Tên: Phạm Dương Minh Nhật

Mã sinh viên: 19IT182

# **Lab 2**

## **AutomobileLibary**

### **Car.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AutomobileLibary.BussinessObject

{

public class Car

{

public int CarID { get; set; }

public string CarName { get; set; }

public string Manufacturer { get; set; }

public decimal Price { get; set; }

public int ReleasedYear { get; set; }

}

}

### **BaseDAL.cs**

using Microsoft.Data.SqlClient;

using Microsoft.Extensions.Configuration;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AutomobileLibary.DataAcess

{

public class BaseDAL

{

public StockDataProvider dataProvider {get; set;} = null;

public SqlConnection connection = null;

public BaseDAL()

{

var connectionString = GetConnectionString();

dataProvider = new StockDataProvider(connectionString);

}

public string GetConnectionString()

{

string connectionString;

IConfiguration config = new ConfigurationBuilder()

.SetBasePath(Directory.GetCurrentDirectory())

.AddJsonFile("appsettings.json", true, true)

.Build();

connectionString = config["ConnectionString:MyStockDB"];

return connectionString;

}

public void CloseConnection() => dataProvider.CloseConnection(connection);

}

}

### **CarDBContext.cs**

using AutomobileLibary.BussinessObject;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.Common;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AutomobileLibary.DataAcess

{

public class CarDBContext:BaseDAL

{

private static CarDBContext instance = null;

private static readonly object instanceLock = new object();

private CarDBContext(){ }

public static CarDBContext Instance

{

get

{

lock (instanceLock)

{

if(instance == null)

{

instance = new CarDBContext();

}

return instance;

}

}

}

public IEnumerable<Car> GetCarList()

{

IDataReader dataReader = null;

string SQLSelect = "Select CarID, CarName, Manufacturer, Price, ReleasedYear from Cars";

var cars = new List<Car>();

try

{

dataReader = dataProvider.GetDataReader(SQLSelect, CommandType.Text, out connection);

while (dataReader.Read())

{

cars.Add(new Car

{

CarID = dataReader.GetInt32(0),

CarName = dataReader.GetString(1),

Manufacturer = dataReader.GetString(2),

Price = dataReader.GetDecimal(3),

ReleasedYear = dataReader.GetInt32(4)

});

}

}

catch (Exception e)

{

throw new Exception(e.Message);

}

finally

{

dataReader.Close();

CloseConnection();

}

return cars;

}

public Car GetCarByID(int carID)

{

Car car = null;

IDataReader dataReader = null;

string SQLSelect = "Select CarID, CarName, Manufacturer, Price, ReleasedYear" + " from Cars where CarID = @CarID";

try

{

var param = dataProvider.CreateParameter("@CarID", 4, carID, DbType.Int32);

dataReader = dataProvider.GetDataReader(SQLSelect, CommandType.Text, out connection, param);

if (dataReader.Read())

{

car = new Car

{

CarID = dataReader.GetInt32(0),

CarName = dataReader.GetString(1),

Manufacturer = dataReader.GetString(2),

Price = dataReader.GetDecimal(3),

ReleasedYear = dataReader.GetInt32(4)

};

}

}

catch (Exception e)

{

throw new Exception(e.Message);

}

finally

{

dataReader.Close();

CloseConnection();

}

return car;

}

public void AddNew(Car car)

{

try

{

Car pro = GetCarByID(car.CarID);

if(pro == null)

{

string SQLInsert = "Insert Cars values(@CarID, @CarName, @Manufacturer, @Price, @ReleasedYear)";

var parameters = new List<SqlParameter>();

parameters.Add(dataProvider.CreateParameter("@CarID", 4, car.CarID, DbType.Int32));

parameters.Add(dataProvider.CreateParameter("@CarName", 50, car.CarName, DbType.String));

parameters.Add(dataProvider.CreateParameter("@Manufacturer", 50, car.Manufacturer, DbType.String));

parameters.Add(dataProvider.CreateParameter("@Price", 50, car.Price, DbType.Decimal));

parameters.Add(dataProvider.CreateParameter("@ReleasedYear", 4, car.ReleasedYear, DbType.Int32));

dataProvider.Insert(SQLInsert, CommandType.Text, parameters.ToArray());

} else

{

throw new Exception("The Car is already exist");

}

} catch (Exception e)

{

throw new Exception(e.Message);

}

finally

{

CloseConnection();

}

}

public void Update(Car car)

{

try

{

Car pro = GetCarByID(car.CarID);

if (pro != null)

{

string SQLUpdate = "Update Cars set CarName = @CarName, Manufuturer = @Manufuturer,"

+ "Price = @Price, ReleasedYear = @ReleasedYear where CarID=@CarID";

var parameters = new List<SqlParameter>();

parameters.Add(dataProvider.CreateParameter("@CarID", 4, car.CarID, DbType.Int32));

parameters.Add(dataProvider.CreateParameter("@CarName", 50, car.CarName, DbType.String));

parameters.Add(dataProvider.CreateParameter("@Manufacturer", 50, car.Manufacturer, DbType.String));

parameters.Add(dataProvider.CreateParameter("@Price", 50, car.Price, DbType.Decimal));

parameters.Add(dataProvider.CreateParameter("@ReleasedYear", 4, car.ReleasedYear, DbType.Int32));

dataProvider.Update(SQLUpdate, CommandType.Text, parameters.ToArray());

}

else

{

throw new Exception("The Car does not already exist");

}

}

catch (Exception e)

{

throw new Exception(e.Message);

}

finally

{

CloseConnection();

}

}

public void Remove(int carID)

{

try

{

Car pro = GetCarByID(carID);

if (pro != null)

{

string SQLDelete = "Delete Cars where CarID = @CarID";

var param = dataProvider.CreateParameter("@CarID", 4, carID, DbType.Int32);

dataProvider.Delete(SQLDelete, CommandType.Text, param);

}

else

{

throw new Exception("The Car does not already exist");

}

}

catch (Exception e)

{

throw new Exception(e.Message);

}

finally

{

CloseConnection();

}

}

}

}

### **StockDataProvider.cs**

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Reflection.PortableExecutable;

using System.Text;

using System.Threading.Tasks;

namespace AutomobileLibary.DataAcess

{

public class StockDataProvider

{

public StockDataProvider() { }

private string ConnectionString { get; set; }

public StockDataProvider(string ConnectionString) => ConnectionString = ConnectionString;

public void CloseConnection(SqlConnection connection) => connection.Close();

public SqlParameter CreateParameter(string name, int size, object value, DbType dbType, ParameterDirection direction = ParameterDirection.Input)

{

return new SqlParameter

{

DbType = dbType,

ParameterName = name,

Size = size,

Direction = direction,

Value = value,

};

}

public IDataReader GetDataReader(string commandText, CommandType commandType, out SqlConnection connection, params SqlParameter[] parameters)

{

IDataReader reader = null;

try

{

connection = new SqlConnection(ConnectionString);

connection.Open();

var command = new SqlCommand(commandText, connection);

command.CommandType = commandType;

if(parameters != null)

{

foreach(var parameter in parameters)

{

command.Parameters.Add(parameter);

}

}

reader = command.ExecuteReader();

}

catch (Exception e)

{

throw new Exception(e.Message);

}

return reader;

}

public void Delete(string commandText, CommandType commandType, params SqlParameter[] parameters)

{

try

{

using var connection = new SqlConnection(ConnectionString);

connection.Open();

using var command = new SqlCommand(commandText, connection);

command.CommandType = commandType;

if (parameters != null)

{

foreach (var parameter in parameters)

{

command.Parameters.Add(parameter);

}

}

command.ExecuteNonQuery();

}

catch (Exception e)

{

throw new Exception( "Data Provider: Delete Method", e.InnerException);

}

}

public void Insert(string commandText, CommandType commandType, params SqlParameter[] parameters)

{

try

{

using var connection = new SqlConnection(ConnectionString);

connection.Open();

using var command = new SqlCommand(commandText, connection);

command.CommandType = commandType;

if (parameters != null)

{

foreach (var parameter in parameters)

{

command.Parameters.Add(parameter);

}

}

command.ExecuteNonQuery();

}

catch (Exception e)

{

throw new Exception(e.Message);

}

}

public void Update(string commandText, CommandType commandType, params SqlParameter[] parameters)

{

try

{

using var connection = new SqlConnection(ConnectionString);

connection.Open();

using var command = new SqlCommand(commandText, connection);

command.CommandType = commandType;

if (parameters != null)

{

foreach (var parameter in parameters)

{

command.Parameters.Add(parameter);

}

}

command.ExecuteNonQuery();

}

catch (Exception e)

{

throw new Exception(e.Message);

}

}

}

}

### **CarRepository.cs**

using AutomobileLibary.BussinessObject;

using AutomobileLibary.DataAcess;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace AutomobileLibary.Repository

{

public class CarRepository:ICarRepository

{

public Car GetCarByID(int carID) => CarDBContext.Instance.GetCarByID(carID);

public IEnumerable<Car> GetCars() => CarDBContext.Instance.GetCarList();

public void InsertCar(Car car) => CarDBContext.Instance.AddNew(car);

public void UpdateCar(Car car) => CarDBContext.Instance.Update(car);

public void DeleteCar(int carID) => CarDBContext.Instance.Remove(carID);

}

}

### **ICarRepository.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using AutomobileLibary.BussinessObject;

namespace AutomobileLibary.Repository

{

public interface ICarRepository

{

IEnumerable<Car> GetCars();

Car GetCarByID(int carId);

void InsertCar(Car car);

void UpdateCar(Car car);

void DeleteCar(int carId);

}

}

### **appsettings.json**

{

"exclude": [

"\*\*/bin",

"\*\*/bower\_components",

"\*\*/jspm\_packages",

"\*\*/node\_modules",

"\*\*/obj",

"\*\*/platforms"

],

"ConnectionStrings": {

"MyStockDB": "Server:=SONNE;Integrated Security=true; Database=MyStock;",

}

}

## **UI**

### **CarDeltaills.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AutomobileLibary.BussinessObject;

using AutomobileLibary.Repository;

namespace UI

{

public partial class CarDeltails : Form

{

public CarDeltails()

{

InitializeComponent();

}

public ICarRepository CarRepository { get; set; }

public bool InsertOrUpdate { get; set; }

public Car CarInfo { get; set; }

private void CarDeltails\_Load(object sender, EventArgs e)

{

cboManufacturer.SelectedIndex = 0;

txtCarID.Enabled = !InsertOrUpdate;

if(InsertOrUpdate == true)

{

txtCarID.Text = CarInfo.CarID.ToString();

txtCarName.Text = CarInfo.CarName;

cboManufacturer.Text = CarInfo.Manufacturer.Trim();

txtPrice.Text = CarInfo.Price.ToString();

txtReleasedYear.Text = CarInfo.ReleasedYear.ToString();

}

}

private void btnSave\_Click(object sender, EventArgs e)

{

try

{

var car = new Car

{

CarID = int.Parse(txtCarID.Text),

CarName = txtCarName.Text,

Manufacturer = cboManufacturer.Text,

Price = decimal.Parse(txtPrice.Text),

ReleasedYear = int.Parse(txtReleasedYear.Text)

};

if(InsertOrUpdate == false)

{

CarRepository.InsertCar(car);

} else

{

CarRepository.UpdateCar(car);

}

} catch (Exception ex)

{

MessageBox.Show(ex.Message, InsertOrUpdate == false ? "Add a new car?" : "Update a car?");

}

}

private void btnCancel\_Click(object sender, EventArgs e)

{

Close();

}

}

}

### **CarManageMent.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Dynamic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using AutomobileLibary.BussinessObject;

using AutomobileLibary.Repository;

namespace UI

{

public partial class CarManagement : Form

{

public CarManagement()

{

InitializeComponent();

}

ICarRepository carRepository = new CarRepository();

BindingSource source;

private void btnLoad\_Click(object sender, EventArgs e)

{

LoadCarList();

}

private void CarManagement\_Load(object sender, EventArgs e)

{

btnDel.Enabled = false;

dtGVListCar.CellDoubleClick += dtGVListCar\_CellDoubleClick;

}

private void dtGVListCar\_DoubleClick(object sender, EventArgs e)

{

}

private void ClearText()

{

txtCarID.Text = String.Empty;

txtCarName.Text = String.Empty;

txtManufacturer.Text = String.Empty;

txtPrice.Text = String.Empty;

txtReleasedYear.Text = String.Empty;

}

private Car GetCarObject()

{

Car car = null;

try {

car = new Car

{

CarID = int.Parse(txtCarID.Text),

CarName = txtCarName.Text,

Manufacturer = txtManufacturer.Text,

Price = decimal.Parse(txtPrice.Text),

ReleasedYear = int.Parse(txtReleasedYear.Text)

};

} catch (Exception ex)

{

MessageBox.Show(ex.Message, "Get Car");

}

return car;

}

public void LoadCarList()

{

var cars = carRepository.GetCars();

try

{

source = new BindingSource();

source.DataSource = cars;

txtCarID.DataBindings.Clear();

txtCarName.DataBindings.Clear();

txtManufacturer.DataBindings.Clear();

txtPrice.DataBindings.Clear();

txtReleasedYear.DataBindings.Clear();

txtCarID.DataBindings.Add("Text", source, "CarID");

txtCarName.DataBindings.Add("Text", source, "CarName");

txtManufacturer.DataBindings.Add("Text", source, "Manufacturer");

txtPrice.DataBindings.Add("Text", source, "Price");

txtReleasedYear.DataBindings.Add("Text", source, "ReleasedYear");

dtGVListCar.DataSource = null;

dtGVListCar.DataSource = source;

if(cars.Count() == 0)

{

ClearText();

btnDel.Enabled = false;

} else

{

btnDel.Enabled = true;

}

} catch (Exception ex)

{

MessageBox.Show(ex.Message, "Load Car List");

}

}

private void btnNew\_Click(object sender, EventArgs e)

{

CarDeltails carDeltails = new CarDeltails

{

Text = "Add Car",

InsertOrUpdate = false,

CarRepository = carRepository

};

if(carDeltails.ShowDialog() == DialogResult.OK)

{

LoadCarList();

source.Position = source.Count - 1;

}

}

private void btnDel\_Click(object sender, EventArgs e)

{

try

{

var car = GetCarObject();

carRepository.DeleteCar(car.CarID);

LoadCarList();

} catch (Exception ex)

{

MessageBox.Show(ex.Message, "Delete Car");

}

}

private void btnClose\_Click(object sender, EventArgs e)

{

Close();

}

private void dtGVListCar\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

CarDeltails carDeltails = new CarDeltails

{

Text = "Update Car",

InsertOrUpdate = true,

CarInfo = GetCarObject(),

CarRepository = carRepository

};

if (carDeltails.ShowDialog() == DialogResult.OK)

{

LoadCarList();

source.Position = source.Count - 1;

}

}

}

}

Graphical user interface

Description automatically generated  
Graphical user interface

Description automatically generated