МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

**ОТЧЁТ**

по курсовой работе

по дисциплине:

ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЁННЫХ БАЗ ДАННЫХ

на тему:

«Приложение с базой данных»

Выполнил: ст. гр. ТКИ-541

Архипов М. С.

Проверила: доцент., к.т.н.

Васильева М. А.

МОСКВА – 2022

Содержание

[Цель: 3](#_Toc122430703)

[Задание: 3](#_Toc122430704)

[Код программы: 3](#_Toc122430705)

[Результаты работы: 14](#_Toc122430706)

[Вывод: 16](#_Toc122430707)

# Цель:

Изучить современные технологии ORM, разработать приложение с базой данных, умеющее отрабатывать операции CRUD.

# Задание:

1. Выбрать предметную область.

В соответствии с заданием на магистерскую диссертацию (или выбрать предметную область самостоятельно) описать предметную область, выделить основные сущности, формализовать задачу будущего приложения.

1. Разработать базу данных.

Разработать ER-диаграмму, разработать структуру таблиц (инфологическое проектирование БД), провести нормализацию БД.

1. Выбрать СУБД для физического проектирования БД. Создать БД.
2. Выбрать язык для разработки приложения.
3. Выбрать технологию ORM.
4. Разработать приложение с учетом выбранной технологии ORM.

Разработать репозиторий для работы с БД.

# Код программы:

Program.cs

using System;

using System.Configuration;

using SportClub;

using DataAccessLayer;

namespace Library

{

class Program

{

static void Main(string[] args)

{

var training = new Training("2345");

var coaches = new Coach("Petrov", "Petr", null);

var clients = new Clients("Ivanov", "Ivan", "Ivanovich", coaches);

clients.AddTraining(training);

Console.WriteLine(clients);

Console.ReadKey(true);

var settings = new Settings();

settings.AddDatabaseServer("AX3RIP4IK\\SQLEXPRESS");

settings.AddDatabase("SportClub");

using var sessionFactory = Configurator.GetSessionFactory(settings, showSql: true);

using (var session = sessionFactory.OpenSession())

{

session.Save(training);

session.Save(coaches);

session.Save(clients);

session.Flush();

}

}

}

}

Client.cs

using System;

using System.Collections.Generic;

namespace SportClub

{

public class Clients

{

public Clients(string lastName, string firstName, string middleName, ISet<Coach> coaches)

{

if (string.IsNullOrWhiteSpace(firstName))

{

throw new ArgumentNullException(nameof(firstName));

}

if (string.IsNullOrWhiteSpace(lastName))

{

throw new ArgumentNullException(nameof(lastName));

}

if ((middleName?.Trim())?.Length == 0)

{

throw new ArgumentOutOfRangeException(nameof(middleName));

}

this.Id = Guid.NewGuid();

this.FirstName = firstName;

this.LastName = lastName;

this.MiddleName = middleName;

this.coaches = coaches;

this.trainings = trainings;

}

[Obsolete("For ORM")]

protected Clients()

{

}

public virtual Guid Id { get; protected set; }

/// <summary>

/// Имя.

/// </summary>

public virtual string FirstName { get; protected set; }

/// <summary>

/// Фамилия.

/// </summary>

public virtual string LastName { get; protected set; }

/// <summary>

/// Отчество.

/// </summary>

public virtual string MiddleName { get; protected set; }

/// <summary>

/// Тренировки.

/// </summary>

public virtual ISet<Coach> coaches { get; protected set; } = new HashSet<Coach>();

public virtual ISet<Training> trainings { get; protected set; } = new HashSet<Training>();

public virtual bool AddTraining(Training trainings)

{

return trainings is null

? throw new ArgumentNullException(nameof(trainings))

: this.trainings.Add(trainings);

}

public Clients(string lastName, string firstName, string middleName, params Coach[] coaches)

: this(lastName, firstName, middleName, new HashSet<Coach>(coaches))

{

}

public override string ToString()

{

return $"{this.LastName},{this.FirstName},{this.MiddleName},{string.Join(", ", this.coaches)}";

}

}

}

Client.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SportClub

{

public class Coach : IEquatable<Coach>

{

public Coach(string lastName, string firstName, string? middleName)

{

if (string.IsNullOrWhiteSpace(firstName))

{

throw new ArgumentNullException(nameof(firstName));

}

if (string.IsNullOrWhiteSpace(lastName))

{

throw new ArgumentNullException(nameof(lastName));

}

if ((middleName?.Trim())?.Length == 0)

{

throw new ArgumentOutOfRangeException(nameof(middleName));

}

this.Id = Guid.NewGuid();

this.FirstName = firstName;

this.LastName = lastName;

this.MiddleName = middleName;

}

[Obsolete("For ORM")]

protected Coach()

{

}

public virtual Guid Id { get; protected set; }

/// <summary>

/// Имя.

/// </summary>

public virtual string FirstName { get; protected set; }

/// <summary>

/// Фамилия.

/// </summary>

public virtual string LastName { get; protected set; }

/// <summary>

/// Отчество.

/// </summary>

public virtual string MiddleName { get; protected set; }

/// <summary>

/// Тренировки.

/// </summary>

public virtual ISet<Clients> Clients { get; protected set; } = new HashSet<Clients>();

public virtual bool Equals(Coach other)

{

return Equals(other);

}

public override bool Equals(object obj)

{

if (object.ReferenceEquals(obj, null))

{

return false;

}

if (object.ReferenceEquals(obj, this))

{

return true;

}

return (obj is Clients other)

&& string.Equals(this.FirstName, other.FirstName, StringComparison.InvariantCulture);

}

public override string ToString()

{

return $"{this.LastName},{this.FirstName},{this.MiddleName}";

}

}

}

Training.cs

using System;

using System.Collections.Generic;

namespace SportClub

{

public class Training

{

public Training(string numTrain)

{

if (string.IsNullOrEmpty(numTrain))

{

throw new ArgumentNullException("name");

}

this.NumTrain = numTrain;

}

[Obsolete("For ORM")]

protected Training()

{

}

public virtual Guid Id { get; protected set; }

public virtual string NumTrain { get; protected set; }

public virtual ISet<Clients> clients { get; protected set; } = new HashSet<Clients>();

public override string ToString()

{

return $"{this.NumTrain}";

}

}

}

ClientsTest.cs

using System.Reflection;

using Xunit;

using SportClub;

namespace SportClubTest;

public class ClientsTests

{

[Fact]

public void CreateClientsTrue()

{

//Arrange

string lastName = "asdf";

string firstName = "asdf";

string middleName = "asdf";

string LastName = "asdf";

string FirstName = "asdf";

string MiddleName = "asdf";

string temp = $"{lastName},{firstName},{middleName},{LastName},{FirstName},{MiddleName}";

//Act

var coaches1 = new Coach(LastName,FirstName,MiddleName);

var clients1 = new Clients(lastName,firstName,middleName,coaches1);

//Assert

Assert.Equal(clients1.ToString(),temp);

}

[Fact]

public void ExceptionCreateFalse()

{

//Arrange

string lastName = "asdf";

string firstName = "asdf";

string middleName = "asdf";

string LastName = "asdf";

string FirstName = "asdf";

string MiddleName = "asdf";

//Act

var coaches1 = new Coach(LastName, FirstName, MiddleName);

var clients1 = new Clients(lastName, firstName, middleName, coaches1);

//Assert

Assert.Throws<ArgumentNullException>(() => new Clients(lastName, null!, middleName, coaches1));

}

[Fact]

public void AddCreateTrue()

{

//Arrange

string lastName = "asdf";

string firstName = "asdf";

string middleName = "asdf";

string LastName = "asdf";

string FirstName = "asdf";

string MiddleName = "asdf";

string numTrain = "12345";

//Act

var training1 = new Training(numTrain);

var coaches1 = new Coach(LastName, FirstName, MiddleName);

var clients1 = new Clients(lastName, firstName, middleName, coaches1);

clients1.AddTraining(training1);

//Assert

Assert.Equal(clients1.trainings.Contains(training1), true);

}

}

CoachTest.cs

using System.Reflection;

using Xunit;

using SportClub;

namespace SportClubTest;

public class CoachTests

{

[Fact]

public void CreateCoachTrue()

{

//Arrange

string LastName = "asdf";

string FirstName = "asdf";

string MiddleName = "asdf";

string temp = $"{LastName},{FirstName},{MiddleName}";

//Act

var coaches1 = new Coach(LastName, FirstName, MiddleName);

//Assert

Assert.Equal(coaches1.ToString(), temp);

}

[Fact]

public void ExceptionCreateFalse()

{

//Arrange

string LastName = "asdf";

string FirstName = "asdf";

string MiddleName = "asdf";

//Act

var coaches1 = new Coach(LastName, FirstName, MiddleName);

//Assert

Assert.Throws<ArgumentNullException>(() => new Coach(null!, FirstName, MiddleName));

}

}

TrainingTest.cs

using System.Reflection;

using Xunit;

using SportClub;

namespace SportClubTest;

public class TrainingTests

{

[Fact]

public void CreateTrainingTrue()

{

//Arrange

string numTrain = "12345";

string temp = $"{numTrain}";

//Act

var training1 = new Training(numTrain);

//Assert

Assert.Equal(training1.ToString(), temp);

}

[Fact]

public void ExceptionCreateFalse()

{

//Arrange

string numTrain = "12345";

//Act

var training1 = new Training(numTrain);

//Assert

Assert.Throws<ArgumentNullException>(() => new Training(null!));

}

}

ConfiguratorTest.cs

namespace DataAccessLayer

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

public class ConfiguratorTests

{

private static Configuration? configuration;

public static ISessionFactory GetSessionFactory(

Assembly? assembly = null,

bool showSql = false)

{

var databaseConfiguration =

SQLiteConfiguration.Standard.InMemory();

if (showSql)

{

databaseConfiguration = databaseConfiguration

.ShowSql()

.FormatSql();

}

return Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings.AddFromAssembly(assembly ??

Assembly.GetExecutingAssembly()))

.ExposeConfiguration(c => configuration = c)

.BuildSessionFactory();

}

public static ISession BuildSessionForTest(bool showSql = true)

{

var session = GetSessionFactory(showSql: showSql).OpenSession();

new SchemaExport(configuration)

.Execute(true, true, false, session.Connection, null);

return session;

}

}

}

MapTest.cs

namespace SportClubTest

{

using DataAccessLayer;

using NHibernate;

using NUnit.Framework;

public class MappingTests

{

protected ISession Session { get; private set; }

[SetUp]

public void Setup()

{

this.Session = ConfiguratorTests.BuildSessionForTest();

}

[TearDown]

public void TearDown()

{

this.Session?.Dispose();

}

}

}

ClientMap.cs

// <copyright file="ClientMap.cs" company="PlaceholderCompany">

// Copyright (c) PlaceholderCompany. All rights reserved.

// </copyright>

namespace DataAccessLayer

{

using FluentNHibernate.Mapping;

using SportClub;

internal class ClientMap : ClassMap<Clients>

{

/// <summary>

/// Initializes a new instance of the <see cref="ClientMap"/> class.

/// </summary>

public ClientMap()

{

this.Table("Clients");

this.Id(x => x.Id);

this.Map(x => x.FirstName).Not.Nullable().Length(255);

this.Map(x => x.LastName).Not.Nullable().Length(255);

this.Map(x => x.MiddleName).Nullable().Length(255);

//this.References(x => x.coaches);

this.HasManyToMany(x => x.coaches)

.Cascade.Delete()

.Not.Inverse();

}

}

}

CoachMap.cs

// <copyright file="ClientMap.cs" company="PlaceholderCompany">

// Copyright (c) PlaceholderCompany. All rights reserved.

// </copyright>

namespace DataAccessLayer

{

using FluentNHibernate.Mapping;

using SportClub;

internal class CoachMap : ClassMap<Coach>

{

/// <summary>

/// Initializes a new instance of the <see cref="CoachesMap"/> class.

/// </summary>

public CoachMap()

{

this.Table("Coaches");

this.Id(x => x.Id);

this.Map(x => x.FirstName).Not.Nullable().Length(255);

this.Map(x => x.LastName).Not.Nullable().Length(255);

this.Map(x => x.MiddleName).Nullable().Length(255);

this.HasManyToMany(x => x.Clients)

.Cascade.Delete()

.Not.Inverse();

}

}

}

DAL.cs

namespace DataAccessLayer

{

using System.Reflection;

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

public class ConfiguratorTests

{

private static Configuration? configuration;

public static ISessionFactory GetSessionFactory(

Assembly? assembly = null,

bool showSql = false)

{

var databaseConfiguration =

SQLiteConfiguration.Standard.InMemory();

if (showSql)

{

databaseConfiguration = databaseConfiguration

.ShowSql()

.FormatSql();

}

return Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings.AddFromAssembly(assembly ??

Assembly.GetExecutingAssembly()))

.ExposeConfiguration(c => configuration = c)

.BuildSessionFactory();

}

public static ISession BuildSessionForTest(bool showSql = true)

{

var session = GetSessionFactory(showSql: showSql).OpenSession();

new SchemaExport(configuration)

.Execute(true, true, false, session.Connection, null);

return session;

}

}

}

DataAccessLayer.cs

using FluentNHibernate.Cfg;

using FluentNHibernate.Cfg.Db;

using NHibernate;

using NHibernate.Cfg;

using NHibernate.Tool.hbm2ddl;

using System.Reflection;

namespace DataAccessLayer

{

public static class Configurator

{

private static FluentConfiguration? fluentConfiguration;

public static ISessionFactory GetSessionFactory(

Settings settings,

Assembly? assembly = null,

bool showSql = false)

{

return GetConfiguration(settings

, assembly ?? Assembly.GetExecutingAssembly(), showSql)

.BuildSessionFactory();

}

private static FluentConfiguration GetConfiguration(

Settings settings,

Assembly assembly,

bool showSql = false)

{

if (fluentConfiguration is null)

{

var databaseConfiguration =

MsSqlConfiguration.MsSql2012.ConnectionString(

x => x

.Server(settings.GetDatabaseServer())

.Database(settings.GetDatabaseName())

.TrustedConnection());

if (showSql)

{

databaseConfiguration = databaseConfiguration

.ShowSql()

.FormatSql();

}

fluentConfiguration = Fluently.Configure()

.Database(databaseConfiguration)

.Mappings(m => m.FluentMappings.AddFromAssembly(assembly))

.ExposeConfiguration(BuildSchema);

}

return fluentConfiguration;

}

private static void BuildSchema(NHibernate.Cfg.Configuration configuration)

{

new SchemaExport(configuration).Execute(true, true, false);

}

}

}

Settings.cs

// <copyright file="Settings.cs" company="PlaceholderCompany">

// Copyright (c) PlaceholderCompany. All rights reserved.

// </copyright>

using System;

namespace DataAccessLayer

{

public sealed class Settings

{

private string databaseServerName;

private string databaseName;

public void AddDatabaseServer(string serverName)

{

this.databaseServerName = serverName;

}

public void AddDatabase(string databaseName)

{

this.databaseName = databaseName;

}

public string GetDatabaseServer()

{

return this.databaseServerName;

}

public void AddDatabaseName(string databaseName)

{

this.databaseName = databaseName;

}

public string GetDatabaseName()

{

return this.databaseName;

}

}

}

TrainingMap.cs

// <copyright file="ClientMap.cs" company="PlaceholderCompany">

// Copyright (c) PlaceholderCompany. All rights reserved.

// </copyright>

namespace DataAccessLayer

{

using FluentNHibernate.Mapping;

using SportClub;

internal class TrainingMap : ClassMap<Training>

{

/// <summary>

/// Initializes a new instance of the <see cref="TrainingsMap"/> class.

/// </summary>

public TrainingMap()

{

this.Table("Trainings");

this.Id(x => x.Id);

this.Map(x => x.NumTrain).Not.Nullable().Length(255);

this.HasManyToMany(x => x.clients)

.Cascade.Delete()

.Not.Inverse();

}

}

}

# Результаты работы:

Изображение выглядит как текст, снимок экрана, монитор, черный

Автоматически созданное описание

Рисунок – Прохождение тестов

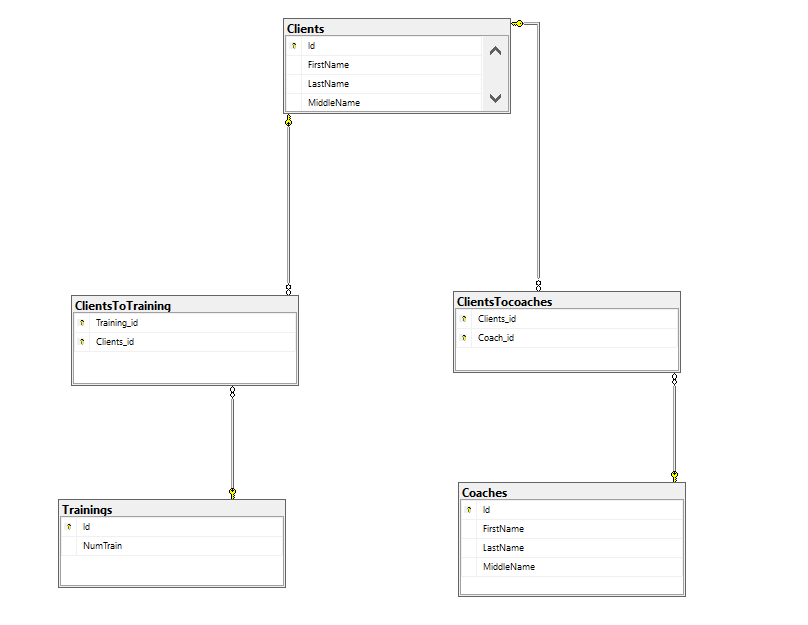


Рисунок – Диаграмма базы данных

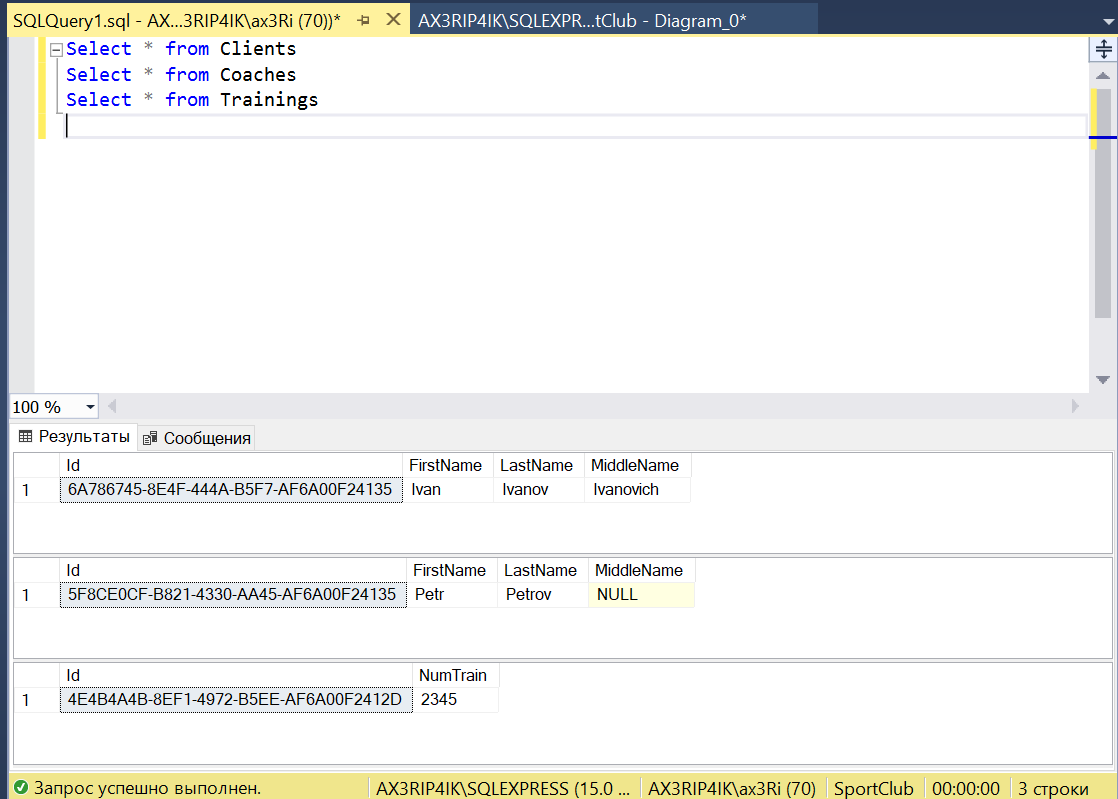


Рисунок – Демонстрация наполнения базы данных

# Вывод:

В результате выполнения работы были изучены современные технологии ORM и разработано приложение с базой данных, умеющее отрабатывать операции CRUD.