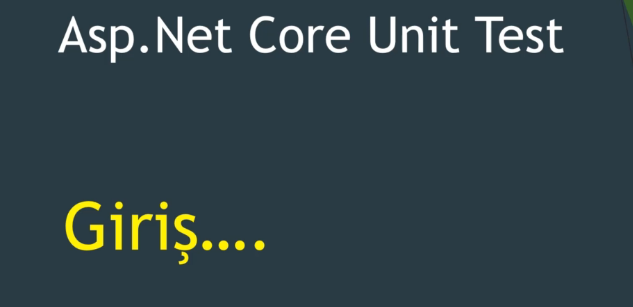
# Unit Test Yazma-Asp.Net Core MVC/API(Sıfırdan)

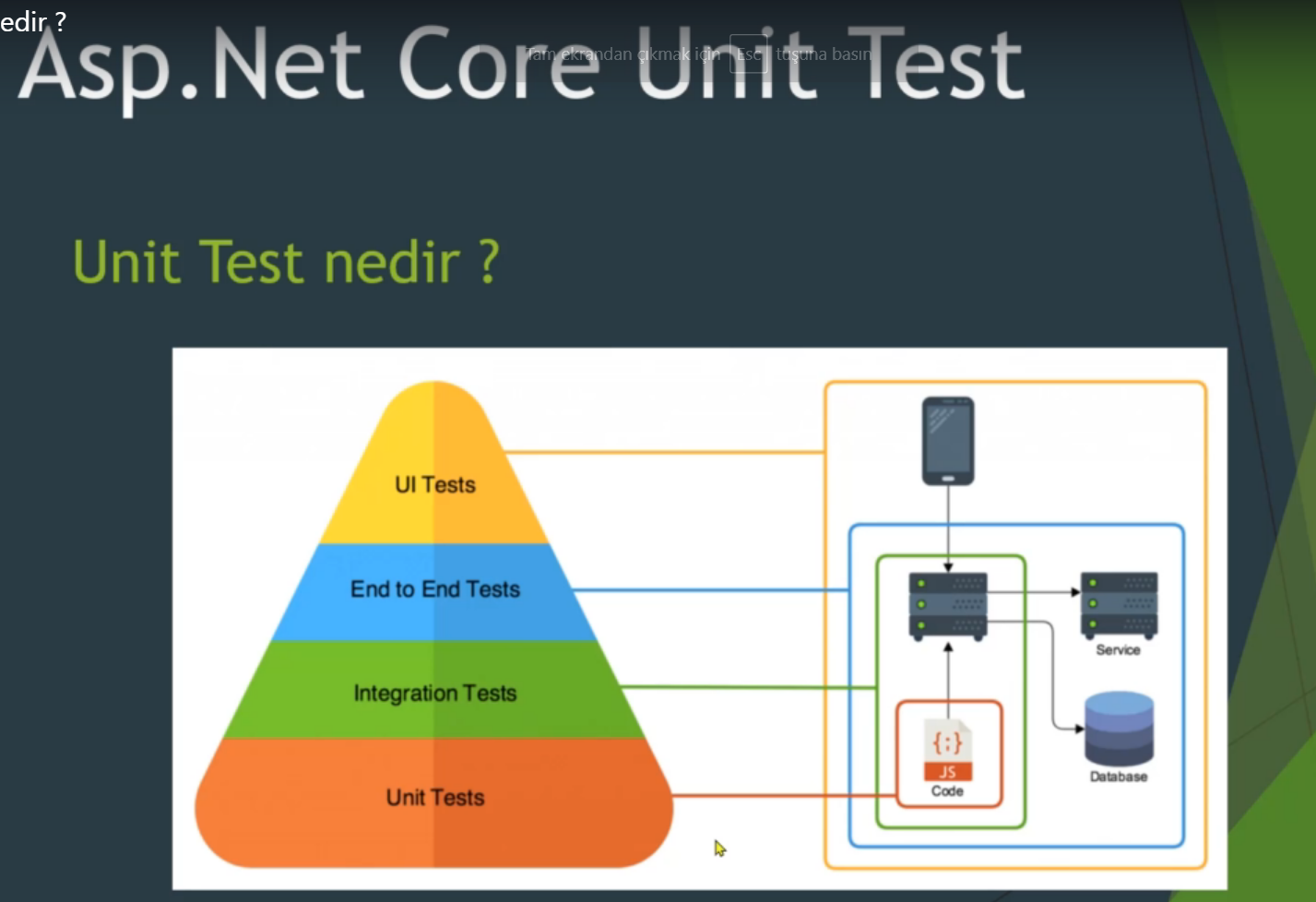
## Section 1: Giriş

### 1. Giriş

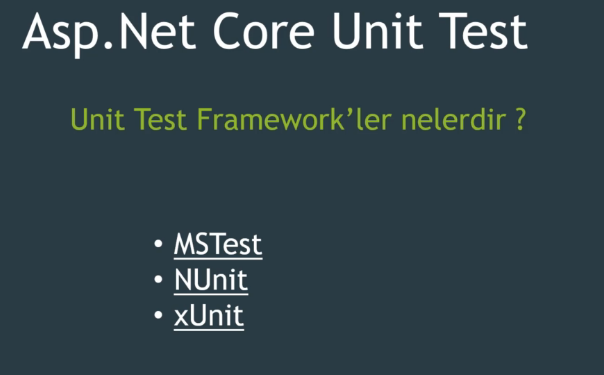


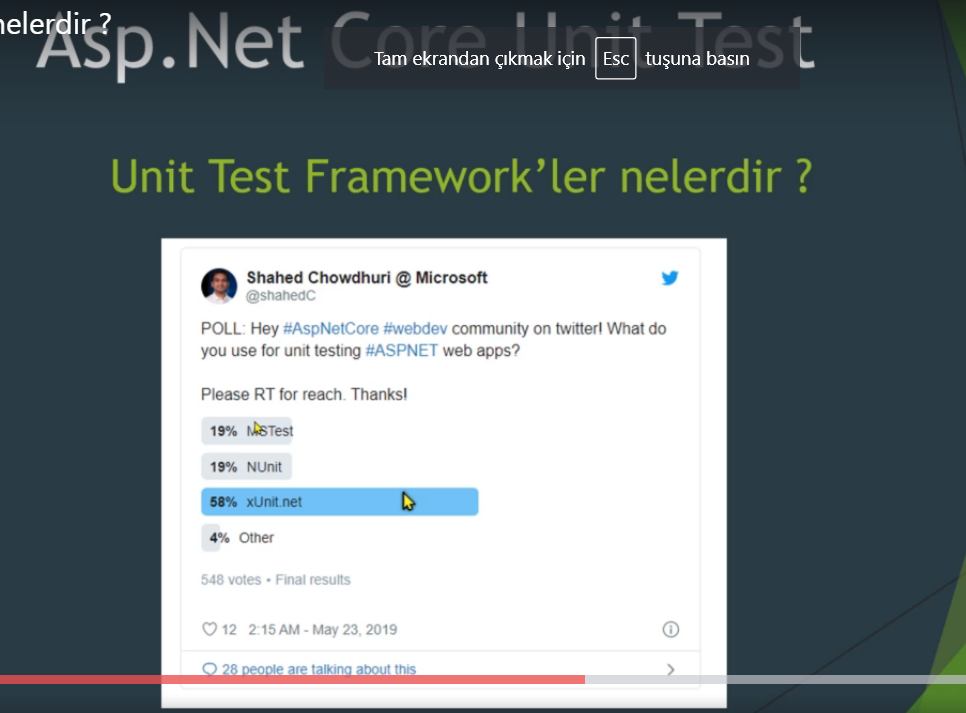
## Section 2: Xunit Framework

### 2. Unit Test nedir ?

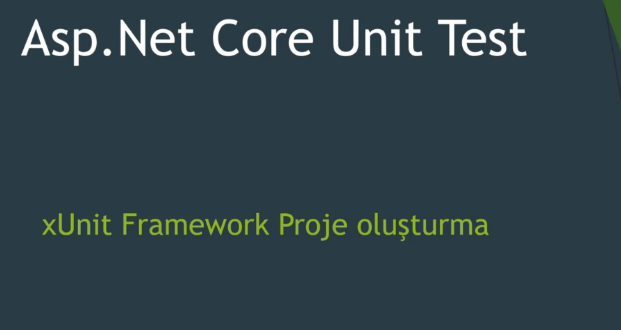


### 3. Unit Test Framework'leri nelerdir ?

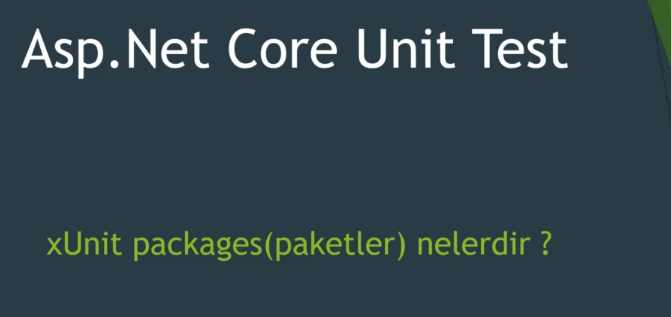




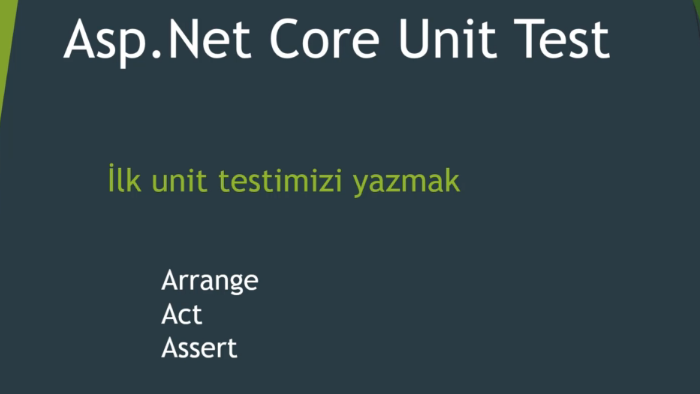
### 4. xUnit Test proje oluşturma



### 5. xUnit packages(paketler) nelerdir ?



### 6. İlk unit testimizi yazmak



**[Fact]** — öznitelik, yazmış olduğumuz metodun test çalıştırıcısı tarafından yürütülmesi gerektiğini belirtir.  
**[Teori]** — özellik, test kodumuza bazı parametreler göndereceğimizi ifade eder. test metoduna parametreler göndereceğimizi ima eder  
**[InlineData]**— özniteliği, test yöntemine gönderdiğimiz parametreleri sağlar. [Teori] özelliğini kullanıyorsanız, [InlineData] öğesini de kullanmalıyız

public class CalculatorTest

{

[Fact]

public void AddTest()

{

// Arrange -> değişlenlerin initialize edildiği yerdir

int a = 5;

int b = 20;

var calculator = new Calculator();

// Act ->

var total = calculator.Add(a, b);

// Assert -> doğrulama evresi

Assert.Equal<int>(25, total);

}

}

## Section 3: xUnit Assert Methodları

### 7. Contain/DoesNotContain

[Fact]

public void AddTestContains()

{

Assert.Contains("Mehmet", "Mehmet Yağcı");

}

[Fact]

public void AddTestContainsList()

{

var names = new List<string>() { "Mehmet", "Şule", "Yasin", "Ümit" };

Assert.Contains(names, x => x == "Şule");

}

[Fact]

public void AddTestContainsList2()

{

var names = new List<string>() { "Mehmet", "Şule", "Yasin", "Ümit" };

Assert.Contains(names, x => x == "Hakan");

}

[Fact]

public void AddTestDoesNotContain()

{

Assert.DoesNotContain("Yasin", "Mehmet Yağcı");

}

### 8. True/False method

[Fact]

public void AssertTrue()

{

Assert.True(5 > 2);

}

[Fact]

public void AssertTrue2()

{

Assert.True("".GetType() == typeof(string));

}

[Fact]

public void AssertFalse()

{

Assert.False(2 > 5);

}

### 9. Match/DoesNotMatch

RegEx ifadesi alır. Uyuyorsa True, uymuyorsa false.

[Fact]

public void AssertMatches()

{

var regEx = "^dog";

Assert.Matches(regEx, "dog fight");

}

[Fact]

public void AssertDoesNotMatch()

{

var regEx = "^dog";

Assert.DoesNotMatch(regEx, "tiger fight");

}

### 10. StartsWith/EndsWith

[Fact]

public void AssertStartsWith()

{

Assert.StartsWith("meh", "mehmet");

}

[Fact]

public void AssertEndsWith()

{

Assert.EndsWith("met", "mehmet");

}

### 11. Empty/NotEmpty

[Fact]

public void AssertEmpty()

{

Assert.Empty(new List<string>());

}

[Fact]

public void AssertNotEmpty()

{

Assert.NotEmpty(new List<string>() { "Mehmet" });

}

### 12. InRange/NotInRange

[Fact]

public void AssertInRange()

{

Assert.InRange(10, 2, 20);

}

[Fact]

public void AssertNotInRange()

{

Assert.NotInRange(30,2,20);

}

### 13. Single

[Fact]

public void AssertSingle()

{

Assert.Single(new List<string>() { "Mehmet" });

Assert.Single<string>(new List<string>() { "Mehmet" });

}

[Fact]

public void AssertSingleFalse()

{

Assert.Single(new List<string>() { "Mehmet", "Şule" });

}

### 14. IsType/IsNotType

[Fact]

public void AssertIsType()

{

Assert.IsType<string>("Mehmet");

}

[Fact]

public void AssertIsNotType()

{

Assert.IsNotType<string>(5);

}

### 15. IsAssignableFrom

Bir tipin bir tipe referans verip veremeyeçeğini dönen metodtur.

[Fact]

public void AssertIsAssignableFrom()

{

Assert.IsAssignableFrom<IEnumerable<string>>(new List<string>());

}

### 16. Null/NotNull

[Fact]

public void AssertNull()

{

string deger = null;

Assert.Null(deger);

}

[Fact]

public void AssertNotNull()

{

string deger = "mehmet";

Assert.NotNull(deger);

}

### 17. Equal/NotEqual

[Fact]

public void AssertEqual()

{

Assert.Equal(2,2);

}

[Fact]

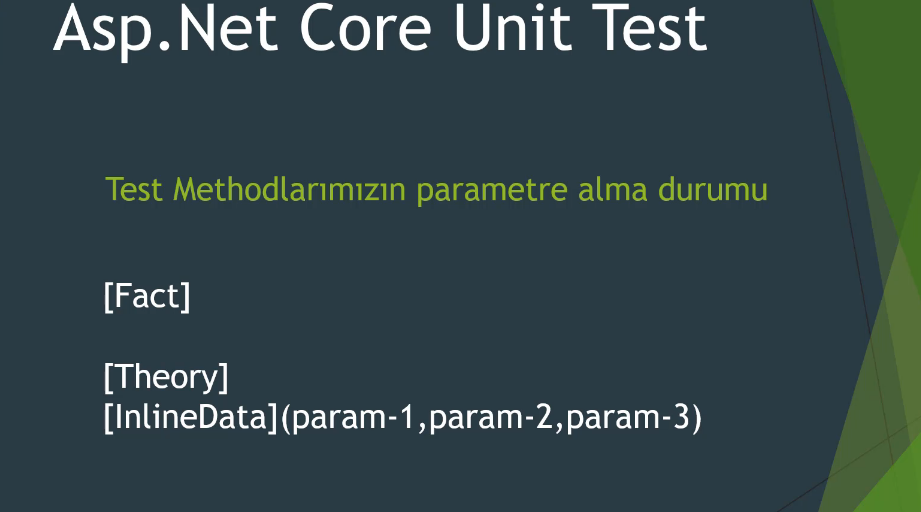
public void AssertNotEqual()

{

Assert.NotEqual(2,5);

}

### 18. Test methodlarımızın parametre alma



Fact – Parametre almaz

Theory – Parametre alır.

[InlineData](param-1,param-2,param-3) ile birlikte kullanılır.

[Theory]

[InlineData(2,5,7)]

public void AddTest2(int a, int b, int expectedTotal)

{

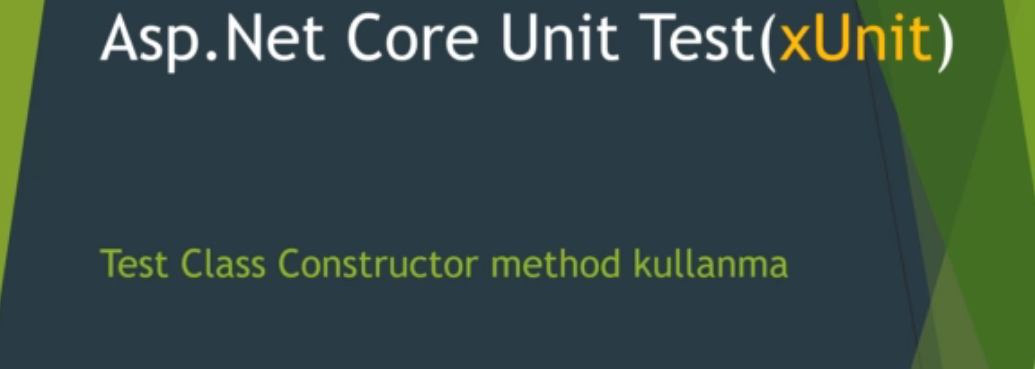
var calculator = new Calculator();

var actualTotal = calculator.Add(a, b);

Assert.Equal<int>(actualTotal, expectedTotal);

}

### 19. Test class'larımızda constructor kullanma



Test class’ında kullanaçağımız nesneleri her metod için new ‘lemek doğru değil.

Constructor ‘ında bir kere new ‘lemek yeterli. Önceden aşağıdaki her method içerisinde Calculator() nesnesi new ‘lniyordu. Bunu test class’ında bir kez yapmak daha doğru.

public class CalculatorTest

{

public Calculator \_calculator { get; set; }

public CalculatorTest()

{

\_calculator = new Calculator();

}

[Fact]

public void AddTest()

{

// Arrange -> değişlenlerin initialize edildiği yerdir

int a = 5;

int b = 20;

// Act ->

var total = \_calculator.Add(a, b);

// Assert -> doğrulama evresi

Assert.Equal<int>(25, total);

}

[Theory]

[InlineData(2,5,7)]

[InlineData(10, 2, 12)]

[InlineData(-3, 15, 12)]

public void AddTest2(int a, int b, int expectedTotal)

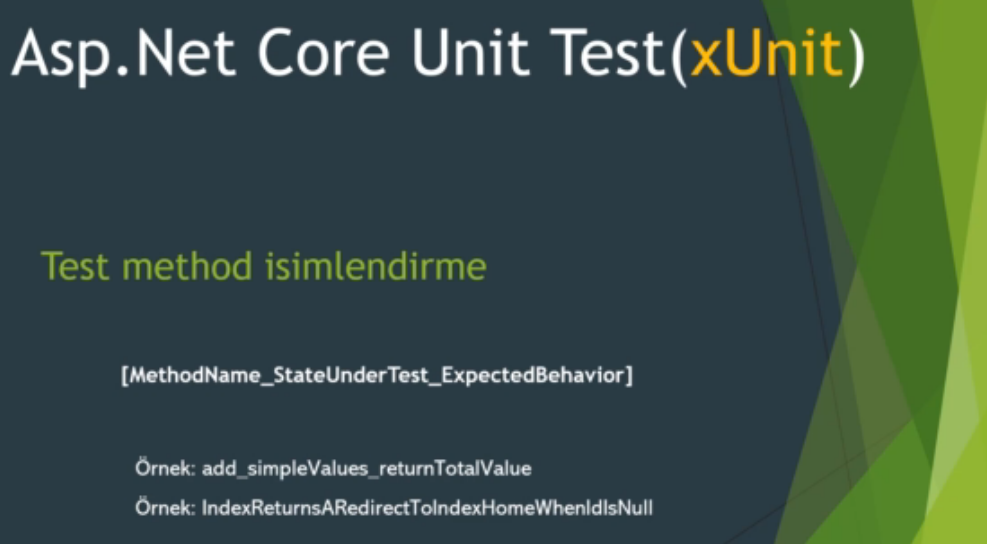
{

var actualTotal = \_calculator.Add(a, b);

Assert.Equal<int>(actualTotal, expectedTotal);

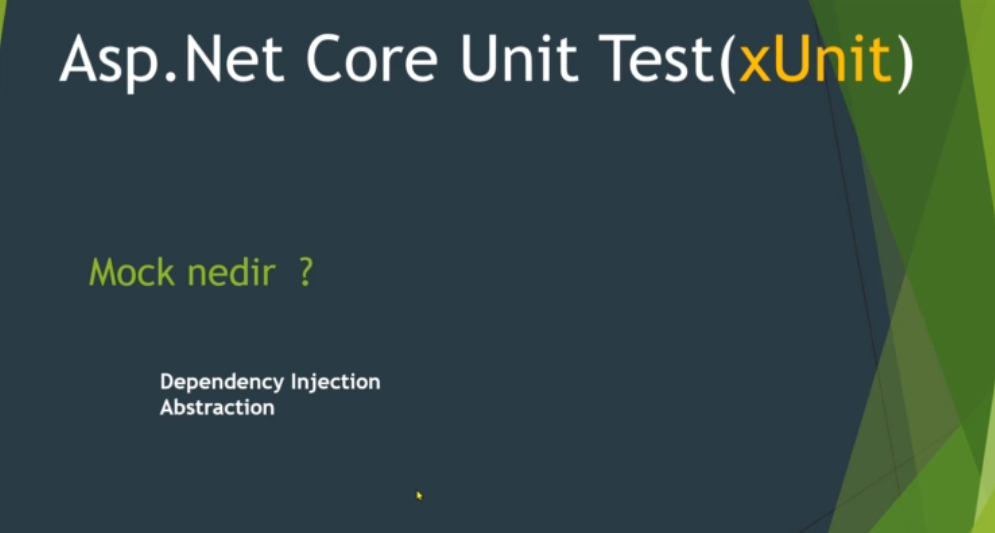
}

### 20. Test method isimlendirme



## Section 4: Moq Framework

### 21. Mock nedir ?



public class Calculator

{

public int Add(int a, int b)

{

// https://www.calculator.com/add/2/3 ortalama her bir istek 5 sn. sürüyor

if(a == 0 || b == 0)

{

return 0;

}

return a + b;

}

}

Tüm 100 tane metodumuz var, tümünü gerçek servislerden test etmek 5-6 saat sürebilir.

Bu durumda Mock ile bu metodun bir kopyasını yerel olarak oluşturup kullanarak 5-6 saatlik süreden tasarruf ediliyor. Mock kullanabilmek için class ‘ların birbirlerine loosely coupled olması gerekiyor.

### 22. Calculator.app projemize DI prensibini uygulamak

### 23. Moq Framework

### 24. Verify() : Bir methodun çalışıp çalışmadığını test etmek

### 25. Throws() : bir method üzerinden hata fırlatmak

### 26. Callback() : bir method üzerinden callback çalıştırmak