Xerris Bootcamp Series

Test Automation with XUnit

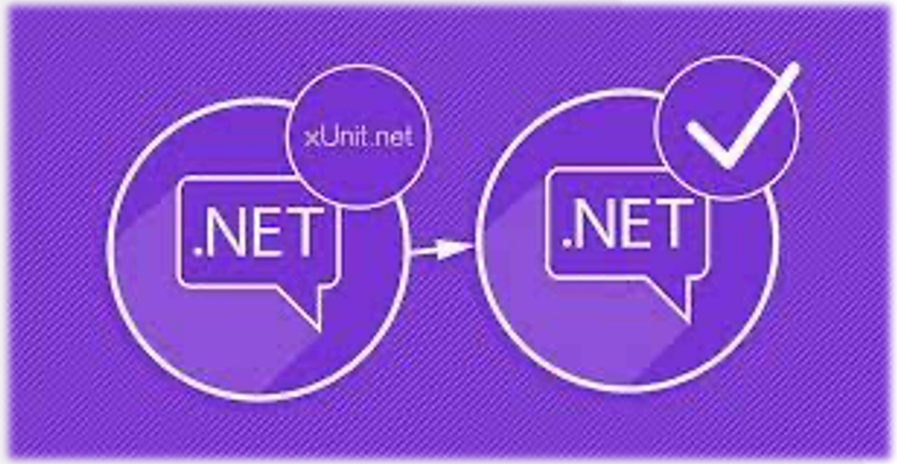
xerris

ACCOLITE DIGITAL
Transforming The Future, Now

What is XUnit?

XUnit

- An open-source framework for DotNet unit testing
- XUnit can be run from the command-line
- Tests can run in Parallel
- Supports shared test fixtures
- Is cross-platform (support for DotNet Core)
- Is installable via NuGet



XUnit – An Open-Source testing framework

A basic XUnit test

```
public class TestClass1   //each class is a test collection
{
    [Fact]                //each test uses the Fact attribute
    public void Test1( )
    {
        Thread.Sleep(500);
    }
}
```

XUnit Test Anatomy

Tests Consist of

- Test Fixture Setup Step (class constructor)
- Testing each scenario with a test method [Fact]
- Test methods are where we invoke the system under test (SUT) method and Assert it produces the correct result(s)
- Test Fixture teardown (Dispose())

```
public class SampleTest : IDisposable
{
    public SampleTest()
    {
        //Test SetUp
    }

    [Fact]
    public void TestMethod_1()
    {
        //Unit test
    }

    [Fact]
    public void TestMethod_2()
    {
        //Unit test
    }

    public void Dispose()
    {
        // test teardown
    }
}
```

XUnit Shared setup/teardown

Tests can share the setup/teardown code.

```
public class StackTests : IDisposable
{
      Stack<int> stack;

      // Create a shared Stack instance
      public StackTests()
      {
          stack = new Stack<int>();
      }

     [Fact]
       ...

     [Fact]
       ...

      //Distroy the shared Stack instance
      public void Dispose() => stack = null;
}
```

XUnit Shared Class Fixture

You can create class Fixtures.

```
public class StackFixture : IDisposable
{
    public Stack<int> Stack { get; set; }

    public StackContext() => Stack = new Stack<int>();

    public void Dispose() => Stack = null;
}
```

When to use: *when you want to create a single test context and share it among all the tests in the class and have it cleaned up after all the tests in the class have finished.*

Test classes can use the shared class fixture

```
public class StackTests : IClassFixture< StackFixture >
{
    private readonly StackContext context;

    public StackTests(StackContext context) =>
        this.context = context;

    [Fact]
    public void AddItem()
    {
        context.Stack. StackPush(1);
        Assert. Single(context.Stack);
    }
}
```

XUnit Shared Collection Fixtures

Collection Fixtures allows you to share across test classes

```
public class ModelFixture : IDisposable
{
    public ModelFixture()
    {
        new CleanModelFactory()
            .Then(new CustomerFactory()).Run();
    }

    public void Dispose() => FactoryGirl.Clear();
}

[CollectionDefinition("Models collection")]
public class StackCollection : ICollectionFixture<ModelFixture>
{
    // This class has no code, and is never created. Its purpose is simply
    // to be the place to apply [CollectionDefinition] and all the
    // ICollectionFixture<> interfaces.
}
```

Test classes use the Collection attribute to identify which Collection Fixture they want to use

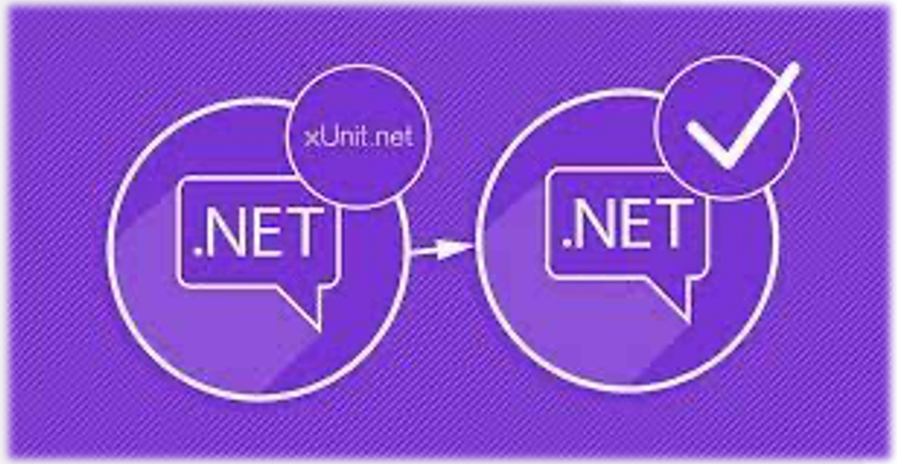
```
[Collection("Models collection")]
public class CustomerTests
{
    [Fact]
    public void TestCollectionDefinition()
    {
        var bobHope = FactoryGirl.Build<Customer>();
        Assert.NotNull(bobHope);
        Assert.Equal("Bob", bobHope.FirstName);
        Assert.Equal("Hope", bobHope.LastName);
    }
}
```

xUnit.net

Test Assertions

Assertions are utility methods to verify conditions in unit tests.

- Assertions an be equality checking
- If an Assertion fails, the test fails.



Verification

Setup any data for a given test scenario

```
public class TestClass1 {
        … setup …
        [Fact]
        public void TestAddCustomer( ) {
            … add customer …
              var foundSanta = customerRepository.Find(“Santa”);

              foundSanda.Should( ).NotBeNull( );
              …
        }
}
```

xerris

Verification

Fluent Assertions

- Open source assertion framework
- Fluent interface makes assertions easy to read

```
1.Should( ).Be(1);

true.Should( ).BeTrue( );

result.Should( ).Be(expectedResult);
```

https://fluentassertions.com/

xerris

XUnit Workshop

XUnit.101 DotNet Project

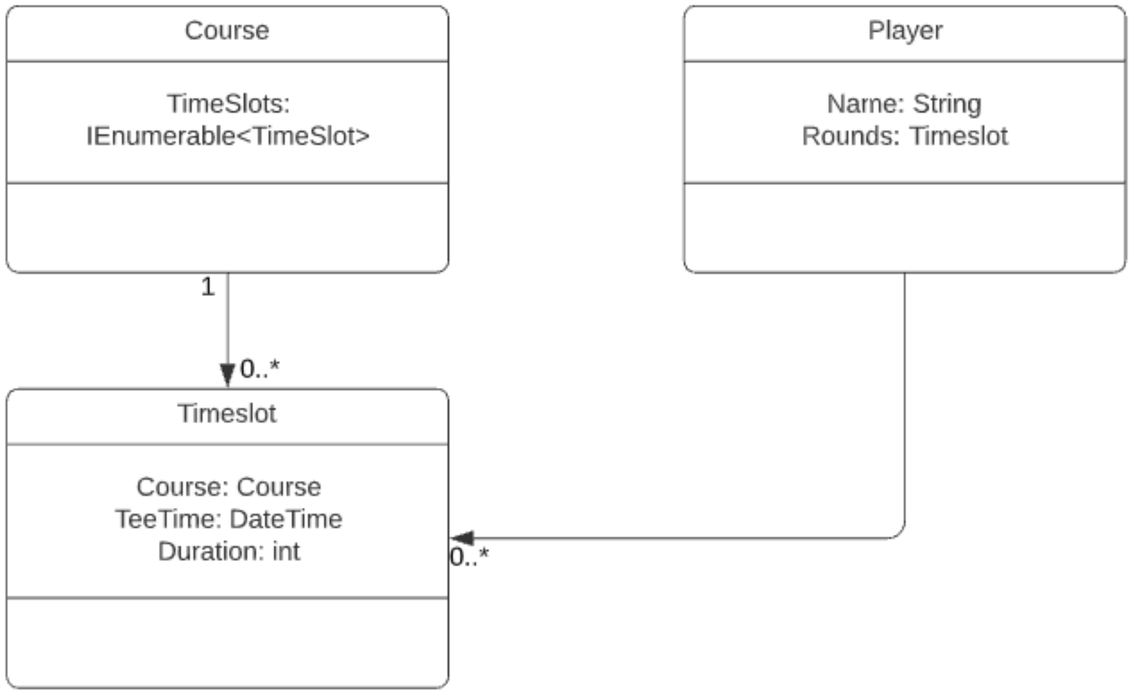
- Load the XUnit.101 project
- Write unit tests for Math.cs
  - Add
  - Subtract
  - Divide
  - Multiply

xerris

Golf Score App

Create a program to:

- Book a Tee time



xerris

XUnit Workshop

Objective

- Add TimeSlots to Course

xerris