Building your Allocated Component

Creating the properties for your table design

[Creating the properties for your table design 1](#_Toc58413716)

[Before you Begin 1](#_Toc58413717)

[Creating the Properties 1](#_Toc58413718)

# Creating the properties for your table design

## Before you Begin

Make sure that you have completed the work for last week and that you have watched the eLectures for this week before starting this work.

Having established the work you need to undertake it is down to you as a team to dive in and have a go yourselves. You need to complete your own components using the following notes as a guide. Remember you need to adapt what I give you to match what you are building and also keep in mind the work-flow.

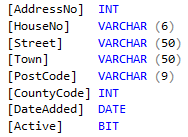
Keep in mind your Git Hub day of work!

* Start with a new folder called todays work
* Clone the master to your local computer
* Create a new branch
* Do some work
* Merge the perfectly formed work to Git Hub
* Delete the branch

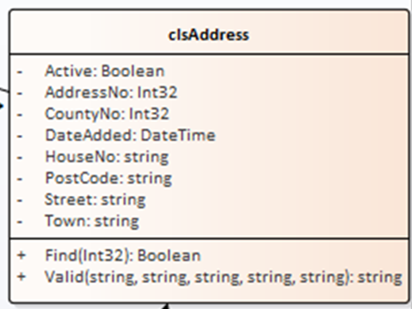
## Creating the Properties

Within your allocated component your table will have a set of attributes that need to be modelled as properties within your new class (in my case clsAddress).

My attributes are as follows…

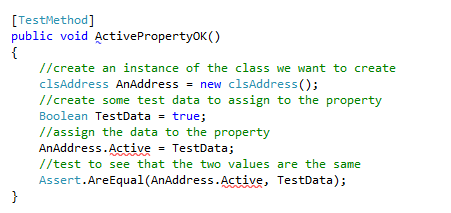


My attributes modelled in the class design are as follows…

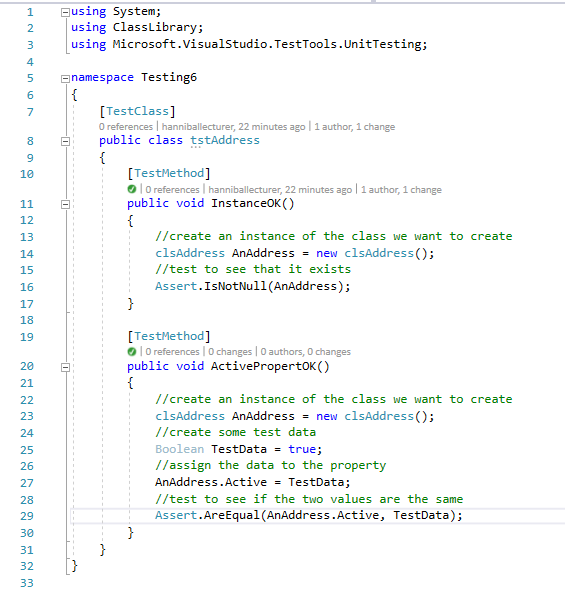


We need to pick off the properties one at a time, create a test in the test class and from there generate the property.

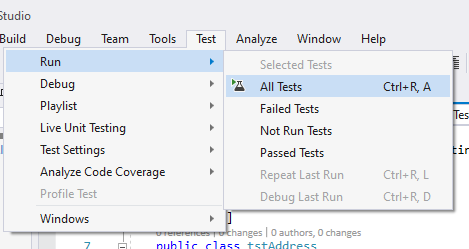
In tstAddress I will create a test for the first property “Active”…



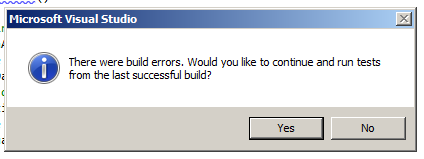
Your class code should look something like this…



Run all test from the main menu…

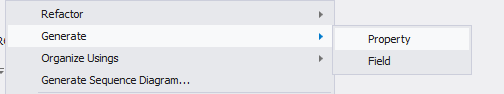


Watch them fail…



(As always press no here!)

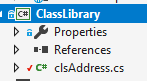
Fix the problem by holding the mouse over the red underlining and generate the new property…



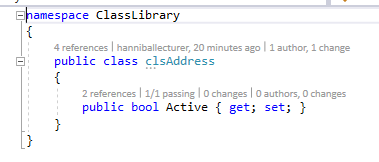
Make sure the test passes…



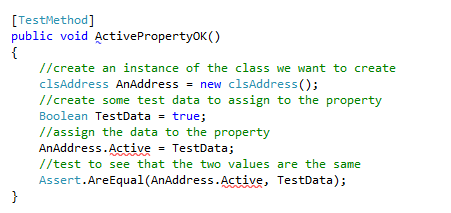
Open the code for clsAddress in the class library…



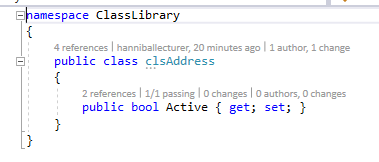
Double click the file and you should see the code for the property created automatically here…



Notice also that having created the correct data type for the test data in your original test…

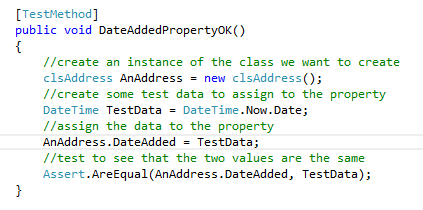


The correct data type has been used in the property…



The next property we shall look at is DateAdded as it allows us to look at some useful code.

Look at the test code for this date property…



The thing about date tests is that we don’t want to be updating the date every time we run the test.

The following line of code gets around this problem.

DateTime TestData = DateTime.Now.Date;

What it does is assign the variable TestData with today’s date taken from DateTime.Now.Date

You will need to create the rest of the tests such that all of the properties are tested and created.

The code for the test class for all properties is as follows…

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Class\_Library;

namespace Testing6

{

[TestClass]

public class tstAddress

{

[TestMethod]

public void InstanceOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//test to see that it exists

Assert.IsNotNull(AnAddress);

}

[TestMethod]

public void ActivePropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

Boolean TestData = true;

//assign the data to the property

AnAddress.Active = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.Active, TestData);

}

[TestMethod]

public void DateAddedPropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

DateTime TestData = DateTime.Now.Date;

//assign the data to the property

AnAddress.DateAdded = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.DateAdded, TestData);

}

[TestMethod]

public void AddressNoPropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

Int32 TestData = 1;

//assign the data to the property

AnAddress.AddressNo = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.AddressNo, TestData);

}

[TestMethod]

public void CountyNoPropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

Int32 TestData = 1;

//assign the data to the property

AnAddress.CountyNo = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.CountyNo, TestData);

}

[TestMethod]

public void HouseNoPropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

string TestData = "21b";

//assign the data to the property

AnAddress.HouseNo = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.HouseNo, TestData);

}

[TestMethod]

public void PostCodePropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

string TestData = "LE1 4AB";

//assign the data to the property

AnAddress.PostCode = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.PostCode, TestData);

}

[TestMethod]

public void StreetPropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

string TestData = "Some Street";

//assign the data to the property

AnAddress.Street = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.Street, TestData);

}

[TestMethod]

public void TownPropertyOK()

{

//create an instance of the class we want to create

clsAddress AnAddress = new clsAddress();

//create some test data to assign to the property

string TestData = "Leicester";

//assign the data to the property

AnAddress.Town = TestData;

//test to see that the two values are the same

Assert.AreEqual(AnAddress.Town, TestData);

}

}

}

To complete this work, you will need to create on test for each attribute in your table schema with an associated property generated from your testing procedure.

Once you have done this make sure you end your Git Hub day by merging your perfect working code to the master repository and delete the unwanted branch.