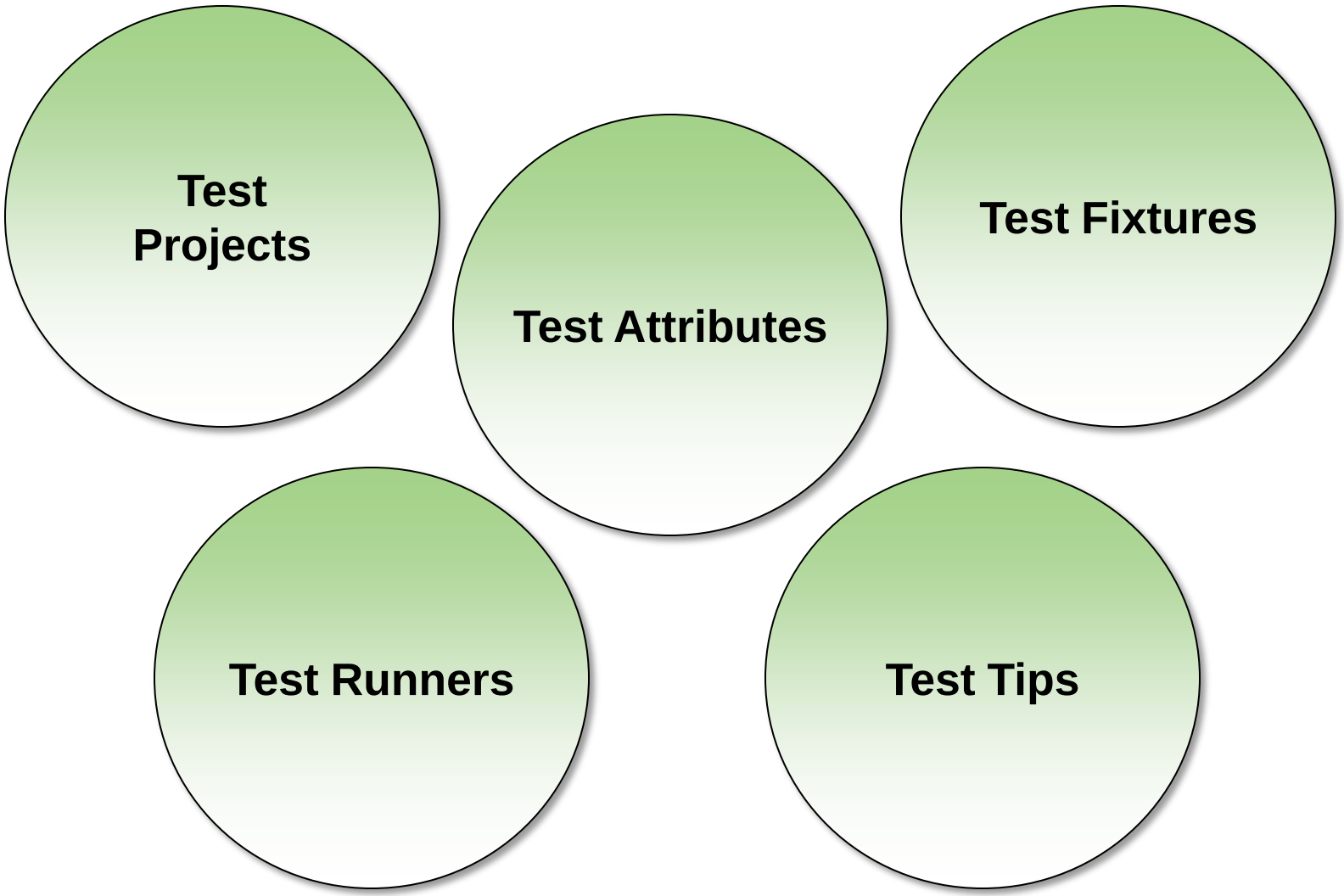
TFD: Writing Unit Tests

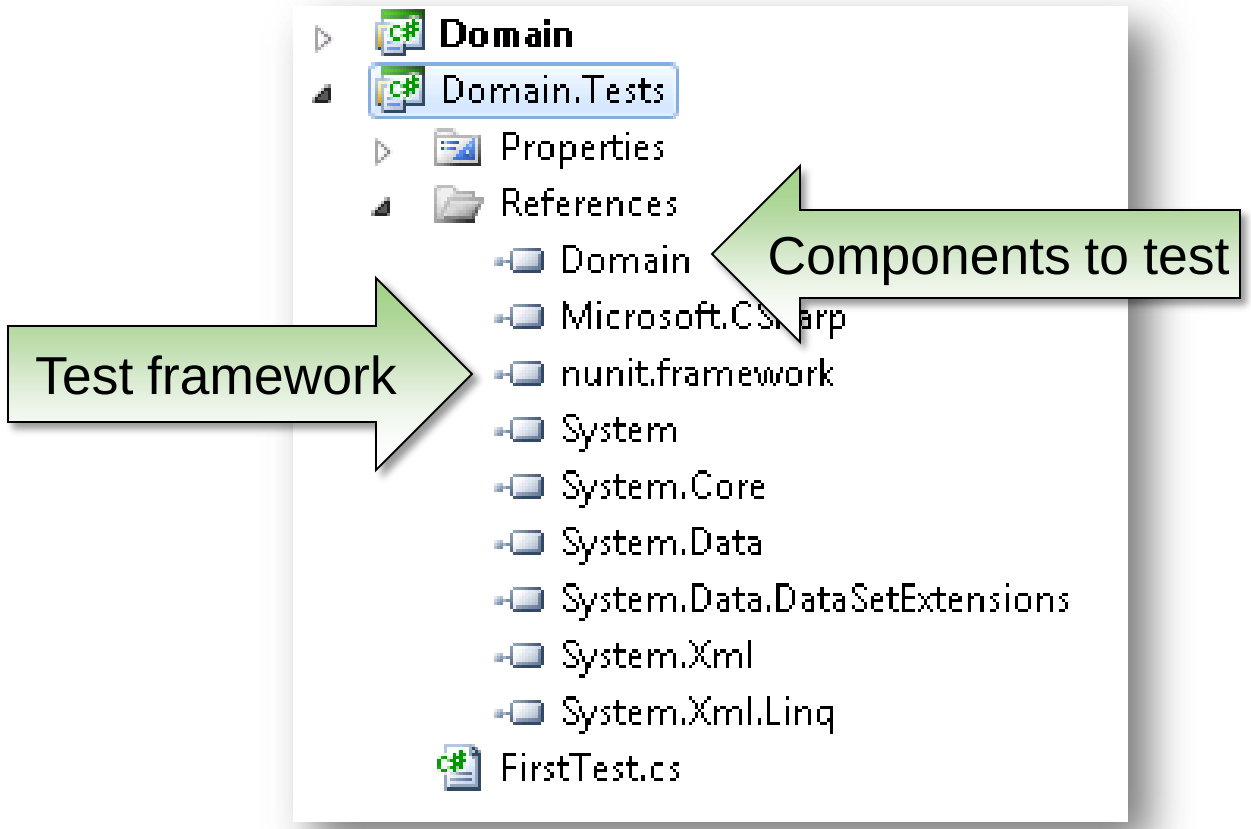
Scott Allen
http://www.pluralsight.com/

Topics


Test Projects
Test Attributes
Test Fixtures
Test Runners
Test Tips

Setting Up A Test Project

- Tests live in a separate class library project



A First Test

Attributes enable unit test functionality

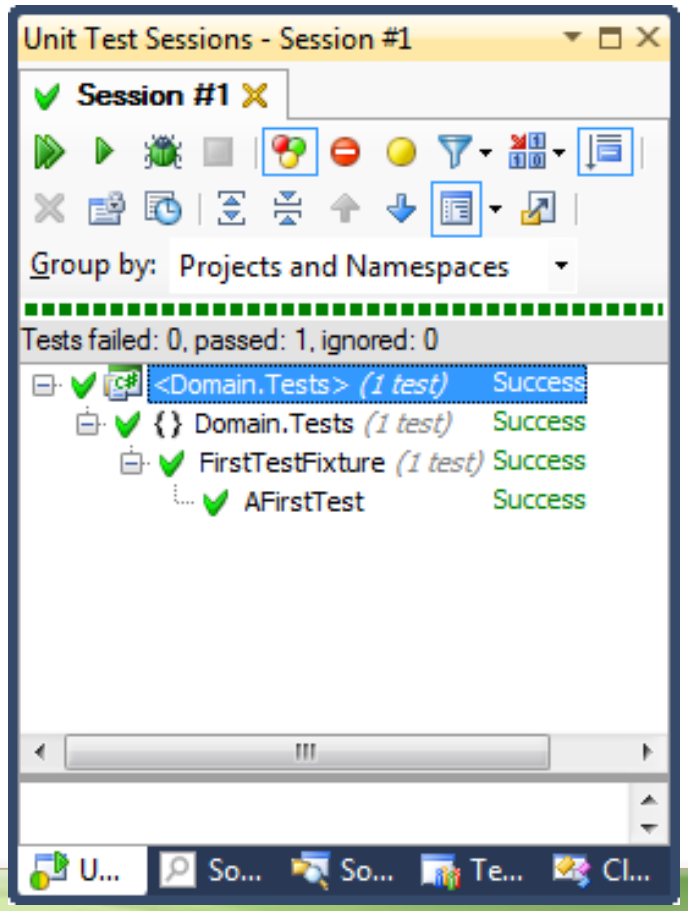
This test passes unless the assert fails

```
using NUnit.Framework;

namespace Domain.Tests
{
    [TestFixture]
    public class FirstTestFixture
    {
        [Test]
        public void AFirstTest()
        {
            Assert.IsTrue(true, “true is true!");
        }
    }
}
```

Running Tests

- **Test runner find, execute, and collect test results**
  - Run all tests or subset of tests
  - Debug into tests



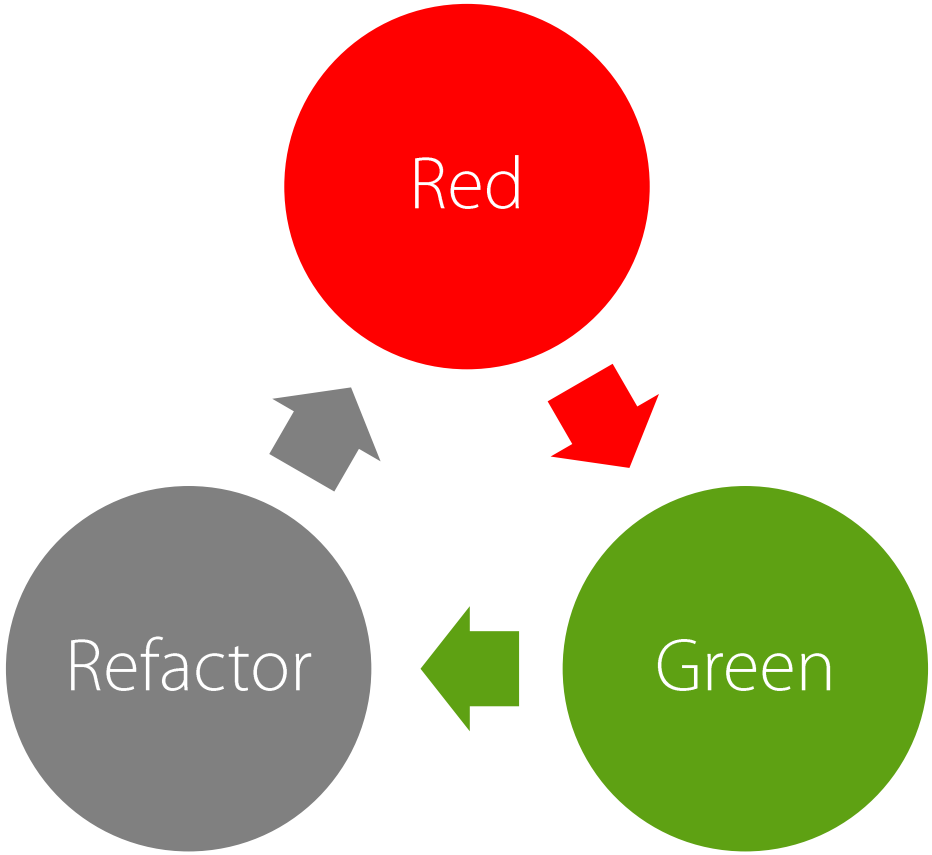
Our TFD Scenario

Take a collection of measurements …

```
public class Measurement
{
    public double HighValue { get; set; }
    public double LowValue { get; set; }
}
```

… and put them in groups of arbitrary size

Test Driven Development



1. **Create a failing test**

   Red

2. **Make the test pass**

   Green

3. **Refactor**

   Improve the internal implementation without changing the external contract or behavior

TFD - Assertions

- **Use one logical assertion per test**
  - Strive for a single statement with a focused Assert

```
[Test]
public void the_order_is_canceled()
{
    var customer = CreateCustomer();
    Assert.IsNotNull(customer);

    customer.PlaceOrder();
    Assert.IsTrue(customer.HasOrder);

    customer.CancelOrder();
    Assert.IsFalse(customer.HasOrder);
}
```

Test Code Is Important, Too

- Keep test code maintainable and DRY.

Test Qualities

Repeatable

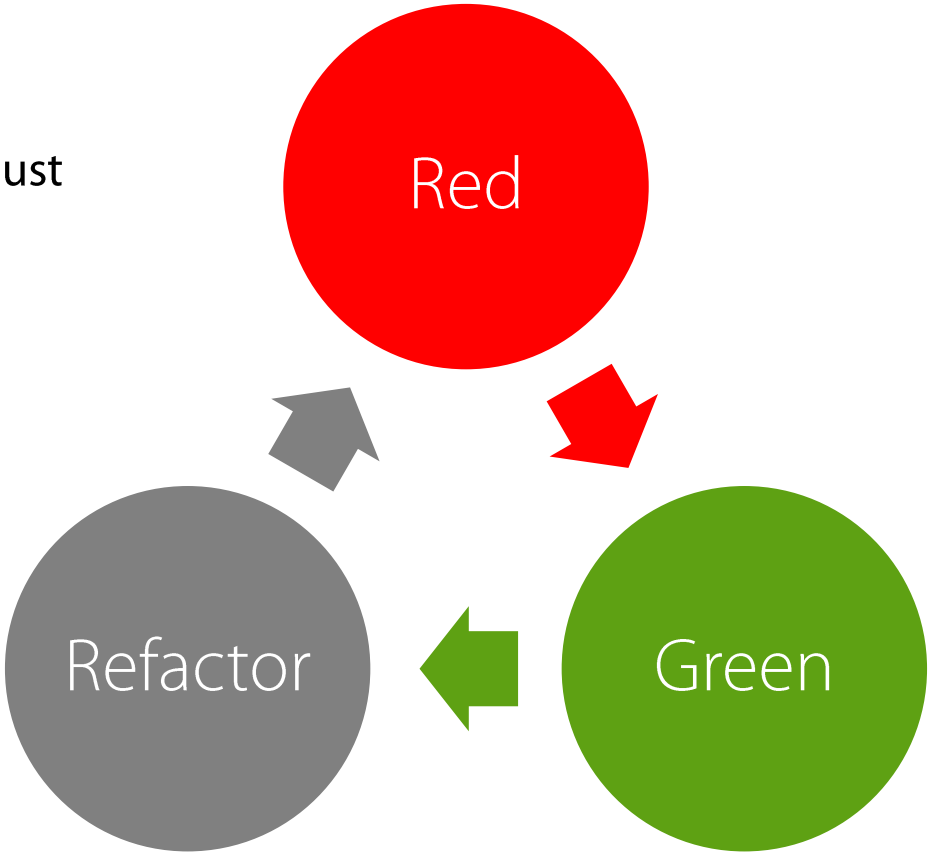
- Tests shouldn't fail after 6 pm

Independent

- Tests shouldn't rely on state from another test

Test First …. Design

- **Test only public members**
  - Makes you think like a client
  - Bonus: make the tests more robust



Summary

Write tests in a separate project

Stick with red-green-refactor

Treat test code with respect

Keep practicing and learning