

Desenvolvimento de Aplicações Java
Plataforma Corporativa

Aplicação WEB
Acesso ao Banco de Dados

Fevereiro 2017

Sumário

1.Introdução.....	3
2. Aplicação WEB – DAO	4
Objetivo.....	4
Método	4
Questões	4
ANEXO 1 - DAO.....	5
UsuarioDAO.....	6
UsuarioDAOImpl	7
UsuarioDAOFactory.....	13
UsuarioConstantes	14
UsuarioNaoEncontradoException.....	15
UsuarioUtil	16
DAORuntimeException	20

1.Introdução

Neste documento, a Aplicação WEB já implementada será trabalhada para realizar acesso ao banco de dados através da API JDBC.

2. Aplicação WEB – DAO

Objetivo

Trabalhar sobre aplicação WEB em desenvolvimento fazendo uso de padrões de projeto e persistência através do uso da API JDBC.

Método

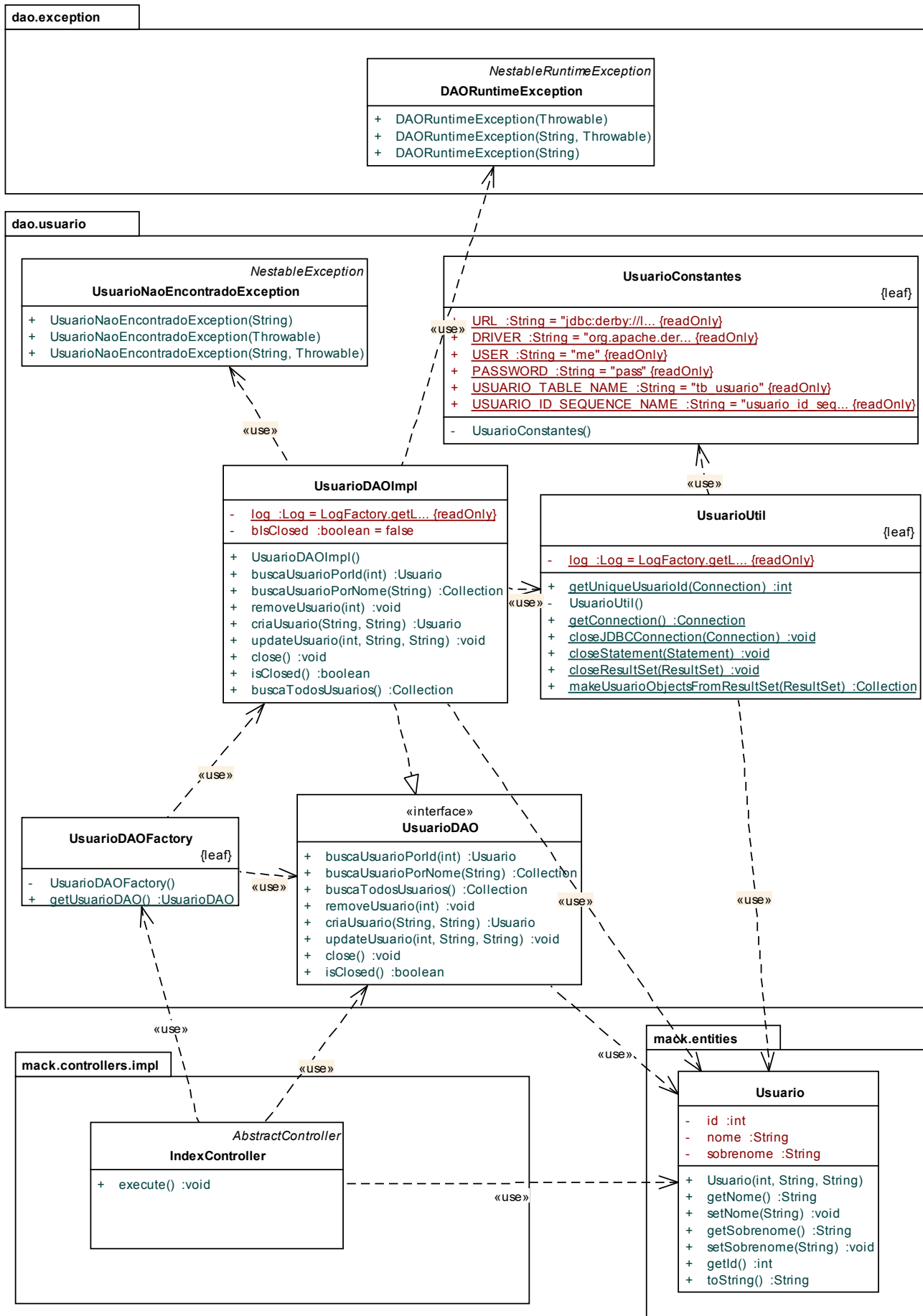
1. Implementar na Aplicação WEB as classes para acesso ao banco de dados do Anexo 1.
 - a. Implementar novas classes e pacotes relativos ao DAO do Usuário.
 - b. Para executar a aplicação, é necessário acrescentar na biblioteca do projeto uma lib (arquivo jar) para cliente do Apache Derby. Tente identifica-lo dentro do pacote do banco de dados que você configurou.
 - c. Executar o servlet de teste.
2. Modifique sua aplicação para que faça uso do DAO ao listar todos os usuários cadastrados e ao realizar uma busca pelo nome.
3. Documente cada classe implementada, indicando propósito e forma com que atinge seus objetivos.
4. Responda as questões.

Questões

1. Explique porque existe a necessidade de utilizar um ID para o usuário.
2. Como geramos o id de um novo usuário?
3. O que foi necessário alterar no que já havia sido implementado na aplicação para o uso do DAO?
4. Quais são os recursos que você deve guardar para restaurar o seu ambiente de desenvolvimento em uma nova máquina? (projeto, dados, etc...).

ANEXO 1 - DAO

```
class DAO Model
```



```
package mack.dao.usuario;

import mack.entities.Usuario;
import java.util.Collection;

public interface UsuarioDAO {

    public Usuario buscaUsuarioPorId(int id) throws UsuarioNaoEncontradoException;
    public Collection buscaUsuarioPorNome(String nome);
    public Collection buscaTodosUsuarios();
    public void removeUsuario(int id) throws UsuarioNaoEncontradoException;
    public Usuario criaUsuario(String nome, String sobrenome);
    public void updateUsuario(int id, String nome, String sobrenome) throws
    UsuarioNaoEncontradoException;
    public void close();
    public boolean isClosed();
}
```

```

package mack.dao.usuario;

import mack.entities.Usuario;
import java.sql.*;
import java.util.*;
import mack.dao.exception.DAORuntimeException;
import org.apache.commons.logging.*;

class UsuarioDAOImpl implements UsuarioDAO {

    static final private Log log = LogFactory.getLog(UsuarioDAOImpl.class);
    private boolean bIsClosed = false;

    public UsuarioDAOImpl() {
        bIsClosed = false;
    }

    @Override
    public Usuario buscaUsuarioPorId(final int id)
        throws UsuarioNaoEncontradoException {
        Connection conn = UsuarioUtil.getConnection();
        Usuario result = null;
        ResultSet rs = null;
        PreparedStatement stmtSelect = null;
        try {
            StringBuilder sbSelect = new StringBuilder();
            sbSelect.append("SELECT usuario_id, nome, sobrenome FROM ");
            sbSelect.append(UsuarioConstantes.USUARIO_TABLE_NAME);
            sbSelect.append(" WHERE usuario_id = ?");
            stmtSelect = conn.prepareStatement(sbSelect.toString());
            stmtSelect.setInt(1, id);
            rs = stmtSelect.executeQuery();
            Collection c = UsuarioUtil.makeUsuarioObjectsFromResultSet(rs);
            if (c.size() != 1) {

```

```

        throw new UsuarioNaoEncontradoException("id = " + id);
    }

    Iterator iter = c.iterator();
    result = (Usuario) iter.next();
} catch (SQLException ex) {
    log.error(ex);
    throw new DAORuntimeException(ex);
} finally {
    UsuarioUtil.closeStatement(stmtSelect);
    UsuarioUtil.closeResultSet(rs);
    UsuarioUtil.closeJDBCCConnection(conn);
}
return result;
}

```

@Override

```

public Collection buscaUsuarioPorNome(final String nome) {
    Connection conn = UsuarioUtil.getConnection();
    Collection result = null;
    ResultSet rs = null;
    PreparedStatement stmtSelect = null;
    try {
        StringBuilder sbSelect = new StringBuilder();
        sbSelect.append("SELECT usuario_id, nome, sobrenome FROM ");
        sbSelect.append(UsuarioConstantes.USUARIO_TABLE_NAME);
        sbSelect.append(" WHERE nome = ?");
        stmtSelect = conn.prepareStatement(sbSelect.toString());
        stmtSelect.setString(1, nome);
        rs = stmtSelect.executeQuery();
        result = UsuarioUtil.makeUsuarioObjectsFromResultSet(rs);
    } catch (SQLException ex) {
        log.error(ex);
        throw new DAORuntimeException(ex);
    } finally {
        UsuarioUtil.closeStatement(stmtSelect);
    }
}

```



```

        UsuarioUtil.closeResultSet(rs);

        UsuarioUtil.closeJDBCCConnection(conn);
    }

    return result;
}

```

@Override

```

public void removeUsuario(final int id)
    throws UsuarioNaoEncontradoException {
    Connection conn = UsuarioUtil.getConnection();
    PreparedStatement stmtDelete = null;
    try {
        StringBuilder sbDelete = new StringBuilder();
        sbDelete.append("DELETE FROM ");
        sbDelete.append(UsuarioConstantes.USUARIO_TABLE_NAME);
        sbDelete.append(" WHERE usuario_id = ?");
        stmtDelete = conn.prepareStatement(sbDelete.toString());
        stmtDelete.setInt(1, id);
        int rows = stmtDelete.executeUpdate();
        if (rows != 1) {
            throw new SQLException("executeUpdate return value: " + rows);
        }
    } catch (SQLException ex) {
        log.error(ex);
        throw new DAORuntimeException(ex);
    } finally {
        UsuarioUtil.closeStatement(stmtDelete);
        UsuarioUtil.closeJDBCCConnection(conn);
    }
}

```

@Override

```

public Usuario criaUsuario(
    final String nome,
    final String sobrenome) {

```

```

Usuario result = null;

PreparedStatement stmtInsert = null;

Connection conn = UsuarioUtil.getConnection();

try {

    int usuario_id = UsuarioUtil.getUniqueUsuarioId(conn);

    StringBuilder sbInsert = new StringBuilder();

    sbInsert.append("INSERT INTO ");

    sbInsert.append(UsuarioConstantes.USUARIO_TABLE_NAME);

    sbInsert.append(" ( usuario_id, nome, sobrenome ) ");

    sbInsert.append(" VALUES (");

    sbInsert.append(" NEXT VALUE FOR ");

    sbInsert.append(UsuarioConstantes.USUARIO_ID_SEQUENCE_NAME);

    sbInsert.append(", ?, ?) ");

    stmtInsert = conn.prepareStatement(sbInsert.toString());

    stmtInsert.setString(1, nome);

    stmtInsert.setString(2, sobrenome);

    log.info("About to execute INSERT: values "

        + nome + ", " + sobrenome);

    int rows = stmtInsert.executeUpdate();

    if (rows != 1) {

        throw new SQLException(

            "executeUpdate return value: "

            + rows);

    }

    result = new Usuario(usuario_id,nome, sobrenome);

} catch (SQLException ex) {

    log.error(ex);

    throw new DAORuntimeException(ex);

} finally {

    UsuarioUtil.closeStatement(stmtInsert);

    UsuarioUtil.closeJDBCCConnection(conn);

}

return result;

}

```

@Override

```
public void updateUsuario(final int id,
    final String nome,
    final String sobrenome) throws UsuarioNaoEncontradoException {

    Connection conn = UsuarioUtil.getConnection();
    PreparedStatement stmtUpdate = null;
    try {
        StringBuilder sbUpdate = new StringBuilder();
        sbUpdate.append("UPDATE ");
        sbUpdate.append(UsuarioConstantes.USUARIO_TABLE_NAME);
        sbUpdate.append(" SET ");
        sbUpdate.append(" nome = ?, ");
        sbUpdate.append(" sobrenome = ? ");
        sbUpdate.append(" WHERE usuario_id = ?");
        stmtUpdate = conn.prepareStatement(sbUpdate.toString());
        stmtUpdate.setString(1, nome);
        stmtUpdate.setString(2, sobrenome);
        stmtUpdate.setInt(3, id);
        int rows = stmtUpdate.executeUpdate();
        if (rows != 1) {
            throw new SQLException("executeUpdate return value: " + rows);
        }
    } catch (SQLException ex) {
        throw new DAORuntimeException(ex);
    } finally {
        UsuarioUtil.closeStatement(stmtUpdate);
        UsuarioUtil.closeJDBCCConnection(conn);
    }
}
```

@Override

```
public void close() {
    log.info("close() called");
}
```

```
        bIsClosed = true;
    }

    @Override
    public boolean isClosed() {
        return bIsClosed;
    }

    @Override
    public Collection buscaTodosUsuarios() {
        Connection conn = UsuarioUtil.getConnection();
        Collection result = null;
        ResultSet rs = null;
        PreparedStatement stmtSelect = null;
        try {
            StringBuilder sbSelect = new StringBuilder();
            sbSelect.append("SELECT usuario_id, nome, sobrenome FROM ");
            sbSelect.append(UsuarioConstantes.USUARIO_TABLE_NAME);
            stmtSelect = conn.prepareStatement(sbSelect.toString());
            rs = stmtSelect.executeQuery();
            result = UsuarioUtil.makeUsuarioObjectsFromResultSet(rs);
        } catch (SQLException ex) {
            log.error(ex);
            throw new DAORuntimeException(ex);
        } finally {
            UsuarioUtil.closeStatement(stmtSelect);
            UsuarioUtil.closeResultSet(rs);
            UsuarioUtil.closeJDBCCConnection(conn);
        }
        return result;
    }
}
```

UsuarioDAOFactory

```
package mack.dao.usuario;

public final class UsuarioDAOFactory {
    private UsuarioDAOFactory() {
    }
    public static UsuarioDAO getUsuarioDAO(){
        return new UsuarioDAOImpl();
    }
}
```

UsuarioConstantes

```
package mack.dao.usuario;

final class UsuarioConstantes{

    static public final String URL = "jdbc:derby://localhost:1527/meuDB";
    static public final String DRIVER = "org.apache.derby.jdbc.ClientDriver";
    static public final String USER="me";
    static public final String PASSWORD="pass";
    static public final String USUARIO_TABLE_NAME = "tb_usuario";
    static public final String USUARIO_ID_SEQUENCE_NAME = "usuario_id_sequence";

    private UsuarioConstantes(){

    }
}
```

UsuarioNaoEncontradoException

```
package mack.dao.usuario;

import org.apache.commons.lang.exception.*;

public class UsuarioNaoEncontradoException extends NestableException {
    public UsuarioNaoEncontradoException(String msg){
        super(msg);
    }
    public UsuarioNaoEncontradoException(Throwable t){
        super(t);
    }
    public UsuarioNaoEncontradoException(String msg, Throwable t){
        super(msg, t);
    }
}
```

```

package mack.dao.usuario;

import mack.entities.Usuario;
import java.sql.*;
import java.util.Collection;
import javax.sql.*;
import mack.dao.exception.DAORuntimeException;
import org.apache.commons.logging.*;

public final class UsuarioUtil {

    static final private Log log = LogFactory.getLog(UsuarioUtil.class);

    static public int getUniqueUsuarioId(final Connection conn)
        throws java.sql.SQLException {

        int id;

        Statement stmtSelect = null;
        ResultSet rs = null;

        StringBuilder sbSelect = new StringBuilder();

        sbSelect.append("SELECT currentvalue FROM SYS.SYSSEQUENCES WHERE SEQUENCENAME='");
        sbSelect.append(UsuarioConstantes.USUARIO_ID_SEQUENCE_NAME.toUpperCase());
        sbSelect.append("'");

        try {
            stmtSelect = conn.createStatement();
            log.info("Ejecutando a query: " + sbSelect.toString());
            rs = stmtSelect.executeQuery(sbSelect.toString());
            if (rs.next()) {
                log.info("OK");
            } else {

```



```

        log.info("NOK");
    }
    long aux = rs.getLong(1);
    id = (int) aux;
    id++;
} finally {
    UsuarioUtil.closeStatement(stmtSelect);
    UsuarioUtil.closeResultSet(rs);
}
return id;
}

```

```

private UsuarioUtil() {
    // this constructor is intentionally private
}

```

```

static public Connection getConnection() {
    Connection conn = null;

    DataSource ds = null;

    try {
        Class.forName(UsuarioConstantes.DRIVER).newInstance();

        conn = DriverManager.getConnection(UsuarioConstantes.URL,
            UsuarioConstantes.USER,
            UsuarioConstantes.PASSWORD);
    } catch (ClassNotFoundException ex) {
        throw new DAORuntimeException(ex);
    } catch (InstantiationException e){
        throw new DAORuntimeException(e);
    }catch (IllegalAccessException e){
        throw new DAORuntimeException(e);
    }catch (SQLException e){

```

```
        throw new DAORuntimeException(e);
    }
    return conn;
}

public static void closeJDBCConnection(final Connection conn) {
    if (conn != null) {
        try {
            conn.close();
        } catch (SQLException ex) {
            log.error(conn, ex);
        }
    }
}

public static void closeStatement(final Statement stmt) {
    if (stmt != null) {
        try {
            stmt.close();
        } catch (SQLException ex) {
            log.error(stmt, ex);
        }
    }
}

public static void closeResultSet(final ResultSet rs) {
    if (rs != null) {
        try {
            rs.close();
        } catch (SQLException ex) {
            log.error(rs, ex);
        }
    }
}
```

```
static public Collection makeUsuarioObjectsFromResultSet(final ResultSet rs)
    throws java.sql.SQLException {
    Collection result = new java.util.ArrayList();

    while (rs.next()) {
        int id = rs.getInt("usuario_id");
        String nome = rs.getString("nome");
        String sobrenome = rs.getString("sobrenome");
        Usuario u = new Usuario(id, nome, sobrenome);
        result.add(u);
    }

    return result;
}
}
```

DAORuntimeException

```
package mack.dao.exception;

import org.apache.commons.lang.exception.*;

public class DAORuntimeException extends NestableRuntimeException{
    public DAORuntimeException(final Throwable cause){
        super(cause);
    }
    public DAORuntimeException( final String msg, final Throwable cause){
        super(msg, cause);
    }
    public DAORuntimeException( final String msg){
        super(msg);
    }
}
```