
10. Faça uma verificação do BD após cada operação.
11. Observe que a alteração e remoção deve ser feita pelo **id** da conta corrente e, portanto, é necessário conhecê-lo de antemão. Para obter o **id**, pode-se inicialmente fazer uma busca para recuperar o registro do BD, neste caso pelo número da conta corrente. Ou seja, basta criar um método em **ContaCorrenteDAO** para buscar um registro pelo número da conta corrente:

```
public ContaCorrente buscaPeloNumero(String numero) {…}
```