Creating the Find Method

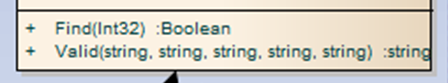
[Creating the Methods 1](#_Toc19266032)

[Adding the Presentation Layer 26](#_Toc19266033)

# Creating the Methods

Having created the testing for the properties of clsAddress we need to think about how to approach building the methods for the class.

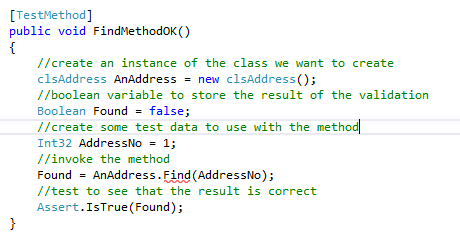
There are two methods in the class Find and Valid.



Let’s have a go at Find first.

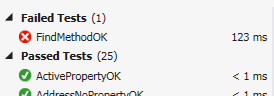
The simple things first, we need to create a test to ensure the existence of the method.

Like so…



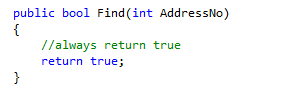
Run the test – watch it fail and then fix it by creating the method.

It should still fail due to the code in the class not being right.



One of the features of TDD is that we are able to put in a quick fix to problems simply to make the test pass.

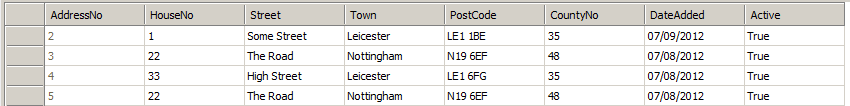
The following code is an example of such a fix.



Modifying the Find method on clsAddress to always return true forces the test to pass but doesn’t exactly fix the problem in the long term.

We need to think a little more about how this method is going to work.

Let’s assume that the table in the database contains the following data…



How the method works is that we will send it the value of a record’s primary key e.g. 2.

If record 2 is found the method will return a value of true, otherwise false.

As well as returning true or false the method also needs to return the data for all fields of the record being searched for.

For example

clsAddress AnAddress = new clsAddress();

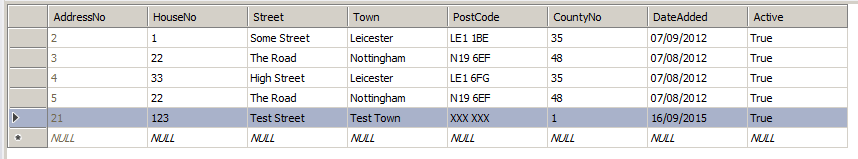
AnAddress.Find(2);

string Street = AnAddress.Street;

Would set the value of the variable Street to “Some Street”.

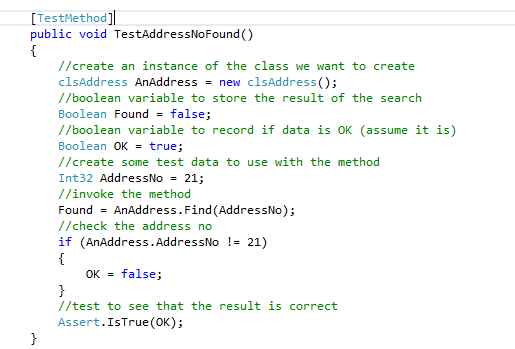
What we will do first is set up some test data in the table that will be used to check that the method is returning the data we think it should be.

Add a new record to the table like so…

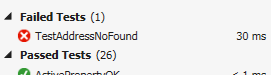


We need to create a series of tests that search for the record and ensure that the correct data is being returned.

The first test is as follows…

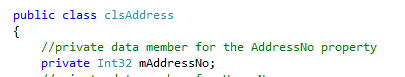


As usual the test should fail…

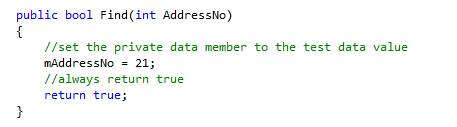


Let’s fix the problem to see if we can resolve the failed test.

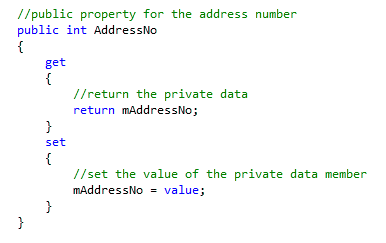
Open the code for clsAddress and add a private data member for the property…



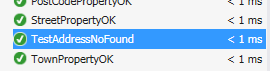
To make this work we will hard code the result into the Find method’s function like so…



Next we need to modify the public AddressNo property so that it returns the value of this private data member…



Let’s try the test again…



The good news is that it passes. The bad news is that the fix is still a bit rubbish as it isn’t drawing its data from the database.

At the moment we don’t care about that!

Let’s get the rest of the tests in place like so…

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Class\_Library;

namespace Test\_Framework

{

[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);

}

[TestMethod]

public void FindMethodOK()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the validation

Boolean Found = false;

//create some test data to use with the method

Int32 AddressNo = 1;

//invoke the method

Found = AnAddress.Find(AddressNo);

//test to see that the result is correct

Assert.IsTrue(Found);

}

[TestMethod]

public void TestAddressNoFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the address no

if (AnAddress.AddressNo != 21)

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestStreetFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the property

if (AnAddress.Street != "Test Street")

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestTownFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the property

if (AnAddress.Town != "Test Town")

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestPostCodeFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the property

if (AnAddress.PostCode != "XXX XXX")

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestCountyNoFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the property

if (AnAddress.CountyNo != 1)

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestDateAddedFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the property

if (AnAddress.DateAdded !=Convert.ToDateTime("16/09/2015"))

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

[TestMethod]

public void TestActiveFound()

{

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

clsAddress AnAddress = new clsAddress();

//boolean variable to store the result of the search

Boolean Found = false;

//boolean variable to record if data is OK (assume it is)

Boolean OK = true;

//create some test data to use with the method

Int32 AddressNo = 21;

//invoke the method

Found = AnAddress.Find(AddressNo);

//check the property

if (AnAddress.Active != true)

{

OK = false;

}

//test to see that the result is correct

Assert.IsTrue(OK);

}

}

}

Here is the code as it stands for the class clsAddress

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Class\_Library

{

public class clsAddress

{

//private data member for the AddressNo property

private Int32 mAddressNo;

//private data member for HouseNo

private string mHouseNo;

//private data member for street

private string mStreet;

//private data member for town

private string mTown;

//private data member for post code

private string mPostCode;

//private data member for county no

private Int32 mCountyNo;

//private date added data member

private DateTime mDateAdded;

//private data member for active

private Boolean mActive;

//public property for active

public bool Active

{

get

{

//return the private data

return mActive;

}

set

{

//set the private data

mActive = value;

}

}

//public property for date added

public DateTime DateAdded

{

get

{

//return the private data

return mDateAdded;

}

set

{

//set the private data

mDateAdded = value;

}

}

//public property for the address number

public int AddressNo

{

get

{

//return the private data

return mAddressNo;

}

set

{

//set the value of the private data member

mAddressNo = value;

}

}

//public property for county no

public int CountyNo

{

get

{

//return the private data

return mCountyNo;

}

set

{

//set the private data

mCountyNo = value;

}

}

//public property for house no

public string HouseNo

{

get

{

//return private data

return mHouseNo;

}

set

{

//set the private data

mHouseNo = value;

}

}

//public property for post code

public string PostCode

{

get

{

//return the private data

return mPostCode;

}

set

{

//set the private data

mPostCode = value;

}

}

//public data member for street

public string Street

{

get

{

//return the private data

return mStreet;

}

set

{

//set the private data

mStreet = value;

}

}

//public data member for Town

public string Town

{

get

{

//return the private data

return mTown;

}

set

{

//set the private data

mTown = value;

}

}

public bool Find(int AddressNo)

{

//set the private data members to the test data value

mAddressNo = 21;

mHouseNo = "123";

mStreet = "Test Street";

mTown = "Test Town";

mPostCode = "XXX XXX";

mCountyNo = 1;

mDateAdded = Convert.ToDateTime("16/9/2015");

mActive = true;

//always return true

return true;

}

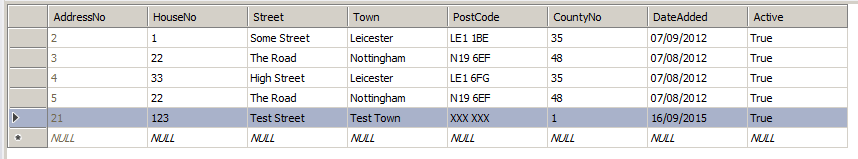
}

}

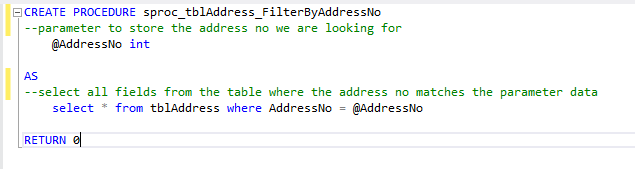
At this stage all tests should pass however there is still one important feature that is missing.

That is we are still not actually drawing the data from the database!

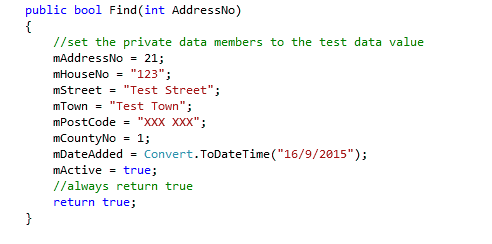
The database contains the following data…



And we will make use of the following stored procedure…



We now need to change the code for the Find method from this…



To this…

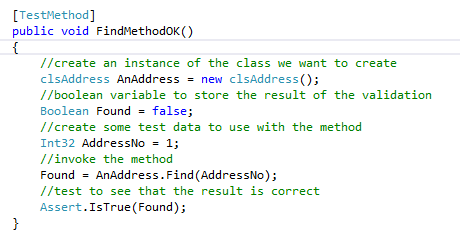


You should get the following results when you run your tests…



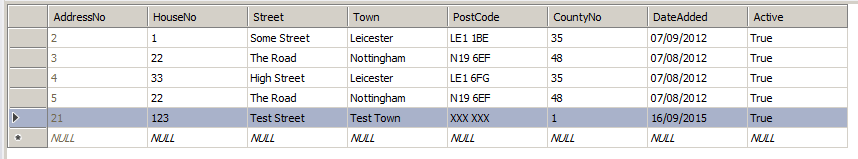
So what is the problem?

Double click the error to see the problem code…



In this test we are testing the find method by searching for AddressNo 1.

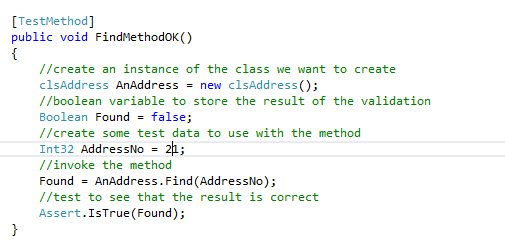
The problem is that in the test data…



There is no record 1!

Clearly we need to ensure that the test data we are using actually exists.

It would make more sense to change this test such that it tests for record 21 the primary key of the test record like so…



## Adding the Presentation Layer