WEEK 4 ASSESSMENT

Doctor Prescription connection with DB and unit testing with Log4net

--------------------------------------------------------------------------------------------------------------------------------------

Programs.cs

--------------------------------------------------------------------------------------------------------------------------------------

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Week4Assessment;

using log4net;

namespace Week4Assessment

{

public class ServerException : Exception

{

public ServerException(string message) : base(message) { }

}

public class DoctorPrescription

{

public int DoctorID { get; set; }

public string PatientName { get; set; }

public string Medication { get; set; }

public double Dosage { get; set; }

public override string ToString()

{

return $"DoctorID : {DoctorID} Patient : {PatientName} Dosage : {Dosage}\n";

}

}

public class DoctorPrescriptionService

{

private static string connectionString = "Data Source=LAPTOP 19GF1AGG\\SQLEXPRESS04;Initial Catalog = Week4AssessmentDb; Integrated Security = True;";

public static void Read(DoctorPrescription[] doctorPrescriptions)

{

try {

using (SqlConnection conn = new SqlConnection(connectionString))

{

string query = "SELECT DoctorID, PatientName, Dosage FROM DoctorPrescription";

SqlCommand cmd = new SqlCommand(query, conn);

conn.Open();

SqlDataReader reader = cmd.ExecuteReader();

for (int i = 0; i < doctorPrescriptions.Length; i++)

{ if (!reader.Read())

{ throw new ServerException("[0101]Server Errror."); }

doctorPrescriptions[i] = new DoctorPrescription

{

DoctorID = (int)reader["DoctorID"],

PatientName = reader["PatientName"].ToString(),

Dosage = (double)reader["Dosage"]

};

} } }

catch (SqlException ex)

{ throw new ServerException($"[0102]ServerErrror.{ex.Message}"); }

catch (ServerException ex) { ex; }

catch (Exception ex)

{ throw new ServerException($"[0103]Server Errror.{ex.Message}"); }

}

public static void Sort(DoctorPrescription[] doctorPrescriptions)

{ int size = doctorPrescriptions.Length;

for (int i = 0; i < size - 1; i++)

{ int minIndex = i;

for (int j = i + 1; j < size; j++)

{

if (string.Compare(doctorPrescriptions[j].PatientName, doctorPrescriptions[minIndex].PatientName) < 0)

{ minIndex = j; }

}

if (minIndex != i)

{

DoctorPrescription temp = doctorPrescriptions[minIndex];

doctorPrescriptions[minIndex] = doctorPrescriptions[i];

doctorPrescriptions[i] = temp;

}

} }

public static DoctorPrescription FindMin(DoctorPrescription[] doctorPrescriptions)

{

Console.Write("Enter doctor id to find the minimum dosage prescribed : ");

int searchId = int.Parse(Console.ReadLine());

var minDosage = double.MaxValue;

DoctorPrescription minDoctorPrescription = null;

foreach (DoctorPrescription item in doctorPrescriptions)

{

if (item.DoctorID == searchId && item.Dosage < minDosage)

{

minDosage = item.Dosage;

minDoctorPrescription = item;

}

}

return minDoctorPrescription;

}

public static DoctorPrescription FindThirdMax(DoctorPrescription[] doctorPrescriptions)

{

int size = doctorPrescriptions.Length;

for (int i = 0; i < size - 1; i++)

{

int minIndex = i;

for (int j = i + 1; j < size; j++)

{

if (doctorPrescriptions[j].Dosage < doctorPrescriptions[minIndex].Dosage)

{ minIndex = j; }

}

if (minIndex != i)

{

DoctorPrescription temp = doctorPrescriptions[minIndex];

doctorPrescriptions[minIndex] = doctorPrescriptions[i];

doctorPrescriptions[i] = temp;

}

}

return doctorPrescriptions[size - 3];

}

public class Programs

{

private static readonly ILog log = LogManager.GetLogger(typeof(Programs));

static void Main()

{

DoctorPrescription[] doctorPrescriptions = new DoctorPrescription[5];

try

{ DoctorPrescriptionService.Read(doctorPrescriptions); }

catch (ServerException ex)

{ Console.WriteLine($"{ex.Message}"); }

DoctorPrescription min = DoctorPrescriptionService.FindMin(doctorPrescriptions);

Console.WriteLine($"Minimum dosage : {min}");

DoctorPrescription thirdMax = DoctorPrescriptionService.FindThirdMax(doctorPrescriptions);

Console.WriteLine($"Third maximum dosage : {thirdMax}");

DoctorPrescriptionService.Sort(doctorPrescriptions);

string output = "";

foreach (var e in doctorPrescriptions)

{ output += $"{e}"; }

Console.WriteLine(output);

} } }

--------------------------------------------------------------------------------------------------------------------------------------

SQL Query

--------------------------------------------------------------------------------------------------------------------------------------

CREATE DATABASE Week4AssessmentDb;

USE Week4AssessmentDb;

CREATE TABLE DoctorPrescription{

DoctorID INT PRIMARY KEY,

PatientName VARCHAR(225),

Dosage FLOAT };

INSERT INTO DoctorPrescription { DoctorID, PatientName, Dosage } VALUES

( 1, ‘Mahesh’, 22),

( 2, ‘Vijesh’, 11),

( 3, ‘Rajesh, 35),

( 4, ‘Rijesh, 23),

( 5, ‘Sujesh, 15);

SELECT DoctorID, PatientName, Dosage FROM DoctorPrescription;

--------------------------------------------------------------------------------------------------------------------------------------

AssemblyInfo.cs

--------------------------------------------------------------------------------------------------------------------------------------

[assembly: log4net.Config.XmlConfigurator]

--------------------------------------------------------------------------------------------------------------------------------------

App.config

--------------------------------------------------------------------------------------------------------------------------------------

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

<section name="log4net" type="log4net.Config.Log4NetConfigurationSectionHandler, log4net" />

</configSections>

<log4net>

<!-- File Appender -->

<appender name="FileAppender" type="log4net.Appender.RollingFileAppender">

<file value="week4assessment\_app\_log.log" />

<appendToFile value="true" />

<rollingStyle value="Size" />

<maxSizeRollBackups value="5" />

<maximumFileSize value="10MB" />

<staticLogFileName value="true" />

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%date [%thread] %-5level %logger - %message%newline" />

</layout>

</appender>

<!-- Console Appender -->

<appender name="ConsoleAppender" type="log4net.Appender.ConsoleAppender">

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%date [%thread] %-5level %logger - %message%newline" />

</layout>

</appender>

<!-- Root logger -->

<root>

<level value="ALL" />

<appender-ref ref="FileAppender" />

<appender-ref ref="ConsoleAppender" />

</root>

</log4net>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

</configuration>

--------------------------------------------------------------------------------------------------------------------------------------

DoctorPrescriptionService Test. cs

--------------------------------------------------------------------------------------------------------------------------------------

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Week4Assessment;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Week4Assessment.Tests

{ [TestClass()]

public class DoctorPrescriptionServiceTests

{ [TestMethod()]

public void SortTest()

{

DoctorPrescription[] doctorPrescriptions = new DoctorPrescription[5];

DoctorPrescriptionService.Read(doctorPrescriptions);

DoctorPrescription expected = new DoctorPrescription()

{

DoctorID = 5,

PatientName = "Sujesh",

Dosage = 15

};

DoctorPrescriptionService.Sort(doctorPrescriptions);

DoctorPrescription actual = doctorPrescriptions[3];

Assert.AreEqual(expected.ToString(), actual.ToString());

}

[TestMethod()]

public void FindMinTest()

{

DoctorPrescription[] doctorPrescriptions = new DoctorPrescription[5];

DoctorPrescriptionService.Read(doctorPrescriptions);

DoctorPrescription expected = new DoctorPrescription()

{

DoctorID = 2,

PatientName = "Vijesh",

Dosage = 11

};

DoctorPrescription actual = DoctorPrescriptionService.FindMin(doctorPrescriptions);

Assert.AreEqual(expected.ToString(), actual.ToString());

}

[TestMethod()]

public void FindThirdMaxTest()

{ DoctorPrescription[] doctorPrescriptions = new DoctorPrescription[5];

DoctorPrescriptionService.Read(doctorPrescriptions);

DoctorPrescription expected = new DoctorPrescription()

{

DoctorID = 1,

PatientName = "Mahesh",

Dosage = 22

};

DoctorPrescription actual = DoctorPrescriptionService.FindThirdMax(doctorPrescriptions);

Assert.AreEqual(expected.ToString(), actual.ToString());

} } }

--------------------------------------------------------------------------------------------------------------------------------------

Output

--------------------------------------------------------------------------------------------------------------------------------------



