TDD

We do it not because we want to test the software. We do it because it’s the way of development. It’s the way how a complex software can be built. Every complex problem can be modeled by small and simple steps. That’s what TDD in meant to.

Never start by complicated/complex objects.

We usually start development from the simplest object. Which object exactly? Usually, the object that has no dependency to other objects, e.g. value objects.

Start with happy path scenarios:

Happy path scenarios usually refer to 3 categories of scenarios including:

. construing the SUT with only required references

. constructing the SUT with both required and optional references

. constructing with in limitation of boundaries

Then we follow the development journey with focusing on exceptional scenarios including:

. boundary limitation

. business invariants and rules

Test Builder

We usually create a test builder for every SUT. One the differences between a test builder and a production builder is that every instance of a test builder initiates by valid required references. This is very useful technique help us to have high maintainable tests.

How is relation of the test builder and production object?

Usually, all production objects are inherited from an abstract/interface, so we simply inherit the test builder from the SUT abstract/interface.

interface ISut

{

String Id;

}

Class Sut : ISut

{

Sut(string id)

{

Id = id;

}

String Id;

}

Class SutTestBuilder : ISut

{

SutTestBuilder (string id)

{

Id = “some id”;

}

String Id;

SutTestBuilder WithId(string value)

{

Id = value;

}

Sut Build()

{

Return this;

}

}