**Lesson 11 - Working with Database using C# & Unit Testing**

# Microsoft SQL Server Management Studio Express

To work with SQL Server 2012, first you have to install the management studio express on your local machine.

When you lunch the management studio express, the following window will appear.



Server name:

In your computer, you may have several instances on of sql server 2005, or different versions of sql server like as 2005, 2008 etc. you will give them a specific name to work with it.

In this case, MAGICBOX is the server name and SQLEXPRESS is the instance name of the sql server 2005 on which we are currently working on.

Authentication:

Sql server provides you couple of types of authentication. In Windows Authentication, server assumes since you logged into the machine, you are authenticated to access on the server. But if you use the other type (SQL Server Authentication), you have to enter the credentials of User name and the Password field of that server instance.

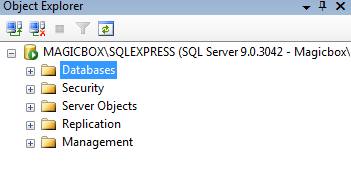
Walkthrough 1: Creating ‘t\_Student’ table using SQL Server Management Studio Express

Steps:

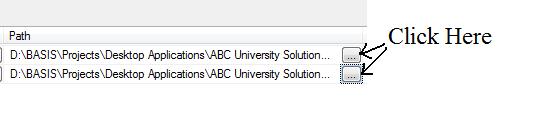
1. Go to: StartMicrosoft SQL Server 2012SQL Server Management Studio Express
2. The following window will appear



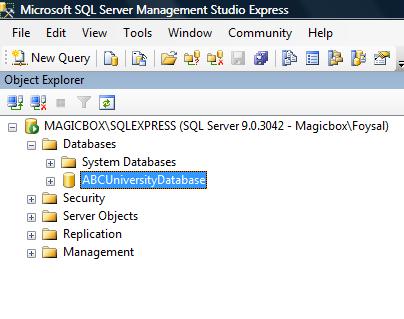
1. Click the ‘Connect’ button
2. There are several nodes you will find on which you can work with. Each of the nodes explains itself.



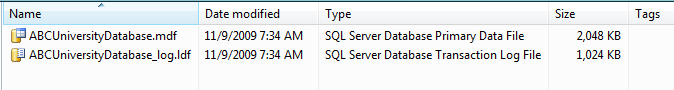
1. Right click on the ‘Databases’ node
2. Click on ‘New Database’
3. Enter the database name meaningful. For our user requirement, we will enter ‘ABCUniversityDatabase’ as the name.
4. Click on the button given in the following figure to select the destination path of the database files



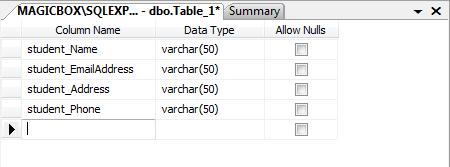
1. Click ‘OK’
2. You will find the following figure on your screen



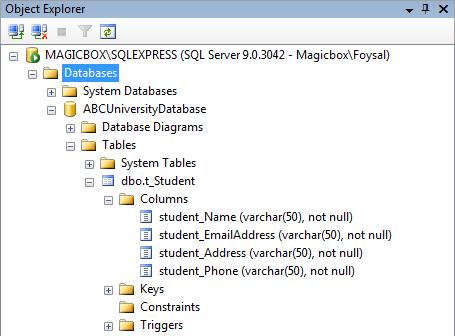
1. Check your HDD file location to check the database file.



1. Now expand the ‘ABCUniversityDatabase’ on your management studio express
2. Right click on the ‘Tables’ node
3. Click on ‘New Table’ option
4. Enter the column names given in the following figure



1. Click on save option and enter the table name as ‘t\_Student’
2. Now expand the ‘Tables’ node from Object Explorer window
3. You will find your just created table in here



Thus you have created a table (t\_Student) what your user wants from you. 

Walkthrough 2: Creating the ABC University Application

Steps:

1. Fire visual studio 2008
2. Create a desktop application named ‘ABCUniversityApplication’
3. Create a windows form named ‘StudentEntryUI’
4. We have found an object of student. So create a class Student which have name, email address, address and phone number as its data. Create a constructor to assign these values when creating a new student object.

public class Student

{

private string name;

private string emailAddress;

private string address;

private string phoneNumber;

public Student(string name,string emailAddress, string address, string phoneNumber)

{

this.name = name;

this.emailAddress = emailAddress;

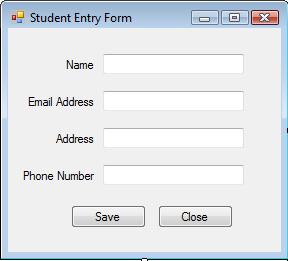
this.address = address;

this.phoneNumber = phoneNumber;

}

}

1. Now create a user interface like the given figure



1. Now take a note. Your ‘Student’ class holds the student data. Your UI class takes data from user. So you need a class which will check the validity of the data inputted by the user. That’s why you need a manager class which will perform different works related with the student object.

Create a ‘StudentManager’ class which will take data from UI and pass it to another class for saving.

public class StudentManager

{

public void SaveStudentManager(Student student)

{

StudentGateway studentGateway=new StudentGateway();

studentGateway.SaveStudentIntoDatabase(student);

}

}

1. Create the ‘StudentGateway’ class for saving the student object into the database.

To do so, you need to use some built in class of C#.

1. Create a connection string which will be used as a parameter to establish a connection with the database

string connectionString = @"server=Magicbox\SQLEXPRESS; Integrated Security = SSPI; database = ABCUniversityDB";

1. To create a connection with the database, you need to use SqlConnection class. To use it, you have to import system’s SqlClient directive into your code
2. Create an object of that class.

SqlConnection connection = new SqlConnection(connectionString);

1. Open the connection of the databse.

connection.Open();

1. Now create the insert query string for execution to save the data into the database.

Since your Student class doesn’t have the get properties of the data, change your Student class adding the get part for each of the data.

string insertQueryString =

@"INSERT INTO t\_Student VALUES (

'" +student.Name + @"',

'" +student.EmailAddress + @"',

'" +student.Address + @"',

'" +student.PhoneNumber + @"'

)";

1. Now you need a SqlCommand object to execute the query you have built to save the data

SqlCommand command = new SqlCommand(insertQueryString, connection);

1. Now execute the command

command.ExecuteNonQuery();

1. So at the end, your StudentGateway class will have the following code snippet

public class StudentGateway

{

public void SaveStudentIntoDatabase(Student student)

{

string connectionString = @"server=Magicbox\SQLEXPRESS; Integrated Security = SSPI; database = ABCUniversityDatabase";

SqlConnection connection = new SqlConnection(connectionString);

connection.Open();

string insertQueryString =

@"INSERT INTO t\_Student VALUES (

'" +student.Name + @"',

'" +student.EmailAddress + @"',

'" +student.Address + @"',

'" +student.PhoneNumber + @"'

)";

SqlCommand command = new SqlCommand(insertQueryString, connection);

command.ExecuteNonQuery();

}

}

1. Now go back to your UI class. Write down the following code into your saveButton method block

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

{

Student student=new Student(nameTextBox.Text,emailAddressTextBox.Text,

addressTextBox.Text,phoneNumberTextBox.Text);

StudentManager studentManager = new StudentManager();

studentManager.SaveStudentManager(student);

MessageBox.Show(student.Name+"'s information is saved into database");

nameTextBox.Text = "";

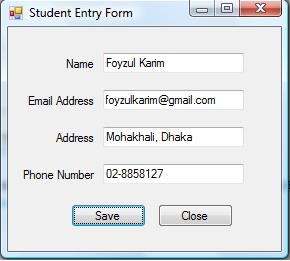
emailAddressTextBox.Text = "";

addressTextBox.Text = "";

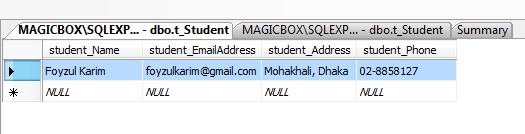
phoneNumberTextBox.Text = "";

}

1. Now build the application and run it. Enter the data given like the following figure and click on the save button



1. If the data is saved, the messagebox will appear showing you the success of the saving of your data.
2. Now go to your Sql Server Management Studio Express and right click on t\_Student table
3. Click ‘Open Table’ option to check whether our data is really saved or not
4. You will see the following figure



1. Thus we have saved a sample data into the database through an application.

User Story:

Mr. X: “Mr. Foysal. I am facing a problem.”

Me: “What is it?”

Mr. X: “First I thought, I wouldn’t need first name and middle name and last name separately. That’s why I told you to store only the name. but now, I need to store these parts separately. Can you do it for me?”

Me: “why not? It’s simpler than you think.”

Mr. X: “thank you. Thank you a lot.”

Me: 

Types of attributes:

There are several kinds of attributes. But selecting any data as attribute depends on the user story. Depending to your user’s story, any attribute can be changed into other types of attribute.

Atomic attributes:

The simple or atomic attributes cannot be further broken or subdivided.

Analysis:

Since our user just wanted to store the data of a student, we didn’t need to select the attributes other than atomic attributes.

Composite attributes:

Two or more atomic attributes make a composite attribute. Or in other words, a composite attribute can be divided into sub parts or multiple different atomic attributes.

Analysis:

Since our user told us to store the first name, middle name and the last name of a student, so we will create composite attribute Name. And this name has multiple attributes.

Notation:

We use ellipse for expressing a multi-valued attribute.

Result:

Middle Name

First Name

Last Name

Name

Student

Table schema:

Though the name attribute itself an attribute, we don’t need to add it in our table schema. The table schema of the given figure just above will be:

t\_Student ( student\_FirstName, student\_MiddleName student\_LastName)

User Story:

Mr. X: “I need to save both t&t number and mobile number of a student in database now. Is it possible?”

Me: “yap”

Multi-valued attributes:

Multi-valued attributes hold more than one value for a particular entity.

Analysis:

In our user’s story, we can find the phone number is now acting as a multi-valued attribute. It has several data for a single student entity.

Notation:

We express multi-valued attributes in a concentric oval or circles.

Result:

Phone

Student

Table schema:

We will create a new table having the multi-valued attributes with a single main attribute of the main table.

t\_StudentPhone(student\_FirstName, student\_Phone)

User Story:

Mr. X: “Mr. Foysal. My university rules are being changed. We have to store student’s date of birth and age now. Please do something for me.”

Me: “Sure. But I hope you know these changes will add a little more charge on your product?”

Mr. X: “Of course I know. And I am very happy to hire you for my software. ”

Me: “thanks”

Derived attributes:

A derived attribute is a result calculated from another attribute.

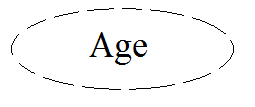
Analysis:

It is not a good choice to let the application user to calculate the age of the student respect to the date of birth. It will consume the time and also not be an errorless calculation. So, we need to add an attribute which will be calculated from the date of birth of the student. This attribute (Age) is called the derived attribute of the student entity.

Notation:

We use a dashed oval to denote a derived attribute.

Result:



Birth Date

Student

Table schema:

A derived attribute will be treated as the other attribute in database table.

t\_Student(student\_BirthDate, student\_Age)

So, after fulfilling the changes made by the user, we will have the following ERD

Last Name

Middle Name

First Name

Date Of Birth

Name

Email Address

Address

Age

Student

And the table schema which will be created by this ERD is given below:

t\_Student(student\_FirstName, student\_MiddleName, student\_LastName, student\_EmailAddres, student\_Address, student\_DateOfBirth, student\_Age)

t\_StudentPhoneNumber(student\_FirstName, student\_PhoneNumber)

Walkthrough : Update the ABC University Application on User's requirement

Steps:

1. Fire your visual studio IDE

2. a. Since the client wants to add couple of more attributes for the student now, we just create the attributes in the student class and make their Get part to access those variables.

b. Again, since our client now wants to add different types of phone number for a student. So we now need a new object type PhoneNumber which will be an attribute of Student object.

3. Create a class named PhoneNumber.cs and write the following code snippet in it

public class PhoneNumber

{

private string landPhoneNumber;

private string mobilePhoneNumber;

public string LandPhoneNumber

{

get { return landPhoneNumber; }

set { landPhoneNumber = value; }

}

public string MobilePhoneNumber

{

get { return mobilePhoneNumber; }

set { mobilePhoneNumber = value; }

}

}

4. Add a PhoneNumber object into the Student class.

private PhoneNumber phoneNumberObject;

5. Add the following code snippet into the constructor of the Student.cs class to assign values at the time of object creation

phoneNumberObject = new PhoneNumber();

phoneNumberObject.LandPhoneNumber = landPhoneNumber;

phoneNumberObject.MobilePhoneNumber = mobilePhoneNumber;

6. Create a Get part of the phoneNumberObject data to access the variable publicly

public PhoneNumber PhoneNumberObject

{

get { return phoneNumberObject; }

}

7. a. Since the architecture of the t\_Student has been changed, and so as the formation of Student class, you need to change the query of the insert method in the gateway class.

b. Since t\_PhoneNumber table is added new, so as PhoneNumber.cs class also been added, you also need to add a new query for saving the phone number data of a student into the newly created table.

c. Your StudentGateway.cs class now should have the following method

public int SaveStudentIntoDatabase(Student student)

{

string connectionString = @"server=Magicbox\SQLEXPRESS; Integrated Security = SSPI; database = ABCUniversityDatabase";

SqlConnection sqlConnection=new SqlConnection(connectionString);

sqlConnection.Open();

string insertStudentQueryString = @"INSERT INTO t\_Student VALUES

(

'" + student.FirstName + @"',

'" + student.MiddleName + @"',

'" + student.LastName + @"',

'" + student.EmailAddress + @"',

'" + student.Address + @"',

'" + student.DateOfBirth + @"',

" + student.Age + @"

)

";

string insertLandPhoneNumberQueryString = @"INSERT INTO t\_StudentPhoneNumber VALUES

(

'"+student.FirstName+@"',

'" +student.PhoneNumberObject.LandPhoneNumber+@"'

)

";

string insertMobilePhoneNumberQueryString = @"INSERT INTO t\_StudentPhoneNumber VALUES

(

'" + student.FirstName + @"',

'" + student.PhoneNumberObject.MobilePhoneNumber + @"'

)

";

SqlCommand insertStudentCommand=new SqlCommand(insertStudentQueryString,sqlConnection);

SqlCommand insertLandPhoneNumberCommand=new SqlCommand(insertLandPhoneNumberQueryString,sqlConnection);

SqlCommand insertMobilePhoneNumberCommand=new SqlCommand(insertMobilePhoneNumberQueryString,sqlConnection);

int effectedRows = insertStudentCommand.ExecuteNonQuery();

effectedRows+= insertLandPhoneNumberCommand.ExecuteNonQuery();

effectedRows += insertMobilePhoneNumberCommand.ExecuteNonQuery();

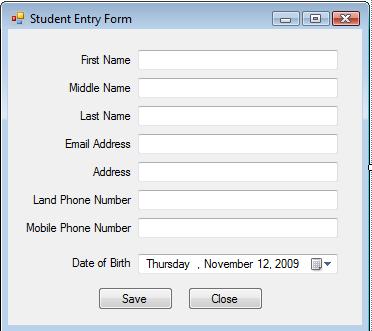
return effectedRows;

}

8. Your manager class don't need to change according to the user's requirement.

9. Now go to your StudentEntryUI.cs class

a. design your user interface like the following figure.

b. Since your user now enters more data than before, the code behind of your Save button must

deal with more data than past. Change the code to hold all of the data of a student object. And throw the object to the manager class. So the code snippet should look like the following code in your Click method of the Save button.

StudentManager studentManager=new StudentManager();

Student student=new Student(firstNameTextBox.Text,middleNameTextBox.Text,lastNameTextBox.Text,emailAddressTextBox.Text,addressTextBox.Text,landPhoneNumberTextBox.Text,mobilePhoneNumberTextBox.Text,dateOfBirthDateTimePicker.Value);

int effectedRows = studentManager.SaveStudentManager(student);

if(effectedRows!=0)

{

MessageBox.Show(student.FirstName + @"'s information is saved into database successfully");

}

else

{

MessageBox.Show("Data did not saved");

}

firstNameTextBox.Text = "";

middleNameTextBox.Text = "";

lastNameTextBox.Text = "";

emailAddressTextBox.Text = "";

addressTextBox.Text = "";

landPhoneNumberTextBox.Text = "";

mobilePhoneNumberTextBox.Text = "";

dateOfBirthDateTimePicker.Value = DateTime.Today;

10. Now build the application and run it.

11. Enter the appropriate data and press the Save button to save the data into database.

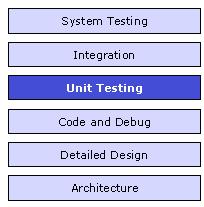
**Unit Testing:**

## Introduction

uction to unit testing and explains a tool used for unit testing.

## What is Unit Testing?

Unit testing is a kind of testing done at the developer side. It is used to test methods, properties, classes, and assemblies. Unit testing is not testing done by the quality assurance department. To know where unit testing fits into development, look at the following image:



*Figure: Unit Testing in Application Development*

Unit testing is used to test a small piece of workable code (operational) called unit. This encourages developers to modify code without immediate concerns about how such changes might affect the functioning of other units or the program as a whole. Unit testing can be time consuming and tedious, but should be done thoroughly with patience.

## What is NUnit?

NUnit is a unit testing framework for performing unit testing based on the .NET platform. It is a widely used tool for unit testing and is preferred by many developers today. NUnit is free to use. NUnit does not create any test scripts by itself. You have to write test scripts by yourself, but NUnit allows you to use its tools and classes to make unit testing easier. The points to be remembered about NUnit are listed below:

1. **NUnit is not an automated GUI testing tool.**
2. **It is not a scripting language, all tests are written in .NET supported languages, e.g., C#, VC, VB.NET, J#, etc.**

NUnit is a derivative of the popular testing framework used by eXtreme Programming (XP). It was created by Philip Craig for .NET. It is also available in the name of jUnit for Java code testing.

NUnit works by providing a class framework and a test runner application. They can be downloaded from [here](http://www.nunit.org/index.php?p=download). The class framework allows to write test cases based on the application. The test is run using the test runner application downloaded from the above link.

## Steps for Using NUnit

First, what one needs to do is download the recent version of the NUnit framework from the above mentioned website.

1. In development studio, create a new project. In my case, I have created a console application.
2. In the program.cs file, write the following code:

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Enter two numbers\n");

int number1;

int number2;

number1 = int.Parse(Console.ReadLine());

number2 = int.Parse(Console.ReadLine());

MathsHelper helper = new MathsHelper();

int x = helper.Add(number1, number2);

Console.WriteLine("\nThe sum of " + number1 +

" and " + number2 + " is " + x);

Console.ReadKey();

int y = helper.Subtract(number1, number2);

Console.WriteLine("\nThe difference between " +

number1 + " and" + number2 + " is " + y);

Console.ReadKey();

}

}

public class MathsHelper

{

public MathsHelper() { }

public int Add(int a, int b)

{

int x = a + b;

return x;

}

public int Subtract(int a, int b)

{

int x = a - b;

return x;

}

}

1. Then to the solution of the project, add a new class library project and name it followed by “.Test” (it is the naming convention used for unit testing). Import the downloaded DLL files into the project and follow the steps given below.
2. In the newly added project, add a class and name it TestClass.cs.
3. In the class file, write the following code:

[TestFixture]

public class TestClass

{

[TestCase]

public void AddTest()

{

MathsHelper helper = new MathsHelper();

int result = helper.Add(20, 10);

Assert.AreEqual(30, result);

}

[TestCase]

public void SubtractTest()

{

MathsHelper helper = new MathsHelper();

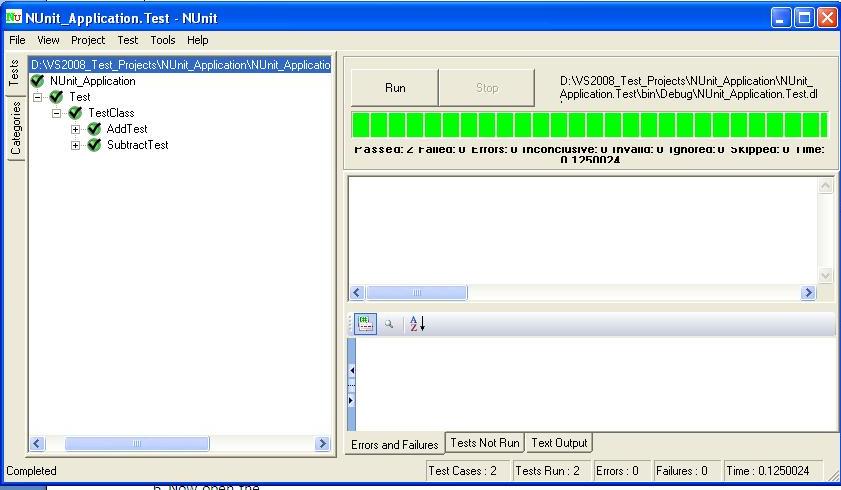
int result = helper.Subtract(20, 10);

Assert.AreEqual(10, result);

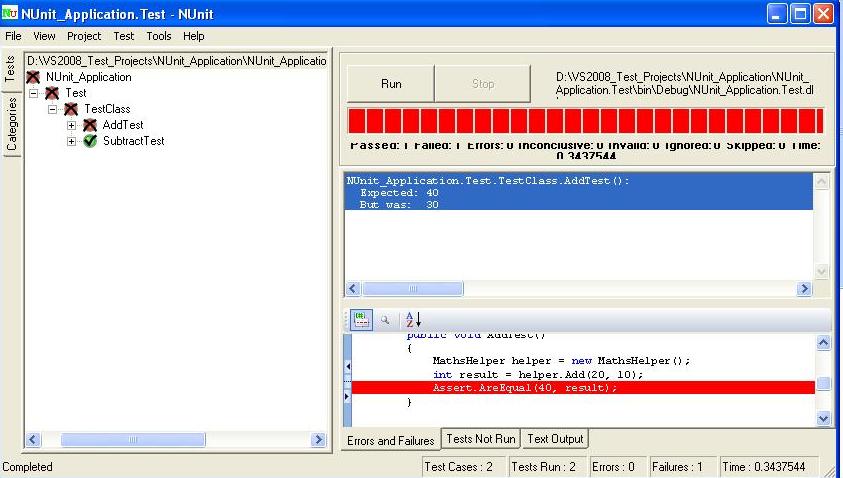
}

}

1. Now open the test runner (test runner is downloaded from the NUnit site along with the NUnit DLLs). In the NUnit API, click File > Open project. A file open dialog appears. Give the path of the NUunit test project DLL. Now run the test. If the test passes, then the following test screen is displayed:



Otherwise, the following screen displays:



### Important Attributes

### 1. [SetUp]

SetUp is generally used for initialization purposes. Any code that must be initialized or set prior to executing a test is put in functions marked with this attribute. As a consequence, it avoids the problem of code repetition in each test.

http://www.codeproject.com/images/minus.gif

[SetUp]

public void Xyz()

{

.

.

}

The code written in the Xyz method is executed before the test runs and it avoids the call of the code inside this method.

### 2. [Ignore]

This is the attribute which is used for the code that needs to be bypassed.

### 3. [ExpectedException]

This attribute is used to test methods that need to throw an exception in some cases. The cases may beFileNotFoundException and others.

http://www.codeproject.com/images/minus.gif

[Test]

[ExpectedException(typeof(MissingFileException))]

public void CheckException()

{

......

......

}

The code shouldn't have any Assert statement. On passing the test, the code should throw an exception, otherwise an exception is not thrown.

### 4. [TearDown]

This is an attribute that acts the opposite of [SetUp]. It means the code written with this attribute is executed last (after passing other lines of code). So, inside this closing is generally done, i.e., closing of file system, database connection, etc.

### Mock Objects

A mock object is a simulation of a real object. Mock objects act just as real objects but in a controlled way. A mock object is created to test the behavior of a real object. In unit testing, mock objects are used to scrutinize the performance of real objects. Simply said, a mock object is just the imitation of a real object. Some important characteristics of mock objects are that they are lightweight, easily created, quick and deterministic, and so on.

### Stub Object

A stub object is an object which implements an interface of a component. A stub can be configured to return a value as required. Stub objects provide a valid response, but it is of static nature meaning that no matter what input is passed in, we always get the same response.

### Mock Object vs. Stub Object

Mock objects vary from stub objects in some ways. They are listed below:

1. The first distinct factor is that mock objects test themselves, i.e., they check if they are called at the proper time in the proper manner by the object being tested. Stubs generally just return stubbed data, which can be configured to change depending on how they are called.
2. Secondly, mock objects are generally lightweight relative to stubs with different instances set for mock objects for every test whilst stubs are often reused between tests.