# eCargo test

I have figured out the blank before the output and the asynchronous programming, others I follow your instructions. Therefore, I don’t submit any assumpation for the test.

Context

[eCargo test 1](#_Toc479027788)

[Unit test 3](#_Toc479027789)

[Describer unit test 3](#_Toc479027790)

[Transformer unit test 7](#_Toc479027791)

[Integration test 16](#_Toc479027792)

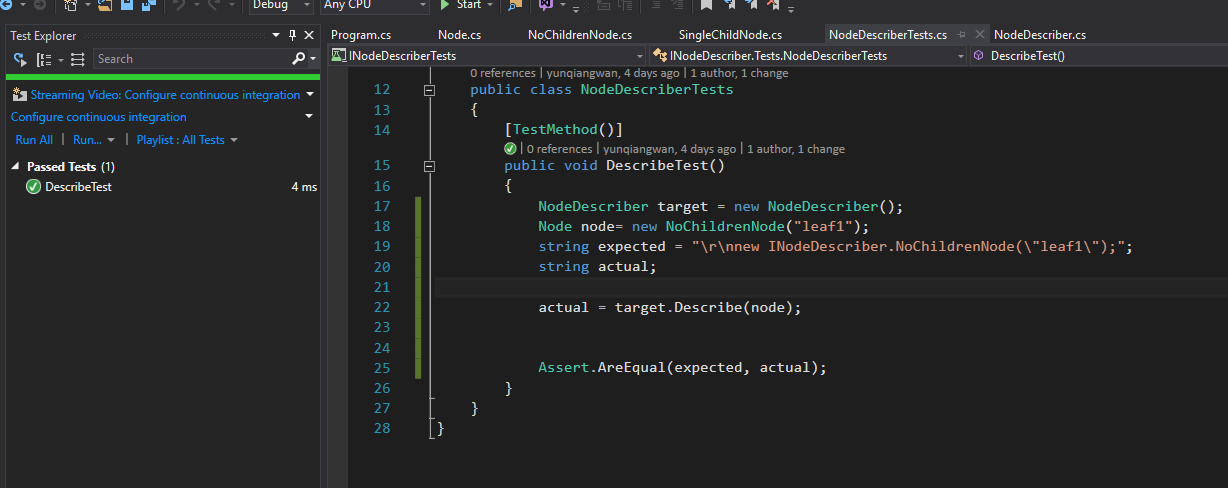
[NodeWriter test 16](#_Toc479027793)

## Unit test

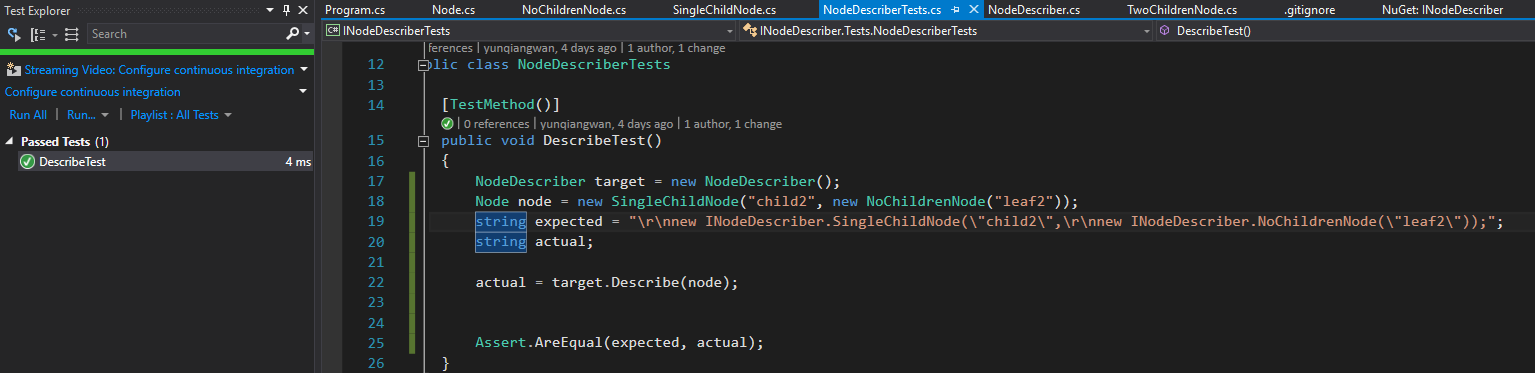
### Describer unit test

#### 1, single objects test

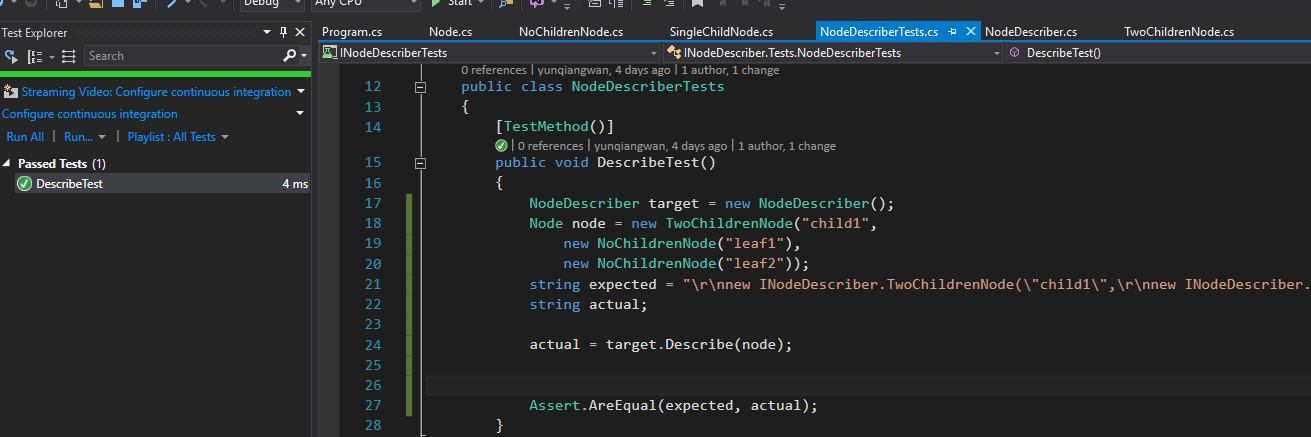
new INodeDescriber.NoChildrenNode(\"leaf1\");



new SingleChildNode("child2", new NoChildrenNode("leaf2"));

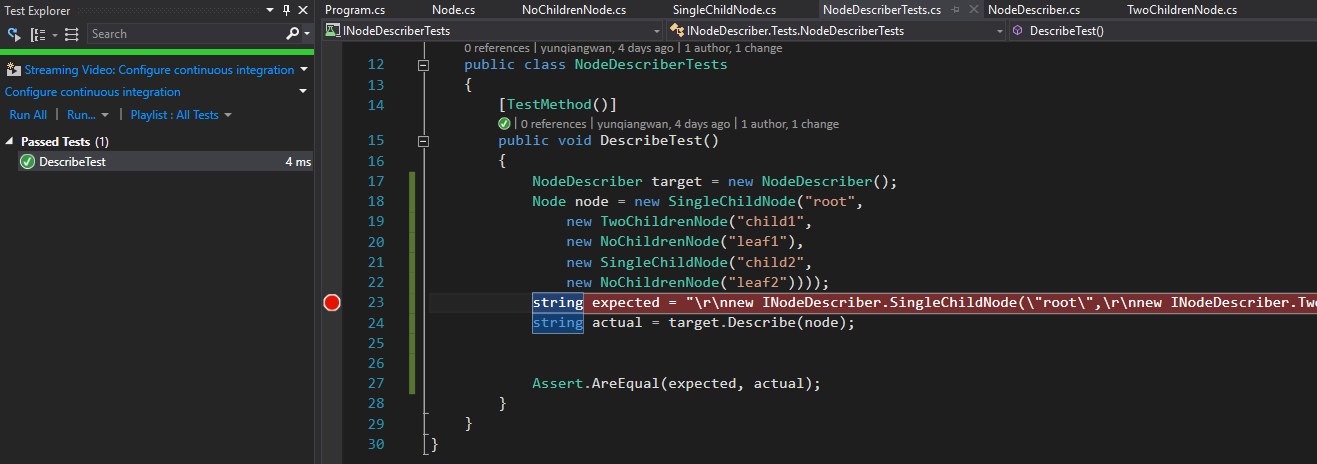


new TwoChildrenNode("child1", new NoChildrenNode("leaf1"), new NoChildrenNode("leaf2"));



#### 2, normal objects test

new SingleChildNode("root", new TwoChildrenNode("child1", new NoChildrenNode("leaf1"), new SingleChildNode("child2", new NoChildrenNode("leaf2"))));



#### 3, with many SingleChildNodes test

new SingleChildNode("root",

new TwoChildrenNode("child1",

new NoChildrenNode("leaf1"),

new SingleChildNode("child2",

new SingleChildNode("child3",

new SingleChildNode("child4",

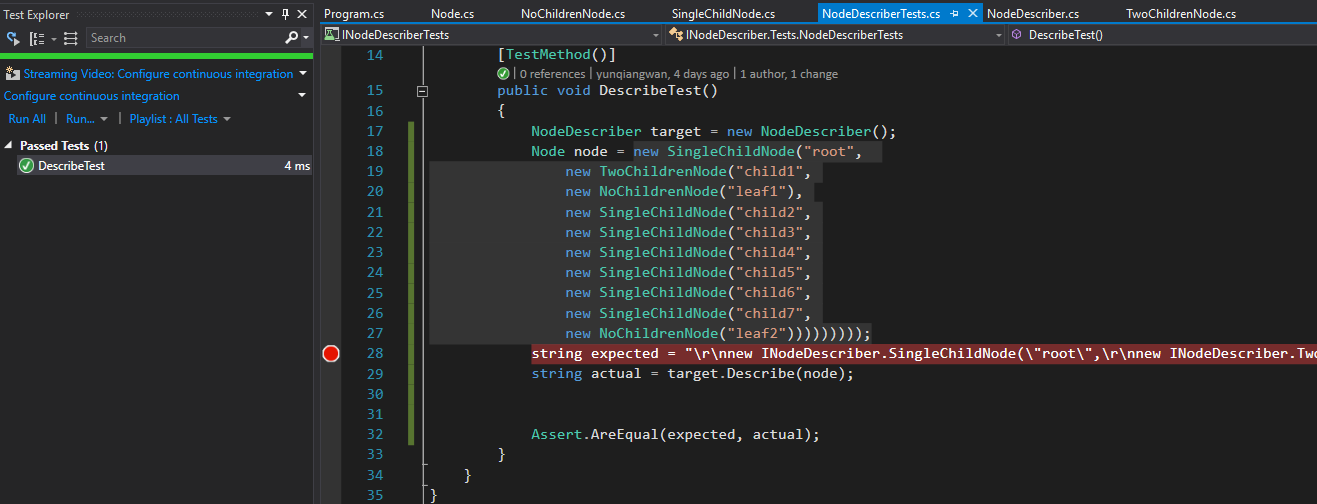
new SingleChildNode("child5",

new SingleChildNode("child6",

new SingleChildNode("child7",

new NoChildrenNode("leaf2")))))))));

"\r\nnew INodeDescriber.SingleChildNode(\"root\",\r\nnew INodeDescriber.TwoChildrenNode(\"child1\",\r\nnew INodeDescriber.NoChildrenNode(\"leaf1\")\r\nnew INodeDescriber.SingleChildNode(\"child2\",\r\nnew INodeDescriber.SingleChildNode(\"child3\",\r\nnew INodeDescriber.SingleChildNode(\"child4\",\r\nnew INodeDescriber.SingleChildNode(\"child5\",\r\nnew INodeDescriber.SingleChildNode(\"child6\",\r\nnew INodeDescriber.SingleChildNode(\"child7\",\r\nnew INodeDescriber.NoChildrenNode(\"leaf2\")))))))));"



#### 4, with many twoChildrenNodes test

new SingleChildNode("root",

new TwoChildrenNode("child1",

new NoChildrenNode("leaf1"),

new TwoChildrenNode("child2",

new NoChildrenNode("leaf2"),

new TwoChildrenNode("child4",

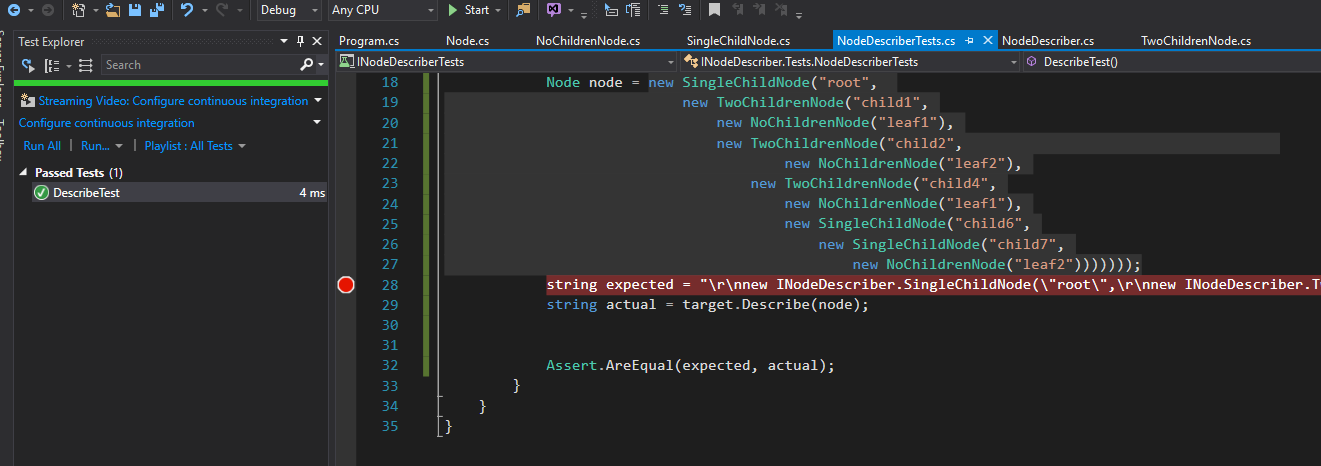
new NoChildrenNode("leaf1"),

new SingleChildNode("child6",

new SingleChildNode("child7",

new NoChildrenNode("leaf2")))))));

"\r\nnew INodeDescriber.SingleChildNode(\"root\",\r\nnew INodeDescriber.TwoChildrenNode(\"child1\",\r\nnew INodeDescriber.NoChildrenNode(\"leaf1\")\r\nnew INodeDescriber.TwoChildrenNode(\"child2\",\r\nnew INodeDescriber.NoChildrenNode(\"leaf2\")\r\nnew INodeDescriber.TwoChildrenNode(\"child4\",\r\nnew INodeDescriber.NoChildrenNode(\"leaf1\")\r\nnew INodeDescriber.SingleChildNode(\"child6\",\r\nnew INodeDescriber.SingleChildNode(\"child7\",\r\nnew INodeDescriber.NoChildrenNode(\"leaf2\")))))));"



### Transformer unit test

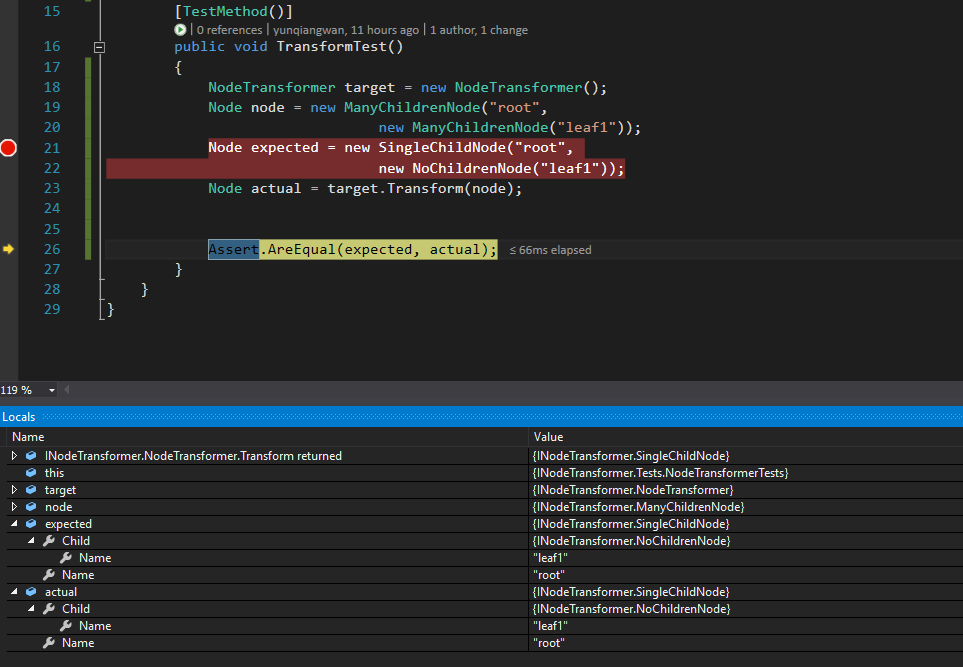
#### 1, single objects test

Node node = new ManyChildrenNode("root",

new ManyChildrenNode("leaf1"));

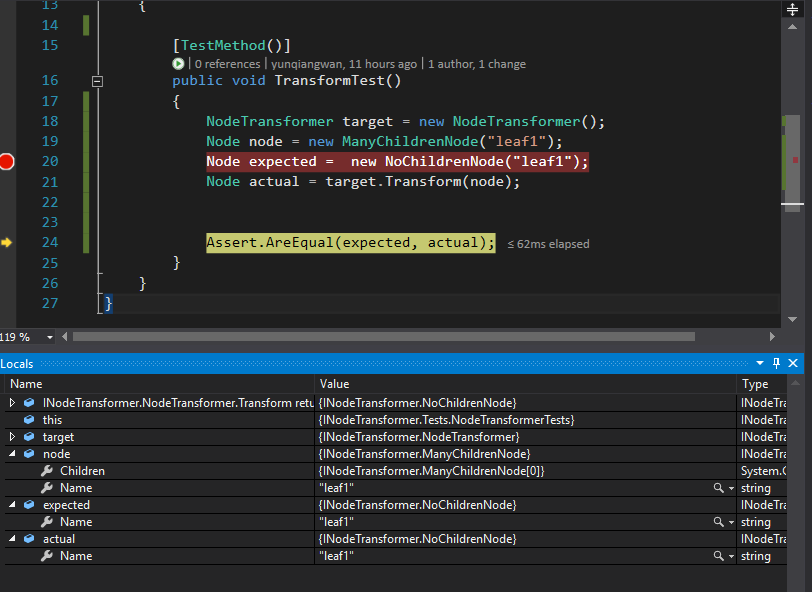
Node expected = new SingleChildNode("root",

new NoChildrenNode("leaf1"));



Node node = new ManyChildrenNode("leaf1");

Node expected = new NoChildrenNode("leaf1");



Node node = new ManyChildrenNode("root",

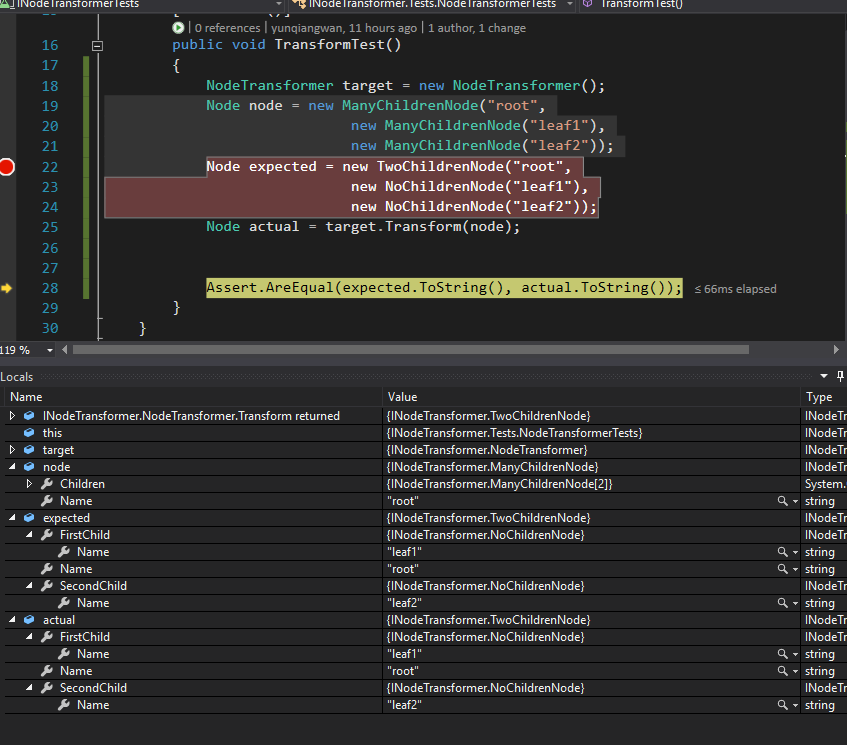
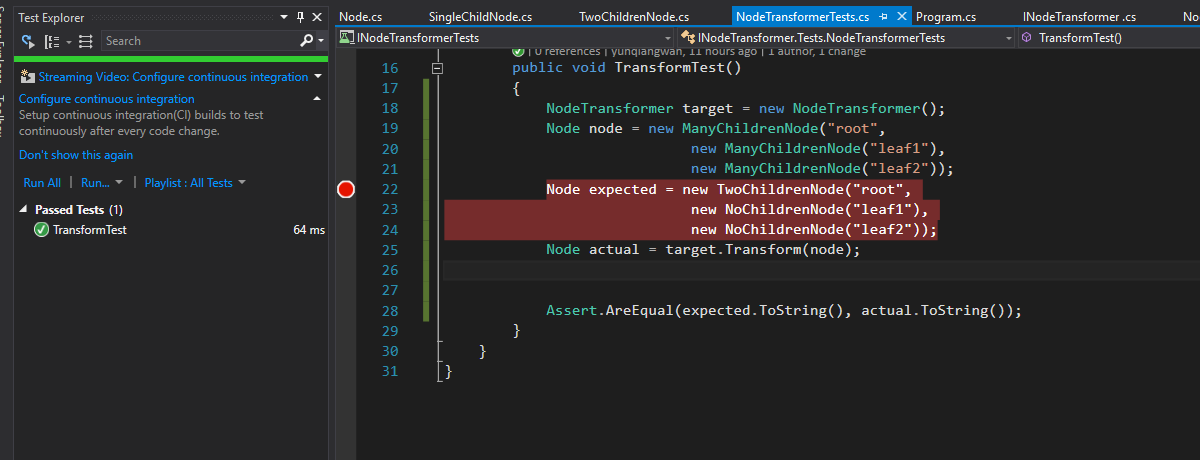
new ManyChildrenNode("leaf1"),

new ManyChildrenNode("leaf2"));

Node expected = new TwoChildrenNode("root",

new NoChildrenNode("leaf1"),

new NoChildrenNode("leaf2"));



#### 2, normal objects test

Node node = new ManyChildrenNode("root",

new ManyChildrenNode("child1",

new ManyChildrenNode("leaf1"),

new ManyChildrenNode("child2",

new ManyChildrenNode("leaf2"),

new ManyChildrenNode("child3",

new ManyChildrenNode("leaf3")))));

Node expected = new SingleChildNode("root",

new TwoChildrenNode("child1",

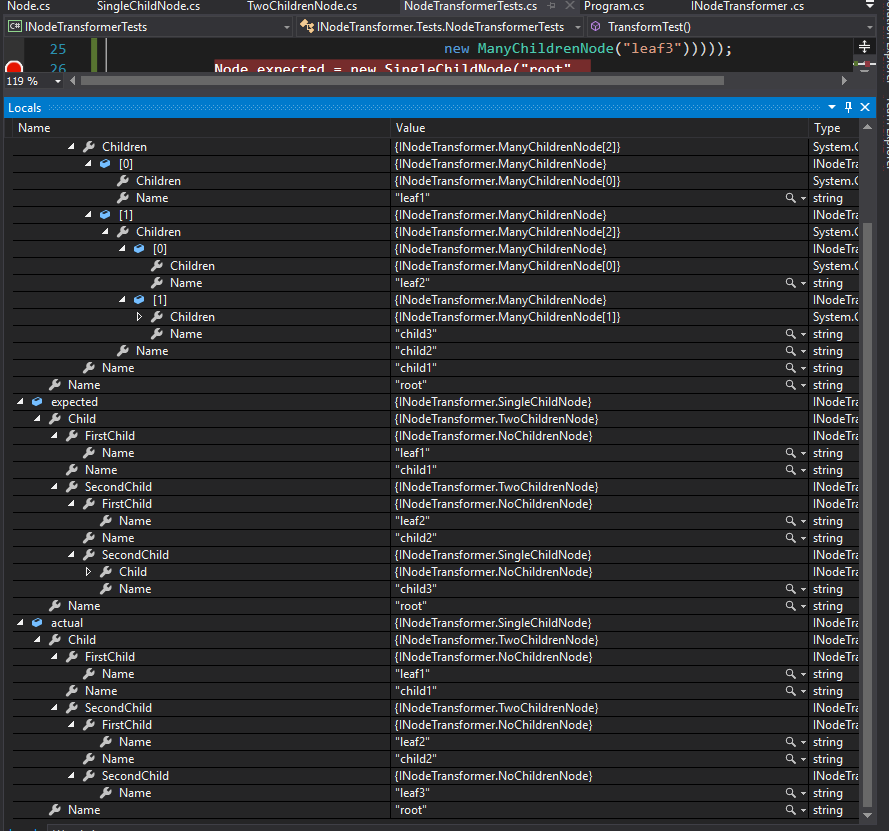
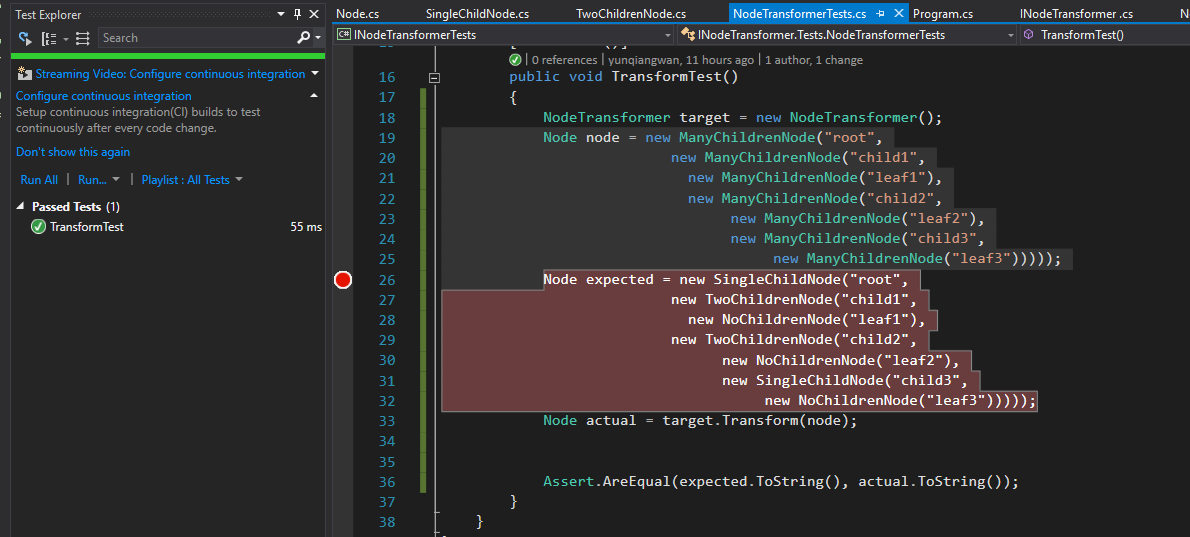
new NoChildrenNode("leaf1"),

new TwoChildrenNode("child2",

new NoChildrenNode("leaf2"),

new SingleChildNode("child3",

new NoChildrenNode("leaf3")))));



#### 3, with many SingleChildNodes test

Node node = new ManyChildrenNode("root",

new ManyChildrenNode("child1",

new ManyChildrenNode("child1",

new ManyChildrenNode("child1",

new ManyChildrenNode("child2",

new ManyChildrenNode("child3",

new ManyChildrenNode("child4",

new ManyChildrenNode("child5",

new ManyChildrenNode("child6",

new ManyChildrenNode("leaf2"),

new ManyChildrenNode("leaf1"))))))))));

Node expected = new SingleChildNode("root",

new SingleChildNode("child1",

new SingleChildNode("child1",

new SingleChildNode("child1",

new SingleChildNode("child2",

new SingleChildNode("child3",

new SingleChildNode("child4",

new SingleChildNode("child5",

new TwoChildrenNode("child6",

new NoChildrenNode("leaf2"),

new NoChildrenNode("leaf1"))))))))));



#### 4, with many twoChildrenNodes test

Node node = new ManyChildrenNode("root",

new ManyChildrenNode("child1",

new ManyChildrenNode("leaf1"),

new ManyChildrenNode("child1",

new ManyChildrenNode("child1",

new ManyChildrenNode("leaf2"),

new ManyChildrenNode("child2",

new ManyChildrenNode("child3",

new ManyChildrenNode("leaf4"),

new ManyChildrenNode("child4",

new ManyChildrenNode("leaf5"),

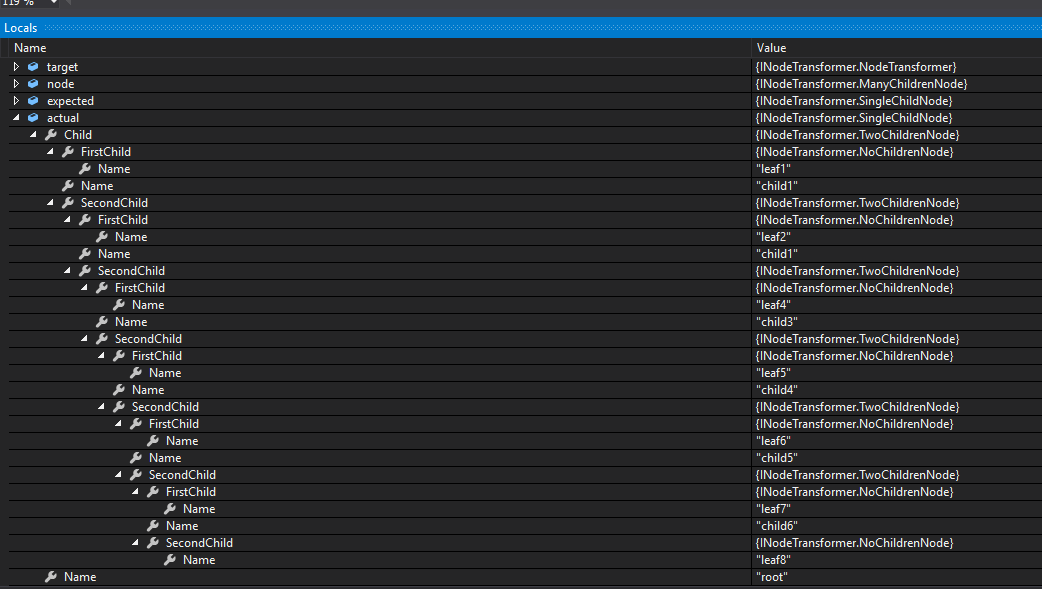
new ManyChildrenNode("child5",

new ManyChildrenNode("leaf6"),

new ManyChildrenNode("child6",

new ManyChildrenNode("leaf7"),

new ManyChildrenNode("leaf8"))))))))));



## Integration test

### NodeWriter test

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Microsoft.Practices.Unity;

using INodeWriter;

using System.IO;

namespace INodeWriterTests

{

[TestClass]

public class IntegrationTest

{

[TestMethod]

public void Intergration()

{

IUnityContainer uContainer = new UnityContainer();

uContainer.RegisterType<INodeWriter.INodeWriter, NodeWriter>();

uContainer.RegisterType<INodeDescriber, NodeDescriber>();

INodeDescriber nodeDescriber = uContainer.Resolve<INodeDescriber>();

INodeWriter.INodeWriter nodeWriter = uContainer.Resolve<INodeWriter.INodeWriter>();

Node node = new SingleChildNode("child1", new NoChildrenNode("leaf1"));

var result = nodeDescriber.Describe(node);

string conf = "..//..//conf.txt";

nodeWriter.WriteToFileAsync(node, conf);

string res = File.ReadAllText(conf);

Assert.IsNotNull(result);

Assert.IsInstanceOfType(node,typeof(SingleChildNode));

Assert.IsFalse(node.Name.Equals("SingleChildNode"));

Assert.AreNotSame(result, res);

Assert.AreNotEqual(result, res);

}

}

}

