TDDBUDDY.COM

Test Naming Guide

Intro

Naming is a hard problem in software development. To maximize the value of the test we write it is important that the given convention of **MethodName_Given{Scenario}_Should{ExpectedResult}** is followed. This convention is preferred because it is easy and quick to understand the failure, group test by functionality and provide a simple yet meaningful approach to test naming.

- MethodName - The method being tested on the object.
- Given{Scenario} – Think about the higher level idea you are checking. Avoid specific technical terms if possible.
- Should{ExpectedResult} – What should the test expected? Allow the read to grok the test without reading the method body.

C# Example

Given the code snippet below for FizzBuzz

```
public class FizzBuzzer
{
    public string FizzBuzz(string input)
    {
        // ...
    }
}
```

some meaningful tests names might be:

- *FizzBuzz_GivenMulitpleOf3_ShouldReturnFizz*
- *FizzBuzz_GivenMulitpleOf5_ShouldReturnBuzz*
- *FizzBuzz_GivenMuliepleOf15_ShouldReturnFizzBuzz*

JavaScript Example

The JavaScript convention is slightly more verbose. The example given is for Jasmine.

```
describe("System Under Test")
        describe("MethodName")
                describe("Secenario") // this can be as simple as 'happy path' or 'sad path'
                        it("ExpectedResult")
```

Given the code snippet below for Fizzbuzz

```
// SUT
function FizzBuzzer(){
        var self = this;

        self.FizzBuzz = function(input){
                // ...
        }
}
```

the spec file could look something like:

TDDBUDDY.COM

```
// Spec file
describe("FizzBuzzer"){
	describe("FizzBuzz"){
		describe("When given mulitple of 3"){
			it("should return Fizz when 6"){
				// ...
			}
		describe("When given mulitple of 5")
			it("should return Buzz when 10"){
				// ...
			}
		describe("When given mulitple of 15")
			it("should return FizzBuzz when 30"){
				// ...
			}
	}
}
```