**Adım Adım Cross Browser Testing**

Birden fazla browser’ı birlikte kullanma.

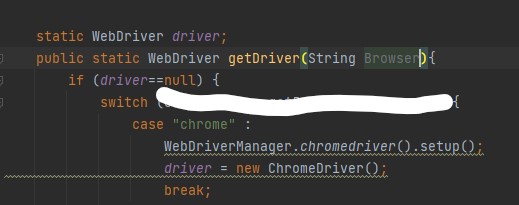
1. POM tasarımındaki Driver’ı bire bir yeni CrossDriver Class’ına kopyalayın.

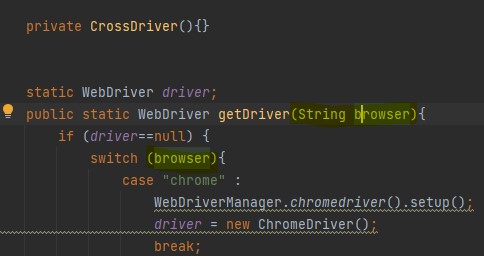


1. Ortak kullanımı şart koşmak için parametresiz private constructor create edin.



1. Cross Browser Testing yapabilmek için Driver classımıza parametre göndermeye ihtiyacımız olacak bu sebeple getDriver methodunu parametreli bir methoda çeviriyoruz ve parametresini (String Browser yapıyoruz.) Bunu yapınca artık switch caseler için configration dosyasına gidip browser için bir value geri getirmemize gerek kalmayacak o yüzden Switch caseleri parametredeki browser’a göre belirliyoruz.





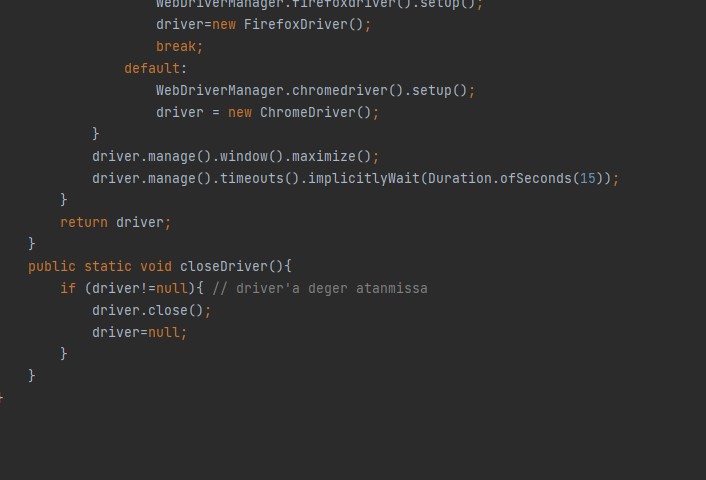
1. Bizim parametre ile getDriver methodunu çağırırken yapabileceğimiz bir hata sebebiyle her koşulda test çalışsın istiyorsak ve ConfigReader+Configration.xml dosyalarımız projenin içinde mevcutsa parametre atanmazsa bile Configration.xml’deki default browser için Driver’ı her koşulda methoda atamak için şu kodu ekleyebiliriz.



5.CrossDriver classı getDriver methodu eklemelerden sonra şu şekilde olacak.



1. Elimizde bir parametreye göre herhangi bir browserı çalıştırabilecek bir getDriver methodu mevcut ama bu methoda doğrudan XML dosyasından parametre gönderemiyoruz. XML’den parametre veri alabilmek için CrossDriver için bir TestBase classı create ediyoruz.

Xml(Parametre)->TestBase(Parametre+ klasik TestBase işlemleri)->getDriver

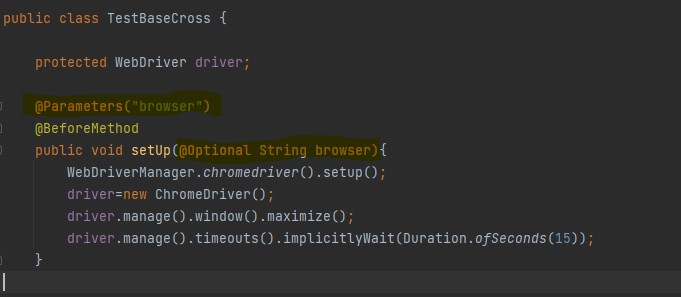
*7. driver*.manage().window().maximize();  
driver.manage().timeouts().implicitlyWait(Duration.*ofSeconds*(15));

Kodlarını TestBase’e taşıyacağımız için getDriver methodundan silebiliriz.

1. TestUnit’de kullandığımız TestBase’in aynısını TestBaseCross için kullanabiliriz.



1. GetDriver methodummuza parametre gönderebilmek için @Parameters notasyonunu kullanıyoruz(TestNG’deki) ve bununla birlikte methodumuzun parametresine @Optional notasyonunu her zaman ekliyoruz.



10.WebDriverManager’ı kullanmıyoruz.

11.Driverımıza CrossDriver classı getDriver methoduna giden browser parametresi değeri veriyoruz.(CrossDriver.getDriver(browser)



12. @AfterMethod  
public static void tearDown(){  
 CrossDriver.*closeDriver*();  
}

Ile Close driverı da uyarlıyoruz.

13.Src üzerinde yeni bir xml dosyası oluşturuyoruz.

14. Ilk satıra şunu kopyalıyoruz.

<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >

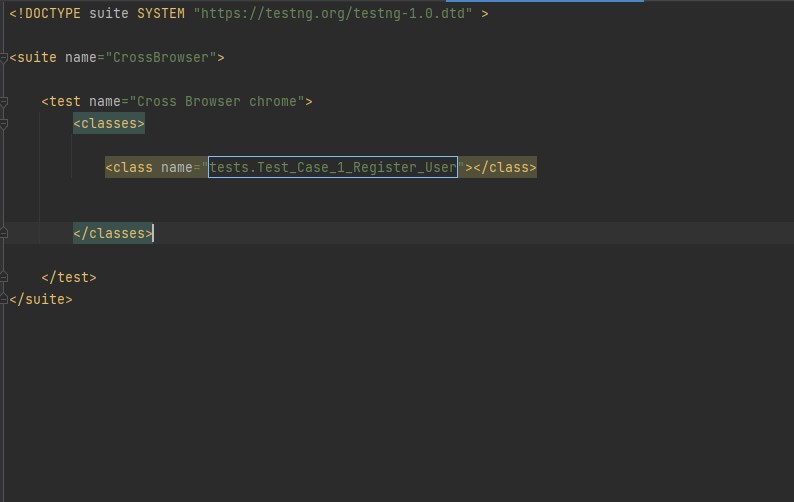
15. < basıyoruz suite öneriliyor,entera basıyoruz. Suitimize isim veriyoruz ve > basıp suite kapatıyoruz.

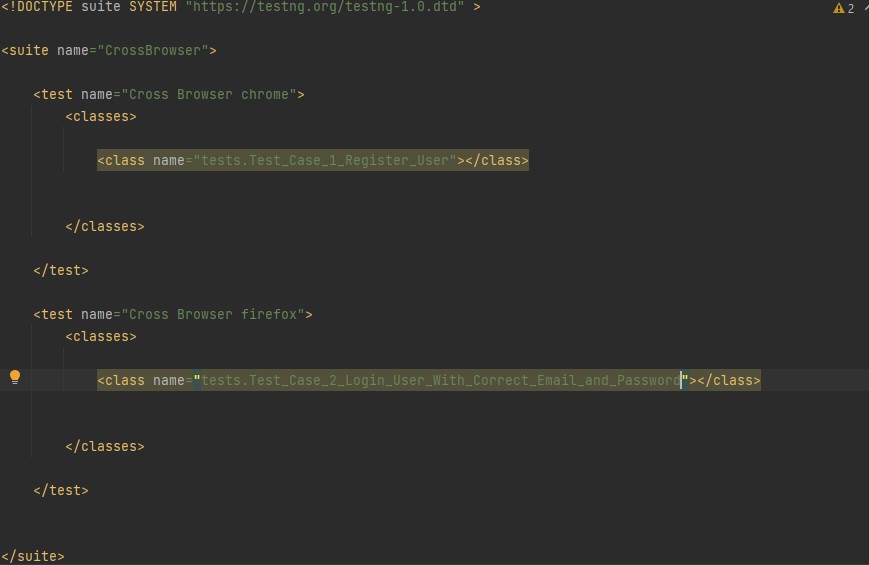
16.Suite taginin içinde < basıyoruz, önerilerden testi seçiyoruz ve teste isim veriyoruz.(Bir class için uygulanacaksa)

17. Testin içinde yine <basıyoruz ve classes seçiyoruz.

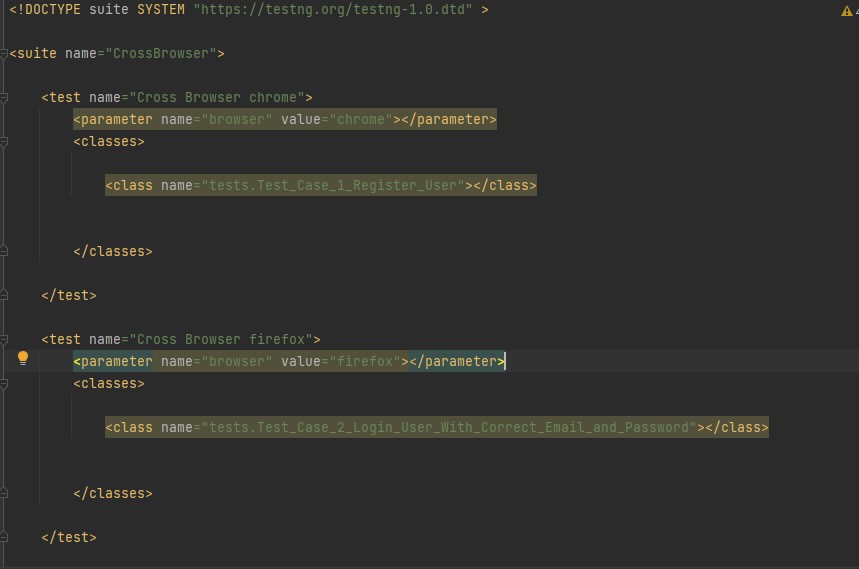
18.Onun içinde yine aynı işlemi yapıp class seçiyoruz ve oradan da classın yolunu gösteriyoruz.

19.Şunun gibi bir görüntü oluşması lazım.



20.Farklı browser için suitin içindekini yine suitin içine aynen kopyalayıp uyarlıyoruz.

21. Testlerimizin içine parametre ekleyeceğiz. Test taginin altında < basıyoruz ve parameter ı seçiyoruz. Name: “browser”(testbase de parametreye ne yazdıysak) value:(seçili testin hangi browser ile çalışmasını istiyorsak o/ yazımın CrossDriver’daki yazıma uygun olmasına dikkat etmemiz gerekiyor.)



22. XML dosyası çalıştırılınca farklı browserlarda testler yapmalıdır.

ÖNEMLİ:

Sorun varsa ;

* Testlerinizde kullandığın bütün classların TestBaseCross’a extend olduğuna emin ol
* Testlerde çalışan bütün classların CrossDriver’daki getDriver methodunu çağırdığından ve bu methodu browser parametresi ile çağırdığından emin ol. Sadece çalışan classlar değil mesela POM’dan yaralanılıyorsa pages paketindeki classlar vs.
* Halen sorun devam ediyorsa kullanılan browser’ın izinleriyle alakalı olabilir alternatif browserlar ile sorunun projede mi yoksa browserda mı olduğundan emin ol.