

xUnit Cheat Sheet

Test

```
[Fact]
public void Test() {}
```

Setup

```
public class TestFixture {
    public TestFixture() {
        // Setup here
    }
}
```

Teardown

```
public class TestFixture : IDisposable {
    public void Dispose() {
        // Teardown here
    }
}
```

Fixtures

```
public class TestFixture : IUseFixture<Context>/*,
    IUseFixture<...> */{
    public void SetFixture(Context data) {
        // i.e. assign to field
    }
}
// Instantiated once per test class
public class Context: IDisposable {
    public Context() {
        // Setup context here
    }
    public void Dispose() {
        // Teardown context here
    }
}
```

Static row tests

```
[Theory]
[InlineData(..., ...)]
...
[InlineData(..., ...)]
public void Test(..., ...) {
    // Parameters must match in type and order
}
```

Dynamic row tests: Excel as DataProvider

```
[Theory]
[ExcelData(@"Path/Data.xls", "select * from NamedRange")]
public void Test(string value, int expected) {
    // ExcelSheet must have range with name NamedRange
}
```

Dynamic row tests: Property as DataProvider

```
public static IEnumerable<object[]> NameOfProperty {
    get {
        yield return new object[] { ..., ... };
        ...
        yield return new object[] { ..., ... };
    }
}

[Theory]
[PropertyData("NameOfProperty")]
public void Test(..., ...) {
    // Parameters must match in type and order
}
```

Dynamic row tests: Class as DataProvider

```
[Theory]
[ClassData(typeof(TestDataProvider))]
public void Test(string value, int expected) {
    // Parameters must match in type and order
}
public class TestDataProvider : IEnumerable<object[]> {
}
```

Dynamic row tests: SqlServer as DataProvider

```
[Theory]
[SqlServerData("serverName", "dbName",
    "user", "pwd"
    "select value, expected...")]
public void Test(string value, int expected) {
    // Select must include parameters in type and order
}
```

Dynamic row tests: OleDbConnection as DataProvider

```
[Theory]
[OleDbData("connection", "select value, expected...")]
public void Test(string value, int expected) {
    // Select must include parameters in type and order
}
```

Dynamic row tests: IDataAdapter as DataProvider

```
public sealed YourSourceDataAttribute :
    DataAdapterDataAttribute {
    // Assign DataAdapter property with your custom
    IDataAdapter
}
```

Dynamic row tests: AnySource as DataProvider

```
public sealed YourSourceDataAttribute :
    DataAdapterAttribute {
    public override IEnumerable<object[]> GetData(...) {
        // Connect to your source of happiness
    }
}
```

Advice:

Use fluent assertions for asserts
fluentassertions.codeplex.com