НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

# Звіт

до лабораторної роботи № 1

Облік машин автосервісу

Виконав:

Студент групи КН-311

Синюк І.І.

Прийняв:

Щербак Сергій

Львів-2020

Мета: розробити DAO рівень для отрмання об’єктів з бази даних.

package com.auto.dao;  
import java.sql.SQLException;  
  
public interface CrudDao<T> extends ReadDao<T>{  
  
  
 int create(T entity) throws SQLException;  
  
 int update(T entity) throws SQLException;  
  
 int delete(Integer id) throws SQLException;  
  
}

package com.auto.dao;  
  
import java.sql.SQLException;  
import java.util.List;  
  
public interface ReadDao<T> {  
 List<T> getAll() throws SQLException;  
  
 T findById(Integer id) throws SQLException;  
}

package com.auto.dao.impl;  
  
import com.auto.dao.CrudDao;  
  
import com.auto.utils.ConnectionManager;  
  
  
import java.sql.\*;  
  
  
  
public abstract class CrudDaoImpl<T> extends ReadDaoImpl<T> implements CrudDao<T> {  
  
 protected CrudDaoImpl() {  
 }  
  
 protected abstract String[] getFields(T entity);  
 protected abstract String[] getUpdateFields(T entity);  
 protected abstract String createQuery();  
 protected abstract String deleteQuery();  
 protected abstract String updateQuery(T entity);  
  
 @Override  
 public int create(T entity) throws SQLException {  
 int result;  
 Connection connection = ConnectionManager.*getInstance*().getConnection();  
 PreparedStatement statement = connection.prepareStatement(createQuery());  
 *setArgsToStatement*(statement, getFields(entity));  
 result = statement.executeUpdate();  
 return result;  
 }  
  
 @Override  
 public int update(T entity) throws SQLException {  
 Connection connection = ConnectionManager.*getInstance*().getConnection();  
 int result;  
 try (PreparedStatement statement = connection.prepareStatement(updateQuery(entity))) {  
 *setArgsToStatement*(statement, getUpdateFields(entity));  
 result = statement.executeUpdate();  
 return result;  
 }  
 }  
  
 @Override  
 public int delete(Integer id) throws SQLException {  
 Connection connection = ConnectionManager.*getInstance*().getConnection();  
 int result;  
 try (PreparedStatement statement = connection.prepareStatement(deleteQuery())) {  
 statement.setInt(1, id);  
 result = statement.executeUpdate();  
 }  
 return result;  
 }  
  
 private static void setArgsToStatement(PreparedStatement preparedStatement, Object... args) {  
 try {  
 for (int i = 0; i < args.length; i++) {  
 if (args[i] == null) {  
 preparedStatement.setNull(i + 1, Types.*NULL*);  
 } else if (String.class.equals(args[i].getClass())) {  
 preparedStatement.setString(i + 1, (String) args[i]);  
 } else if (Long.class.equals(args[i].getClass())) {  
 preparedStatement.setLong(i + 1, (Long) args[i]);  
 } else if (Integer.class.equals(args[i].getClass())) {  
 preparedStatement.setInt(i + 1, (Integer) args[i]);  
 } else if (Date.class.equals(args[i].getClass())) {  
 preparedStatement.setDate(i + 1, (Date) args[i]);  
 } else {  
  
 }  
 }  
 } catch (SQLException e) {  
  
 }  
 }  
}

package com.auto.dao.impl;  
  
import com.auto.dao.CrudDao;  
import com.auto.dao.ReadDao;  
import com.auto.utils.ConnectionManager;  
  
import java.sql.Connection;  
import java.sql.PreparedStatement;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
import java.util.ArrayList;  
import java.util.List;  
  
public abstract class ReadDaoImpl<T> implements ReadDao<T> {  
 protected final static String *QUERY\_NOT\_FOUND* = "Query not found %s";  
 protected final static String *EMPTY\_RESULTSET* = "Empty ResultSet by Query %s";  
 protected final static String *DATABASE\_READING\_ERROR* = "Database Reading Error";  
  
 protected abstract T createInstance(String[] strings);  
 protected abstract String getAllQuery();  
 protected abstract String getByIdQuery();  
 protected abstract String getIdQuery();  
 protected abstract String getByFieldQuery(String field);  
 protected String getByBrandQuery(String field){  
 return "Select \* from checks inner join cars on cars.id = checks.car\_id where brand like '%" + field + "%'";  
 }  
  
 protected List<T> getQueryResult(String query) {  
 List<T> all = new ArrayList<T>();  
 PreparedStatement statement = null;  
 ResultSet resultSet = null;  
 String[] queryResult;  
 if (query == null) {  
 throw new RuntimeException(String.*format*(*QUERY\_NOT\_FOUND*));  
 }  
 try {  
 statement = ConnectionManager.*getInstance*().getConnection().prepareStatement(query);  
 resultSet = statement.executeQuery();  
 queryResult = new String[resultSet.getMetaData().getColumnCount()];  
 while (resultSet.next()) {  
 for (int i = 0; i < queryResult.length; i++) {  
 queryResult[i] = resultSet.getString(i + 1);  
 }  
 all.add(createInstance(queryResult));  
 }  
 } catch (SQLException e) {  
 System.*out*.println("501 problem");  
 throw new RuntimeException(*DATABASE\_READING\_ERROR*, e);  
 } finally {  
 if (resultSet != null) {  
 try {  
 resultSet.close();  
 } catch (Exception ex) {  
 System.*out*.println("500 problem");  
 }  
 }  
 if (statement != null) {  
 try {  
 statement.close();  
 } catch (Exception ex) {  
 System.*out*.println("502 problem");  
 }  
 }  
 }  
 if (all.isEmpty()) {  
 throw new RuntimeException(String.*format*(*EMPTY\_RESULTSET*, query));  
 }  
 return all;  
 }  
  
 public List<T>getByField(String field){  
 return getQueryResult(getByFieldQuery(field));  
 }  
 public List<T> getByBrand(String field){  
 return getQueryResult(getByBrandQuery(field));  
 }  
  
 public Integer getId() throws SQLException {  
 Integer key = null;  
 Connection connection = ConnectionManager.*getInstance*().getConnection();  
 PreparedStatement statement = connection.prepareStatement(getIdQuery());  
 ResultSet resultSet = statement.executeQuery();  
 if (resultSet.next()) {  
 key = (int) resultSet.getLong(1);  
 }  
 return key;  
 }  
  
 public List<T> getAll() {  
  
 return getQueryResult(  
 getAllQuery());  
 }  
  
 @Override  
 public T findById(Integer id) {  
  
 return getQueryResult(String.*format*(getByIdQuery(),id)).get(0);  
 }  
}

Висновок : розробив рівень DAO .