Unit test

ساخت نرم افزار مورد انتظار کاربران و خروجی با کیفیت تری تحویل بدهیم.

تست تمیز سبب ایجاد یک داکیومنت کامل از نرم افزار خود می شود.

حالت های مختلف متد خود را بررسی می کنیم.

اگر کدی را تغییر بدهیم و جای دیگه را تحت تاثیر قرار دهد متوجه می شویم و اصلاح می کنیم.

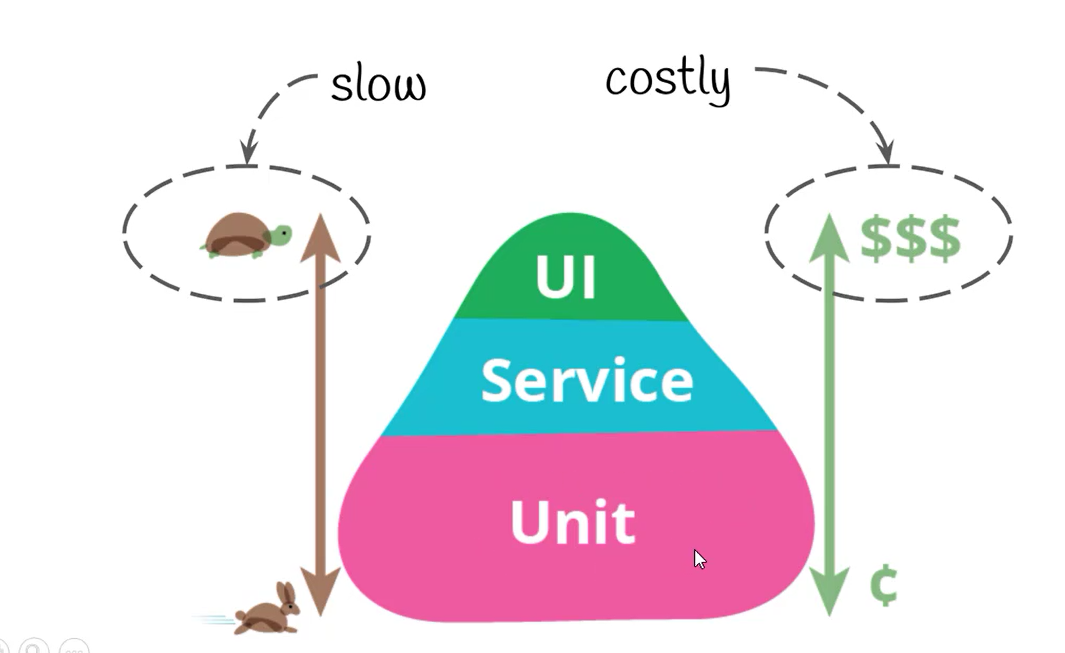
نتیجه تست را سریع و راحت به ما نمایش می دهد.

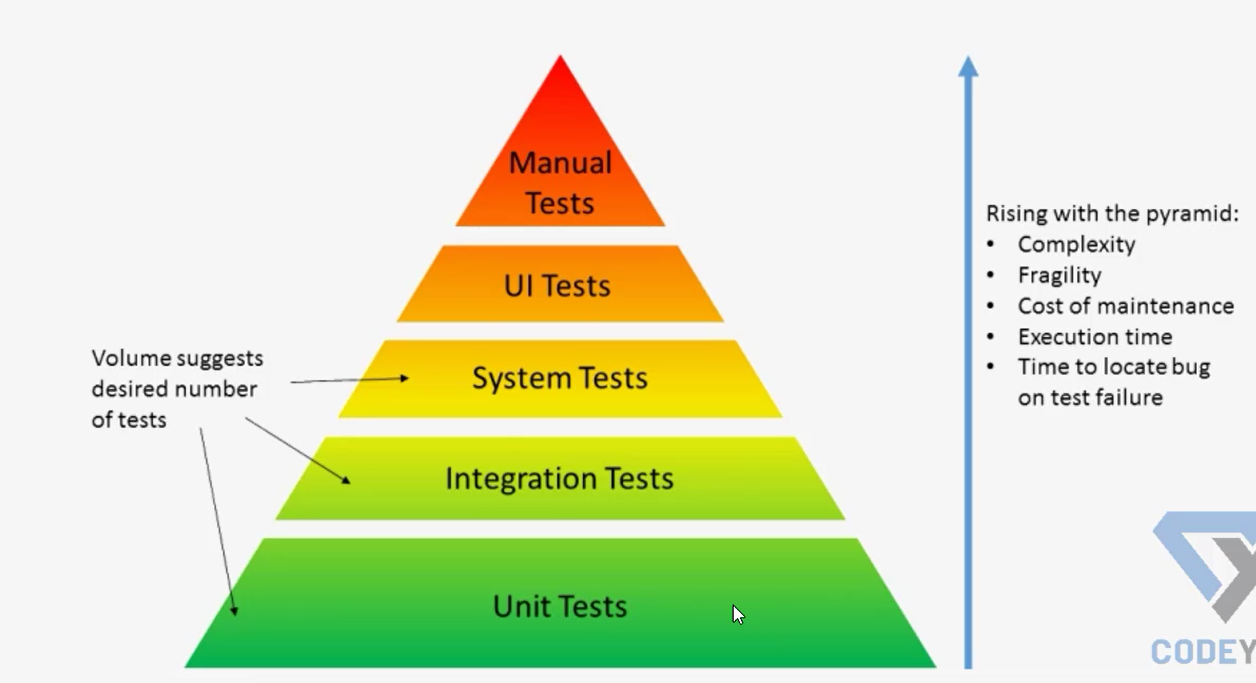
زمان و هزینه را در دراز مدت کاهش می دهد.

unitTest یک واحد کوچک از نرم افزار را تست می کند.

Integrated test مثلا برای زمانی که یک سرویسی تست شود و کار دیتابیسی نیاز است و دیتا فچ شود.

Ui تست نیز از endpoint شروع می کند و تا انتهای عملیات را تست می کند





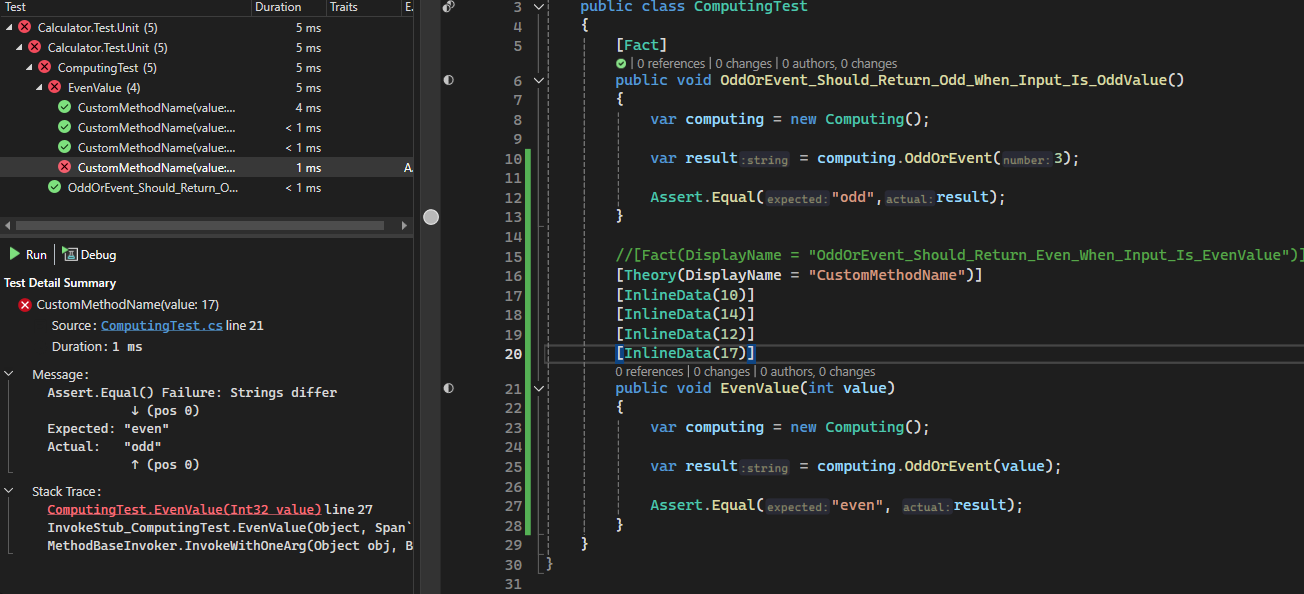
بیشتر تست ها از نوع unit است و هر جه بالا برویم هزینه و پیچیدگی اضافه می شود.

فریم ورک برای تست نوشتن در سی شارپ xunit است و در قدیم nunit بوده است. و اجرای تست ها توسط vs انجام می شود. Visual studio test runner

نکته : اسم متد در xunit باید بیان گر اسم متدی باشد که برای آن تست می خواهیم بنویسیم.

نکته : زمانی که تست خود را کامل کردیم می توانیم تنیجه ای که انتظار داریم برای پاس شدن تست را بدهیم و اگر درست باشد تست pass می شود

Assert.Equal(“Odd”,result);



Ncrunch

یک testRunner به صورت live است که بعد از بروز تغییرات به صورت اتوماتیک تست را انجام می دهد.

قسمت های از کد که بیشتر از حد معمول زمان میبرد را علامت زرد می زند.

Live unit testing

کلیک راست در پروژه و انتخاب گزینه Live Unit testing و آن را پلی کرده. با هر بار ذخیره کردن تست را انجام می دهد مجدد با هر تغییر.

[Fact]

برای تست های ساده و ثابت کاربرد دارد که هیچ پارامتری ندارد و به یک صورت اجرا می شودparameter less

[Theory]

برای تعریف تست های که با داده های مختلف سرکار دارد استفاده می شود و زمانی که متد شما parameter ورودی دارد و اگر داده تعریف نکنید خطا می دهد

Fluent Assertion

فریم ورکی برای افزایش خوانایی کد به جای استفاده از Assert.Equal می توانیم از متد های این کتابخانه استفاده کنیم به طور مثال

Result.Should().Be();

Should.Equal();

Should.BeTypeOf();

**نکته** برای تست Exception

var result = new Action(() =>

{

computing.CalculateAge(0, 2025);

});

چهار فاز اصلی هر تست

Setup fixture - arrange

زمانی که داریم بستر را برای تست آماده می کنیم یا system under test - sut

Exercise - act

عملیاتی که لازم است را روی بستر خود یا sut انجام می دهیم

Verify - asserts

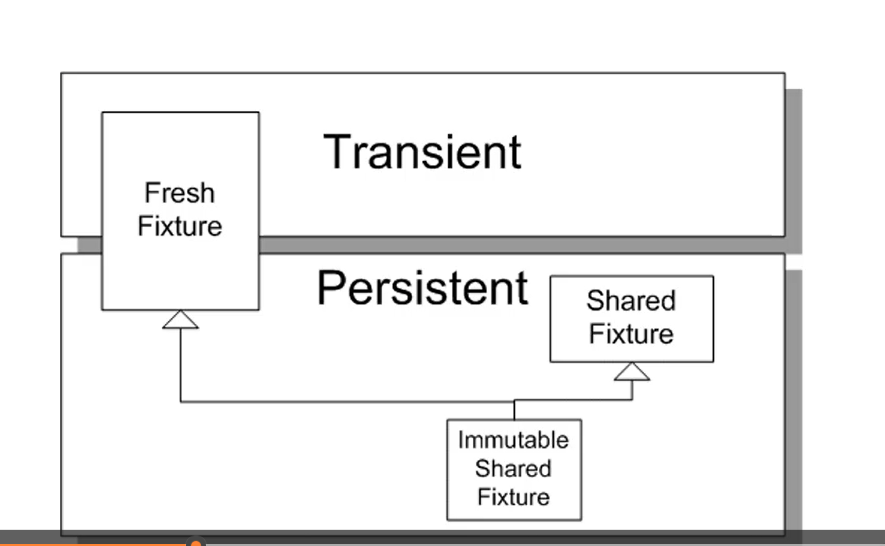
بررسی می کنیم که خواسته های ما را برآورده کرده است یا نه

Teardown

زمانی که بخواهیم یک سری اثر های جانبی تست های خود را خنثی کنیم

مثلا یک یوزر در تست به دی بی اضافه شده که لازم است بعدا حذف شود.

2 بخش اصلی Fixture = Transient and persistent



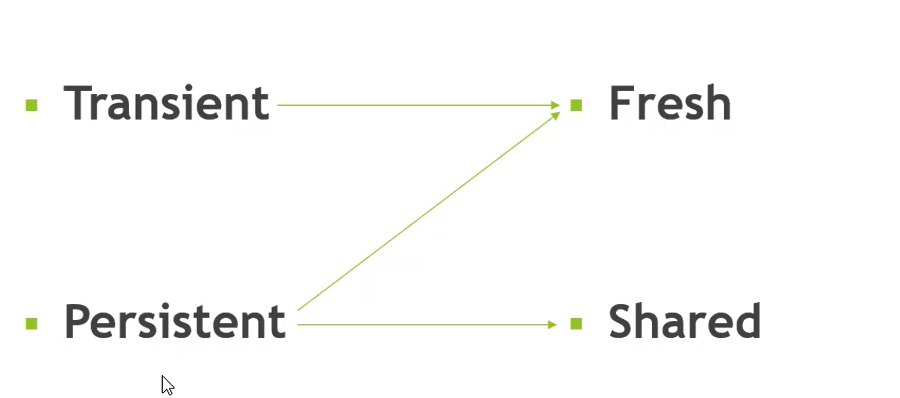
Transient

برای یک تست ساخته می شود و از بین می رود

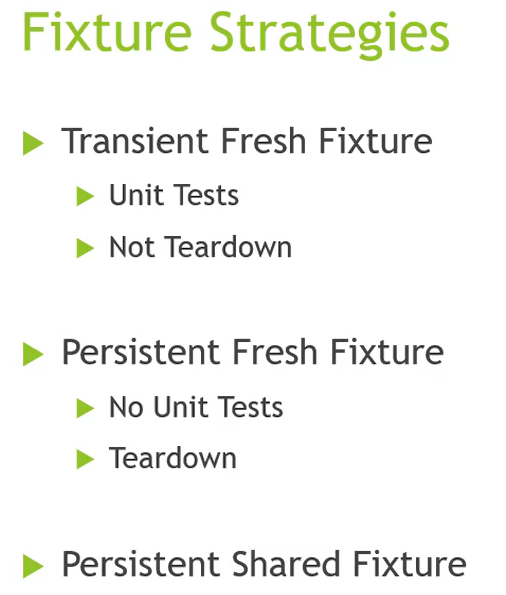
Persistent

چیز های که در دیتابیس هستند و ذخیره می شوند و می مانند در سیستم

استراتژی های Fixture



انوع حالت ها و ترکیب آن ها در ادامه می بینیم.



انواع روش های setup fixture

Inline

با دلیگیت در خود متد new کرده و استفاده می کنیم.

Implicit

توسط فریم ورک هوک ها ساخته می شود مثل Xunit , nunit , junit

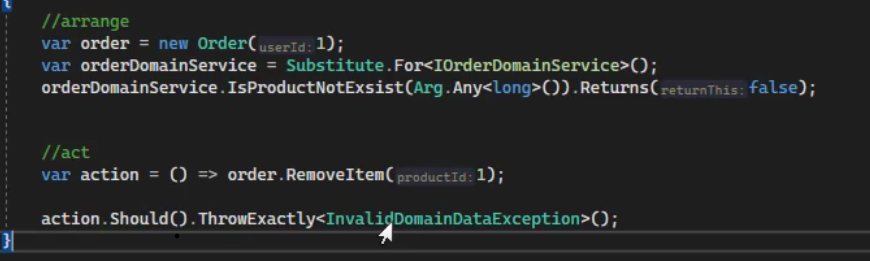


### NSubstitute چیست؟

**NSubstitute** یک کتابخانه‌ی mocking (ماکینگ) برای تست واحد (Unit Testing) در دات‌نت است. وقتی شما دارید تست واحد می‌نویسید، ممکن است بخواهید رفتار بخشی از کد را شبیه‌سازی کنید (mock کنید) تا بتوانید قسمت موردنظر را به‌صورت جداگانه تست کنید. NSubstitute کمک می‌کند تا به‌سادگی اشیاء جایگزین (substitute) برای وابستگی‌ها بسازید و رفتارهای مختلفی را شبیه‌سازی کنید.

### کاربردها و ویژگی‌ها:

* ایجاد آسان اشیاء mock بدون نیاز به تنظیمات پیچیده
* پشتیبانی از استاب (Stub)، ماک (Mock)، و اسپای (Spy)
* قابل استفاده با اکثر فریمورک‌های تست مثل NUnit، xUnit و MSTest
* نوشتن کد تست خواناتر و ساده‌تر



Anonymous method

Action<int, int> add = (a, b) => Console.WriteLine(a + b);

add(5, 3); // خروجی: 8

Action<int, int> add = delegate(int a, int b)

{

Console.WriteLine(a + b);

};

add(5, 3); // خروجی: 8

