



Wise Quarter
first class IT courses

Selenium Team113

Ders-01

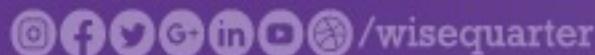
Selenium Giriş
WebDriver Method'ları



The future at your fingertips

+1 912 888 1630

www.wisequarter.com



Software Testing Nedir ?

Software testing var olan veya gelistirilmekte olan bir uygulamanin, tasirim asamasinda planlanan ozellikleri tasiyip tasimadiginin belirlenmesi icin yapılan faaliyetlerin butunudur.

Software testi icin tasirim asamasinda belirlenen sonuclar (**Expected Result**) ile uygulamanin kendisinden alınan sonuclar (**Actual Result**) karsilastirilir.

Expected ve actual result birbirine esit ise test basarili (**Test Passed**), Expected ve actual result birbirine esit degilse test basarisiz (**Test Failed**) olarak raporlanir.

Test gelistirme dongusunde developer'lar bir feature icin kod yazmaya basladiklarinda, tester'larda o feature'i analiz ederek acceptance criteria cercevesinde expected result'lari tespit etmeli, yazılımin bu ihtiyacları karşıladıgından emin olmak icin positive ve negative test senaryolari olusturmali ve bu testleri otomasyonla yapacak test case'leri olusturmalidir.



Software Testing Neden Önemlidir ?

Gunumuzdeki rekabetci piyasa kosullari, uygulamalari bugs - free olmaya zorlamaktadir.

Ayrıca developer'larin user case'den anladıklari ile end – user'larin kullanım alışkanlıklarını her zaman uyusmayabilir.

Uygulamaya sonradan eklenen bazı feature'lar çalışan uygulamada bazı işlevleri negatif etkileyebilir.



Kullanıcının bekleyenlerini karşılamayan uygulamalar başarısız olur.

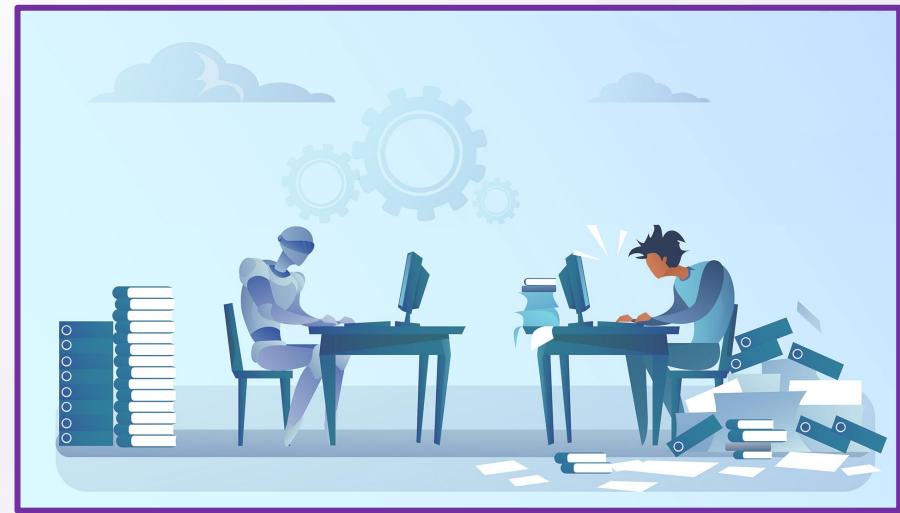
- Ürünün end-user kullanımına hazır olduğundan emin olmak
- End-user tarafından karşılaşılan sorunlarda düzeltme ve yeniden yapma maliyetlerini azaltmak
- Uygulamanın marketteki algısının üst seviyede olmasını sağlamak
- Sonradan eklenen işlevlerin eski işlevleri bozmadığından emin olmak için software testing yazılım geliştirme sürecinin vazgeçilmez bir parçası olmuştur.

Manual Software Testing Nedir ?

Manual testing, uygulamanın planlanan sonuçlara uygun çalışıp çalışmadığını hiç bir otomasyon aracı kullanmadan, bir end-user gibi test edilmesidir.

Ancak insan gücüyle yapılan bu testler için hem çok fazla insan gücüne ve zamana ihtiyaç duyulur hem de insanın özelliklerimizden dolayı testlerde yanlışlıklar yapılabilir.

Zaman ve ihtiyaç duyulan insan gücünü azaltmak, testler çalıştırılırken ortaya çıkabilecek hataları minimum'a indirmek için test otomasyonu gereklidir.



Manual tester'lar uygulama üzerinde çok fazla zaman harcadıkları ve her adımı manual yaptıkları için uygulama bilgileri daha iyidir.

Automation tester'lar uygulamayı daha iyi anlamak ve sistem ihtiyaçlarını görmek için başlangıçta bir kaç kez manual test yapıp sonra otomasyon yapmalıdır.

Test Otomasyonu Nedir ?

Test otomasyonu, insan gucu ile klavye ve mouse kullanilarak yapilabilecek bir yazılım testinin, bir otomasyon araci kullanilarak kodlar aracılıgi ile yapılmasidir.

Test otomasyon sayesinde

- klavye ve mouse kullanilarak yapilabilecek islemlerin cogu yapilabilir,
- yapılan islemler sonucunda gerceklesen sonuclar kaydedilebilir
- Elde edilen sonuclarla, expected sonuclar karsilastirilip, testin sonunu bulunabilir,
- Ve istenirse otomatik raporlar olusturulabilir



Otomasyonu yapılan bir test, istenen aralıklarla tekrar calistirilabilir. Hatta belirli aralıklarla olusturulan tum testler calistirilarak sistemin saglikli olarak calismaya devam ettiginden emin olunabilir (Regression Test)

Test otomasyonu insan gucu ihtiyacini azaltmasi, sorunsuz calismasi gibi ozellikleri sayesinde her gecen gun daha cok talep gormektedir.

Automation & Manual Testing

Yandaki kod sizce nedir ?

- A- Test Case
- B- Manuel tester icin test adimlari
- C- Otomasyon ile test yapan kodlar

Feature: US1010 herokuapp Delete testi

@heroku @sirali @pr1

Scenario: TC15 herokuapp'dan delete butonu calismali

Given kullanici "herokuappUrl" anasayfasinda

And add element butonuna basar

And kullanici 3 sn bekler

Then Delete butonu gorunur oluncaya kadar bekler

And Delete butonunun gorunur oldugunu test eder

Then Delete butonuna basar

And Delete butonunun gorunmedigini test eder

And sayfayı kapatir

Test otomasyonu sayesinde herkesin anlayacagi test case'ler olusturabilir, daha kisa surede, daha az insan kaynagi ile testlerinizi gerceklestirebilir, istediginiz raporlari otomatik olarak olusturabilirsiniz.

Manuel Test sayesinde kod bilgisi olmasa bile insanlara test yapabilir, temel test ihtiyaclarinizi karsilayabilirsiniz.



Automation & Manual Testing

MANUAL TESTING

VS

AUTOMATION TESTING

EXECUTION TIME



PEOPLE



INFRESTRUCTURE



TOOLS



TURNAROUND TIME



TRAINING



En Çok Kullanılan Tool'lar

En çok kullanılan test otomasyon tool'lari



Selenium Nedir ?

About Selenium

Selenium is a suite of tools for automating web browsers.

Selenium browser'lari otomasyon ile calistiracak tool'larin calismasi icin olusturulmus bir suite'dir.

Selenium farkli programlama dilleri ile calisarak gunumuzde kullanilan browser'larin tamamini otomasyon ile calistirabilmek icin olusturulmus class ve method'lara sahiptir.

Selenium'u kullanabilmek icin bu class'lar calisilan projeye eklenmelidir.

Selenium'un class'larini, kendi sitesinden indirecegimiz jar dosyalarini projeye ekleyerek projemize dahil edebilir veya bu isi bizim adimiza yapacak maven gibi tool'lari kullanarak class'lari direk projemize ekleyebiliriz.

Selenium Nedir ?

Selenium automates browsers. That's it!

What you do with that power is entirely up to you.

Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that.

Boring web-based administration tasks can (and should) also be automated as well.

Selenium browser'ları otomasyonla calistirir, bu otomasyon gucu ile ne yapacagini tamamen size kalmistir.

Selenium web uygulamalarini test etmek icin kullanilan acik kaynakli, ucretsiz bir uygulamadir.

2021 yilinda Selenium 4 piyasaya ciktigı ve Selenium'a yeni yetenekler(method'lar) kazandirdi.

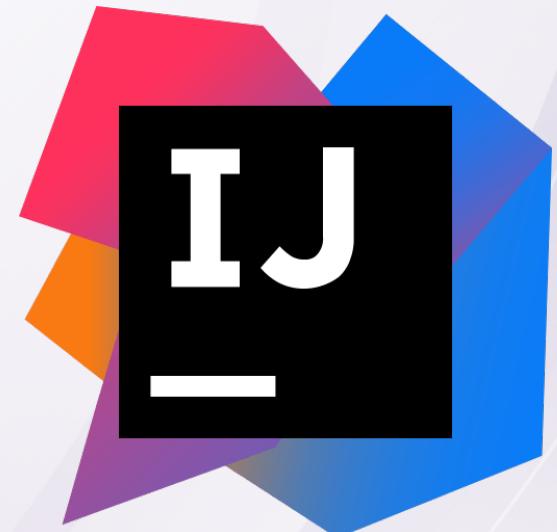
Selenium, Java, Phyton, .Net gibi en çok kullanılan programlama dilleri ile kullanılabilir.

IntelliJ Nedir ?

High Level programlama dilleri calisabilmek icin derleyicilere ihtiyac duyarlar.

IntelliJ IDEA 2000 yılında kurulmuş olan JetBrains firmasına ait olan, popüler bir kod gelistirme ortamı (**Integrated Development Environment**) dir.

IntelliJ uretkenligi en ust duzeye tasiyacak akilli kodlama yardimi, kod tamamlama ve ergonomic tasarim gibi ozelliklerle kod yazimini sadece verimli degil, ayni zamanda keyifli hale getirmistir.



Bir cok framework ve plugin ile calisma imkani saglar.

Kisaca, intelliJ IDE ihtiyaclarınızı tahmin eder ve sıkıcı ve tekrarlayan geliştirme görevlerini otomatikleştirir, böylece büyük resme odaklanabilirsiniz.

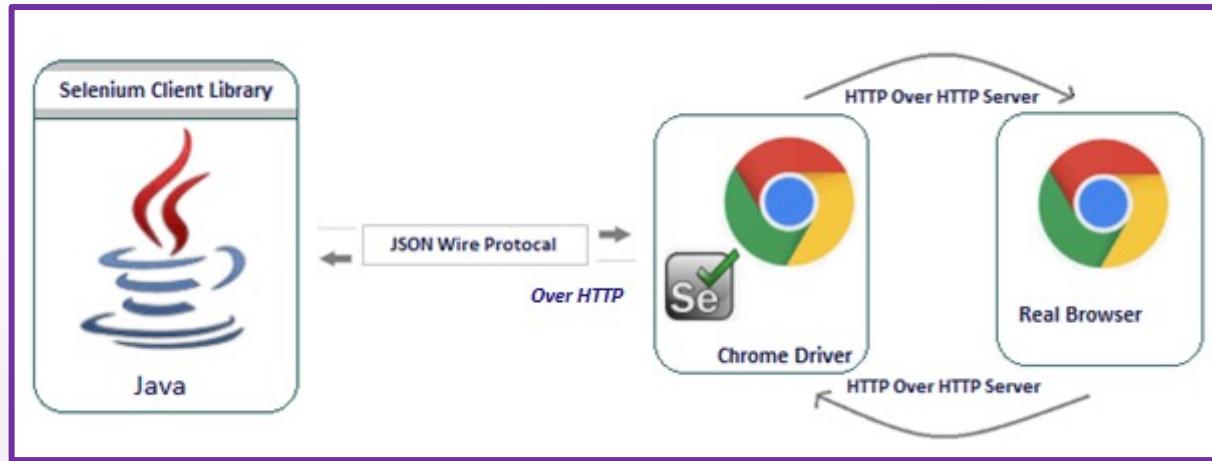
Selenium Bileşenleri

Selenium'un dört bileşeni vardır;

- Selenium Integrated Development Environment (IDE) (Selenyum Entegre Geliştirme Ortamı (IDE))
- Selenium Remote Control (RC)(Selenyum Uzaktan Kumanda (RC))
- WebDriver (Biz Selenium WebDriver kullanacağız)
- Selenium Grid (paralel test için kullanılıyor)



Selenium Nasil Calisir ?



Selenium test otomasyonunu WebDriver ile gerceklestirir.

Java ile yazdigimiz kodlar ile kullanilacak browser'a uygun bir webDriver objesi olusturulur.

Selenium kullanarak WebDriver class'indan olusturulan driver objesi bizim elimiz, gozumuz gibi calisir. Gonderildigi web sayfasinda klavye ve mouse ile yapabilecek islemleri yapar, elementlere tıklama, yazi gonderme, elementler uzerindeki yazilari alma gibi pek çok islemi gerceklestirir. Elde ettigi sonucları Java kodlarinin oldugu ortama döndürür.

Selenium'un Avantajlari & Dezavantajlari



- 1) Ücretsiz ve acik kaynaklidir. (Open source)
- 2) Bir çok programlama dilini destekler
(Java, Python, PHP, C#, Ruby vs.)
- 3) Çoklu işletim sistemleriyle çalışır.
Multiple operating systems (Windows, MacOS, Linux)
- 4) Birden çok tarayıcı ile çalışır.
Multiple browsers (Edge, Safari, Chrome, Firefox vs.)

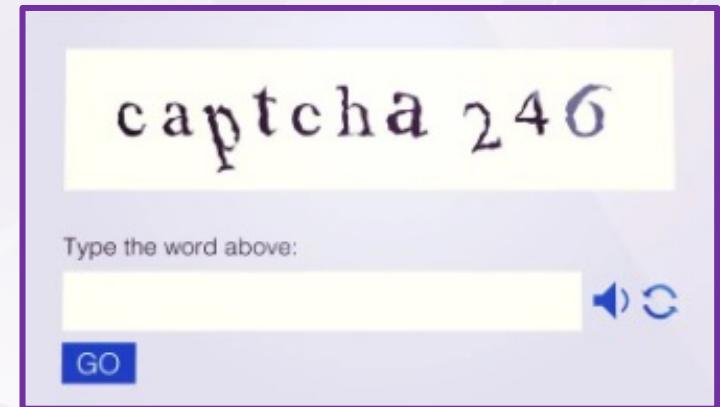
Programlama bilgisi gerektirir (Biz Java biliyoruz)

Yalnızca web tabanlı uygulamaları test eder

Profesyonel desteği sahip değil

performans testleri yapamaz

Captcha'yı asamaz(düger tüm otomasyon araçları gibi)



Framework Nedir ?

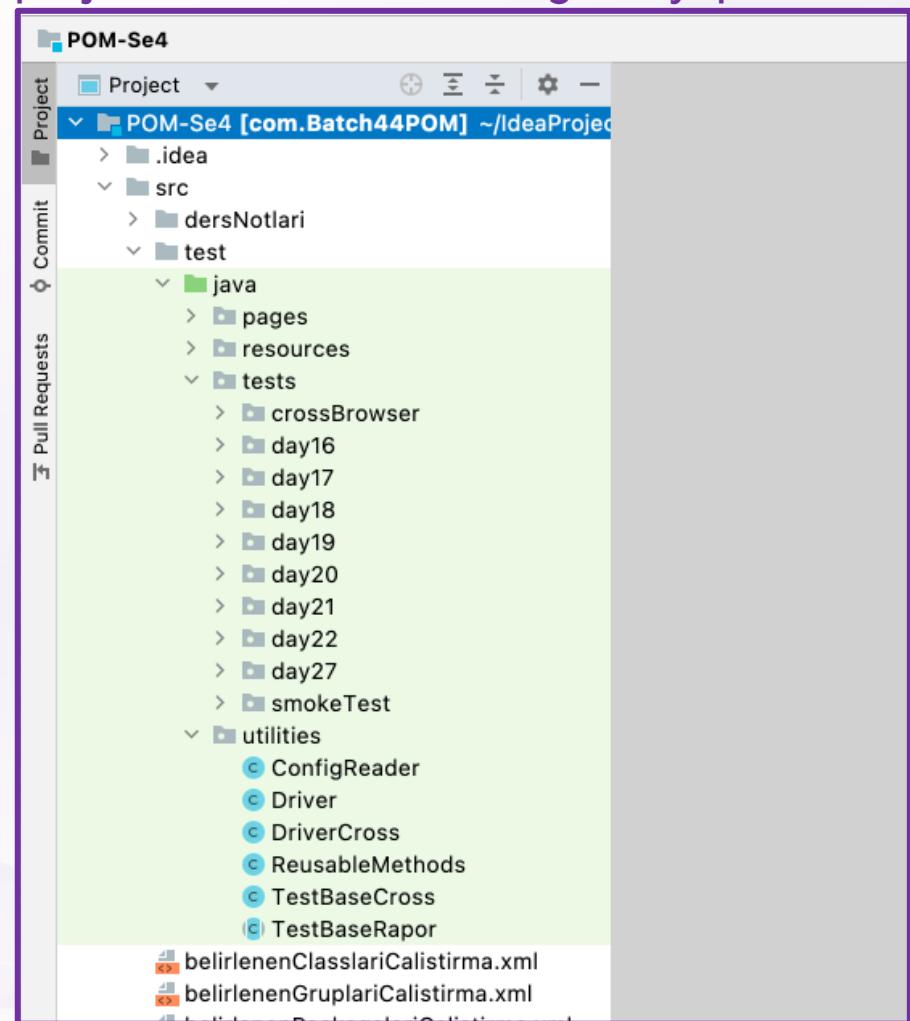
Framework, uzerine kodlarimizi yazarak projelerin olusturulabilecegi bir yapidir.

Test otomasyonu yaparken herseye sıfırdan baslayıp, herseyi sıfırdan kurgulayıp olusturmak yerine,

Herkesin anlayabileceği, test oluşturma ve gözden geçirme süreçlerini kolaylastırmak, tüm ekibin ortak çalışabileceği bir yapı oluşturmak için test framework'ları oluşturulmuştur.

Framework, test yapmak için kullandığımız tüm enstrumanları birleştirir.

Framework ile UI, API ve Database testleri yapılandırılabilir.



Jar Dosyaları ile Selenium Kurulumu

- 1) <https://www.selenium.dev/downloads/> adresine gidin
- 2) Selenium Client & WebDriver Language Bindings altında Java driver'ini download edin
- 3) Browsers altında Chrome documentation linkini tıklayalım

Chrome'un kendi sayfasına gidip Current stable release'i tıklayıp size uygun olanı download edin

Indirilen surum ile bilgisayarınızdaki Chrome browser surumunun aynı olduğundan emin olun

- 4) src altında resources director'si oluşturun
- 5) Bu klasor altında drivers ve libraries klasorleri oluşturun
- 6) Indirdigimiz chromedriver'i drivers klasorune, selenium-java dosyasını ise libraries klasorune çıkartın
- 7) intelliJ 'de yeni project / package / class oluşturualım ve class içinde main method oluşturun
- 8) File/Project Structure/Modules/Dependencies kısmından jar dosyalarını yükleyin

WebDriver Objesi Olusturma

Selenium jar dosyaları ile projeye eklendiğinde, kullanmak istenen tüm browser'ların driver'larının da projeye eklenmesi gerekmektedir.

Kullanılacak browser'a ait driver projeye eklendikten sonra, her class'da bilgisayardaki browser'i yönetecek bir WebDriver objesi oluşturulur ve o obje yardımıyla WebDriver Class'ındaki hazır method'lar kullanılabilir.

WebDriver objesi oluşturmak ve objeye kullanılacak browser'a uygun değeri atamak için main method içerisinde

- 1) Java'daki setProperty("webdriver.chrome.driver", "driverPath"); ile sistem ayarları yapılır.

```
System.setProperty("webdriver.chrome.driver", "src/driver/chromedriver"); /MAC
```

```
System.setProperty("webdriver.chrome.driver", "src/driver/chromedriver.exe");  
\WINDOWS
```

- 2) WebDriver driver= new ChromeDriver(); ile webdriver objesi oluşturulur ve istenen browser'a uygun değer ataması yapılır.



Wise Quarter
first class IT courses

Selenium Team113

Ders-02

WebDriver Method'lari

WebElements

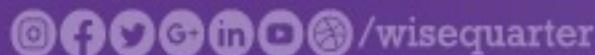
Locators



The future at your fingertips

+1 912 888 1630

www.wisequarter.com



WebDriver Objesi Kullanma

Selenium ile otomasyon yapabilmenin ilk adımı WebDriver Class'ından obje oluşturmaktr.

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class C01_DriverMethods {

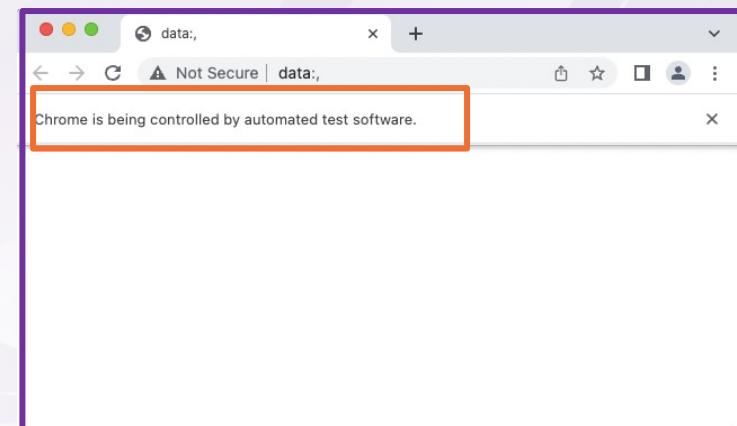
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "src/resources/chromedriver");

        WebDriver driver= new ChromeDriver();
    }
}
```

İlgili ayarları yapıp bir driver objesi oluşturduğumuzda, Selenium bu driver objesi sayesinde otomasyon yapabileceğimiz bir browser acar.

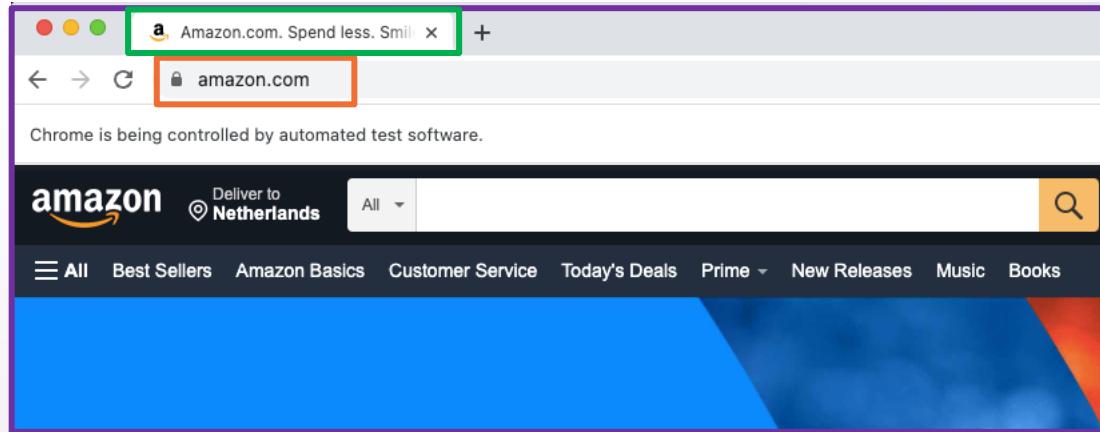
Bu browser'un Selenium tarafından kontrol edildiği yazılıdır.

Chrome dışında bir browser kullanılabaksa, o browser'in driver'ini da projeye eklemeli ve class içindeki ayarlarda o driver'in dosya yolu driver'a gösterilmelidir.





driver.get...() Method'ları



- 1- `driver.get("https://www.amazon.com");` driver'i istenen url'e goturur.
- 2- `driver.getCurrentUrl();` Gidilen Web sayfasinin URL bilgisini döndürür.
- 3- `driver.getTitle();` Gidilen Web sayfasinin title (baslik) bilgisini döndürür.
- 4- `driver.close();` Acılmış olan driver'i kapatır.
- 5- `driver.quit();` Test sırasında birden fazla window açılmışsa, tümünü kapatır.



driver.get...() Method'lari

6- driver.getPageSource()

Gidilen sayfanın kaynak kodlarını döndürür.

Sayfa kaynak kodları çok test otomasyonunda çok kullanılmaz, sadece özel olarak bu kodlarda bir kelimenin var olup olmadığı gibi özel bir test istenirse sayfa kodları String olarak kaydedilip, istenen arama yapılır.

```
ine wrap□
1 <!doctype html><html lang="en-us" class="a-no-js" data-19ax5a9jf="dingo"
2 <head><script>var aPageStart = (new Date()).getTime();</script><meta cha
3 <!-- sp:end-feature:head-start -->
4 <!-- sp:feature:csm:head-open-part1 -->
5
6 <script type='text/javascript'>var ue_t0=ue_t0||+new Date();</script>
7 <!-- sp:end-feature:csm:head-open-part1 -->
8 <!-- sp:feature:cs-optimization -->
9 <meta http-equiv='x-dns-prefetch-control' content='on'>
10 <link rel="dns-prefetch" href="https://images-na.ssl-images-amazon.com">
11 <link rel="dns-prefetch" href="https://m.media-amazon.com">
12 <link rel="dns-prefetch" href="https://completion.amazon.com">
13 <!-- sp:end-feature:cs-optimization -->
14 <!-- sp:feature:csm:head-open-part2 -->
15 <script type='text/javascript'>
16 window.ue_ihb = (window.ue_ihb || window.ueinit || 0) + 1;
17 if (window.ue_ihb === 1) {
18
19 var ue_csm = window,
20     ue_hob = +new Date();
21 (function(d){var e=d.ue||{},f=Date.now||function(){return+new Date}
22
23
24     var ue_err_chan = 'jserr-rw';
25     (function(d,e){function h(f,b){if(!!(a.ec>a.mxe)&&f){a.ter.push(f);b=b||e
26 pec:0,ts:0,erl:[],ter:[],mxe:50,startTimer:function(){a.ts++;setInterval
```

7- driver.getWindowHandle()

CDwindow-8C07925B8CBA4C8EF3039F660C30DDA1

Açılan window'a işletim sistemi tarafından verilen unique bir değer olan **window handle** değerini döndürür.

8- driver.getWindowHandles()

Test sırasında driver birden fazla window actıysa , bir **Set** olarak açılan tüm window'ların **window handle** değerlerini döndürür.

İlk Test Otomasyonu

Software testi için tasarım aşamasında belirlenen sonuçlar (**Expected Result**) ile uygulamanın kendisinden alınan sonuçlar (**Actual Result**) karşılaştırılır.

Expected ve actual result birbirine eşit ise test başarılı (**Test Passed**), Expected ve actual result birbirine eşit değilse test başarısız (**Test Failed**) olarak raporlanır.

Test aşamalarının ve test sonuçlarını anlasılabilir olması, testin kısa olmasından önemlidir.

```
String expectedTitleIcerik="amazon";
String actualTitle= driver.getTitle();

// url test yapalim

if (actualUrl.contains(expectedUrlIcerik)){
    System.out.println("Url test PASSED");
} else {
    System.out.println("Url test FAILED");
    System.out.println("actual Url : " + actualUrl);
    System.out.println("Actual Url aranan " + expectedUrlIcerik + " kelimesini icermiyor");
}
```

Ornegin; gidilen sayfanın title değerinin belirli bir kelimeyi içerdigi test edilmek isteniyorsa, expected ve actual değerler kaydedilip, bir if else blogu içerisinde istenen test yapılp, sonuc yazdırılabilir.

WebDriver Method'ları

1. Yeni bir package olusturalim : day01
2. Yeni bir class olusturalim : C03_GetMethods
3. Amazon sayfasina gidelim. <https://www.amazon.com/>
4. Sayfa basligini(title) yazdirin
5. Sayfa basliginin “Amazon” icerdigini test edin
6. Sayfa adresini(url) yazdirin
7. Sayfa url’inin “amazon” icerdigini test edin.
8. Sayfa handle degerini yazdirin
9. Sayfa HTML kodlarinda “alisveris” kelimesi gectigini test edin
10. Sayfayı kapatın.

driver.navigate...() Method'lari

9- `driver.navigate().to(url: "https://www.amazon.com");`

driver'i verile URL'e götürür. driver.get()'den farkı navigate method'lari ile gidilen sayfaların back, forward gibi fonksiyonları saglayabilmektedir.

10- `driver.navigate().back();` Gidilen web sayfasını bir önceki sayfaya döndürür.

11- `driver.navigate().forward();` Gidilen web sayfasından navigate().back() ile bir önceki sayfaya dönülmüşse yeniden ilk sayfaya götürür.

12- `driver.navigate().refresh();` İçinde olunan web sayfasını yeniler.

driver.navigate...() Method'lari

1. Yeni bir Class olusturalim.CO5_NavigationMethods
2. Youtube ana sayfasina gidelim . <https://www.youtube.com/>
3. Amazon soyfasina gidelim. <https://www.amazon.com/>
4. Tekrar YouTube'sayfasina donelim
5. Yeniden Amazon sayfasina gidelim
6. Sayfayı Refresh(yenile) yapalim
7. Sayfayı kapatalim / Tüm sayfaları kapatalim

driver.manage()... Method'lari

13- `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));`

driver'in gittiği sayfayı açması ve orada kullanacağı her bir web elementi bulması için tanımlanan maximum bekleme süresini tanımlar.

Wait konusu ayrı bir konu olarak anlatılacak, ancak yazılan otomasyon yapılmırken, internet bağlantısı veya bilgisayarın hızı gibi sebeplerle gecikmeler yaşanması durumunda ne yapacağını net olması için her testin başında max.bekleme süresi belirlenmelidir.

14- `driver.manage().window().maximize();`

Açılan driver'i tam sayfa yapar.

`driver.manage().window()`.... ile kullanılabilen farklı method'lar vardır ancak açılan web sayfasında tüm webelement'lerin görülebilir ve ulaşılabilir olması için her testin başında `maximize()` method'u kullanılmamasında fayda vardır.

driver.manage()... Method'ları

15-

```
driver.manage().window().fullscreen();
driver.manage().window().maximize();
driver.manage().window().minimize();
```

Acilan driver'i onceden belirlenmis standart buyukluklere getirir.

16-

```
driver.manage().window().setSize(new Dimension(width: 1000, height: 700));
driver.manage().window().setPosition(new Point(x: 100, y: 100));
```

Acilan driver'i kullanicinin istedigi ozel olculere getirir ve istenen noktaya tasir.

17-

```
driver.manage().window().getPosition();
driver.manage().window().getSize();
```

Acilan driver'in bulundugu pozisyonu ve boyutlarini döndürür.

driver.manage()... Method'lari

1. Yeni bir Class olusturalim.C06_ManageWindow
2. Amazon soyfasina gidelim. <https://www.amazon.com/>
3. Sayfanin konumunu ve boyutlarini yazdirin
4. Sayfayi simge durumuna getirin
5. simge durumunda 3 saniye bekleyip sayfayı maximize yapın
6. Sayfanin konumunu ve boyutlarini maximize durumunda yazdirin
7. Sayfayı fullscreen yapın
8. Sayfanin konumunu ve boyutlarini fullscreen durumunda yazdirin
9. Sayfayı kapatın

driver.manage()... Method'lari

1. Yeni bir Class olusturalim.C07_ManageWindowSet
2. Amazon soyfasina gidelim. <https://www.amazon.com/>
3. Sayfanin konumunu ve boyutlarini yazdirin
4. Sayfanin konumunu ve boyutunu istediginiz sekilde ayarlayın
5. Sayfanin sizin istediginiz konum ve boyuta geldigini test edin
8. Sayfayı kapatın

WebDriver Method'ları

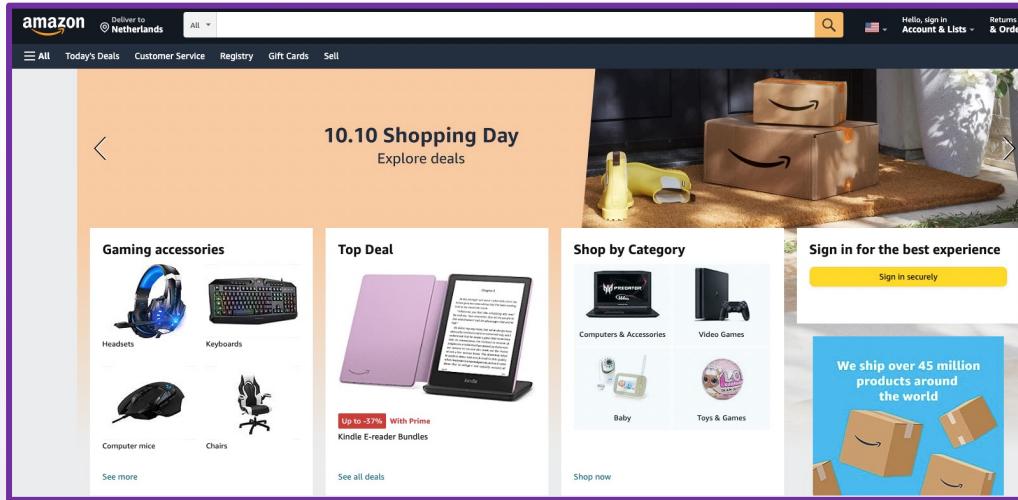
1. Yeni bir class olusturalim (Homework)
2. ChromeDriver kullanarak, facebook sayfasina gidin ve sayfa basliginin (title) "facebook" oldugunu dogrulayin (verify), degilse dogru basligi yazdirin.
3. Sayfa URL'inin "facebook" kelimesi icerdigini dogrulayin, icermiyorsa "actual" URL'i yazdirin.
4. <https://www.walmart.com/> sayfasina gidin.
5. Sayfa basliginin "Walmart.com" icerdigini dogrulayin.
6. Tekrar "facebook" sayfasina donun
7. Sayfayı yenileyin
8. Sayfayı tam sayfa (maximize) yapın
9. Browser'i kapatın

WebDriver Method'ları

1. Yeni bir class olusturun (TekrarTesti)
2. Youtube web sayfasına gidin ve sayfa başlığının "youtube" olup olmadığını doğrulayın (verify), eğer değilse doğru başlığı(Actual Title) konsolda yazdırın.
3. Sayfa URL'sinin "youtube" içerip içermediğini (contains) doğrulayın, içermiyorsa doğru URL'yi yazdırın.
4. Daha sonra Amazon sayfasına gidin <https://www.amazon.com/>
5. Youtube sayfasına geri donun
6. Sayfayı yenileyin
7. Amazon sayfasına donun
8. Sayfayı tamsayfa yapın
9. Ardından sayfa başlığının "Amazon" içerip içermediğini (contains) doğrulayın, Yoksa doğru başlığı(Actual Title) yazdırın.
- 10.Sayfa URL'sinin <https://www.amazon.com/> olup olmadığını doğrulayın, degilse doğru URL'yi yazdırın
- 11.Sayfayı kapatın

WebElements

Bir web sayfasında kullanılan herseye web element denir.



Her web element farklı özelliklerde olur. Link, açılır menü, button gibi etkilesimli web elementler olduğu gibi resim, background gibi etkilesimsiz web elementler de olur.

Görünüş her web element aslında developer'lar tarafından yazılan bir HTML kodun görselleştirilmiş halidir.

Selenium WebDriver görsel elementleri değil, HTML kodları kullanır.

Otomasyon sırasında kullanılmak istenen web elementler HTML kodları kullanılarak unique olarak WebDriver'a tarif edilmelidir.

WebElements



Her bir web element yapısına uygun olarak farklı tag ve attribute'ler bulundurur.

Web elementi unique olarak tarif edebilmek için tag ve attribute'ler tekil olarak kullanılabilir.

Tekil kullanım unique tarif için yeterli olmazsa, birden fazla bilginin kombinasyonu kullanılır.

Tag : input

Attributes : type, id, value, name, autocomplete, placeholder, class, dir, tabindex, aria-label

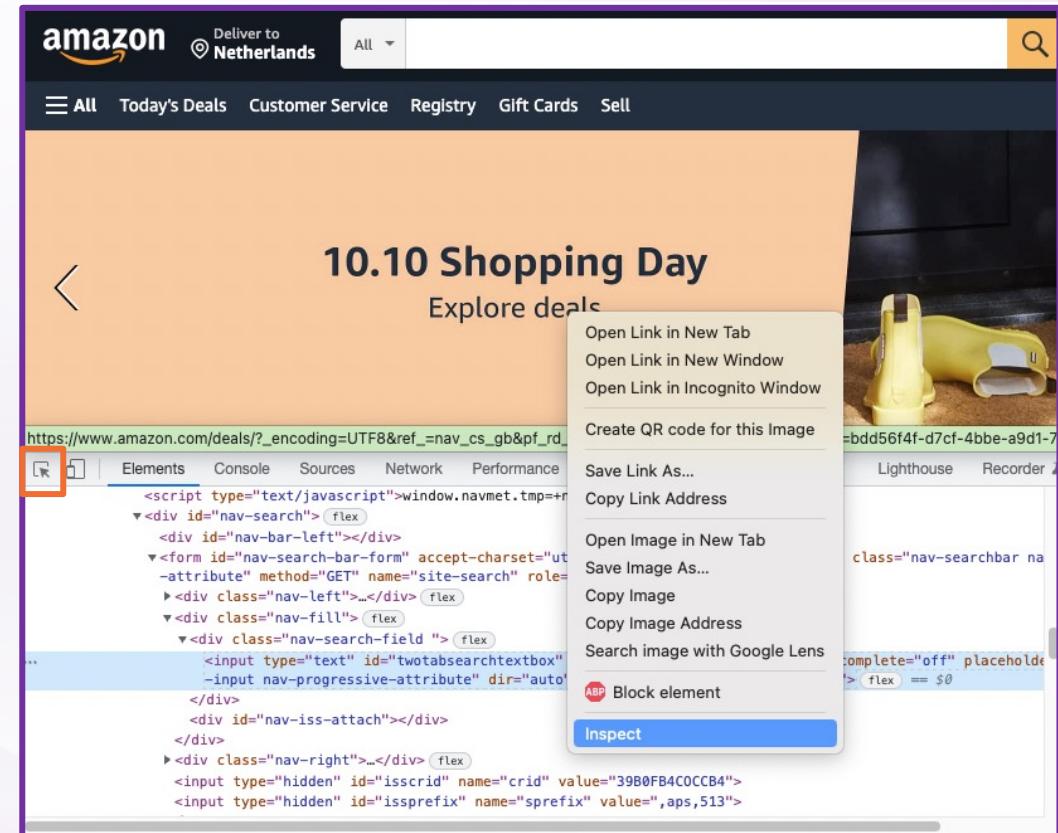
WebElements

Web sayfasında HTML kodlarını görebilmek için mouse'da sağ click yapıp inspect/incele seçilmelidir.

Istenen web elementin HTML kodunu bulmak için, o element üzerinde yeniden sağ click yapıp inspect denebilir,

Veya menudeki → işaretini seçip, mavi iken mouse ile istenilen element seçilebilir.

HTML kodları açık iken ctrl+f tuslarına basılıncaya acılan bölümde webelement'in özellikleri aratılırsa, o özellikte kaç webelement bulunduğu görülebilir.



Locators

Selenium LOCATORS, web sayfasındaki web öğelerini tanımlamak için kullanılır.

Selenium'da; metin kutuları, onay kutuları, linkler, radyo butonları, liste kutuları ve diğer **tüm web öğeler** üzerinde eylemler gerçekleştirmek için LOCATORS'a ihtiyacımız vardır.

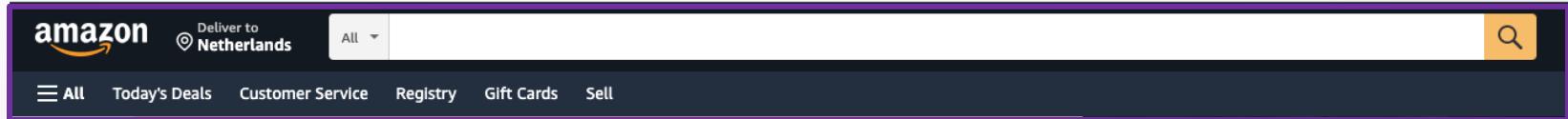
Konum belirleyiciler bize web elementleri tanımlamada yardımcı olur.

Web Elementlerine ulaşmak için tag veya bazı attribute'lerin kullanıldığı 6 adet locators bulunur, bunlarla ulaşamayan webelementleri için özel olarak tanımlanan Xpath ve css locator'lari kullanılır.



SELENIUM LOCATORS

Locators



1- By.id("uniqueld")

Web elementi tanimlamak icin ilk bakacagimiz locator id olabilir.

Id genellikle unique oldugu icin locate etmekte sıkça kullanılır. Ancak developer'ların aynı id ile birden fazla webelementi tanımlayabilecekleri de unutulmamalıdır.

Hangi locator kullanılırsa kullanılsın, web sayfasının HTML kodlarında locator aratılarak, unique sonuca ulaşıldığı gözlemlenmelidir.



driver.findElement() Method'u

The screenshot shows the top navigation bar of an Amazon page. It includes the Amazon logo, a dropdown for 'Deliver to Netherlands', a search bar with a magnifying glass icon, and a menu bar with links for 'All', 'Today's Deals', 'Customer Service', 'Registry', 'Gift Cards', and 'Sell'. Below this, a larger box highlights the HTML code for the search input field:

```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

Unique locator'i tespit edilen web element kullanilmak icin, driver objesi ile LOCATE edilib, WebElement class'indan olusturulan objeye atanmalidir.

Driver.findElement(By.locator ("uniqueLocatorDegeri"))

```
WebElement amazonAramaKutusu = driver.findElement(By.id("twotabsearchtextbox"));
```

Driver, findElement() ile objeyi bulamazsa **NoSuchElementException** verir.

findElement() ile locate edilib, objeye atanen WebElement testler sirasinda kullanilabilir.

webElement bir obje oldugu icin direkt yazdirilamaz, hazir method'lar kullanilarak manipule edilebilir.



WebElement Objesi Olusturma

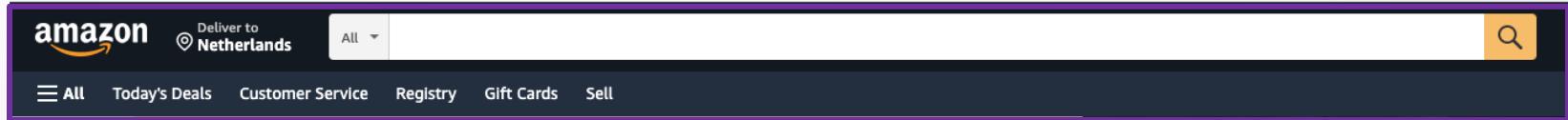
The screenshot shows the top navigation bar of the Amazon website. It includes the Amazon logo, a dropdown menu for 'Deliver to' set to 'Netherlands', a search bar with the placeholder 'All', and a search button. Below the main navigation, there are links for 'All', 'Today's Deals', 'Customer Service', 'Registry', 'Gift Cards', and 'Sell'. A large callout box highlights the HTML code for the search input field.

```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

Amazon Arama Testi

- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.amazon.com> adresine gidin
- 3- amazon arama kutusunu locate edin
- 4- arama kutusuna "Nutella" yazdirin
- 5- arama islemini yapabilmek icin ENTER tusuna basin

Locators



```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

2- By.name("uniqueName")

WebElement'in HTML kodlarında name attribute'u varsa ve unique ise locate etmek için By.name() kullanılır

```
WebElement amazonAramaKutusu= driver.findElement(By.name("field-keywords"));
```



Locators

CATEGORY

WOMEN	+
MEN	+
KIDS	+

3- By.className("uniqueClassName")

class attribute'u genellikle benzer ozellikleri barindiran web elementleri gruplandirmak icin kullanilir.

```
<div class="panel-group category-products" id="accordion">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Women" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Men" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">...</div>
    <div id="Kids" class="panel-collapse collapse">...</div>
  </div>
</div>
```

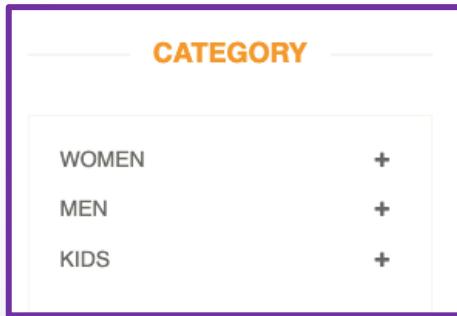
Bu sebeple class attribute'u ile yapacagimiz locate islemleri genellikle 1 web element degil, birden fazla element döndürür.

bu elementleri store edebilmek icin bir web element degil, web elementlerinden olusan bir list gereklidir.

NOT : class value'sunde bosluk (space) varsa By.className ile locate islemlerinde sorunlar yasanabilir.



driver.findElements() Method'u



driver.findElements(....)

Locator'a uygun tüm web elementlerini döndürür.

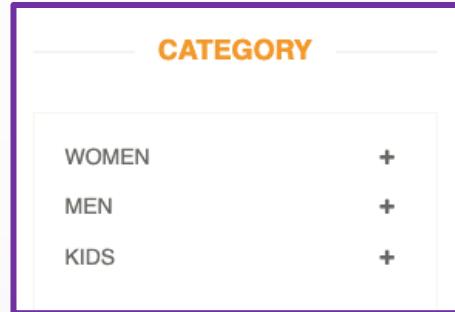
```
<div class="panel-group category-products" id="accordion">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Women" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Men" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">...</div>
    <div id="Kids" class="panel-collapse collapse">...</div>
  </div>
<div class="brands_products">
```

findElements() birden fazla web element döndürebileceği için dönen elementleri store etmek için bir list kullanılmalıdır.

Locator'a uyan hicbir webelement olmasa da exception olusmaz, bos bir list olusur.

List'teki tüm elementler web element oldugu için direkt yazdırılamaz, bir for-each loop kullanılarak elementlere istenen işlemler yapılabilir.

Locators



Automation Exercise Category Testi

- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.automationexercise.com/> adresine gidin
- 3- Category bolumundeki elementleri locate edin
- 4- Category bolumunde 3 element oldugunu test edin
- 5- Category isimlerini yazdirin
- 6- Sayfayı kapatın

Locators

4- By.tagName("tagName")

```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

```
List<WebElement> inputTagList =driver.findElements(By.tagName("input"));
```

Bir web sayfasında herhangi bir tagName'in unique olması nadiren karşılaşılmabilir bir durumdur.

Tag ismi ile yapılan locate'ler unique bir elemente ulaşmaktan daha çok sayfadaki tüm link'leri bulmak gibi amaclarla kullanılabilir.

Birden fazla web element döndürecegi için driver.findElements(..) ile kullanılması daha çok karşılaşılan bir durumdur.



Locators

```
▼<a href="/view_cart"> == $0
▶<i class="fa fa-shopping-cart">...
    " Cart"
</a>
```

5- By.linkText("linkYazisininTamami")

6- By.partialLinkText("linkYazisininBirBolumu")

Sadece link'ler icin kullanilabilirler.

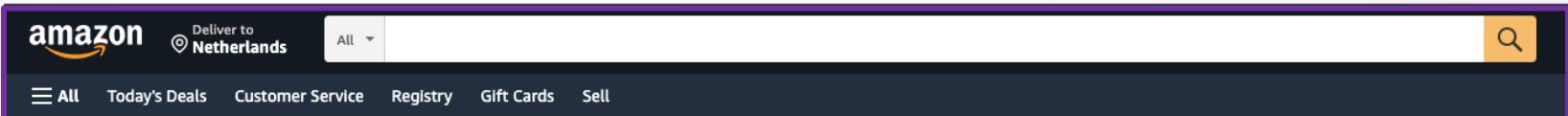
Her link uzerinde bulunan yazi kullanilarak locate yapmamizi saglar.

Link uzerinde bulunan yazi String data turunde oldugundan case sensite'dir.

By.linkText () icin bosluklar da dikkate alınarak tum metin yazılmalıdır.

Tum metnin yazılaması, yazının kısmı olarak kullanılması isteniyorsa
By.partialLinkText () kullanılmalıdır.

WebElement Method'lari



```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

Bir web element'i locate etmek her testin vazgecilmez bir adimidir.

Webelement'i locate ettikten sonra variable atamak veya atamadan direk kullanmak test adimlarina ve belirlenen genel test stratejisine baglidir.

Webelement ile yapabilecegimiz islemler icin hazir method'lari kullaniriz.

1- `webElement.click();`

Web element'e click yapar.

2- `webElement.sendKeys(...keysToSend: "Istenen Metin");`

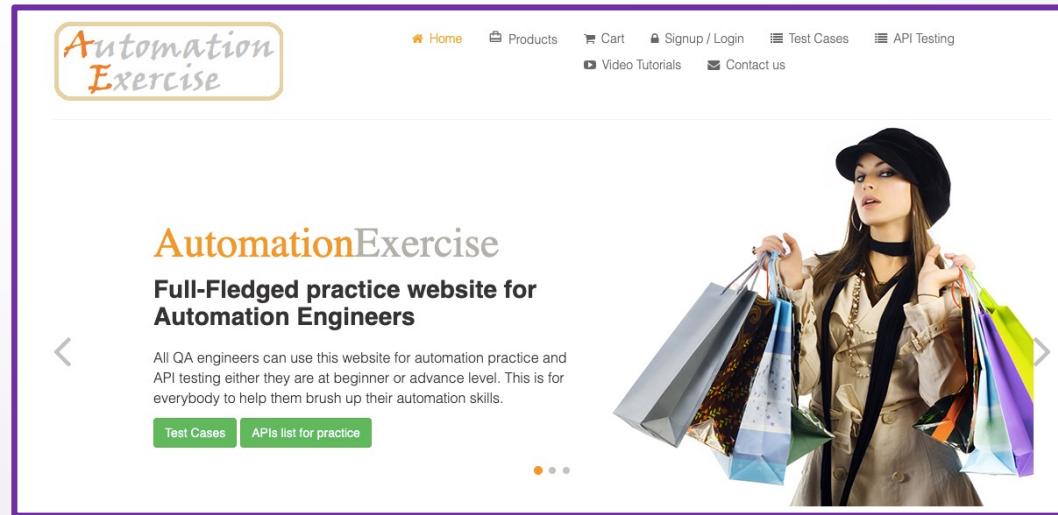
Web element'e istenen metni yollar

WebElement Method'lari



- 1- `webElement.submit();` Web element ile işlem yaparken ENTER tusuna basma işlemini yapar.
- 2- `webElement.sendKeys(...keysToSend: "Istenen Metin" + Keys.ENTER);` Web element'e istenen metni yollayip, sonra ENTER tusuna basar
- 3- `webElement.isEnabled();` Web element erişilebilir ise true, yoksa false döner.
- 4- `webElement.isDisplayed();` Web element gorunuyor ise true, yoksa false döner.
- 5- `webElement.isSelected();` Web element secili ise true, yoksa false döner.

Locators



Automation Exercise Link Testi

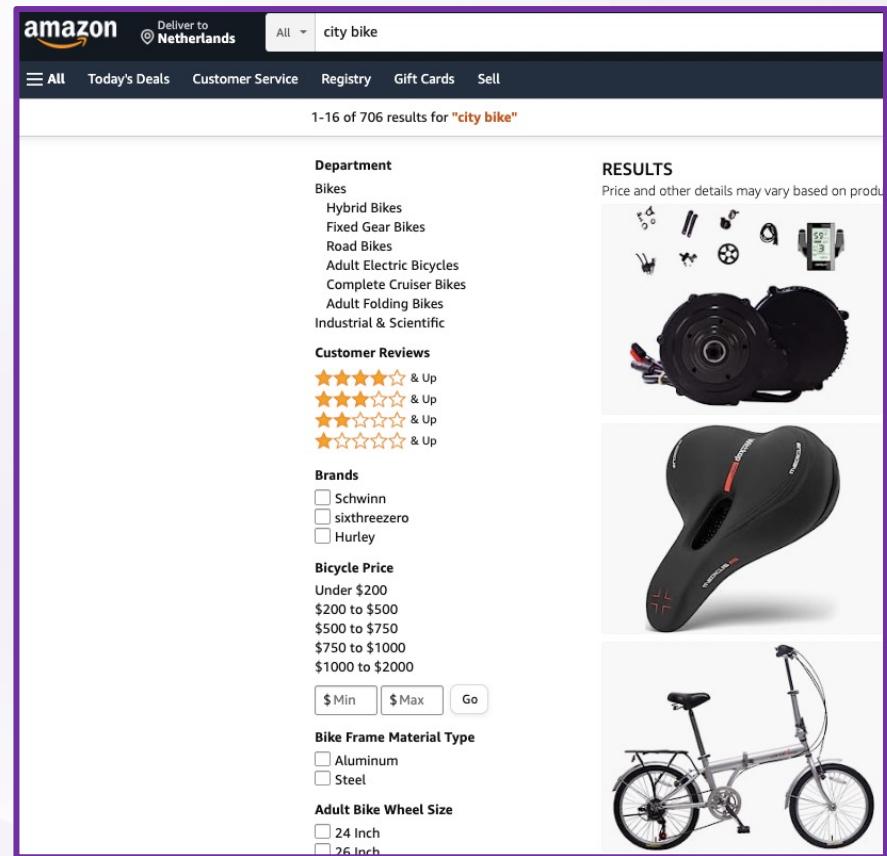
- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.automationexercise.com/> adresine gidin
- 3- Sayfada 147 adet link bulundugunu test edin.
- 4- Products linkine tiklayin
- 5- special offer yazisinin gorundugunu test edin
- 6- Sayfayi kapatın

Locators

	<code>findElement()</code>	<code>findElements()</code>
websayfasında birden fazla Web Element Locator ile uyusursa	İlk elemani dondurur	Tüm elemanları ondurur
websayfasında hiçbir Web Element Locator ile uyuşmazsa	NoSuchElementException fırlatır	Exception fırlatmaz, boş bir liste dondurur
Return Type	WebElement	List<WebElement>
Elemana erişim	Direk ulaşılabilir	Liste'den index veya iterator ile ulaşılabilir

Locators

- 1- <https://www.amazon.com/> sayfasına gidin.
- 2- Arama kutusuna “city bike” yazıp aratin
- 3- Görüntülenen sonuçların sayısını yazdırın
- 4- Listededen ilk ürünün resmine tıklayın.



The screenshot shows the Amazon search results for "city bike". The search bar at the top has "city bike" entered. Below the search bar, there are filters for Department, Customer Reviews, Brands, Bicycle Price, Bike Frame Material Type, and Adult Bike Wheel Size. On the right side, there is a sidebar titled "RESULTS" showing various cycling-related items like a chain, a seat, and a folding bike.

Department:

- Bikes
 - Hybrid Bikes
 - Fixed Gear Bikes
 - Road Bikes
 - Adult Electric Bicycles
 - Complete Cruiser Bikes
 - Adult Folding Bikes
 - Industrial & Scientific

Customer Reviews:

- ★★★★★ & Up
- ★★★★☆ & Up
- ★★★☆☆ & Up
- ★☆☆☆☆ & Up

Brands:

- Schwinn
- sixthreezero
- Hurley

Bicycle Price:

- Under \$200
- \$200 to \$500
- \$500 to \$750
- \$750 to \$1000
- \$1000 to \$2000

Bike Frame Material Type:

- Aluminum
- Steel

Adult Bike Wheel Size:

- 24 Inch
- 26 Inch



Wise Quarter
first class IT courses

Selenium

Ders-03

Locators

+1 912 888 1630
www.wisequarter.com


[/wisequarter](#)

 /wisequarter



The future at your fingertips

www.wisequarter.com

Locators - Xpath

Bir WebElement'i locate etmek için kullanabileceğimiz en etkin yöntemdir.

```
WebElement webElement= driver.findElement(By.xpath( xpathExpression: "bulunan xpath"));
```

Onceki 6 locator, HTML element oluşturulurken developer'in yazdığı kodlara göre yapılır.

Ornegin; HTML element'de id attribute'u varsa By.id() method'u kullanabilir ama developer id attribute'u koymedi ise kullanamayız.

Aynı şekilde HTML elementi bir link ise By.linkText() veya By.partialLinkText() kullanabiliriz, link degilse kullanamayız.



Xpath de HTML kodu kullanır ancak farklı kombinasyonlar kullanıldığı için dinamiktir ve her webelement için mutlaka bir xpath bulunabilir.

2 çeşit Xpath yazılabilir

1. **Absolute xpath** (mutlak)

2. **Relative xpath** (bağılı)

Locators

HTML kodlarda Parent – Child – Sibling ilişkisi

```
<div class="navFooterLine navFooterLinkLine navFooterDescLine" role="navigation" aria-label="More on Amazon.com">
  <table class="navFooterMoreOnAmazon" cellspacing="0">
    <tbody>
      <tr>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">
          <a href="https://advertising.amazon.com/?ref=footer_advtsing_amzn_com" class="nav_a">
            "Amazon Advertising"
            <br>
            <span class="navFooterDescText">...</span>
          </a>
        </td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
```

Her HTML sayfasi <Html> </Html> taglari arasina yazilir.

Bir HTML taginin arasina yeni bir tag acildiginda konum olarak bir tab icerden baslar.

HTML taginin dusey hizasina bakarak parent-child-sibling ilişkisi anlasilabilir.



Locators

1. Absolute Xpath

```
<div class="navFooterLine navFooterLinkLine navFooterDescLine" role="navigation" aria-label="More on Amazon.com">
  <table class="navFooterMoreOnAmazon" cellspacing="0">
    <tbody> ← // div/ table/ tbody
      <tr>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">
          <a href="https://advertising.amazon.com/?ref=footer_advtsing_amzn_com" class="nav_a">
            "Amazon Advertising"
            <br>
            <span class="navFooterDescText">...</span> ← // tbody / tr / td[3] // span
          </a>
        </td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
        <td class="navFooterDescItem">...</td>
        <td class="navFooterDescSpacer" style="width: 4%"></td>
```

Absolute xpath yazmak için en basa // sonraki her adımda / yazarak hedef web element'e kadar tüm tag'lar yazılır.

Eğer aynı path'e sahip birden fazla element varsa index kullanılabilir. [2] gibi

Eğer bir parent'in grand child'lari içinde unique bir tag varsa parent // grand child yazılabilir

Locators

2.Relative Xpath



```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords"  
autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute"  
dir="auto" tabindex="0" aria-label="Search">
```

Bir web element'in 3 bileseni bulunur.

1- Tag : input

2- Attributes : type, id, value, name, autocomplete, placeholder, class, dir, tabindex, aria-label

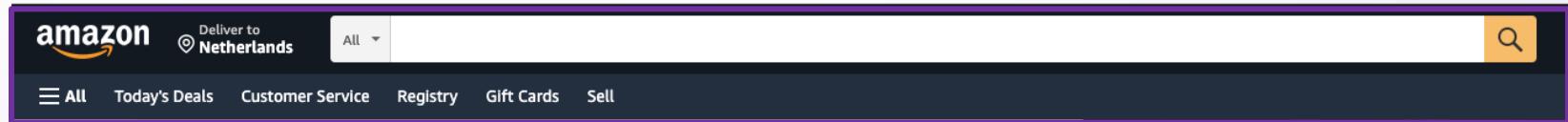
3- Attribute Values : text, twotabsearchtextbox, field-keywords

Relative Xpath bu 3 bilesenin belirlenen sekilde birlikte kullanilmasi ile olusur. Her Xpath ile unique bir sonuc elde edilemeyebilir ancak unique bir deger mutlaka bulunur.

//tagName[@attributelsmi='attributeValue']

Locators

2.Relative Xpath



The screenshot shows the top navigation bar of an Amazon search results page. The search bar contains the text "Search term". Above the search bar, the Amazon logo and the text "Deliver to Netherlands" are visible. Below the search bar, there are links for "All", "Today's Deals", "Customer Service", "Registry", "Gift Cards", and "Sell". To the right of the search bar is a yellow search button with a magnifying glass icon.

```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

Unique deger icin 3 bilesenin tamami kullanilmak zorunda degildir.

Hedef unique olarak webelement'i locate etmektir. Daha az bilesenle de unique degere ulasabilen Xpath'ler de kullanilabilir.

```
driver.findElement(By.xpath( xpathExpression: "//input" ));
```

Sadece tag ismi ile

```
driver.findElement(By.xpath( xpathExpression: "// * [@type='text']" ));
```

tag ismi farketmeksizin

```
driver.findElement(By.xpath( xpathExpression: "//input[@ *= 'text']" ));
```

Attribute farketmeksizin

```
driver.findElement(By.xpath( xpathExpression: "//input[@type]" ));
```

Attribute value farketmeksizin

```
driver.findElement(By.xpath( xpathExpression: "//div[@id='logo' or class='flex-col logo' ] "));
```

```
driver.findElement(By.xpath( xpathExpression: "//div[@id='logo' and class='flex-col logo' ] "));
```

Locators

2. Relative Xpath

Link disindaki bazi webelementler'inde de text bulunabilir.

Bu text'ler o webelement'e ozel oldugu icin unique bir xpath elde etmek icin kullanisli olabilirler.

Text ile locate yazmak icin kullanilan genel syntax :

```
//tagName[text()='yazinin tamami']"
```

Genel xpath kullanimina uygun olarak tagname veya attribute ismi yazilmadan da text ile xpath yazilabilir.

```
//tagname[.='yazinin tamami']"
```

```
//*[.='yazinin tamami'] "
```

Metnin sadece bir kismi kullanilacaksa

```
//*[contains(text(),'yazinin bir bolumu')] "
```

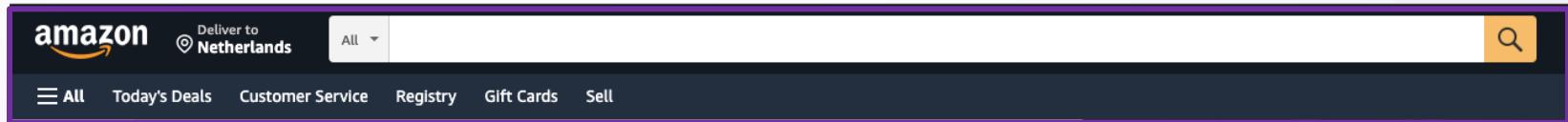
Locators

Relative Xpath Soru

- 1- https://the-internet.herokuapp.com/add_remove_elements/ adresine gidin
- 2- Add Element butonuna basin
- 3- Delete butonu'nun gorunur oldugunu test edin
- 4- Delete tusuna basin
- 5- “Add/Remove Elements” yazisinin gorunur oldugunu test edin

Locators

8.cssSelector



```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

cssSelector de xpath'e benzer bir kullanıma sahiptir. Tag ismi, attribute ismi ve attribute value ile yapılacak kombinasyonlarla oluşturulur.

```
driver.findElement(By.cssSelector("input[id='twotabsearchtextbox']"));
```

Hedef unique olarak webelement'i locate etmektir. Daha az bilesenle de unique degere ulasılabilen Xpath'ler de kullanılabilir.

cssSelector özellikle class ve id attribute'leri ile kısa şekilde yazılabilir

```
driver.findElement(By.cssSelector("#twotabsearchtextbox));
```

Id attribute ile

```
driver.findElement(By.cssSelector(".nav-input nav-progressive-attribute"));
```

Class attribute ile



Tekrar Sorusu

- 1- bir class olusturun
- 2- [https://www.amazon.com/ adresine gidin](https://www.amazon.com/)
- 3- Browseri tam sayfa yapın
- 4- Sayfayı "refresh" yapın
- 5- Sayfa basliginin "Spend less" ifadesi içerdigini test edin
- 6- Gift Cards sekmesine basin
- 7- Birthday butonuna basin
- 8- Best Seller bolumunden ilk urunu tiklayin
- 9- Gift card details'den 25 \$'i secin
- 10-Urun ucretinin 25\$ oldugunu test edin
- 11-Sayfayı kapatın

Relative Locators

Relative Locators nedir ?

Bir web elementi direk locate edemedigimiz durumlarda gunluk hayatimizda kullandigimiz sekilde o web elementi etrafindaki web elementlerin referansi ile tarif edebiliriz.

Ornegin yandaki resimde Berlin icin bir cok relative locator tanimlayabiliriz.

- Boston'in saginda , Sailor'in ustunde
- NYC'nin altinda, Bay Area'nin solunda
- Boston yakinlarinda Bay Areanin solunda ve Toronto'nun saginda vb..



Bu ozellik Selenium 4 ile gelen yeniliklerden biridir.

<https://www.diemol.com/selenium-4-demo/relative-locators-demo.html>

Relative Locators

Class Work: Relative Locators

- 1) <https://www.diemol.com/selenium-4-demo/relative-locators-demo.html>
adresine gidin
- 2) Berlin'i 3 farkli relative locator ile locate edin
- 3) Relative locator'larin dogru calistigini test edin





Relative Locators

```
driver.get("https://www.diemol.com/selenium-4-demo/relative-locators-demo.html#");

WebElement boston=driver.findElement(By.id("boston"));

WebElement sailor = driver.findElement(By.id("sailor"));

WebElement berlin = driver.findElement(with(By.tagName("li")).above(sailor).toRightOf(boston));

WebElement mountie=driver.findElement(with(By.className("ui-li-has-thumb")).below(boston));
```



Maven



Apache Maven yazılım projelerinin kolay anlasılmasına ve yönetilmesine odaklanmış bir tool'dur.

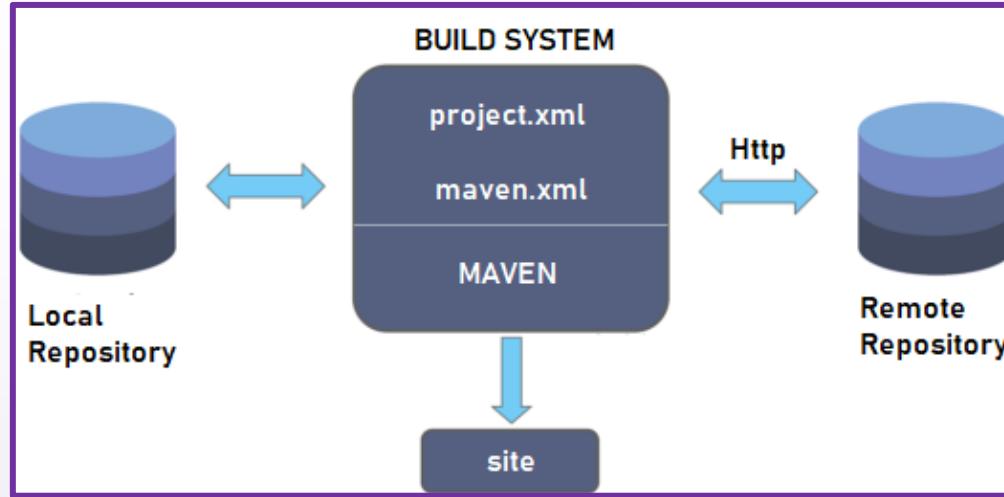
Proje nesne modeli (POM) konseptine dayalı olarak Maven, bir projenin inşasını, raporlamasını ve dokümantasyonunu merkezi bir bilgi parçasından(**dependency**) yönetebilir.

- 1- Projeleri oluşturma için standart belirleme,
- 2- projenin nelerden olduğunu net olarak tanımlama,
- 3- proje bilgilerini yayinallyamanın kolay bir yolunu bulma
- 4- JAR dosyalarını farklı projeler arasında paylaşma

Amaçları çerçevesinde başlayan bir çalışma sonucu ortaya çıktı.

Sadece Java tabanlı projeleri oluşturmak ve yönetmek için kullanılabilecek bir araçtır.

Maven



Maven bir Java oluşturma aracıdır (**build tool**). Maven proje otomasyon ve yönetim aracıdır (**automation and management tool**).

Maven, konfigürasyon için pom.xml dosyasını kullanır.

pom.xml projenin insası , raporlaması ve dokümantasyonu için gerekli bütün bilgileri içerir (dependencies , plugins v)

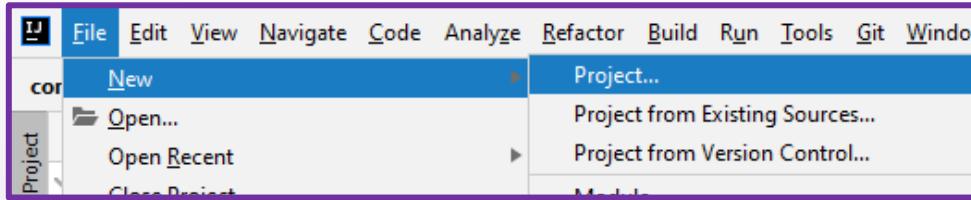
Maven ile çalışan tüm projeler aynı konfigürasyonu kullandığı için yeni bir projede çalışmaya başlandığında projenin anlaşılması çok kolay olacaktır.

Neden Maven ?

- 1- Proje yönetiminde oluşturma, belgeleme, yayınılama ve dağıtım gibi tüm süreçlerin yönetilmesine yardımcı olur.
- 2- Projenin ve yapılm sürecinin performansını artırır.
- 3- Jar dosyalarını ve diğer bağımlılıkları (dependencies) indirme görevi otomatik olarak yapılır
- 4- Gerekli tüm bilgilere kolay erişim sağlar
- 5- Geliştiricinin, bağımlılıklar, süreçler vb. hakkında endişelenmeden farklı ortamlarda bir proje oluşturmasını kolaylaştırır.
- 6- Open source olduğundan ücretsizdir, geniş bir kullanıcı tabanı olduğu için karşıtlan sorunlara çözüm bulmakta zorluk yaşanmaz.



Maven Proje Olusturma

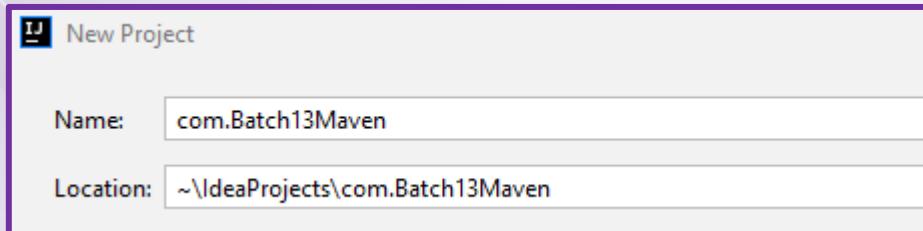


1. Create Project: File -> New -> Project



2. Select Maven -> click next

Java versiyonunun bilgisayarinizdaki version ile ayni oldugunu control edin



3. Name: com.projeAdi -> click finish

EnableAutoImport sorarsa click

4. Package olusturun, name : day04

5. Classolusturun, name : FirstMavenClass

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>

<groupId>org.example</groupId>
<artifactId>SeleniumInt</artifactId>
<version>1.0-SNAPSHOT</version>

<properties>
  <maven.compiler.source>11</maven.compiler.source>
  <maven.compiler.target>11</maven.compiler.target>
</properties>

<dependencies>
  <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
  <dependency>...</dependency>

  <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
  <dependency>...</dependency>
</dependencies>
</project>
```

Project Object Model (pom) Maven projesinin temelini oluşturur.

pom.xml proje hakkında bazi bilgiler ve Maven tarafından projeyi olusturmak icin kullanılan konfigurasyon detaylarini gösterir.

Bir projenin hangi framework'u kullandigini anlamak icin pom.xml'e bakmak yeterlidir.

POM'da belirtilebilecek yapılandırmalardan bazıları proje bağımlılıkları(dependencies), yürütülebilecek eklentiler(plugin) veya hedefler(goal), yapı profilleri vb. Proje sürümü, açıklama, geliştiriciler (artifact id, group id, version) ve benzeri gibi diğer bilgiler de belirtilebilir.

Projeye eklenecek dependency'ler mvnrepository.com'dan bulunabilir.

Istenen dependency icin version secilirken guncellik, stabil versiyon olma ve kullanılma sayiları dikkate alınmalıdır.



```
<properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
</properties>

<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency...>

    <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
    <dependency...>
</dependencies>
</project>
```

- 1- pom.xml'de </properties> kapanis tagi ile </project> kapanis tagi arasinda <dependencies> acilis ve </dependencies> kapanis tag'larini olusturun.
- 2- mvnrepositories.com adresinden WebDriverManager ve Selenium Java dependency'lerini kopyalayip, dependencies taglari arasina yapistirin
- 3- Bu dependency'leri projenize ilk defa yuklediginiz icin versiyon bolumleri kirmizi cikacaktir. Kirmiziliği gidermek icin 4. adimi takip edin.

```
<version>4.5.0</version>
```

pom.xml'e Dependency Ekleme



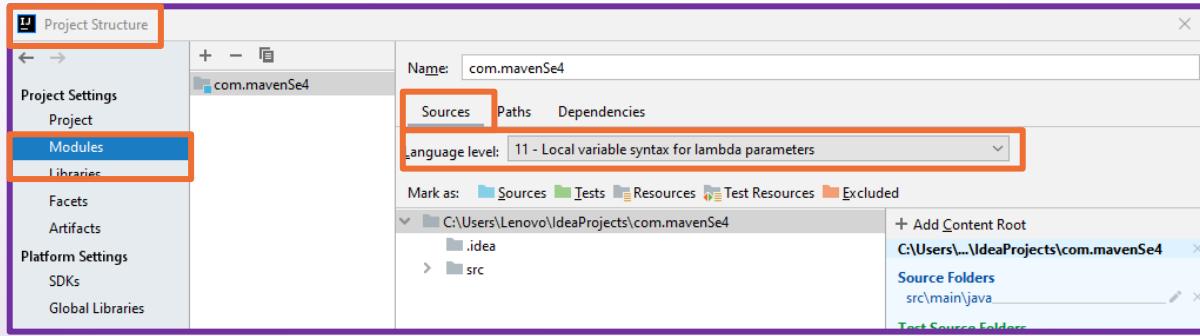
4- projenin sag ust kisminda bulunan Maven yazisini tiklayin, acilan bolumde yenile butonuna tiklayin ve yuklediginiz dependency'lerin projenize eklendigini kontrol edin.

The screenshot shows an IDE interface with several tabs at the top: Test.java, pom.xml (SeleniumInt), C03_linkText.java, C05_noSuchElementExc.java, C06_webElementMethodlari.java, C01_Xpath.java, C01_Xpath2.java, and Maven. The Maven tab is active, showing the Maven tool window on the right. The window has sections for Seleniumpit, Lifecycle, Plugins, Dependencies, and Notifications. The Dependencies section lists org.seleniumhq.selenium:selenium-java:4.4.0 and io.github.bonigarcia:webdrivermanager:5.3.0. The pom.xml file in the editor has its dependencies section highlighted with a red box. Inside this box, the first dependency for selenium-java is also highlighted with a red box. The XML code for the highlighted dependency is:

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>4.4.0</version>
</dependency>
```



pom.xml'e Dependency Ekleme

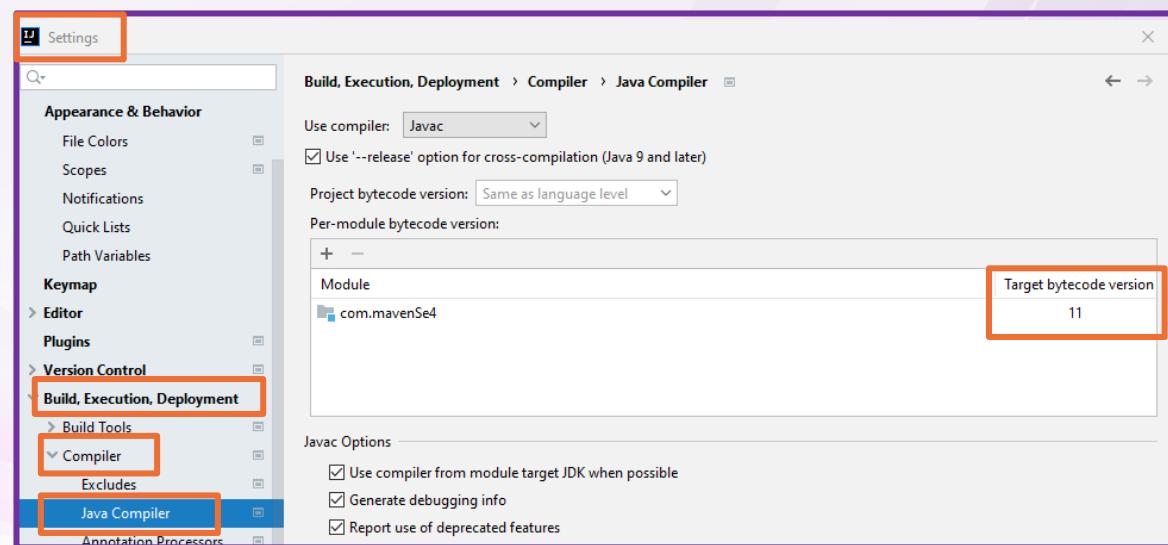


File

Project Structure
Modules
Sources

Language level : min 8 yapın,
bilgisayarınızda kurulu java 8
veya üstü ise java versiyonu
aynı olmalı

File
Settings
Build,Execution,Deployment
Compiler
Java Compiler
Target bytecode version : min 8
yapın, bilgisayarınızda kurulu
java 8 veya üstü ise java
versiyonu aynı olmalı



Maven WebDriver Oluşturma

Maven ile Selenium Java ve WebDriverManager dependency'lerini projemize eklediği için, her seferinde driver.exe dosyasını driver'a tanıtma mecburiyeti kalmadı.

```
System.setProperty("webdriver.chrome.driver", "src/resources/chromedriver");
WebDriver driver=new ChromeDriver();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
driver.manage().window().maximize();
```

Bundan sonra driver ayarları yapılarken ilk satırda sistem ayarlarını bonigarcia WebDriverManager kullanarak yapacağız.

```
WebDriverManager.chromedriver().setup();
WebDriver driver=new ChromeDriver();
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
```

Chrome之外のブラウザを使用する場合は、1. と 2. 行で Chrome ではなくその他のブラウザを選択する必要があります。



Maven ClassWork

Class Work Amazon Search Test

- 1- <https://www.amazon.com/> sayfasina gidelim
- 2- arama kutusunu locate edelim
- 3- “Samsung headphones” ile arama yapalim
- 4- Bulunan sonuc sayisini yazdiralim
- 5- Ilk urunu tiklayalim
- 6- Sayfadaki tum basliklari yazdiralim



Wise Quarter
first class IT courses

Selenium Team108 Ders-04

Maven
Junit Framework

+1 912 888 1630
www.wisequarter.com

/wisequarter

/wisequarter



The future at your fingertips

www.wisequarter.com



Onceki Dersten Aklimizda Kalanlar

- 1- Selenium browser'lari otamate edecek tool'larin calisabilecegi bir suite'dir. Kendi sitesinde "Selenium browser'lari otomasyonla calistirir, that's it!, bu gucle ne yapacagini size kalmis"
- 2- Selenium bu otomasyonu WebDriver sayesinde yapar. WebDriver herhangi bir browser'in ozelliklerini tasir, bunun icin oncelikle hangi browser ile calismak istiyorsak, onun ozelliklerini WebDriver objesine yuklemeliyiz.
- 3- Bu islem sayesinde WebDriver istenen browser'in ozelliklerine sahip kopya bir browser olusturur ve bu kopya browser ile istedigimiz islemleri yapar.
- 4- Maven ise proje build tool'dur. Selenium ile browser'lari otomasyon yaparken Selenium kutuphanesi ve ihtiyacimiz olan baska kutupaneleri projemize eklememiz gerekir.
- 5- Kutupaneleri projemize yuklemek, onlari guncel tutmak ve ihtiyacimiz oldugunda degistirmek jar dosyalari ile cok zor olur. Maven projelerin build ve maintenance islemlerini ustlenmistir.
- 6- Maven'in diger bir amaci da sonradan projemizi inceleyen bir collaborator'in projemizi daha Rahat bir sekilde anlamasini saglamaktir

Maven ClassWork



1. <http://zero.webappsecurity.com> sayfasina gidin
2. Signin buttonuna tiklayin
3. Login alanine “username” yazdirin
4. Password alanina “password” yazdirin
5. Sign in buttonuna tiklayin
6. Back tusu ile sayfaya donun
7. Online Banking menusunden Pay Bills sayfasina gidin
8. amount kismina yatirmak istediginiz herhangi bir miktarı yazin
9. tarih kismina “2020-09-10” yazdirin
10. Pay buttonuna tiklayin
11. “The payment was successfully submitted.” mesajinin ciktigini test edin

Maven Tekrar Testi

- 1- C01_TekrarTesti isimli bir class olusturun
- 2- <https://www.google.com/> adresine gidin
- 3- cookies uyarisini kabul ederek kapatın
- 4- Sayfa basliginin “Google” ifadesi icerdigini test edin
- 5- Arama cubuguna “Nutella” yazip aratin
- 6- Bulunan sonuc sayisini yazdirin
- 7- sonuc sayisinin 10 milyon’dan fazla oldugunu test edin
- 8- Sayfayı kapatın

Maven Tekrar Testi

1. “<https://www.saucedemo.com>” Adresine gidin
2. Username kutusuna “standard_user” yazdirin
3. Password kutusuna “secret_sauce” yazdirin
4. Login tusuna basin
5. Ilk urunun ismini kaydedin ve bu urunun sayfasina gidin
6. Add to Cart butonuna basin
7. Alisveris sepetine tiklayin
8. Sectiginiz urunun basarili olarak sepete eklendigini control edin
9. Sayfayı kapatın

JUnit

Java ile olusturulabilecek en temel Test Framework'udur.

Open-source bir kutuphanedir.

Developer'lar tarafindan yapılan unit test'ler icin de kullanilir.

Testlerimizi yaparken bize kolaylik saglamak amaciyla olusturulan Assert, Test, Before, After gibi bir çok class'a sahiptir.

JUnit, Maven altyapisini kullanir. JUnit framework icin ihtiyacimiz olan kutuphaneleri mvn.repository.com adresinden bulabilecegimiz dependency'ler ile projemize ekleyebiliriz.

JUnit'de testler icin ihtiyac duyulan pek çok olsa da TestNG next generation olarak daha fazla ozellik ile piyasaya cikmis ve piyasayı domine etmistir.



Annotations

JUnit ile Java class'larinin vazgecilmez ogesi olan main method ihtiyaci ortadan kalkar

Annotations ile compiler'a talimatlar verilebilir

Annotations kodların daha iyi anlasilabilmesi ve daha iyi bir yapı kurulabilmesi icin gerekli meta-data (basit bilgi)'lari Java'ya iletmek icin kullanılır.

Testler sirasinda en cok kullanılan Junit notasyonları

@Test

@Ignore

@Before , @After

@BeforeClass , @AfterClass

@BeforeClass

@Before

@Test

@After

@Before

@Test

@After

@AfterClass

@Test ve @Ignore Annotations

`@Test` notasyonu bir method'un bagimsiz olarak calisabilmesini saglar.

`@Test` notasyonu olan method'un calismasi icin Java'daki gibi method call yapilmasina ihtiyac yoktur, bagimsiz olarak calistirilabilir.

Class level'daki run butonu ile class'daki tum test method'lari da birlikte calistirilabilir.

Junit'in method'lari calistirma sirasi ongorulemez. Bunun icin her test method'u bagimsiz olarak calistirilabilir olmalidir.

`@Ignore` notasyonun tanimli olduğu metotlar test sırasında calistirilmayacaktır. Ayrca istenilirse `@Ignore("açıklama")` şeklinde yazilarak metodun neden test edilmesini istemedigimizide yazabiliriz.

```
6  public class C01_JUnitIlkTest {  
7  
8      @Test  
9      public void test01(){  
10         System.out.println("test01");  
11     }  
12  
13      @Test @Ignore  
14      public void test02(){  
15         System.out.println("test02");  
16     }  
17  
18      @Test  
19      public void test03(){  
20         System.out.println("test03");  
21     }  
22 }
```



JUnit

@Test Annotation

JUnit standart olarak, calistirilan her test method'unun sorunsuz olarak calisip calismadigini raporlar.

Calistirilan testlerin kac tanesinin passed, kac tanesinin failed oldugunu yazar.

Passed olan testler yesil tik ile, failed olan testler kirmizi carpi isareti ile, ignore edilen testler ise gri bir daire uzerine bir cizgi cekilerek isaretlenir,

Ancak JUnit'in dogru raporlama yapabilmesi icin Assert class'indan method'lar kullanilmalidir. Aksi takdirde sadece test method'unun sorunsuz calismis oldugunu raporlar.

```
@Test  
public void test01(){  
    System.out.println("test01");  
}
```

```
@Test @Ignore  
public void test02(){  
    System.out.println("test02");  
}
```

```
@Test  
public void test03(){  
    Assert.assertEquals( expected: 5, actual: 7);  
    System.out.println("test03");  
}
```

The screenshot shows a JUnit test run window. On the left, a tree view lists the test class 'C01_JUnitIlikTest' with three children: 'test01' (green checkmark), 'test02' (grey circle), and 'test03' (red X). To the right, the results are displayed:

- test01: Test passed.
- test02: Test ignored.
- test03: Test failed. The output shows:

```
java.lang.AssertionError:  
Expected :5  
Actual   :7  
<Click to see difference>
```

@Before - @After Annotations

@Before ve @After notasyonuna sahip method'lar her @Test method'undan once ve sonra calisirlar.

Bu method'lar sayesinde driver olusturulurken yaptigimiz ayarlari ve test method'u bittikten sonra driver'in kapatilmasi gibi gorevler icin tekrar tekrar kod yazmak mecburiyeti ortadan kalkar.

Genellikle @Before notasyonu kullanan method icin **setup**, @After notasyonu kullanan method icin **teardown** isimleri kullanilir.

```
WebDriver driver;
@Before
public void setUp(){
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
}
@Test
public void amazonTest(){
    driver.get("https://www.amazon.com");
    System.out.println(driver.getTitle());
}
@Test
public void facebookTest(){
    driver.get("https://www.facebook.com");
    System.out.println(driver.getTitle());
}
@Test
public void bestbuyTest(){
    driver.get("https://www.bestbuy.com");
    System.out.println(driver.getTitle());
}
@After
public void tearDown(){
    driver.close();
}
```



Test Vs Test Method'u

Test method'u @Test notasyonu kullanilarak olusturulan, tek basina da veya baska test methodlari ile birlikte calistirilabilen bir test case'dir.

Test ise genellikle birden fazla method, class veya package icerebilen, belirli bir amac icin calistirilan test method'larinin bütünüdür.

Ornek olarak, smoke test, regression test veya end2end testlerini söyleyebiliriz.

```
WebDriver driver;
@Before
public void setUp(){
    WebDriverManager.chromedriver().setup();
    driver = new ChromeDriver();
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
}
@Test
public void amazonTest(){
    driver.get("https://www.amazon.com");
    System.out.println(driver.getTitle());
}
@Test
public void facebookTest(){
    driver.get("https://www.facebook.com");
    System.out.println(driver.getTitle());
}
@Test
public void bestbuyTest(){
    driver.get("https://www.bestbuy.com");
    System.out.println(driver.getTitle());
}
@After
public void tearDown(){
    driver.close();
}
```

@BeforeClass - @AfterClass Annotations

Eger bir class'daki birden fazla test method'u icin driver'in bir kere olusturulmasi ve tum testleri bittikten sonra driver'in kapatilmasi yeterli olacaksa kullanilirlar.

Boylece driver'i tekrar tekrar acip kapatmak zorunda kalmaz.

NOT : @BeforeClass ve @AfterClass notasyonu kullanacak method'lar static olmak zorundadir.

```
// amazon sayfasina gidin
// uc ayri test method'u olusturup
// Nutella, java ve Selenium icin arama yapip, arama sonuclarini yazdirin

static WebDriver driver;

@BeforeClass
public static void setup(){
    WebDriverManager.chromedriver().setup();
    driver=new ChromeDriver();
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));
    driver.manage().window().maximize();
}

@Test
public void testNutella(){...}

@Test
public void testJava(){...}

@Test
public void testSelenium(){...}

@AfterClass
public static void teardown(){
    driver.close();
}
```

Annotations

Bir class'da hem @Before - @After notasyonları, hem de @BeforeClass-@AfterClass notasyonları birlikte kullanılabılır.

Birlikte kullanımda method'ların yapacakları işlemler ve yapıları, çalışma sırası ve notasyon özelliklerine göre düzenlenmelidir.

Ornek : 2 Test method'u

@Before - @After notasyonları

@BeforeClass-@AfterClass notasyonları olan bir class çalıştırılırsa, yandaki gibi bir çalışma olacak ve toplam 8 method çalıştırılmış olacaktır.





JUnit

Assertions

JUnit standart olarak, calistirilan her test method'unun sorunsuz olarak calisip calismadigini raporlar.

JUnit Assert Class'indan static method'lar kullanilarak Actual Ressult - Expected Result karsilastirilir.

Bu karsilastirma bir boolean sonuc uretir.

Assert islemi boolean sonucun **true** veya **false** olmasina degil, beklenen sonuc ile ayni olup olmadigina gore FAILED veya PASSED sonunu uretir.

```
public class Assertions {  
    int sayi1= 10;  
    int sayi2= 20;  
    int sayi3= 30;  
  
    @Test  
    public void test02(){  
        Assert.assertEquals(sayi3, sayi1+sayi2); // PASSED  
    }  
    @Test  
    public void test03(){  
        Assert.assertNotEquals(sayi3, sayi2); // PASSED  
    }  
  
    @Test  
    public void test04(){  
        Assert.assertTrue(sayi3==sayi2); // FAILED  
    }  
  
    @Test  
    public void test05(){  
        Assert.assertFalse(sayi3==sayi2); // Passed  
    }  
}
```

Run: Assertions x

Test	Status
test02	Passed
test03	Passed
test04	Failed
test05	Passed

Tests failed: 1, passed: 3 of 4 tests – 5 ms

/Users/ahmetbulutluoz/Library/Java/J

java.lang.AssertionError <3 internal
at ders06_junit.Assertions.test04()

Process finished with exit code 255

Assertions

JUnit ile assertion icin en cok kullanilan 4 method vardir.

```
Assert.assertEquals(a, c);
Assert.assertNotEquals(b, c);
Assert.assertTrue( condition: a>b );
Assert.assertFalse( condition: a<=b );
```

Kullanilacak method secilirken, assert isleminin icerisine yazilan boolean islem sonucunun ne olmasi beklendiği incelenmeli, assert method'u da beklenen sonuca uyumlu olarak secilmelidir.

Sonucun 20 oldugunu test edin → Assert.assertEquals(sonuc , 20)

Sonucun succesfull olmadigini test edin. → Assert.assertNotEquals(sonuc, "succesfull")

Sayinin 50'den buyuk oldugunu test edin. → Assert.assertTrue(sayi>50)

Sayinin 50'den buyuk olmadigini test edin. → Assert.assertFalse(sayi>50)



JUnit

```
@Test
```

```
public void test02(){  
    int a= 10, b=20, c=30;  
    String str1="Ali", str2="ALI";
```

```
    Assert.assertEquals(str1,str2);
```

Not equals

Failed

```
    Assert.assertNotEquals(str1,str2);
```

Not equals

Passed

```
    Assert.assertTrue(str1.equalsIgnoreCase(str2));
```

True

Passed

```
    Assert.assertFalse(str1.equals(str2));
```

False

Passed

```
    Assert.assertFalse(condition: a>b);
```

False

Passed

```
    Assert.assertEquals(c, actual: a+b);
```

equals

Passed

```
    Assert.assertNotEquals(b,c);
```

Not equals

Passed

```
    Assert.assertTrue(condition: c>b);
```

true

Passed

Assertions

- 1) Bir class oluşturun: BestBuyAssertions
- 2) <https://www.bestbuy.com/> Adresine gidin farkli test method'lari olusturarak asagidaki testleri yapin
 - o Sayfa URLinin <https://www.bestbuy.com/> 'a esit oldugunu test edin
 - o titleTest => Sayfa başlığının “Rest” içermediğini(contains) test edin
 - o logoTest => BestBuy logosunun görüntünlendiğini test edin
 - o FrancaisLinkTest => Fransizca Linkin görüntülendiğini test edin



Wise Quarter
first class IT courses

Selenium Team108 Ders-05

Assertion
Check Box – Radio Button
Handle Dropdown

+1 912 888 1630

www.wisequarter.com



The future at your fingertips

Assertions

- 1) Bir class oluşturun: YoutubeAssertions
- 2) <https://www.youtube.com> adresine gidin
- 3) Aşağıdaki adları kullanarak 4 test metodu oluşturun ve gerekli testleri yapın
 - titleTest => Sayfa başlığının “YouTube” olduğunu test edin
 - imageTest => YouTube resminin görüntünlendiğini (isDisplayed()) test edin
 - Search Box 'in erisilebilir olduğunu test edin (isEnabled())
 - wrongTitleTest => Sayfa basliginin “youtube” olmadığını doğrulayın

Assertions

1. Bir Class olusturalim YanlisEmailTesti
2. <http://automationpractice.com/index.php> sayfasina gidelim
3. Sign in butonuna basalim
4. Email kutusuna @isareti olmayan bir mail yazip enter'a bastigimizda "Invalid email address" uyarisi ciktigini test edelim

Check Box

Gerekli yapıyi olusturun ve aşağıdaki görevi tamamlayın.

- a. Verilen web sayfasına gidin.

<https://the-internet.herokuapp.com/checkboxes>

- b. Checkbox1 ve checkbox2 elementlerini locate edin.
c. Checkbox1 seçili değilse onay kutusunu tıklayın
d. Checkbox2 seçili değilse onay kutusunu tıklayın
e. Checkbox1 ve Checkbox2'nin seçili olduğunu test edin

Specialty Food Type

- GMO-Free
- Organic
- USDA Organic

Calories Per Serving

- 0 Calories
- Up to 40 Calories
- 40 to 100 Calories
- 100 to 200 Calories
- 200 to 300 Calories

Fat Calories Per Serving

- 40-100 Calories

Nutrition Facts Per Serving

- Fat Free (<0.5g)
- Low Fat (<3g)
- Free of Saturated Fat (<0.5g)
- Free of Trans Fat (0g)
- Cholesterol Free (<2mg)
- Sodium Free (<5mg)
- Low Sodium (<140mg)
- Carbohydrate Free (<0.5g)
- Sugar Free (<0.5g)
- Dietary Fiber (>10g)
- Protein (>10g)

Food Diet Type

- Vegan

Availability

- Include Out of Stock

JUnit

Radio Buttons

RADIO BUTTONS	CHECKBOXES
<input type="radio"/> _____	<input type="checkbox"/> _____
<input checked="" type="radio"/> _____	<input checked="" type="checkbox"/> _____
<input type="radio"/> _____	<input type="checkbox"/> _____
<input type="radio"/> _____	<input checked="" type="checkbox"/> _____

Function: Mutually exclusive **Function:** Mutually inclusive

Usage: Option list selection **Usage:** Option list selection

Gerekli yapıyi olusturun ve aşağıdaki görevi tamamlayın.

- a. Verilen web sayfasına gidin.

<https://facebook.com>

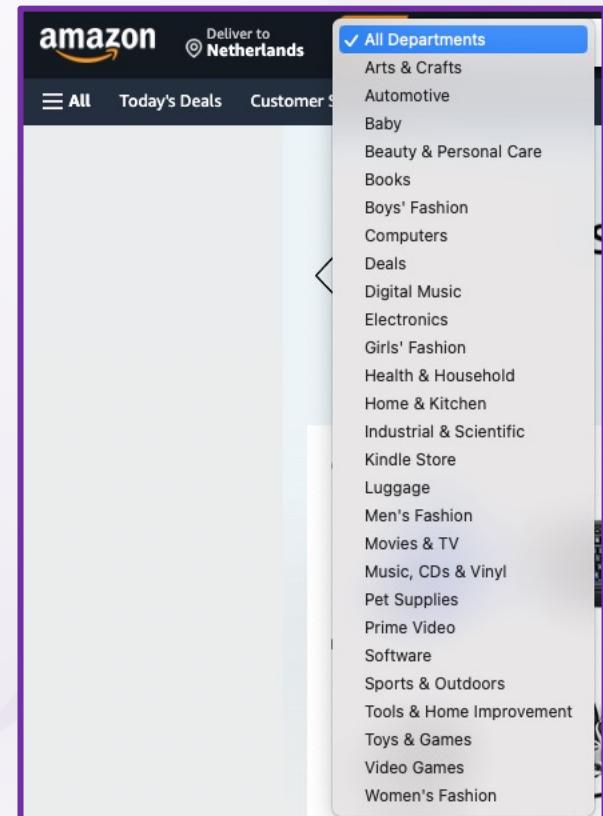
- b. Cookies'i kabul edin
- c. Create an account buton'una basin
- d. Radio button elementlerini locate edin ve size uygun olanı secin
- e. Sectiginiz radio button'un seçili, ötekilerin seçili olmadigini test edin

Handle Dropdown

Dropdown(acilir menu) ozel bir HTML kodu ile olusturulur.

HTML sayfalarda farkli acilir menuler yapilabilir. Dropdown'i digerlerinden ayiran tag'inin **<select>** olmasi ve secilebilecek opsiyonların da **<option>** tag'i ile olusturulmasidir.

Selenium dropdown menu ile islem yapilabilmesi icin ozel bir Select class'i olusturmustur. Select class'indan olusturacak obje yardimi ile bu class'daki method'lar kullanilabilir.



Handle Dropdown

Dropdown'daki opsiyonlardan birini secmek icin 3 islem yapilir.

1- Dropdown webelement'i locate edin

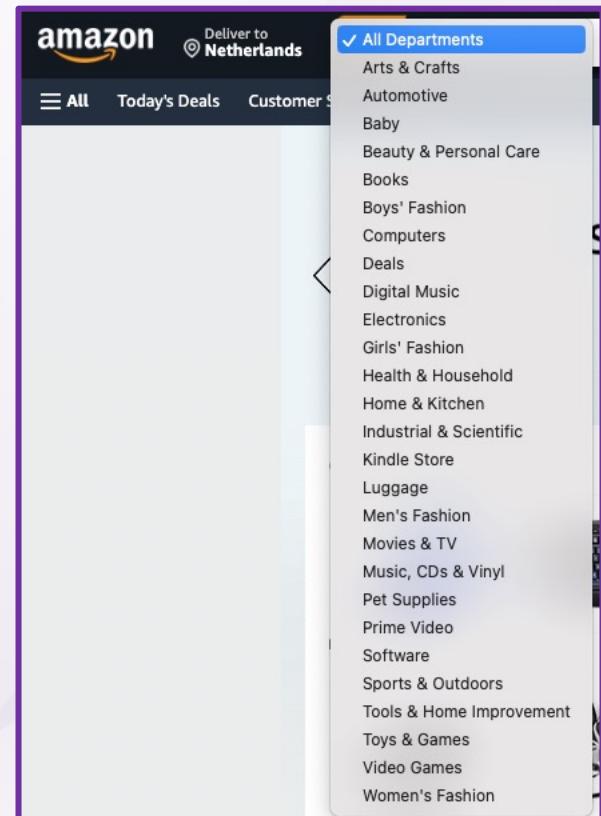
```
WebElement ddm= driver.findElement(By.id("searchDropdownBox"));
```

2- Select class'indan bir obje olusturun ve locate edilen dropdown elementi parametre olarak yazin

```
Select select= new Select(ddm);
```

3- select objesi ile Select class'inda bulunan method'lardan uygun olani ile istediginiz option'i secin.

```
select.selectByVisibleText("Electronics");
```



Select Class Method'ları

1- istenen option'i secme

```
select.selectByIndex(1);  
select.selectByValue("2");  
select.selectByVisibleText("Option 1");
```

2- Bir option secildikten sonra secilen option'i
webelement olarak döndürme

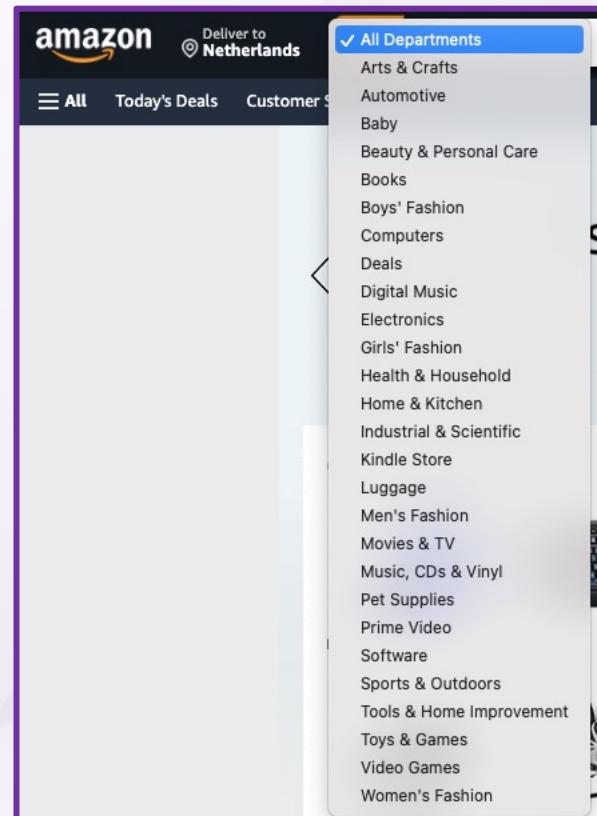
```
select.getFirstSelectedOption()
```

3- Bir option secildikten sonra secilen option'i text
olarak döndürme

```
select.getFirstSelectedOption().getText()
```

4- Tum option'ları döndürüp kaydetme

```
List<WebElement> optionsList= select.getOptions();
```



Handle Dropdown

- Bir class oluşturun: **DropDown**
- <https://the-internet.herokuapp.com/dropdown> adresine gidin.
 1. **Index** kullanarak Seçenek 1'i (Option 1) seçin ve yazdırın
 2. **Value** kullanarak Seçenek 2'yi (Option 2) seçin ve yazdırın
 3. **Visible Text**(Görünen metin) kullanarak Seçenek 1'i (Option 1) seçin ve yazdırın
 4. Tüm dropdown değerleri(value) yazdırın
 5. Dropdown'un boyutunun 4 olduğunu test edin

Handle Dropdown

- Bir class oluşturun: C3_DropDownAmazon
- <https://www.amazon.com/> adresine gidin.
 - Test 1

Arama kutusunun yanindaki kategori menusundeki kategori sayisinin 45 oldugunu test edin

-Test 2

1. Kategori menusunden Books secenegini secin
2. Arama kutusuna Java yazin ve aratin
3. Bulunan sonuc sayisini yazdirin
4. Sonucun Java kelimesini icerdigini test edin



Handle Dropdown

1. <http://zero.webappsecurity.com/> Adresine gidin
2. Sign in butonuna basin
3. Login kutusuna “username” yazın
4. Password kutusuna “password.” yazın
5. Sign in tusuna basin
6. Pay Bills sayfasına gidin
7. “Purchase Foreign Currency” tusuna basin
8. “Currency” drop down menusunden Eurozone’u secin
9. “amount” kutusuna bir sayı girin
10. “US Dollars” in secilmədigini test edin
11. “Selected currency” butonunu secin
12. “Calculate Costs” butonuna basin sonra “purchase” butonuna basin
13. “Foreign currency cash was successfully purchased.” yazısının cıktığını kontrol edin.

JUnit

TestBase Class

Her test method'u calisirken webDriver objesine ve bu obje icin ayarlara ihtiyac vardır. Bu islemleri her class icin yeniden yasmak yerine Java'daki **Inheritance** ozelligi kullanilabilir.

Bu islemlerin yapildigi **@Before** ve **@After** notasyonuna sahip method'lar olusturulan bir TestBase class'ina konulabilir.

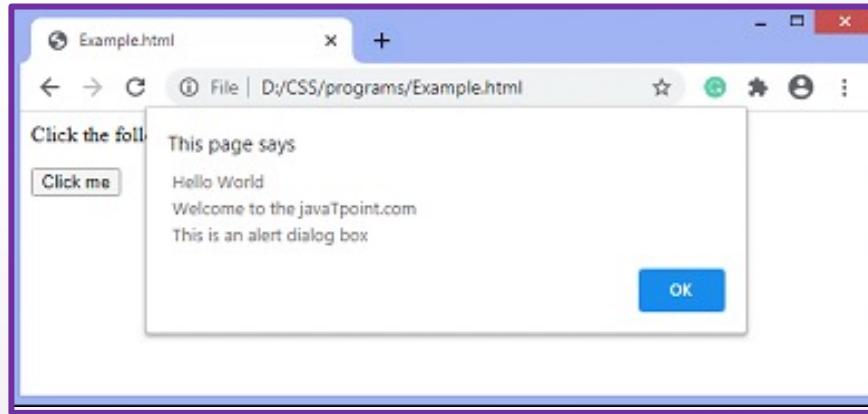
Testlerin yapilacagi class'lar extends keyword ile TestBase class'ina child class yapip, oradaki setup ve teardown method'lari direk kullanilabilir.

```
public class TestBase {  
  
    protected WebDriver driver;  
  
    @Before  
    public void setup(){  
        WebDriverManager.chromedriver().setup();  
        driver=new ChromeDriver();  
        driver.manage().window().maximize();  
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));  
    }  
  
    @After  
    public void teardown(){  
  
        driver.close();  
    }  
}
```

TestBase class'inda setup ve teardown method'lari disinda tekrar tekrar yapacagimiz islemleri yapan hazir method'lar da konulabilir.

Olusturulan driver objesi sadece child class'larin kullanabilmesi icin protected yapilabilir

JS Alerts



Alert Nedir?

Alert kullanıcıya bir tür bilgi vermek veya belirli bir işlemi gerçekleştirmek izni istemek için ekran bildirimini görüntüleyen küçük bir mesaj kutusudur. Uyarı amacıyla da kullanılabilir.

1- HTML Alerts

Bir alert çıktığında sağ click ile inspect yapabiliyorsak html alert'dir ve extra bir işlem gereklidir.

2- Js Alerts

Js alerts inspect yapılamaz, ekstra işlemi ihtiyaç vardır.

JS Alerts

1. Simple Alert : Bu basit alert ekranında bazı bilgiler veya uyarılar görüntüler. Ok denilerek kapatılır

2. Confirmation Alert : Bu onay uyarısı bir tür işlem yapma izni ister. Alert onaylanyorsa OK, onaylanmıyorsa Cancel butonuna basılır.

3. Prompt Alert : Bu Prompt Uyarısı kullanıcıdan bazı girdilerin girilmesini ister ve selenium webdriver metni sendkeys ("input....") kullanarak girebilir.

https://the-internet.herokuapp.com/javascript_alerts



JS Alerts

1. `accept()` : Alert üzerindeki OK butonuna basmak için kullanılır.

```
driver.switchTo().alert().accept();
```

2. `Dismiss()` : Alert üzerindeki OK butonuna basmak için kullanılır.

```
driver.switchTo().alert().dismiss();
```

3. `getText()` : Alert üzerindeki yazıyı döndürür.

```
driver.switchTo().alert().getText();
```

4. `sendKeys("istenen yazı")` : Alert üzerindeki text kutusuna istenilen metni yazdırır.

```
driver.switchTo().alert().sendKeys( keysToSend: "Deneme");
```

Click for JS Alert

Click for JS Confirm

Click for JS Prompt

3 test method'u olusturup asagidaki gorevi tamamlayin

1. Test

- https://the-internet.herokuapp.com/javascript_alerts adresine gidin
- 1.alert'e tiklayın
- Alert'deki yazının "I am a JS Alert" olduğunu test edin
- OK tusuna basıp alert'i kapatın

2. Test

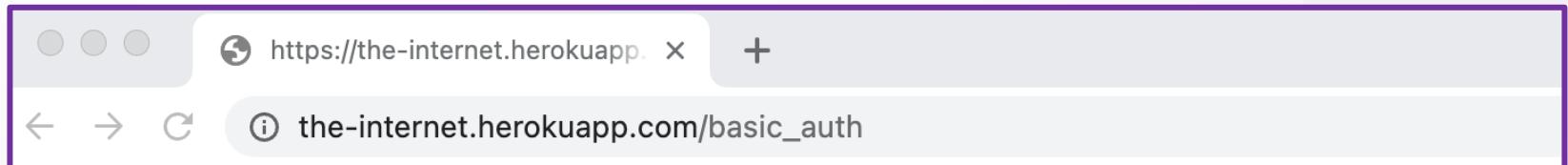
- https://the-internet.herokuapp.com/javascript_alerts adresine gidin
- 2.alert'e tiklayalım
- Cancel'a basıp, çıkan sonuc yazısının "You clicked: Cancel" olduğunu test edin

3. Test

- https://the-internet.herokuapp.com/javascript_alerts adresine gidin
- 3.alert'e tiklayalım
- Cikan prompt ekranına "Abdullah" yazdırıyalım
- OK tusuna basarak alert'i kapatıyalım
- Cikan sonuc yazısının Abdullah içerdigini test edelim

JUnit

Basic Authentication



Authentication Nedir?

Kısaca, herhangi bir internet kullanıcısının, uygulamanın ya da programın, söz konusu sisteme dahil olup olamayacağını belirleyen formu ifade eder.

Uygulama ana sayfalarındaki kullanıcı adı ve password istemek de bir authentication'dır.

End user'lar için tasarılanmayan uygulamalarda(Ornegin API sorgularında) bu authentication HTML komutları ile de yapılabilir.

Bu authentication'i yapabilmek için uygulamanın kullanıcılarına authentication'i nasıl yapacağına dair bilgilendirme yapmış olması gereklidir.

Ornegin yandaki uygulama için authentication aşağıdaki gibi yapılabilir.

<https://username:password@URL>



Wise Quarter
first class IT courses

Selenium Team108 Ders-06

Handle Iframe

Handle Windows

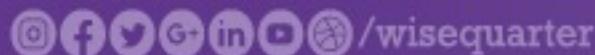
Actions Class

Faker Class



+1 912 888 1630

www.wisequarter.com





JUnit

Basic Authentication

The screenshot shows a web browser window with the URL https://the-internet.herokuapp.com/basic_auth. A modal dialog box titled "Sign in" is displayed, prompting for "Username" and "Password". Below the dialog is a "Cancel" button and a "Sign In" button.

1- Bir class olusturun : BasicAuthentication

2- https://the-internet.herokuapp.com/basic_auth sayfasina gidin

3- asagidaki yontem ve test datalarini kullanarak authentication'i yapin

4- Basarili sekilde sayfaya girildigini dogrulayin

Html komutu : <https://username:password@URL>

Username : admin

password : admin

```
driver.get("https://admin:admin@the-internet.herokuapp.com/basic_auth");
```

JUnit

Handle IFrame

HTML kodlarda kullanılan <iframe> tag'i bir HTML sayfasının içerisinde başka bir HTML sayfası gömmek(embed) için kullanılır.

Iframe'ler genellikle videoları, haritaları ve diğer medyaları bir web sayfasına gömmek için kullanılır. Ancak sadece bunlarla sınırlı degildir, her turlu HTML sayfası <iframe> tag'i ile kullanılabilir.

<iframe> tag'i içerisinde header ve body bulunur. Bir HTML sayfasında iframe varsa, HTML kodlar içerisinde birden fazla header veya body olacaktır.

Ornek iframe icin : <https://html.com/tags/iframe/>



JUnit

Handle IFrame

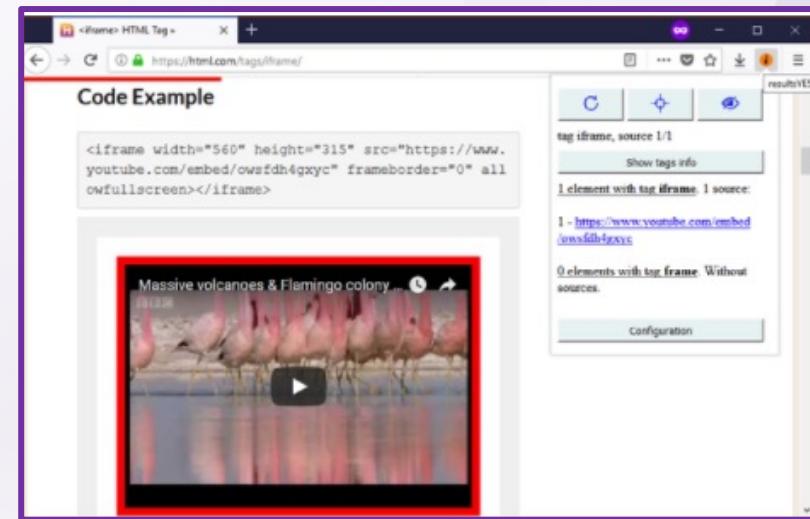
Bir websayfasında locate işlemi doğru yapıldığı halde istenen webelement'e ulaşamıyorsa, aranan elementin bir iframe içinde olup olmadığı kontrol edilmelidir.

Bir iframe içerisindeki webelementi kullanabilmek için driver'i iframe'e switch yapmak gereklidir.

Webdriver'i istenen iframe'e switch yapabilmek için iframe'i driver'a tanıtmak gereklidir. Bu tanıtma 3 farklı yolla yapılabilir.

1) Iframe'i webelement olarak locate ederek

```
driver.switchTo().frame(iframeElementi);
```



2) Iframe'in id veya name attribute value'su kullanılarak

3) Iframe'in index'i biliniyorsa, index kullanılarak

JUnit

Handle IFrame

Iframe icerisindeki webelement'e ulasmak icin iframe'e switch edildigi gibi, iframe icerisine gecis yaptiktan sonra iframe disindaki bir elemente erisebilmek icin de yeniden iframe'den anasayfaya gecis yapmak gereklidir.

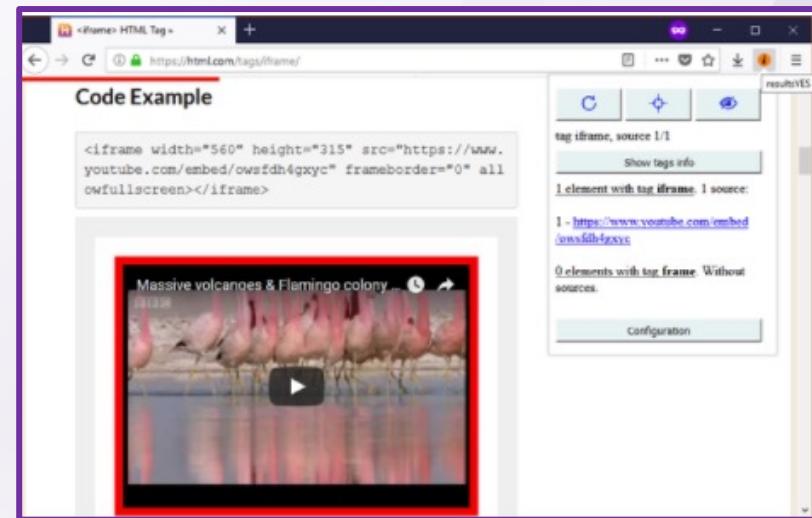
Iframe icerisindeyken oradan cikmak icin 2 yontem kullanilabilir.

1- Direk anasayfaya gecis yapmak icin

```
driver.switchTo().defaultContent();
```

2- Icice birden fazla iframe varsa, bir ust iframe'e cikmak icin

```
driver.switchTo().parentFrame();
```



JUnit

Handle IFrame

1) <https://the-internet.herokuapp.com/iframe> adresine gidin.

2) Bir metod olusturun: iframeTest

- “An IFrame containing....” textinin erisilebilir oldugunu test edin ve konsolda yazdirin.
- Text Box'a “Merhaba Dunya!” yazin.
- TextBox'in altinda bulunan “Elemental Selenium” linkini textinin gorunur oldugunu dogrulayin ve konsolda yazdirin.

- 1) <http://demo.guru99.com/test/guru99home/> sitesine gidiniz
- 2) sayfadaki iframe sayısını bulunuz.
- 3) ilk iframe'deki (Youtube) play butonuna tıklayınız.
- 4) ilk iframe'den çıkışp ana sayfaya dönünüz
- 5) ikinci iframe'deki (Jmeter Made Easy) linke
(<https://www.guru99.com/live-selenium-project.html>) tıklayınız

JUnit

Handle Windows

Opening a new window

[Click Here](#)

Powered by [Elemental Selenium](#)

<https://the-internet.herokuapp.com/windows>

Bir HTML sayfasında test yaparken, bazen isteyerek veya bir link tiklayarak yeni bir tab veya windows açılabilir.

Test sırasında test için yapılan tüm eylemleri webdriver objesi yaptığı için, yeni açılan sayfada işlem yapılabilmesi için webdriver'in yeni sayfaya geçis yapması gereklidir.

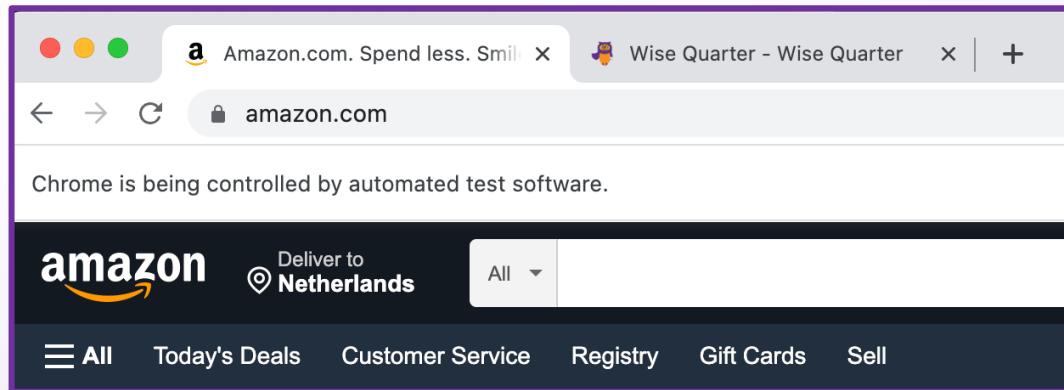
Selenium4 ile yeni gelen bir özellik olarak, test sırasında yeni bir tab veya window açılabilir.

```
driver.switchTo().newWindow(WindowType.TAB);
```

Bu method kullanıldığında yeni sayfa/tab driver ile açılduğundan driver otomatik olarak yeni sayfaya geçis yapar.

JUnit

Handle Windows



Verilen görevi yaparken tıkladığımız bir link, otomatik olarak yeni bir window açıyorsa driver'in yeni window'a geçmesi için kod yazmamız gereklidir.

```
driver.switchTo().window(ilkSayfaHandleDegeri);
```

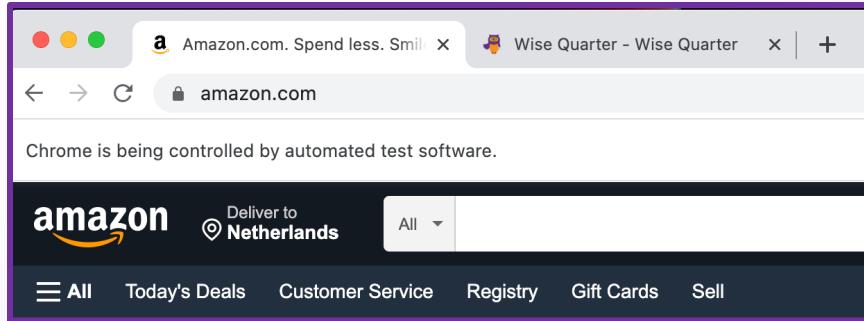
Bir window'a geçiş yapabilmek için o window'un sayfa handle değerine ihtiyaç vardır.

```
driver.getWindowHandle();
```

Bir window'un window handle değeri getWindowHandle() ile alınabilir, ancak bu method'u çalıştırabilmek için önce o sayfada olmak gereklidir.

JUnit

Handle Windows



Ikinci sayfa acildiginda o sayfaya gecmek icin window handle degerini driver ilk sayfada iken bulmamiz gerekir. Bunun icin hazir bir method yoktur ama yazacagimiz kod ile 3 adimla bu islemi yapmamiz mumkundur.

1- Ilk window'un window handle degerini kaydedin.

```
String ilkSayfaHandleDegeri= driver.getWindowHandle();
```

2- Ikinci window acildiktan sonra, iki sayfanin window handle degerini kaydedin.

```
Set<String> windowHandlesSeti= driver.getWindowHandles();
```

3- For-each loop ile set'deki window handle degerlerini kontrol edin, ilk sayfanin window handle degerine esit olmayani ikinci sayfanin window handle degeri olarak kaydedin.

- Yeni bir class olusturun: WindowHandle
- Amazon anasayfa adresine gidin.
- Sayfa'nin window handle degerini String bir degiskene atayin
- Sayfa title'nin "Amazon" icerdigini test edin
- Yeni bir tab olusturup, acilan tab'da wisequarter.com adresine gidin
- Sayfa title'nin "wisequarter" icerdigini test edin
- Yeni bir window olusturup, acilan sayfada walmart.com adresine gidin
- Sayfa title'nin "Walmart" icerdigini test edin
- Ilk acilan sayfaya donun ve amazon sayfasina dondugunu test edin

- Tests package'ında yeni bir class olusturun: WindowHandle2
- <https://the-internet.herokuapp.com/windows> adresine gidin.
- Sayfadaki textin “Opening a new window” olduğunu doğrulayın.
- Sayfa başlığının(title) “The Internet” olduğunu doğrulayın.
- Click Here butonuna basın.
- Acilan yeni pencerenin sayfa başlığının (title) “New Window” olduğunu doğrulayın.
- Sayfadaki textin “New Window” olduğunu doğrulayın.
- Bir önceki pencereye geri döndükten sonra sayfa başlığının “The Internet” olduğunu doğrulayın.

Actions class'i kullanılarak mouse ve klavye ile yapabilecek tüm islevler gerçekleştirilebilir.

Actions Class birçok kullanışlı mouse ve klavye method'una sahiptir.

- Çift tıklama (double click),
 - sürükleme ve bırakma(drag and drop)
 - mouse'u objeye getirme
(move to element)
- gibi karmaşık mouse eylemleri



veya Keyboard ile yapabileceğimiz pageUp, pageDown, shift, arrowDown gibi işlemleri Actions class'ından object ureterek driver ile yapabiliriz.

JUnit

Actions Class

1.Adım: Actions class'ta bir object oluşturulur.

```
Actions actions= new Actions(driver);
```

2. Adım: Üzerinde çalışmak istediğiniz WebElement locate edilir.

```
WebElement accountListElementi= driver.findElement(By.xpath( xpathExpression: "xpath"));
```

3.Adım : Ardından bu webelement üzerinde action gerçekleştirilir.

Örneğin Mouse'u istenen webelement'in üzerine getirmek için

```
actions.moveToElement(accountListElementi).perform();
```

NOT : Action Class'ini her kullanmak istedigimizde yeniden obje olusturmamız gerekmeyez.

NOT 2 : action objesi'ni bir kere olusturunca, istediginiz kadar action. ile baslayan komut yazar ve calismasi icin sonuna perform() yazariz.

action objesi kullanilarak baslayan her komut, calismak icin perform() bekler.

JUnit

Mouse Base Actions

doubleClick (): WebElement'e çift tıklama yapar

clickAndHold (): WebElement üzerinde click yapılı olarak bizden komut bekler.

dragAndDrop (): WebElement'i bir noktadan diğerine sürüklər ve bırakır

moveToElement (): Mouse'u istedigimiz WebElement'in üzerinde tutar

contextClick (): Mouse ile istedigimiz WebElement'e sağ tıklama yapar.





JUnit

Actions Class

- 1- Yeni bir class olusturalim: MouseActions1
- 2- https://the-internet.herokuapp.com/context_menu sitesine gidin
- 3- Cizili alan üzerinde sag click yapin
- 4- Alert'te cikan yazinin “You selected a context menu” oldugunu test edin.
- 5- Tamam diyerek alert'i kapatalim
- 6- Elemental Selenium linkine tiklayalim
- 7- Acilan sayfada h1 taginda “Elemental Selenium” yazdigini test edelim



JUnit

Actions Class

Yeni bir class olusturalim: MouseActions2

- 1- <https://demoqa.com/droppable> adresine gidelim
- 2- “Drag me” butonunu tutup “Drop here” kutusunun ustune birakalim
- 3- “Drop here” yazisi yerine “Dropped!” oldugunu test edin



Wise Quarter
first class IT courses

Selenium Team108 Ders-07

Actions Class

Faker Class

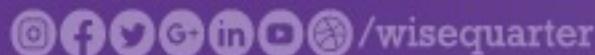
File Testleri



The future at your fingertips

+1 912 888 1630

www.wisequarter.com





JUnit

Actions Class

Yeni bir class olusturalim: MouseActions3

- 1- <https://www.amazon.com/> adresine gidin
- 2- Sağ üst bolumde bulunan “Account & Lists” menusunun acilmasi icin mouse'u bu menunun ustune getirin
- 3- “Create a list” butonuna basin
- 4- Acilan sayfada “Your Lists” yazisi oldugunu test edin

JUnit

Keyboard Base Actions

Action Class'indaki hazır method'lar ile klavyedeki tuslar kontrol edilebilir.

Klavyede çok fazla tus vardır ama tüm tuslar 3 temel işlev ile kontrol edilebilir.

1) **sendKeys ()**: Öğeye bir dizi anahtar gönderir

2) **keyDown ()**: Klavyede tuşa basma işlemi gerçekleştirir

3) **keyUp ()** : Klavyede tuşu serbest bırakma işlemi gerçekleştirir



JUnit

Keyboard Base Actions



- 1- Bir Class olusturalim KeyboardActions1
- 2- <https://www.amazon.com> sayfasina gidelim
- 3- Arama kutusuna actions method'larine kullanarak Samsung A71 yazdirin ve Enter'a basarak arama yaptirin
- 4- aramanin gerceklestigini test edin



JUnit

Keyboard Base Actions



- 1- <https://www.facebook.com> adresine gidelim
- 2- Yeni hesap olustur butonuna basalim
- 3- Ad, soyad, mail ve sifre kutularina deger yazalim ve kaydol tusuna basalim
- 4- Kaydol tusuna basalim

JUnit

Faker Kutuphanesi

Faker class'i testlerimizi yaparken ihtiyav duyduğumuz isim, soyisim, adres vb bilgiler icin fake degerler uretmemize imkan tanir.

Faker degerler uretmek icin Faker class'indan bir obje uretir ve var olan method'lari kullaniriz.

1. "https://facebook.com" Adresine gidin
2. "create new account" butonuna basin
3. "firstName" giris kutusuna bir isim yazin
4. "surname" giris kutusuna bir soyisim yazin
5. "email" giris kutusuna bir email yazin
6. "email" onay kutusuna emaili tekrar yazin
7. Bir sifre girin
8. Tarih icin gun secin
9. Tarih icin ay secin
10. Tarih icin yil secin
11. Cinsiyeti secin
12. Isaretlediginiz cinsiyetin secili, diger cinsiyet kutusunun secili olmadigini test edin.
13. Sayfayı kapatın

Keyboard Actions Homework

Yeni Class olusturun ActionsClassHomeWork

- 1- "http://webdriveruniversity.com/Actions" sayfasina gidin
- 2- Hover over Me First" kutusunun ustune gelin
- 3- Link 1" e tiklayin
- 4- Popup mesajini yazdirin
- 5- Popup'i tamam diyerek kapatin
- 6- "Click and hold" kutusuna basili tutun
- 7- "Click and hold" kutusunda cikan yaziyi yazdirin
- 8- "Double click me" butonunu cift tiklayin

Keyboard Actions Homework

1- Bir Class olusturalim KeyboardActions2

2- <https://html.com/tags/iframe/> sayfasina gidelim

3- video'yu gorecek kadar asagi inin

4- videoyu izlemek icin Play tusuna basin

5- videoyu calistirdiginizi test edin

Genel Tekrar Homework

Test01 :

- 1- amazon gidin
- 2- Arama kutusunun solundaki dropdown menunu handle edip listesini ekrana yazdırın
- 3- dropdown menude 40 eleman olduğunu doğrulayın

Test02

- 1- dropdown menuden elektronik bölümü seçin
- 2- arama kutusuna iphone yazıp aratin ve bulunan sonuç sayısını yazdırın
- 3- sonuc sayisi bildiren yazinin iphone içerdigini test edin
- 4- ikinci ürüne relative locater kullanarak tıklayın
- 5- ürünün title'ni ve fiyatını variable'a assign edip ürünü sepete ekleyelim

Test03

- 1- yeni bir sekme açarak amazon anasayfaya gidin
- 2-dropdown'dan bebek bölümüne secin
- 3-bebek puset aratıp bulundan sonuç sayısını yazdırın
- 4-sonuç yazsının puset içerdigini test edin
- 5-üçüncü ürüne relative locater kullanarak tıklayın
- 6-title ve fiyat bilgilerini assign edelim ve ürünü sepete ekleyin

Test 4

- 1-sepetteki ürünlerle eklediğimiz ürünlerin aynı olduğunu isim ve fiyat olarak doğrulayın

JUnit

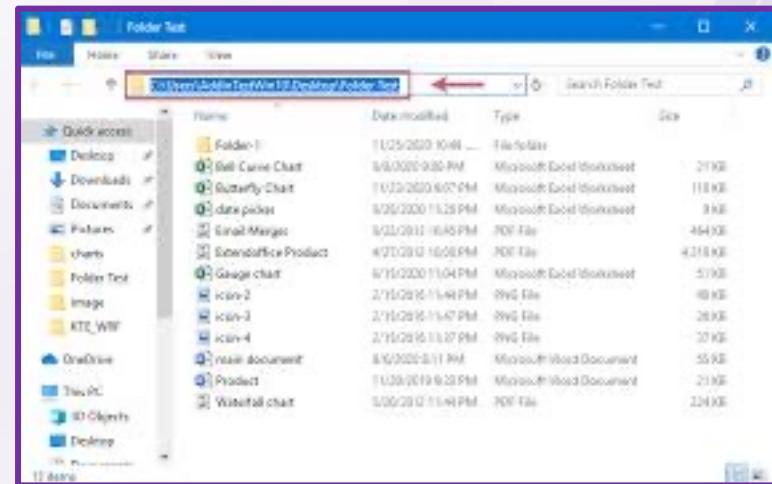
File Exist

Selenium webDriver objesi üzerinden calisir ve local bilgisayard işlem yapamaz. Local bilgisayardaki dosyalara erişmek veya test etmek için JAVA kullanılabilir.

Local bilgisayarda bir dosyaya ulaşmak ve olup olmadığını(exist) kontrol etmek için dosya yoluna ihtiyaç vardır.

Her bilgisayarın ismi ve kullanıcı ismi farklı olacağınından, bir bilgisayarda yazılan dosya yolu başka bilgisayarda çalışmaz.

Java'daki getProperty() method'u ile her bilgisayarda farklı olan kısım, testin çalıştığı bilgisayardan alınabilir. Bu temel path'den sonrası tüm bilgisayarlarda aynı olacağı için kod dinamik olur.



JUnit

File Exist

getProperty() method'u iki farkli parametre ile calisabilir.

- 1- System.getProperty ("user.dir"); icinde bulunulan klasörün yolunu (Path) verir
- 2- System.getProperty ("user.home"); bilgisayarimizda bulunan user klasörünü verir

```
String dosyaYolu= System.getProperty("user.home")+"Desktop/FileTesti/deneme.txt";
```

Seklinde oluşturulup kaydedilen dosya yolu dinamik olduğu için kodların calisacagi tum bilgisayarlarda sorunsuz kullanılabilir.

String olarak dosya yolu oluşturulan bir dosyanın bilgisayarda var olup olmadığını test etmek için Files class'ından exist() method'u kullanılır.

```
Files.exists(Paths.get(dosyaYolu))
```

Bu kod bize boolean bir sonuç döndürür. Bu sonuc kullanılarak test gerçekleştirilebilir.



File Exist

1. Tests packagenin altına bir class oluşturalım : C04_FileDownload
2. <https://the-internet.herokuapp.com/download> adresine gidelim.
3. logo.png dosyasını indirelim
4. Dosyanın başarıyla indirilip indirilmediğini test edelim

```
@Test
public void downloadTesti(){

    driver.get("https://the-internet.herokuapp.com/download");

    driver.findElement(By.xpath( xpathExpression: "//*[text()='logo.png']")).click();

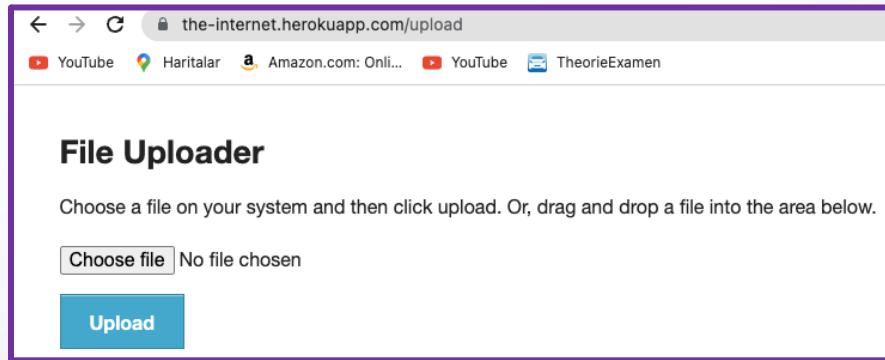
    bekle( beklenenSaniye: 5);

    String dosyaYolu= System.getProperty("user.home")+ "/Downloads/logo.png";

    Assert.assertTrue(Files.exists(Paths.get(dosyaYolu)));
}
```

JUnit

File Upload



<https://the-internet.herokuapp.com/upload>

Local bilgisayardaki bir dosyayı bir web uygulamasına yüklemek için, bilgisayarda dosyalar arasında gezinmek ve dosyayı tıklayarak seçmek gerekebilir.

Ancak selenium ile local dosyalara ulaşamayacağı için yine java'daki dosya kullanma yöntemleri kullanılabilir. Dosyayı yüklemek için

1- Dosya yolunu oluşturup kaydedin

2- Choose file butonunu locate edip dosya yolunu bu element'e gonderin

3- Upload butonunu locate edip tıklayın

File Upload

1. Tests packagenin altina bir class olusturun : C05_UploadFile
2. <https://the-internet.herokuapp.com/upload> adresine gidelim
3. chooseFile butonuna basalim
4. Yuklemek istediginiz dosyayı secelim.
5. Upload butonuna basalim.
6. “File Uploaded!” textinin goruntulendigini test edelim.

JUnit

Synchronization- Waits

Synchronization(Senkronizasyon), UI (kullanıcı arayüzü) üzerinde planlanan bir testin sorunsuz calismasi icin mutlaka dikkate alınması gereken bir konudur.

Bir sayfanın uygulama sunucusu veya web sunucusu çok yavaşsa veya internet ağı çok yavaşsa, web sayfasındaki öğelerin (webelement) yüklenmesi beklenenden uzun sürebilir.

Bu durumda, komut dosyanız (test script) öğeyi bulmaya çalıştığında, öğeler yüklenmez.

Bu yüzden test komut dosyası(test script) öğeyi bulamaz ve başarısız olur ve kod NoSuchElementException verip, calismayı durdurur.



JUnit

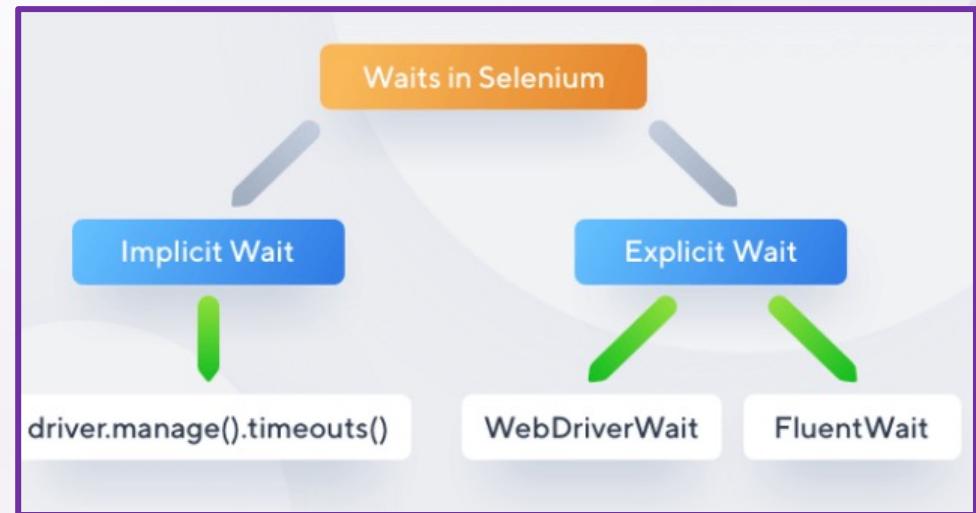
Synchronization- Waits

Driver ile cihaz veya internet arasında yasanan senkronizasyon sorunlarını çözmek için driver'i belirli yöntemler ile bekletmek(wait) gereklidir.

1) Java tabanlı wait

Thread.sleep : Javadan gelir ve kodları yazılan sure kadar bekletir. Sure dolduktan sonra alt satırdan işlemeye devam eder

2) Selenium tabanlı wait'ler



Implicitly Wait: Sayfadaki tüm öğeler için global bir zaman aşımıdır(timeout).

Explicitly Wait: Çoğunlukla belirli öğeler için belirli bir koşul(expected condition) için kullanılır.

Implicitly Wait

Bir sayfanın yüklenmesi veya sayfadaki her bir öğenin locate edilebilmesi için driver'i bekletir.

Selenium tabanlı wait'lerde verilen sure max. bekleme süresidir, işlem daha önce biterse surenin bitmesi beklenmez, kod çalışmaya devam eder.

Genellikle otomasyon frameworklerinde olası senkronizasyon problemleri için default olarak implicitly wait ile kullanılır.

Implicitly wait TestBase class'da kullanılabilir.

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```



Bu kod, driver'in sayfadaki herhangi bir weblement için maximum 10 saniye beklemesi istediği anlamına gelir.

Implicitly Wait

```
driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
```

Webelement 10 saniyeden kısa sürede yüklenirse driver bulur ve devam eder.

Örneğin, Webelement 3 saniye içinde yüklenirse, driver sadece 3 saniye bekleyecektir ve bir sonraki satırı geçecektir.

Webelement 10 saniye içinde yüklenmezse, test case başarısız olur ve NoSuchElementException uyarısı verir.





Wise Quarter
first class IT courses

Selenium

Ders-8

Selenium Waits

Cookies

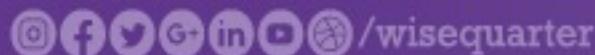
Web Tables



The future at your fingertips

+1 912 888 1630

www.wisequarter.com



JUnit

Explicitly Wait

Beklenen bir durum(expected condition) olduğunda explicit wait kullanılabılır.

Implicitly wait ile cozulebilecek durumlar için explicitly wait kullanımına ihtiyac yoktur.

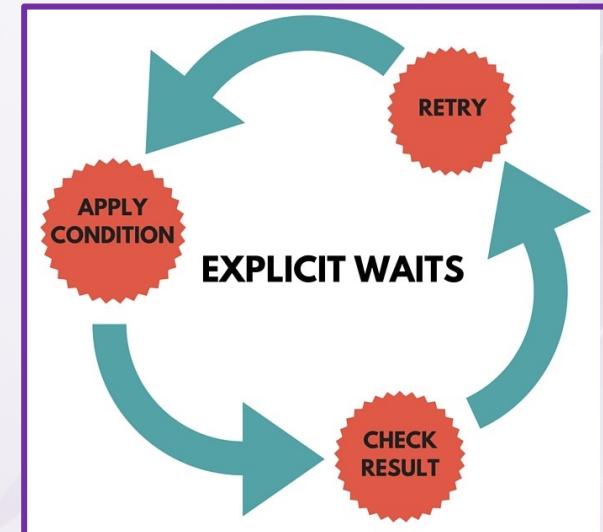
Nadiren karşılaşılan ve daha uzun bekleme süresi gerektiren işlem uygulanan webelementler için explicitly wait kullanılır.

İlk olarak belirli miktarda bekleme süresi ile wait object create edilir.

```
WebDriverWait wait= new WebDriverWait(driver, Duration.ofSeconds(20));
```

Explicit wait'de hem webelement, hem de beklenen condition kullanılır. Cunku olmayan bir webelement'in locate edilmesi mümkün olamayabilir.

```
WebElement itsBackElementi= wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("message")));
```



Explicit Wait / Expected Conditions

1.alertIsPresent()

2.elementSelectionStateToBe()

3.elementToBeClickable()

4.elementToBeSelected()

5.frameToBeAvailableAndSwitchToIt()

6.invisibilityOfTheElementLocated()

7.invisibilityOfElementWithText()

8.presenceOfAllElementsLocatedBy()

9.presenceOfElementLocated()

10.textToBePresentInElement()

11.textToBePresentInElementLocated()

12.textToBePresentInElementValue()

13.titleIs()

14.titleContains()

15.visibilityOf()

16.visibilityOfAllElements()

17.visibilityOfAllElementsLocatedBy()

18.visibilityOfElementLocated()

Explicit Wait

1. Bir class olusturun : WaitTest

2. Iki tane metod olusturun : implicitWait() , explicitWait()

Iki metod icin de asagidaki adimlari test edin.

3. https://the-internet.herokuapp.com/dynamic_controls adresine gidin.

4. Remove butonuna basin.

5. “It’s gone!” mesajinin goruntulendigini dogrulayin.

6. Add buttonuna basin

7. It’s back mesajinin gorundugunu test edin

JUnit

Explicit Wait

1. Bir class olusturun : EnableTest
2. Bir metod olusturun : isEnabled()
3. https://the-internet.herokuapp.com/dynamic_controls adresine gidin.
4. Textbox'in etkin olmadigini(enabled) doğrulayın
5. Enable butonuna tıklayın ve textbox etkin oluncaya kadar bekleyin
6. “It’s enabled!” mesajinin goruntulendigini doğrulayın.
7. Textbox'in etkin oldugunu(enabled) doğrulayın.

Actions Class Homework

1. "<http://webdriveruniversity.com/Actions>" sayfasina gidin
2. "Hover over Me First" kutusunun ustune gelin
3. "Link 1" e tiklayin
4. Popup mesajini yazdirin
5. Popup'i tamam diyerek kapatin
6. "Click and hold" kutusuna basili tutun
7. "Click and hold" kutusunda cikan yaziyi yazdirin
8. "Double click me" butonunu cift tiklayin

Iframe Homework

1. "http://webdriveruniversity.com/IFrame/index.html" sayfasina gidin
2. "Our Products" butonuna basin
3. "Cameras product"i tiklayin
4. Popup mesajini yazdirin
5. "close" butonuna basin
6. "WebdriverUniversity.com (IFrame)" linkini tiklayin
7. "http://webdriveruniversity.com/index.html" adresine gittigini test edin



Window Handle Homework

- 1."<http://webdriveruniversity.com/>" adresine gidin
- 2."Login Portal" a kadar asagi inin
- 3."Login Portal" a tiklayın
- 4.Diger window'a gecin
- 5."username" ve "password" kutularina deger yazdirin
- 6."login" butonuna basin
- 7.Popup'ta cikan yazinin "validation failed" oldugunu test edin
- 8.Ok diyerek Popup'i kapatın
- 9.Ilk sayfaya geri donun
- 10.Ilk sayfaya donuldugunu test edin

JUnit

Cookies

Çerezler, belirli kullanıcıları tanımlamak ve bu kullanıcıların göz atma deneyimini iyileştirmek için kullanıcının bilgisayarı ile web sunucusu arasında takas edilen, kullanıcı adı ve parola gibi küçük veri parçalarını içeren dosyalardır.

İnternette gezinirken ziyaret ettiğiniz web sayfaları, bilgisayarınıza ve telefonunuza küçük bilgi dosyaları kaydeder. Bu dosyalar telefon veya bilgisayarınızın hafızasında saklanır. Daha sonra aynı siteleri ziyaret ettiğinizde bu kayıtlı bilgi dosyaları sayesinde siteler sizi tanıyabilir.

Bilgileriniz bu dosyalara yazıldığından dolayı tekrar aynı web sayfalarını ziyaret ettiğinizde bilgilerinizi yeniden girmeye gerek duymazsınız.

Cookies, kişisel bilgiler de dahil olmak üzere birçok bilgiyi içerebilir. Web siteleri, ancak sizin izin verdığınız bilgilere erişebilir. Bu web sayfaları, sizin vermediğiniz bilgilere erişemez ya da bilgisayarlarınızdaki diğer dosyaları görüntüleyemez.



JUnit

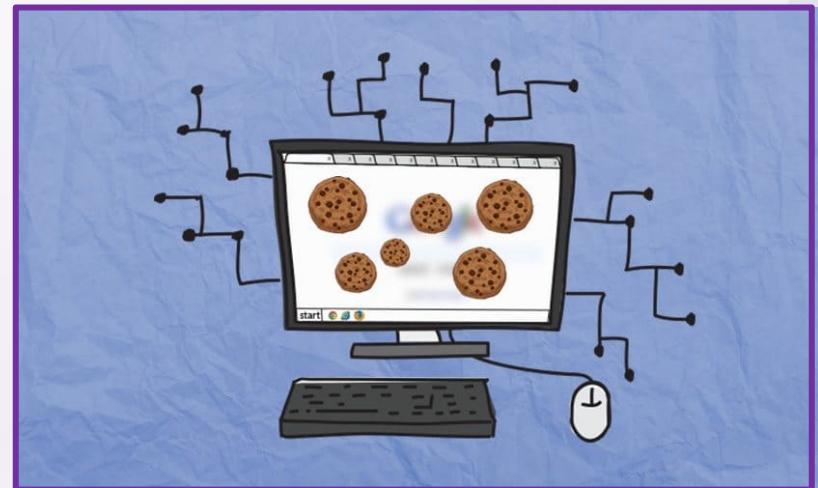
Cookies

Birkaç farklılıkla, siber dünyadaki çerezlerin oturum çerezleri ve kalıcı çerez olmak üzere iki çeşidi vardır.

Oturum çerezleri yalnızca bir web sitesinde gezinirken kullanılır. Bunlar rastgele erişimli bellekte saklanır ve hiçbir zaman sabit sürücüye yazılmasın. Oturum sona erdiğinde oturum çerezleri otomatik olarak silinir.

Kalıcı çerezler bir bilgisayarda sonsuza kadar kalır ancak birçoğunun bir son kullanma tarihi olup bu tarihe gelindiğinde otomatik olarak kaldırılırlar.

Üçüncü taraf çerezler daha sıkıntılıdır. Bunlar, genellikle kullanıcıların halihazırda gezdiği web sayfalarındaki reklamlarla bağlantılı olduklarından bu sayfalardan farklı web siteleri tarafından oluşturulur.



JUnit

Cookies

Selenium ile cookies otomasyonu yapabiliyoruz.

```
driver.manage().cook
    (m) addCookie(Cookie cookie)           void
    (m) getCookies()                     Set<Cookie>
    (m) deleteAllCookies()              void
    (m) deleteCookie(Cookie cookie)      void
    (m) getCookieNamed(String name)      Cookie
    (m) deleteCookieNamed(String name)   void
    ...
    Ctrl+Down and Ctrl+Up will move caret down and up in the editor  Next Tip
```

Driver.manage(). method'u ile cookie'leri

- listeleyebilir
- Isim ile cagirabilir
- Yeni cookie ekleyebilir
- Var olanlari ismi silebilir
- Var olan tum cookie'leri silebiliriz

Yeni bir class olusturun : cookiesAutomation

- 1- Amazon anasayfaya gidin
- 2- tum cookie'leri listeleyin
- 3- Sayfadaki cookies sayisinin 5'den buyuk oldugunu test edin
- 4- ismi i18n-prefs olan cookie degerinin USD oldugunu test edin
- 5- ismi "en sevdigim cookie" ve degeri "cikolatali" olan bir cookie olusturun ve sayfaya ekleyin
- 6- eklediginiz cookie'nin sayfaya eklendigini test edin
- 7- ismi skin olan cookie'yi silin ve silindigini test edin
- 8- tum cookie'leri silin ve silindigini test edin



JUnit

Web Tables

First Name	Last Name	Age	Email	Salary	Department	Action
Cierra	Vega	39	cierra@ex...	10000	Insurance	
Alden	Cantrell	45	alden@ex...	12000	Compliance	
Kierra	Gentry	29	kierra@ex...	2000	Legal	

```
<table>           → table
  <thead>          → header
    <tr>            → Header row
      <th>          → Header data
      </th>
    </tr>
  </thead>

  <tbody>          → body
    <tr>            → row
      <td>          → data
      </td>
    </tr>
  </tbody>
</table>
```



- 1."<https://www.amazon.com>" adresine gidin
- 2.Sayfanın en altına inin
- 3.Web table tüm body'sini yazdırın
- 4.Web table'daki satır sayısının 9 olduğunu test edin
- 5.Tüm satırları yazdırın
6. Web table'daki sutun sayısının 13 olduğunu test edin
7. 5.sutunu yazdırın
- 8.Satır ve sutun sayısını parametre olarak alıp, hücredeki bilgiyi döndüren bir method oluşturun

Web Tables

Bir Class olusturun D19_WebtablesHomework

1. “<https://demoqa.com/webtables>” sayfasina gidin
2. Headers da bulunan basliklari yazdirin
3. 3.sutunun basligini yazdirin
4. Tablodaki tum datalari yazdirin
5. Tabloda kac tane bos olmayan cell (data) oldugunu yazdirin
6. Tablodaki satir sayisini yazdirin
7. Tablodaki sutun sayisini yazdirin
8. Tablodaki 3.kolonu yazdirin
9. Tabloda "First Name" i Kierra olan kisinin Salary'sini yazdirin
10. Page sayfasinda bir method olusturun, Test sayfasindan satir ve sutun sayisini girdigimde bana datayi yazdirinsin



Wise Quarter
first class IT courses

Selenium

Ders-9

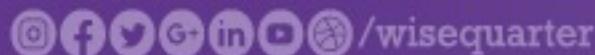
Excel Automation
Get Screenshot
Js Executors



The future at your fingertips

+1 912 888 1630

www.wisequarter.com



Excel Automation

Excel icin daha once inceledigimiz Web Table yapisina benzer bir yapı vardır.

Excel'de bir hucredeki bilgiye ulasmak icin dosya/sayfa/satir/sutun sirasiyla ilerlemeliyiz

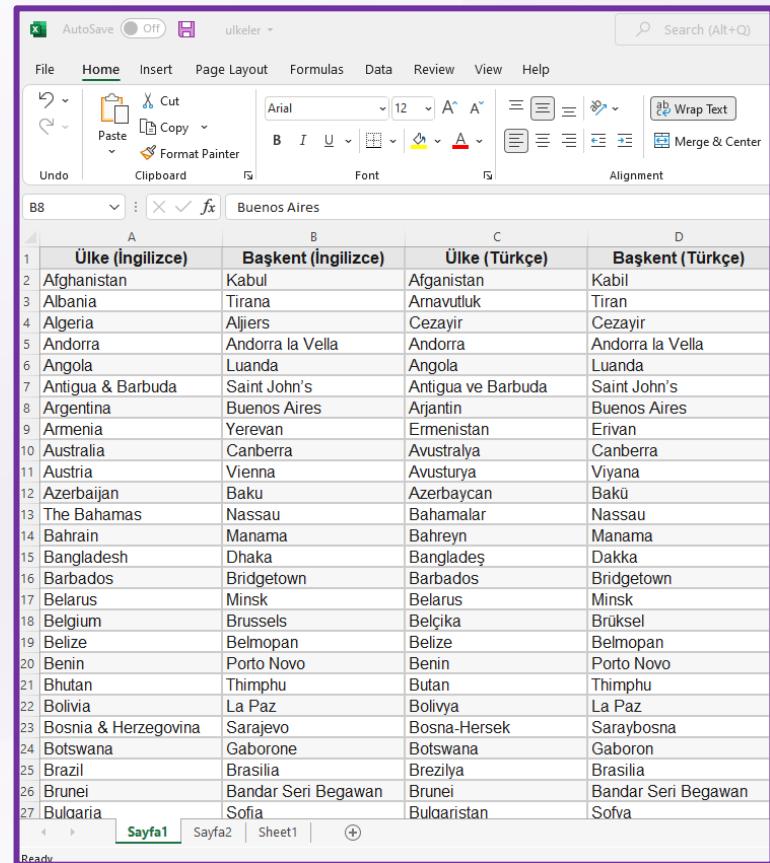
Excel ile ilgili otomasyonda web table'da oldugu gibi sutun yapisi yoktur, ihtiyac duyarsak kodla sutunu elde edebiliriz ancak bir dataya ulasmak icin sutun kullanamayiz

Workbook excel dosyamız

Sheet Her açık excel sekmesi (Sheet1, etc)

Row(satır) Java, yalnızca içindeki veri varsa satırları sayar. Default olarak, Java perspektifinden satır sayısı 0'dır

Cells (hücre) Java her satıra bakar ve yalnızca hücrede veri varsa hücre sayısını sayar.



	A	B	C	D
1	Ülke (İngilizce)	Başkent (İngilizce)	Ülke (Türkçe)	Başkent (Türkçe)
2	Afghanistan	Kabul	Afganistan	Kabil
3	Albania	Tirana	Arnavutluk	Tiran
4	Algeria	Aljiers	Cezayir	Cezayir
5	Andorra	Andorra la Vella	Andorra	Andorra la Vella
6	Angola	Luanda	Angola	Luanda
7	Antigua & Barbuda	Saint John's	Antigua ve Barbuda	Saint John's
8	Argentina	Buenos Aires	Arjantin	Buenos Aires
9	Armenia	Yerevan	Ermenistan	Ervan
10	Australia	Canberra	Avustralya	Canberra
11	Austria	Vienna	Avusturya	Viyana
12	Azerbaijan	Baku	Azerbaycan	Bakü
13	The Bahamas	Nassau	Bahamalar	Nassau
14	Bahrain	Manama	Bahreyn	Manama
15	Bangladesh	Dhaka	Bangladeş	Dakka
16	Barbados	Bridgetown	Barbados	Bridgetown
17	Belarus	Minsk	Belarus	Minsk
18	Belgium	Brussels	Belika	Brüksel
19	Belize	Belmopan	Belize	Belmopan
20	Benin	Porto Novo	Benin	Porto Novo
21	Bhutan	Thimphu	Butan	Thimphu
22	Bolivia	La Paz	Bolivya	La Paz
23	Bosnia & Herzegovina	Sarajevo	Bosna-Hersek	Saraybosna
24	Botswana	Gaborone	Botswana	Gaboron
25	Brazil	Brasilia	Brezilya	Brasilia
26	Brunei	Bandar Seri Begawan	Brunei	Bandar Seri Begawan
27	Bulgaria	Sofia	Bulgaristan	Sofya

JUnit

Apache POI

Apache POI, microsoft ofis dokumanlarına erişmek için kullanılan Java API'ıdır.

Poi.apache.com sayfasından official dokumanlar incelenebilir.

```
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>5.2.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml</artifactId>
    <version>5.2.2</version>
</dependency>
```

Read Excel

1. apache poi dependency'i pom file'a ekleyelim
2. Java klasoru altinda resources klasoru olusturalim
3. Excel dosyamizi resources klasorune ekleyelim
4. excelAutomation isminde bir package olusturalim
5. ReadExcel isminde bir class olusturalim
6. readExcel() method olusturalim
7. Dosya yolunu bir String degiskene atayalim
8. FileInputStream objesi olusturup,parametre olarak dosya yolunu girelim
9. Workbook objesi olusturalim,parameter olarak fileInputStream objesini girelim
10. WorkbookFactory.create(fileInputStream)
11. Worksheet objesi olusturun workbook.getSheetAt(index)
12. Row objesi olusturun sheet.getRow(index)
13. Cell objesi olusturun row.getCell(index)

Read Excel

Yeni bir test method olusturalim readExcel2()

- 1.satirdaki 2.hucreye gidelim ve yazdiralim
- 1.satirdaki 2.hucreyi bir string degiskene atayalim ve yazdiralim
- 2.satir 4.cell'in afganistan'in baskenti oldugunu test edelim
- Satir sayisini bulalim
- Fiziki olarak kullanilan satir sayisini bulun
- Ingilizce Ulke isimleri ve baskentleri bir map olarak kaydedelim

Write Excel

- 1) Yeni bir Class olusturalim WriteExcel
- 2) Yeni bir test method olusturalim writeExcelTest()
- 3) Adimlari takip ederek 1.satira kadar gidelim
- 4) 4.hucreye yeni bir cell olusturalim
- 5) Olusturdugumuz hucreye “Nufus” yazdiralim
- 6) 2.satir nufus kolonuna 1500000 yazdiralim
- 7) 10.satir nufus kolonuna 250000 yazdiralim
- 8) 15.satir nufus kolonuna 54000 yazdiralim
- 9) Dosyayı kaydedelim
- 10) Dosyayı kapatalım

Get Screenshot / Tüm Sayfa

1.Adım : Bir TakeScreenshot objesi olusturup driver'imizi TakeScreenshot'a cast yapalim

```
TakesScreenshot tss= (TakesScreenshot) driver;
```

2.Adım : kaydettigimiz ekran goruntusunu projede istedigimiz yere kaydedebilmek icin path ile yeni bir File olusturalim

```
File tumSayfaSShot= new File( pathname: "target/ScreenShot/tumSayfaScreenshot.jpeg");
```

3.Adım : Takescreenshot objesini kullanarak getScreenshotAs() methodunu calistiralim ve gelen resmi gecici bir file'a assign edelim

```
File geciciResim= tss.getScreenshotAs(OutputType.FILE);
```

4.Adım : Kaydettigimiz goruntuyu, saklamak istedigimiz dosyaya kopyalayalim

```
FileUtils.copyFile(geciciResim,tumSayfaSShot);
```

Get Screenshot / Spesific Webelement

Selenium 4 ile gelen guzel ozelliklerden bir tanesi de istedigimiz WebElement'in fotografini almamiza imkan tanimasi

1.Adım : Istenilen webelement'i locate edin

```
WebElement sonucYaziElementi= driver.findElement(By.xpath( xpathExpression: "locator"));
```

2.Adım : kaydettigimiz ekran goruntusunu projede istedigimiz yere kaydedebilmek icin path ile yeni bir File olusturalim

```
File istenenElementSShot= new File( pathname: "target/ScreenShot/SonucyazisiScreenshot.jpeg");
```

3.Adım : Istenen webelement'i kullanarak getScreenshotAs() methodunu calistirralim ve gelen resmi gecici bir file'a assign edelim

```
File geciciResim= sonucYaziElementi.getScreenshotAs(OutputType.FILE);
```

4.Adım : Gecici resmi, saklamak istedigimiz dosyaya kopyalayalim

```
FileUtils.copyFile(geciciResim,istenenElementSShot);
```

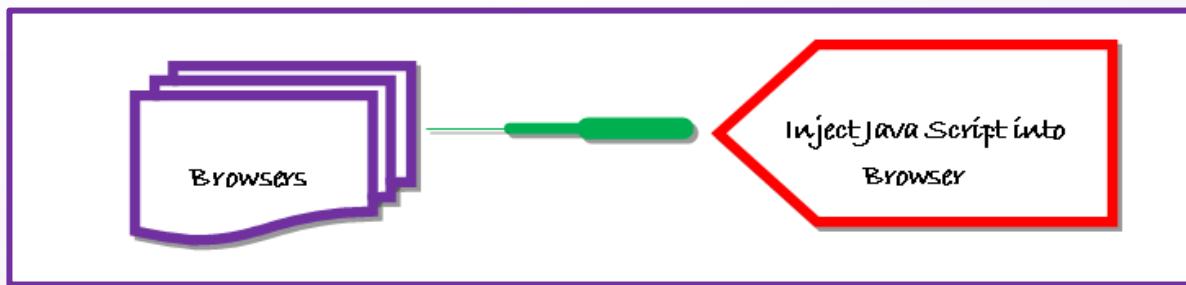


Yeni bir class olusturun : amazonNutellaSearch

- 1- amazon anasayfaya gidin
- 2- amazon anasayfaya gittiginizi test edin ve tum sayfanin goruntusunu kaydedin
- 3- Nutella icin arama yapin
- 4- sonucun Nutella icerdigini test edin ve ilk urunun goruntusunu alin

JUnit

JS Executors



Selenium ile web elementlerinin kontrollerini sağlarken selenium komutları yetersiz kalabilir veya sorunlarla karşılaşabiliriz.

Bu durumlarda alternatif olarak üstesinden gelmek için JavascriptExecutor class'ını dahil edebiliriz.

JavaScript HTML kodlara direk erişip yönetebilen bir script dili olduğundan bize çok fazla kolaylık sağlayabilir.

JUnit

JS Executors

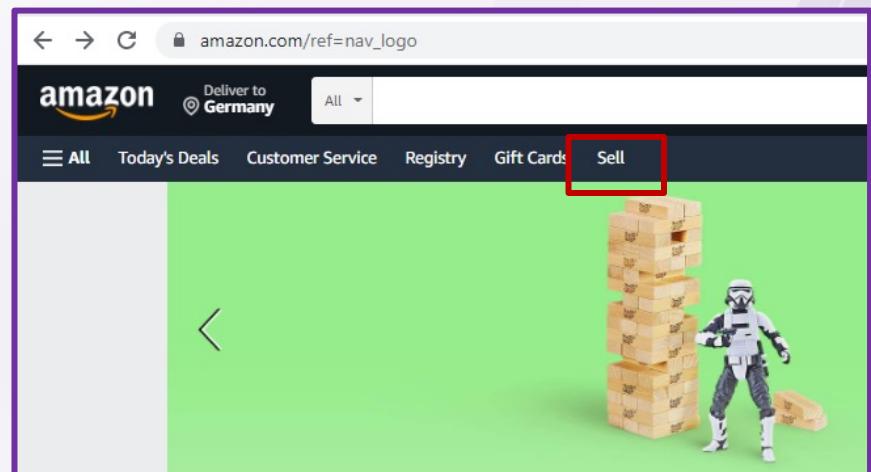
1.Adım : JavascriptExecutor kullanmak için ilk adım olarak driver'imizi JavascriptExecutor'a cast edelim

```
JavascriptExecutor jse= (JavascriptExecutor) driver;
```

2.Adım : Kullanacağımız WebElement'i locate edelim

3.Adım : Js driver ile executeScript method'unu calistiralim, icine parameter olarak ilgili script ve webelement'i yazalim

```
jse.executeScript( script: "ilgili script", ...args: "web element");
```



Ornegin Sell elementine click yapmak için

```
jse.executeScript("arguments[0].click();",sellLinki);
```

JUnit

JS Executors

Istenen Webelement'e kadar asagi inmek icin

```
jse.executeScript("arguments[0].scrollIntoView();",hedefWebelement);
```

Ornek : wisequarter anasayfasina gidin alt kisimda bulunan "Go To Career Page" butonuna kadar asagi inin ve bu butona click yapin



Career opportunities

Career opportunities are provided to our students by the career management service department.



Career Inspiration Sessions

Inspirational moments at the career inspiration sessions for our current and alumni students.

 Go To Career Page

JS alert kullanarak uygulama'dan kullaniciya mesaj vermek icin

```
jse.executeScript("alert('yasasinnnn');");
```



Wise Quarter
first class IT courses

Selenium

Ders-10

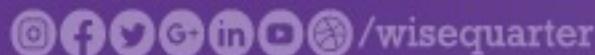
TestNG Framework
Annotations
Priority



The future at your fingertips

+1 912 888 1630

www.wisequarter.com

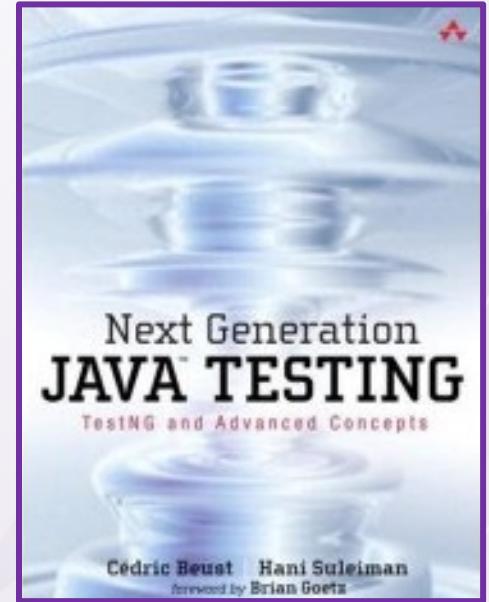


TestNG

Nedir ?

TestNG, JUnit ve NUnit'ten ilham alan ancak onu daha güçlü ve kullanımı daha kolay hale getiren bazı yeni işlevler sunan bir test framework'udur, örneğin:

- Annotations
- Testlerinizi, kurallarını kendinizin belirleyeceği yöntemlerle, toplu çalıştırılabilme
- Esnek test konfigürasyonu.
- Veri odaklı test desteği (@DataProvider ile).
- Parametreler için destek.
- Çeşitli tool ve plug-ins tarafından desteklenir (Eclipse, IDEA, Maven, vb...).



TestNG, tüm test kategorilerini kapsayacak şekilde tasarlanmıştır: birim(unit), işlevsel(functional), uçtan uca(E2E), entegrasyon, vb...

TestNG

Neler Saglar ?

TestNG tester'lara daha fazla kontrol imkani verir ve testleri daha etkili yapmamizi saglar.

Tester'lar TestNG'yi etkili bir framework tasarlamak ve test case'leri TestNG annotation'ları ile organize etmek için kullanırlar.

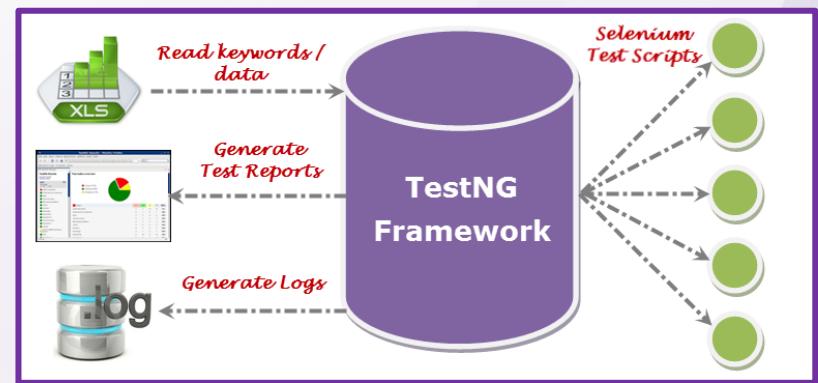
Test caseleri siralama ozelligi (**priority**) ve test caselerin birbirine bagimlilik (**dependsOnMethod**) bize testleri organize etmeye yardım eder.

Yazilan test case'lerin paralel olarak çalıştırılmasına, birden fazla browser kullanılmamasına imkan tanır.

Error mesajlarının daha detaylı bir şekilde gösterilmesini sağlar.

Paralel ve Cross-Browser Test yapmamiza imkan tanır

Kullanicili HTML veya xml raporları olusturma imkani vardır





TestNG

Nasıl Yuklenir ?

Mvnrepository.com adresinden org.testng dependency eklemek yeterlidir.

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.1.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
    <dependency>
        <groupId>io.github.bonigarcia</groupId>
        <artifactId>webdrivermanager</artifactId>
        <version>5.0.3</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.testng/testng -->
    <dependency>
        <groupId>org.testng</groupId>
        <artifactId>testng</artifactId>
        <version>7.4.0</version>
        <scope>test</scope>
    </dependency>
```

TestNG

Framework Yapisi

TestNG ile olusturacagimiz bu framework, isyerinde kullanilabilecek kullanisli bir framework olacak.

TestNG ile uygulayacagimiz POM (Page Object Model) icin Test'lerimizin oldugu class'lar disinda bazi package ve class'lar daha olusturulacaktir.

Olusturulacak POM (Page Object Model) ile

- tum ekip ile sorunsuz sekilde calisabilecegimiz,
- test datalarini kolayca yonetebilecegimiz
- Kod tekrarlarinden kurulacagimiz

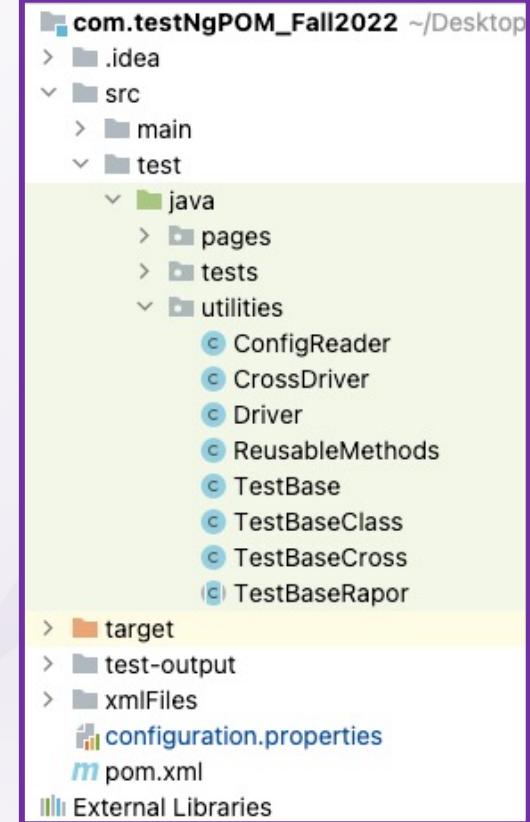
bir yapı olusturulacaktır.

1- File menusunden New, Project secip yeni proje olusturun.

2- src/test/java package'i altinda iki package olusturun

- tests
- utilities

3- Yeni olusturdugumuz tests package'i altinda gunluk package ve test class'i olusturun



TestNG

Annotations

@Test Annotation

En çok kullanılan TestNG notasyonudur.
Bir method'u bagimsiz olarak calisabilecek
bir test method'u haline getirir.

Test notasyonu ile birlikte kullanılabilecek

- priority
- dataProvider
- groups

gibi bircok ozellik bulunur

@Test	Marks a class or a method as part of the test.
alwaysRun	If set to true, this test method will always be run even if it fails.
dataProvider	The name of the data provider for this test method.
dataProviderClass	The class where to look for the data provider. If no class is specified, the current class is used.
dependsOnGroups	The list of groups this method depends on.
dependsOnMethods	The list of methods this method depends on.
description	The description for this method.
enabled	Whether methods on this class/method are enabled.
expectedExceptions	The list of exceptions that a test method is expected to throw.
groups	The list of groups this class/method belongs to.
invocationCount	The number of times this method should be invoked.
invocationTimeOut	The maximum number of milliseconds this test should take to run.
priority	The priority for this test method. Lower priorities will be run first.
successPercentage	The percentage of success expected from this method.
singleThreaded	If set to true, all the methods on this test class are run sequentially. This attribute used to be called sequential (now deprecated).
timeOut	The maximum number of milliseconds this test should take to run.
threadPoolSize	The size of the thread pool for this method. The maximum number of threads used by this method.

@Before @After Annotations

TestNG testlerimizi calistirirken yapacagimiz on hazirliklari koordine edebilmek icin daha fazla secenek sunmaktadır.

@BeforeMethod: The annotated method will be run before each test method.

@AfterMethod: The annotated method will be run after each test method.

JUnit'deki @Before ve @After notasyonlarina benzer her test method'undan once ve sonra calisir.

@BeforeSuite: The annotated method will be run before all tests in this suite have run.

@AfterSuite: The annotated method will be run after all tests in this suite have run.

@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the <test> tag is run.

@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the <test> tag have run.

@BeforeGroups: The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

@AfterGroups: The list of groups that this configuration method will run after. This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.

@AfterClass: The annotated method will be run after all the test methods in the current class have been run.

Belirlenen grplardan once ve sonra calisir.

TestNG

Priority

TestNG test method'larini alfabetik siraya gore calistirir.

Eger hangi test method'unun hangi sira ile calisacagini belirlemek isterseniz priority kullanabilirsiniz.

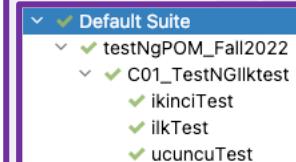
Kurallar:

1- Priority kucukden buyuge dogru calisir.

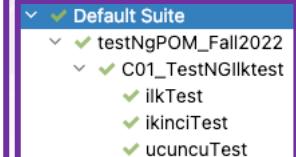
2- Priority verilmeyen test method'unun priority degeri 0 olarak kabul edilir.

3- Esit priority degerine sahip olan test method'lari isim sirasina gore calisir.

```
@Test  
public void ilkTest(){  
    driver.get("https://www.amazon.com");  
}  
  
@Test  
public void ikinciTest(){  
    driver.get("https://www.youtube.com");  
}  
  
@Test  
public void ucuncuTest(){  
    driver.get("https://www.wisequarter.com");  
}
```



```
@Test(priority = 1)  
public void ilkTest(){  
    driver.get("https://www.amazon.com");  
}  
  
@Test(priority = 12)  
public void ikinciTest(){  
    driver.get("https://www.youtube.com");  
}  
  
@Test(priority = 35)  
public void ucuncuTest(){  
    driver.get("https://www.wisequarter.com");  
}
```





Bir class oluşturun: YoutubeAssertions

- 1) <https://www.youtube.com> adresine gidin
- 2) Aşağıdaki adları kullanarak 4 test metodu oluşturun ve gerekli testleri yapın
 - titleTest => Sayfa başlığının “YouTube” olduğunu test edin
 - imageTest => YouTube resminin görüntüülendiğini (isDisplayed()) test edin
 - Search Box 'in erişilebilir olduğunu test edin (isEnabled())
 - wrongTitleTest => Sayfa basliginin “youtube” olmadığını doğrulayın

TestNG

dependsOnMethods

dependsOnMethods test method'larinin siralamasi ile ilgili degildir. Siralamayi priority ile tanimlayabiliriz.

dependsOnMethods ile baska method'a baglanan bir method calismaya baslamadan once baglandigi method'un calisip, PASSED oldugunu kontrol eder.

- Bagli oldugu test calisti ve PASSED olduysa testi **calistirir**.
- Bagli oldugu test calisti ve FAILED olduysa testi **ignore eder**.
- Bagli oldugu test calismadi ise oncelikle **bagli oldugu testi calistirir**, sonucuna gore kendisi calisir veya ignore eder

Bagli oldugu testi calistirma sadece 2 method icindir. 3 Method birbirine baglandiginda, 3.method calistirilmak istendiginde 1.method'a kadar gitmez.

```
@Test  
public void test01(){  
  
    System.out.println("1.test calisti");  
}  
  
@Test(dependsOnMethods = "test01")  
public void test02(){  
  
    System.out.println("2.test calisti");  
}  
  
@Test(dependsOnMethods = "test02")  
public void test03(){  
  
    System.out.println("3.test calisti");  
}
```

TestNG

dependsOnMethods

```
@Test  
public void amazonTesti(){...}  
  
@Test(priority = 2,dependsOnMethods = "amazonTesti")  
public void nutellaTesti(){...}  
  
@Test(priority = 5,dependsOnMethods = "nutellaTesti")  
public void aramaSonucTesti(){...}
```

- Bir class oluşturun: **DependsOnTest**
- <https://www.amazon.com/> adresine gidin.
 1. Test : Amazon ana sayfaya gittiginizi test edin
 2. Test : 1.Test basarili ise search Box'i kullanarak “Nutella” icin arama yapin ve aramanizin gerceklestigini Test edin
 3. Test : “Nutella” icin arama yapıldıysa ilk urunu tiklayın ve fiyatının \$16.83 olduğunu test edin

Java'da Class Uyelerini Kullanma

Java ile olusturulan bir proje'de farkli class'daki class uyelerine erisim ve kullanma farkli yontemlerle yapilabilir.

1- inheritance (Miras)

kullandigimiz Class'i extends anahtar kelimesi (keyword) ile istedigimiz Class'in child'i yapabiliriz.

Bu durumda object olusturmaya gerek kalmadan Parent class'a ulasabilir ve oradaki variable ve methodlari kullanabiliriz. (Test Base gibi)

Inheritance ile variable ve method kullanirken static keyword'e dikkat etmek gerekir. Static olarak tanimlanmis bir variable veya method static olmayan method icinden kullanilamaz.

```
public class Okul {  
    String okulIsmi="Yildiz Koleji";  
    static int ogrenciSayisi=120;  
  
    public void okulMethod(){  
        System.out.println("Yildiz Koleji");  
    }  
}
```

```
public class Ogrenci extends Okul {  
  
    public void ogrenciMethod(){  
        System.out.println(okulIsmi);  
        okulMethod();  
  
        System.out.println(ogrenciSayisi);  
    }  
}
```

extends



Java'da Class Uyelerini Kullanma

2- Object olusturarak

Bir class'dan obje olusturarak istedigimiz class'a ulasabilir ve o class'daki variable ve methodlari object'imizi aracılıgiyla kullanabiliriz

ornek: Servis class'indan Okul class'ina ulasmak istiyorsak

- Okul class'indan bir obje olustururuz
- obje uzerinden variable veya method'lara ulasabiliriz

3- Static Class Uyeleri :

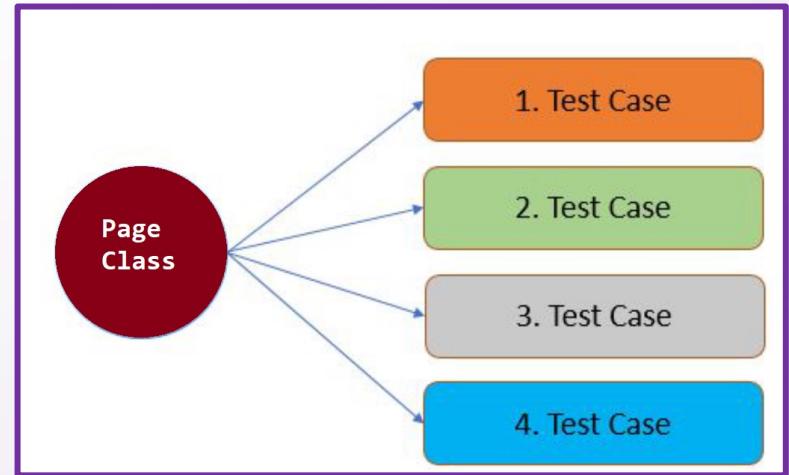
Eger kullanacagimiz variable veya method static ise object olusturmadan direk class ismi ile variable veya method'a ulasabiliriz.

```
public class Okul {  
    String okulIsmi="Yildiz Koleji";  
    static int ogrenciSayisi=120;  
  
    public void okulMethod(){  
        System.out.println("Yildiz Koleji");  
    }  
}
```

```
public class Servis {  
    public static void main(String[] args) {  
        Okul okulObj = new Okul();  
        System.out.println(okulObj.okulIsmi);  
        okulObj.okulMethod();  
  
        System.out.println(Okul.ogrenciSayisi);  
    }  
}
```

Page Object Model Framework Design

Bir sirkette test framework'u olusturdugumuzda kullanici adi, sifresi, gidilecek web adresi gibi test datalari tum testler icin gecerlidir. Ayrıca surekli kullanmamız gereken variable ve method'lar olacaktır.



Daha kullanisli bir **Framework** olusturmak icin temel hedefimiz, tekrar tekrar yaptigimiz islemleri ve testlerimizde kullandigimiz **Test Data**'larini onceden hazırladigimiz dosyalarda tutmaktır.

Bu sekilde testlerimizde ihtiyac duydugumuzda bu verilere kolayca ulasabilir veya test datalari ile ilgili bir degisiklik yapmamız gerektiginde sadece kaynak dosyadan bir degeri degistirerek tum test case'leri guncellemis oluruz.

Page Object Model Framework Design

POM çok popüler bir **Framework Design Pattern** 'dir.

Test suitlerimizde çok fazla testimiz olduğunda, test caseleri ve kodları korumak daha karmaşık hale gelir.

Bu nedenle,

sürdürülebilir(**maintainable**),

yeniden kullanılabilir(**reusable**),

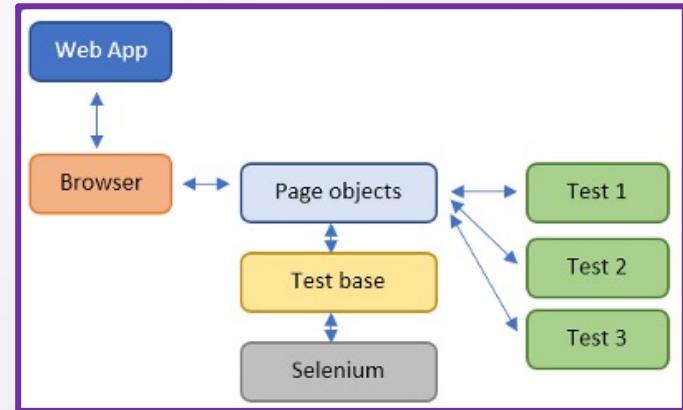
daha hızlı(**faster**),

anlaşılabilir(**understandable**)

daha iyi bir framework dizaynına ihtiyacımız vardır.

Page object model ile, sayfaya özgü elementleri veya methodları **page class** içinde tutar, ve bunları gerçek **test class**larından uzak tutarız.

POM ile ihtiyacımız olan class üyelerini sadece bir kez create edip birden çok kez kullanabiliriz.



Page Object Model Framework Design

Framework'un verimliliğini artırmak için core Java ve Selenium konseptini kullanarak temel olarak **page classları** ve **test classları** oluşturacağız.

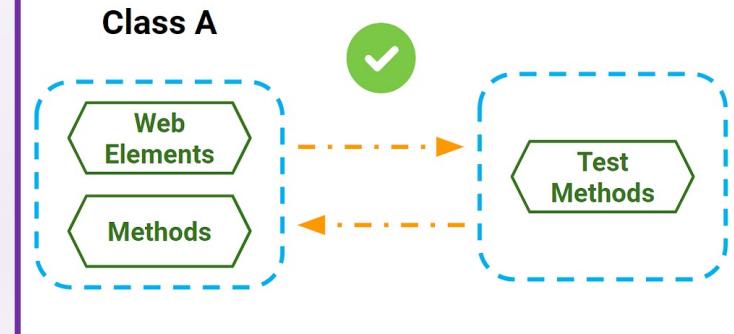
Tüm şirketler page object model dizaynını kullanmaz, ancak herkes bunu bilir ve daha da popüler hale gelmektedir.

Daha iyi bir tasarım, testin yürütme süresini daha hızlı hale getirir.

Bir uygulamanın(application) işlevselliği değiştiğinde, kodu düzeltmek için framework kontrol edilmesini ve gerekli düzeltmelerin yapılmasını kolaylaştırır.

Page Object Design daha bağımsız test senaryoları oluşturmamıza yardımcı olur, böylece test komut dosyalarında(script) hata ayıklamak daha kolay olacaktır.

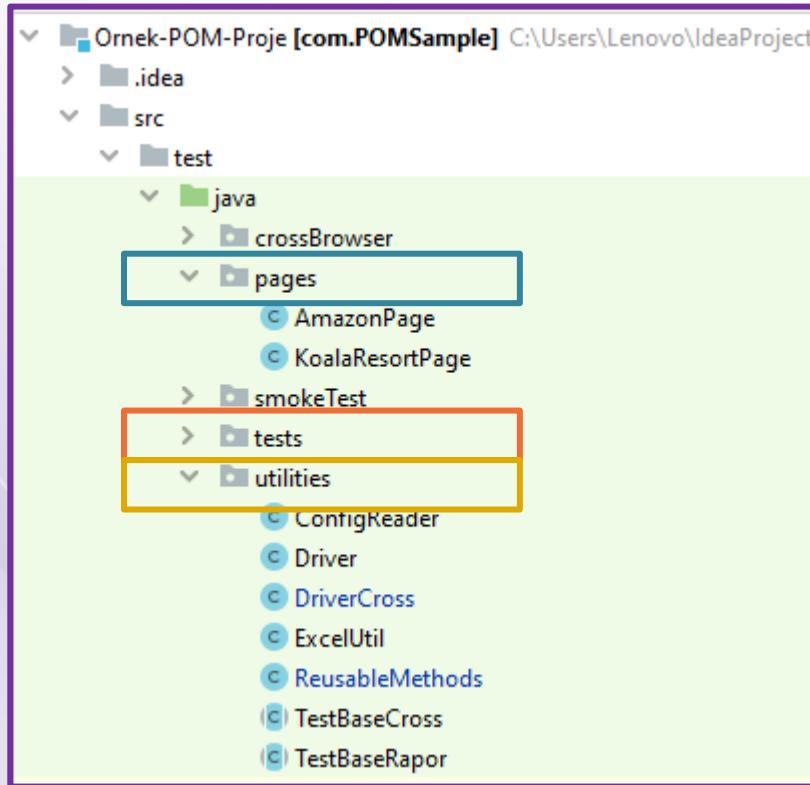
Design Patterns in Automated Testing series
Page Object Model





TestNG

Page Object Model Framework Design



Page Object Model temelde 3 package icerir

Tests : Sadece testleri execute etmek icin gerekli adimlari yazacagimiz class'lar icerir. Hicbir data girişi yapmayacagiz

Pages : Test yapacagimiz sayfalardaki Web Elementlerini locate etmek ve temel methodlari olusturmak icin kullanilir.

Utilities : Driver, TestBase ve ConfigurationReader class'larini icerir

Page Object Model Framework Design

<https://www.qualitydemy.com/>

Login

Provide your valid login credentials

Email

Password
 Forgot password?

Login

Do not have an account? [Sign Up](#)

Calistigimiz uygulamanin login ile girilen bir uygulama oldugunu dusunelim.

Yaptiginiz tum testlerde oncelikle login olmaniz gerekecektir.

Bu durumda her test icerisinde username, password kutularini ve login butonunu locate etmek zorunda kalirsınız

Yine her test icin gecerli kullanici adi ve sifre bilgilerini girmeniz gerekir.

Gecerli kullanici adi ve sifre bilgisi test datalari her degistiginde kullanilmis oldugu testleri bulmak ve tamamini guncellemek gerekecektir.

POM locator'lari sadece 1 kere yapip, ihtiyacimiz oldugunda kullanma, test datalarini bir kere olusturup kullanacagimiz zaman oradan kullanip, update yapacagimiz zaman oradan update yapma gibi imkanlar tanir.

Page Object Model Framework Design

```

@FindBy(xpath = "//a[text()='Log in']")
public WebElement loginLinki;

@FindBy(xpath = "//input[@id='login-email']")
public WebElement emailKutusu;

@FindBy(xpath = "//input[@id='login-password']")
public WebElement passwordKutusu;

@FindBy(xpath = "//button[text()='Login']")
public WebElement loginButonu;
    
```

```

@Test
public void pozitifLoginTesti(){
    // Mycoursesdemmy anasayfasina gidel
    Driver.getDriver().get(ConfigReader.getProperty("myUrl"));
    // login linkine basin
    MyCoursesdemmyPage myCoursesdemmyPage=new MyCoursesdemmyPage();
    myCoursesdemmyPage.loginLinki.click();
    // Kullanici email'i olarak valid email girin
    myCoursesdemmyPage.emailKutusu.sendKeys(ConfigReader.getProperty("myGecerliEmail"));
    // Kullanici sifresi olarak valid sifre girin
    myCoursesdemmyPage.passwordKutusu.sendKeys(ConfigReader.getProperty("myGecerliPassword"));
    // Login butonuna basarak login olyun
    myCoursesdemmyPage.loginButonu.click();
    // Basarili olarak giris yapildigini test edin

    Assert.assertTrue(myCoursesdemmyPage.coursesLinki.isDisplayed());

    ReusableMethods.bekle(saniye: 2);
    Driver.closeDriver();
}
    
```

Page Object Model / Basic Driver Class

Temel Hedefimiz: Test methodlarimizda kullanacagimiz WebDriver driver'i utilities altindaki Driver Class'inda olusturmak ve testlerimizde bu class ismi üzerinden driver'a ulasip olusturma, kullanma ve kapatma islemlerini yapmak.

Su anki haliyle TestBase Class'imizla ayni olacak ama inheritance kullanmak yerine static method'lar ile driver'i kullanacagiz. Driver class'ini daha sonra tum browser'lar ile calisacak hale getirecegiz

```
public class Driver {  
  
    private static WebDriver driver;  
  
    public static WebDriver getDriver() {  
  
        if (driver == null) {  
            WebDriverManager.chromedriver().setup();  
            driver = new ChromeDriver();  
        }  
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));  
        driver.manage().window().maximize();  
  
        return driver;  
    }  
}
```

```
public class C02 {  
  
    @Test  
    public void test01(){  
  
        Driver.getDriver().get("https://www.amazon.com");  
    }  
}
```

Test Class(driver->Driver.getDriver())

Page Object Model / Page Factory

Bir page class'i olusturdugumuzda ilk isimiz bir constructor olusturup, pageFactory class'indan initElements() method'unu kullanmak olmalidir.

1. PageFactory, page object dizayni icin bir önemli classtır.
2. Page objelerini instantiate(ilk deger atama) için page classlarında PageFactory kullanıyoruz.
3. PageFactory.initElements(driver,this);
this => page instance
driver => bizim gonderecegimiz driver
4. Aslinda PageFactory class'ina, elementlere ilk degeri atayan initElements() metodunu kullanmak icin ihtiyacimiz var

```
public class HotelMyCampPage {  
    public HotelMyCampPage(){  
  
        PageFactory.initElements(Driver.getDriver(), page: this);  
    }  
  
    @FindBy(xpath = "//a[text()='Log in']")  
    public WebElement ilkLoginLinki;  
  
    @FindBy(xpath = "//input[@id='UserName']")  
    public WebElement usernameBox;  
  
    @FindBy(xpath = "//input[@id='Password']")  
    public WebElement passwordBox;  
  
    @FindBy(xpath = "//input[@id='btnSubmit']")  
    public WebElement loginButonu;  
  
    @FindBy(xpath = "//div[@class='validation-summary-errors']")  
    public WebElement girisYapilamadiYaziElementi;  
  
    @FindBy(xpath="//span[text()='ListOfUsers']")  
    public WebElement basariliGirisYaziElementi;
```

Page Object Model / @FindBy Annotation

Page class'larinda webElementleri locate etmek icin simdiye kadar kullandigimiz driver.findElement() method'unu kullanmayacagiz.

1. @FindBy notasyonu test class'larinda kullanacagimiz Web Elementlerini Page sayfasinda locate etmek icin kullanilir
2. Bunun icin kullanacagimiz Web Elementini public olarak olusturmali, sonra da @FindBy notasyonu ile locate etmeliyiz
3. Bu islemi yaptiktan sonra hangi test methodumuzda bu web elemente ihtiyac duyarsak page class'indan uretecegimiz obje uzerinden rahatlikla kullanabiliriz

```
public class HotelMyCampPage {  
    public HotelMyCampPage(){  
  
        PageFactory.initElements(Driver.getDriver(), page: this);  
    }  
  
    @FindBy(xpath = "//a[text()='Log in']")  
    public WebElement ilkLoginLinki;  
  
    @FindBy(xpath = "//input[@id='UserName']")  
    public WebElement usernameBox;  
  
    @FindBy(xpath = "//input[@id='Password']")  
    public WebElement passwordBox;  
  
    @FindBy(xpath = "//input[@id='btnSubmit']")  
    public WebElement loginButonu;  
  
    @FindBy(xpath = "//div[@class='validation-summary-errors']")  
    public WebElement girisYapilamadiYaziElementi;  
  
    @FindBy(xpath="//span[text()='ListOfUsers']")  
    public WebElement basariliGirisYaziElementi;
```

Page Object Model

1 - <https://www.facebook.com/> adresine gidin

2- POM'a uygun olarak email, sifre kutularini ve giris yap butonunu locate edin

3- Faker class'ini kullanarak email ve sifre degerlerini yazdirip, giris butonuna basin

4- Basarili giris yapılamadigini test edin



The screenshot shows a login form with a purple border. It contains two input fields: 'E-posta veya Telefon Numarası' (Email or Phone Number) and 'Şifre' (Password). Below these is a large blue button labeled 'Giriş Yap' (Log In). To the right of the password field is a link 'Şifreni mi Unuttun?'. At the bottom right is a green button labeled 'Yeni Hesap Oluştur' (Create New Account).



Wise Quarter
first class IT courses

Selenium Ders-11

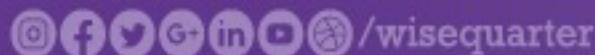
POM Configuration File Assertions



The future at your fingertips

+1 912 888 1630

www.wisequarter.com





Onceki Dersten Aklimizda Kalanlar

- 1- TestNG , Next Generation Testing iddiasi ile ortaya cikmistir, JUnit'e kiyasla tester'lara daha fazla kontrol imkani ve daha fazla GUC vadetmektedir.
- 2- TestNG ile daha fazla notasyon, toplu calistirma, parallel ve cross browser ile calistirma, raporlama, testlerin calismasini gruplar olarak ayarlama gibi bir çok ekstra imkan vardir.
- 3- TestNG ile birlikte Page Object Model (POM) kullaniriz.
- 4- POM bizi tekrarlardan kurtarir ve test datalarini kolayca yonetmemizi saglar
- 5- POM iyi bir orkestrasyon ister, temel olarak testlerimiz disinda 4 dosyayı kullanarak, testlerimizi daha dinamik ve tekrarlardan kurtulmus olarak yazariz
 - Page : locater'lar ve login gibi kucuk islevler yapan method'lar
 - Driver : bize islem yapacagimiz webDriver objesini saglar. Amac projemizde test datasi olarak yazacagimiz browser turune gore tum testlerin calistirilmasidir.
 - configuration.properties : test datalarimizi tuttugumuz basit bir txt gorunumundedir. Ancak uzantisi sayesinde Java icerisinden bu dosyaya erisim imkani olacak
 - ConfigReader Class : her test method'unda configuration.properties dosyasini okumaya calismak yerine, bir Java class'i olan bu dosyayı araci yapariz.

TestNG

POM Properties File

configuration.properties Test datalarini tuttugumuz .properties uzantılı bir dosyadır.

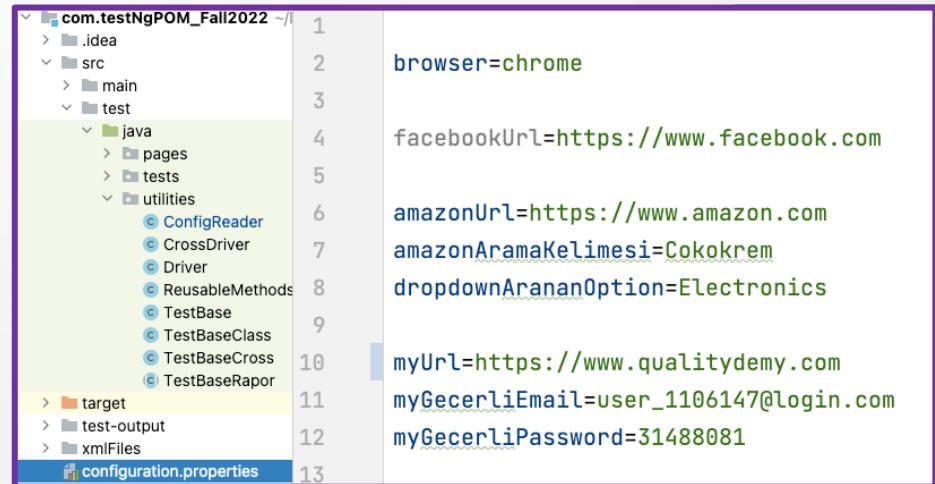
Bu dosya , test datalarina erişimi ve test datalarinda yapılan değişikliklerin tum test case'lere islenmesini kolaylastirir.

Örneğin :

Her test method'unda
driver.get("https://www.amazon.com")
yazmak yerine configuration
dosyamiza url'i tanımlayıp test
classında sadece driver.get(url)
kullanırız.

Temel olarak key(anahtar) ve value(değer) çiftlerini kullanırız ve ihtiyaç duyduğumuzda key kullanarak value cagırırız

- url=https://www.amazon.com/
- browser=chrome
- username=manager
- password=3145665
- name=Ali



The screenshot shows a file explorer window with the following structure:

- com.testNgPOM_Fall2022
- .idea
- src
 - main
 - test
 - java
 - pages
 - tests
 - utilities
 - ConfigReader
 - CrossDriver
 - Driver
 - ReusableMethods
 - TestBase
 - TestBaseClass
 - TestBaseCross
 - TestBaseRapor
 - target
 - test-output
 - xmlFiles
- configuration.properties

The configuration properties file contains the following entries:

browser=chrome
facebookUrl=https://www.facebook.com
amazonUrl=https://www.amazon.com
amazonAramaKelimesi=Cokokrem
dropdownArananOption=Electronics
myUrl=https://www.qualitydemy.com
myGecerliEmail=user_1106147@login.com
myGecerliPassword=31488081

TestNG

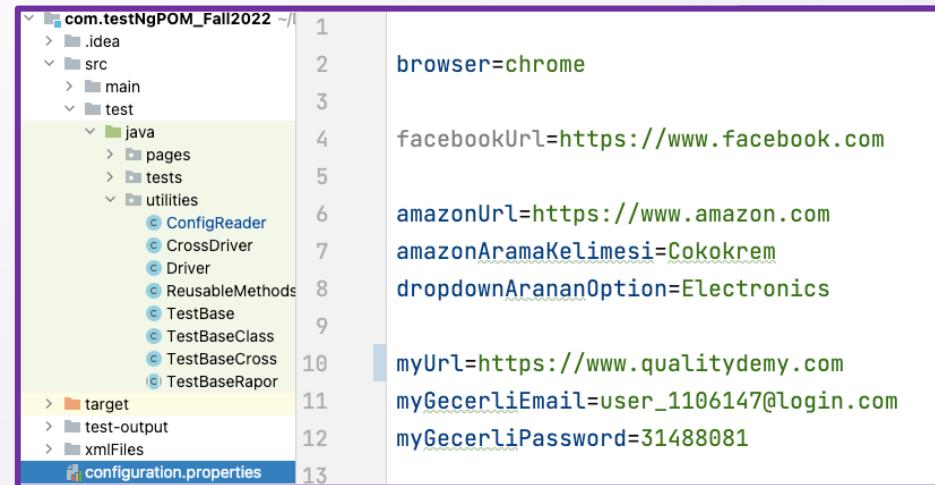
POM Properties File

configuration.properties file Olusturmak icin project'imize sag click yapin

New => File => isim : configuration.properties

Dosya olustururken bizim icin onemli olan ismi degil, uzantisidir (.properties)

Bu uzanti sayesinde Java kutuphanesinden Properties Class'ini kullanabilir, olusturacagimiz obje yardimiyla configuration.properties dosyasindaki key-value ikililerine ulasabiliriz



```
com.testNgPOM_Fall2022 ~/
  > .idea
  > src
    > main
    > test
      > java
        > pages
        > tests
        > utilities
          < ConfigReader
          < CrossDriver
          < Driver
          < ReusableMethods
          < TestBase
          < TestBaseClass
          < TestBaseCross
          < TestBaseRapor
      > target
      > test-output
      > xmlFiles
  configuration.properties
```

browser=chrome
facebookUrl=https://www.facebook.com
amazonUrl=https://www.amazon.com
amazonAramaKelimesi=Cokokrem
dropdownArananOption=Electronics
myUrl=https://www.qualitydemy.com
myGecerliEmail=user_1106147@login.com
myGecerliPassword=31488081

properties file test dataları saklar.

Bu dosyayı kullanmanın amacı, kodu sabit(hard coded) değil, dinamik yapmaktır. Bu dosya sayesinde tüm kullanıcılar verilere kolayca ulaşabilir ve update edebilir.

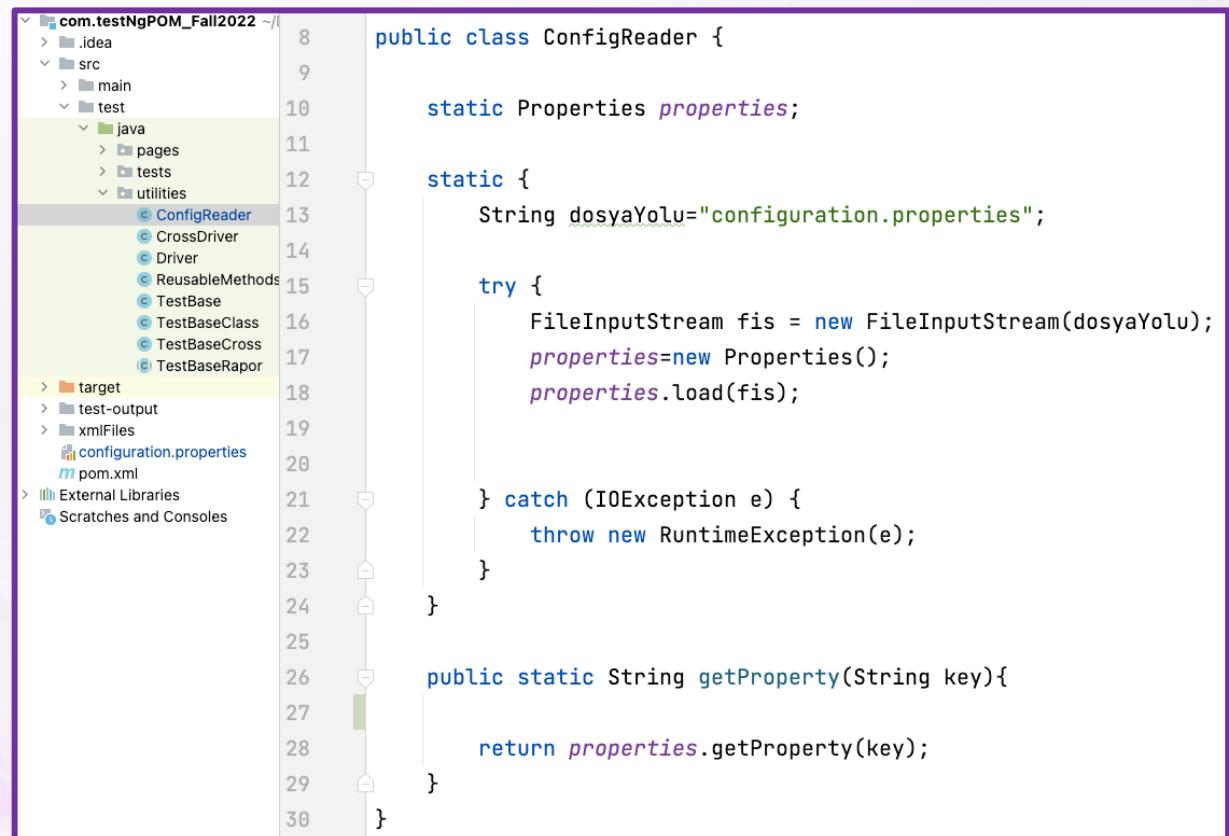
POM ConfigurationReader

ConfigurationReader class test method'larimiz ile Configuration.properties arasında iletisimi saglar.

Bu class'da test class'larinden kolayca ulasmak icin static method bulunur.

Method static oldugundan method icerisinden cagiracagimiz variable da static olmalidir.

Kullanicagimiz static variable'a ilk degeri atamak icin(instantiate) de static block kullaniriz.



The screenshot shows a Java code editor with the following code:

```
public class ConfigReader {
    static Properties properties;
    static {
        String dosyaYolu="configuration.properties";
        try {
            FileInputStream fis = new FileInputStream(dosyaYolu);
            properties=new Properties();
            properties.load(fis);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    public static String getProperty(String key){
        return properties.getProperty(key);
    }
}
```

The code defines a static Properties variable named `properties`. It contains a static block that reads the `configuration.properties` file into this variable using a `FileInputStream` and the `Properties.load` method. If an `IOException` occurs during this process, it is caught and a `RuntimeException` is thrown. Finally, a static method `getProperty` is provided to retrieve values from the `properties` object.

POM ConfigurationReader

Test Class

```
public class TestClass {
    @Test
    public void test1(){
        String url = ConfigReader.getProperty("amazon_url")
    }
}
```

1) Test method'undan

“amazon_url” key'e ait
value'yu kullanmak
istedigimizde

2) ConfigReader Class'indan
getProperty() method'unu
kullaniriz.

3) getProperty() method'u
configuration.properties
dosyasina gidip istedigimiz
key'e ait value'yu bize
dondurur.

ConfigurationReader Class

```
public class ConfigReader {
    private static Properties properties;

    static {
        String path="configuration.properties";
        try {
            FileInputStream fileInputStream=new FileInputStream(path);
            properties =new Properties();
            properties.load(fileInputStream);

            fileInputStream.close();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static String getProperty(String key){
        return properties.getProperty(key);
    }
}
```

Not: Properties
dosyasında
olmayan bir
anahtar(key)
alırsak, null
degeri döndürür

Configuration.properties file

amazon_url=https://www.amazon.com
aranan_kelime=Amazon
aranan_hucre=Home Services



TestNG

Page Object Model

PositiveTest

1- <https://www.qualitydemy.com/> anasayfasina gidin

2- login linkine basin

3- Kullanici email'i olarak valid email girin

4- Kullanici sifresi olarak valid sifre girin

5- Login butonuna basarak login olun

6- Basarili olarak giris yapilabildigini test edin



TestNG

Page Object Model

NegativeTest

1- <https://www.qualitydemy.com/> anasayfasina gidin

2- login linkine basin

3- Kullanvalid 3 farkli test method'u olusturun.

- gecerli username, gecersiz password
- gecersiz username, gecerli password
- gecersiz username, gecersiz password.

4- Login butonuna basarak login olun

5- Basarili olarak giris yapilamadigini test edin

TestNG

Assertions

Test otomasyonunun temel amacı Test Case'lerde belirlenen her bir adimin test edilmesidir.

TestNG testleri yaparken bize daha fazla tercih imkani verir.

- Junit'deki gibi ilk failed olan assertion'da calismayi durdurmak
- Biz raporla diyene kadar tum testleri gerceklestirip, raporla deyince kac testin FAILED oldugunu raporlayip, sonra calismayi durdurmak.

```
public class C01 {  
  
    WebDriver driver;  
  
    @Test  
    public void amazonTest(){  
        WebDriverManager.chromedriver().setup();  
        driver=new ChromeDriver();  
        driver.get("https://www.amazon.com");  
        Assert.assertTrue(driver.getTitle().contains("amazon"));  
    }  
    @Test (dependsOnMethods = "amazonTest")  
    public void amazonAnasayfaTest(){  
  
        SoftAssert softAssert =new SoftAssert();  
        WebElement aramaKutusu=driver.findElement(By.id("twotabsearchtextbox"));  
        aramaKutusu.sendKeys( ...keysToSend: "java"+ Keys.ENTER);  
        WebElement ilkUrun=driver.findElement(By.xpath("//span[@class='a-size-base-plus a-color-base a-text-normal'][1]"));  
        softAssert.assertTrue(ilkUrun.getText().contains("java"), message: "ilk urun java icermiyor");  
        softAssert.assertAll();  
    }  
}
```

Hard Assert

JUnit'te Öğrendiğimiz Assertion ile aynidir. TestNG'de soft assertion da oldugundan, ayristirmak icin bu isim kullanilmistir.

JUnit'ten bildigimiz uzere kullanabilecegimiz 3 cesit hard assertion turu vardir

- i. Assert.assertEquals()
- ii. Assert.assertTrue()
- iii. Assert.assertFalse()

Hard assertion herhangi bir assertion FAILED olursa, test method'nun calismasini durdurur ve kalan kodlari yürütmmez (stops execution).

Test case'in nerede FAILED olduğunu hemen anlamak ve kod'a direkt müdahale etmek istenirse hard assertion tercih edilebilir.

TestNG

Soft Assert (Verification)

SoftAssert doğrulama (verification) olarak da bilinir.

softAssert kullandığımızda, assert FAILED olsa bile test method'unun istediğiniz kısmını durdurmaz ve yürütmeye devam eder. if else statement'da olduğu gibi.

Test method'unun istediğimiz bolumde yapılan tüm testleri raporlar

Eger assertion'lardan FAILED olan varsa raporlama yapılan satirdan sonrasini calistirmaz.

SoftAssert class'indaki method'lari kullanmak icin kullanabilmemiz için object olusturmamiz gereklidir.



TestNG

Soft Assert (Verification)

1

SoftAssert objesi olusturalim

```
SoftAssert softAssert= new SoftAssert();
```

2

Istedigimiz sayida verification'lari yapalim

```
// 3- sifre bosluk icermemeli
softAssert.assertFalse(sifre.contains(" "), message: "Sifre bosluk icermemeli");
// 4- uzunlugu en az 8 karakter olmali
softAssert.assertTrue(condition: sifre.length()>=8, message: "uzunluk en az 8 karakter olmali");
```

3

SoftAssert'in durumu raporlamasini isteyelim

```
softAssert.assertAll();
```

Soft Assert vs Hard Assert

Ortak Ozellikler :

SoftAssert ve HardAssert method'ları TestNG'den gelmektedir.

Kullanma amaçları farklı olsa da method'lar aynıdır.

Farklar :

Hard Assertion fail olursa test method'larının execute etmesi durur. Ve FAILED olarak tanımlanır.

Eğer softAssert FAIL olursa test method'ları execute etmeye devam eder. assertAll dedigimizde FAILED olan assertion varsa execution durur.

Soft Assert (Verification)

Yeni bir Class Olusturun : C03_SoftAssert

1. "http://zero.webappsecurity.com/" Adresine gidin
2. Sign in butonuna basin
3. Login kutusuna "username" yazın
4. Password kutusuna "password" yazın
5. Sign in tusuna basin
6. Online banking menusu icinde Pay Bills sayfasına gidin
7. "Purchase Foreign Currency" tusuna basin
8. "Currency" drop down menusunden Eurozone'u secin
9. soft assert kullanarak "Eurozone (euro)" secildigini test edin
10. soft assert kullanarak DropDown listesinin su secenekleri oldugunu test edin
"Select One", "Australia (dollar)", "Canada (dollar)", "Switzerland (franc)", "China (yuan)", "Denmark (krone)", "Eurozone (euro)", "Great Britain (pound)", "Hong Kong (dollar)", "Japan (yen)", "Mexico (peso)", "Norway (krone)", "New Zealand (dollar)", "Sweden (krona)", "Singapore (dollar)", "Thailand (baht)"



Wise Quarter
first class IT courses

Selenium Ders-12

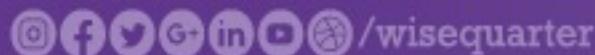
POM Driver Class
Xml File Kullanımı
Reusable Methods
TestNG Reports



The future at your fingertips

+1 912 888 1630

www.wisequarter.com





TestNG

POM Driver Class

Temel Hedefimiz: Test methodlarimizda kullanacagımız WebDriver driver'i utilities altindaki Driver Class'ında olusturmak ve testlerimizde bu class ismi üzerinden driver'a ulasip olusturma, kullanma ve kapatma islemlerini yapmak.

Basic Driver Class

```
public class Driver {  
  
    private static WebDriver driver;  
  
    public static WebDriver getDriver() {  
  
        if (driver == null) {  
            WebDriverManager.chromedriver().setup();  
            driver = new ChromeDriver();  
        }  
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));  
        driver.manage().window().maximize();  
  
        return driver;  
    }  
}
```

```
public class C02 {  
  
    @Test  
    public void test01(){  
  
        Driver.getDriver().get("https://www.amazon.com");  
    }  
}
```

Test Class(driver->Driver.getDriver())

Tum Browserlar icin Kullanim

Driver Classımızı tüm tarayıcılar(browser) için geliştireceğiz.

Driver objesi create etmeden önce farklı tarayıcı koşullarını kontrol etmek için switch statement kullanıyoruz.

Driver Class'i singleton pattern'e uygun dizayn ederek tüm projede farklı driver üretilmesinin onune geceriz

Sonra TestBasede yaptığımız gibi "wait" koyabiliriz.

Ardından driver'i kapatmak için method oluşturuyoruz.

Şu andan itibaren TestBase Classını değil Driver Classını kullanacağız.

TestNG

POM Driver Class

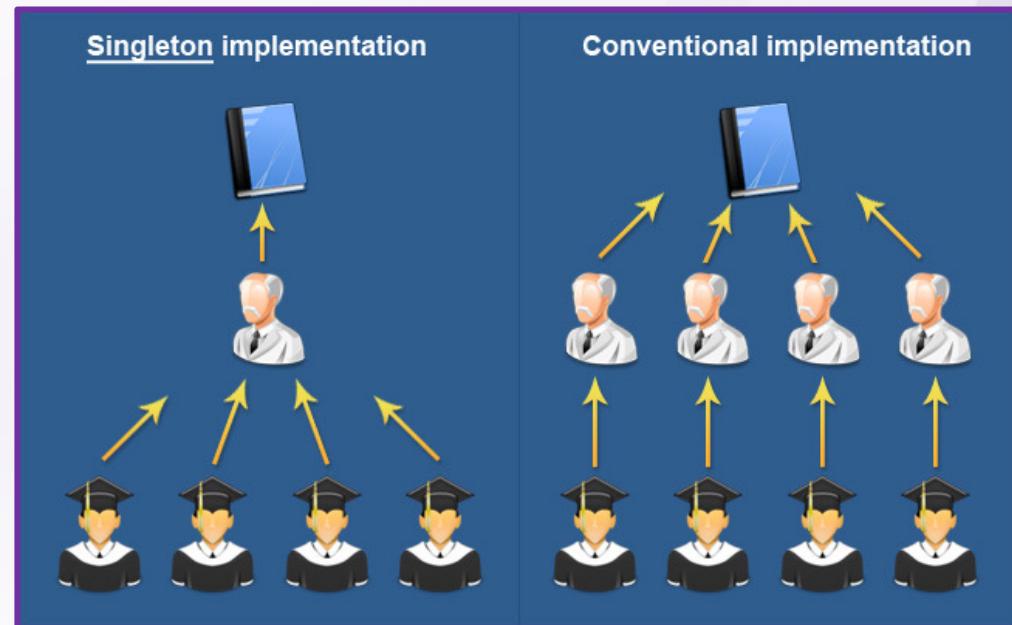
```
public class Driver {  
    private Driver(){  
    }  
    private static WebDriver driver;  
    public static WebDriver getDriver() {  
        if (driver == null) {  
            switch (ConfigReader.getProperty("browser")) {  
                case "chrome":  
                    WebDriverManager.chromedriver().setup();  
                    driver = new ChromeDriver();  
                    break;  
                case "firefox":  
                    WebDriverManager.firefoxdriver().setup();  
                    driver = new FirefoxDriver();  
                    break;  
                case "safari":  
                    WebDriverManager.getInstance(SafariDriver.class);  
                    driver = new SafariDriver();  
                    break;  
                case "opera":  
                    OperaDriverManager.operadriver().setup();  
                    driver = new OperaDriver();  
                    break;  
            }  
        }  
        driver.manage().timeouts().implicitlyWait( 15, TimeUnit.SECONDS);  
        driver.manage().window().maximize();  
        return driver;  
    }  
    public static void closeDriver(){  
        if (driver !=null){  
            driver.close();  
            driver=null;  
        }  
    }  
}
```

Singleton Pattern (Tekli Kullanım)

Herhangi bir Java classından obje kullanımını sınırlayabiliriz. Buna Singleton pattern(tekli kullanım) denir.

Singleton pattern, class'ı tek bir instance ile kısıtlayan bir software dizayn kalıbıdır.

Proje genelinde farklı objeler oluşturulmadan tek bir instance gereksinimimiz olduğunda Singleton class kullanırız.



Sadece bir obje oluşturmak ve buna her ihtiyaç duyduğumuzda kullanmamız performans ve bellek kullanımı için de faydalıdır.

1- Page Object Model : Testlerimizi daha kolay ve düzenli olarak hazırlamamız ve calistirmamız icin olusturulmus bir modeldir.

Framework icin uretilmis benzer modeler olmakla birlikte en guncel olan ve cok kullanilan model oldugu icin POM'i ogrendik

2- POM dosya yapisi :

- Pages : Test yapacagimiz web page'ler icin Pages package'in altında bir class olusturuyoruz. Bu class'larda mutlaka yapmamız gereken sey driver'i olusturdugumuz Driver clasindan alip PageFactory.initElements ile ilk deger atamasi yapmaktir. Sonrasinda web sayfamizda kullanacagimiz WebElementlerin tamamini public olarak olusturmak ve @FindBy notasyonu ile locate etmektir. Eger istersek login gibi bazi adimlari yapacak methodlari da bu class'da olusturabiliriz.

Test clasimizdan Page sayfasındaki variable ve method'lara obje olusturup erisim saglariz.

- **configuration.properties** : Bu dosyayı testlerimizde kullanacağımız url,test dataları gibi kullanıcının aldığımız dataları dinamik yapmak için kullanırız.

Tüm testlerimizi bu sayfadan alacağımız datalara göre dizayn ederiz. Böylece bu dosyada yapacağımız bir değer değişikliği ile tüm testCase'lerindeki test datalarını güncelleleyebiliriz.

Bu sayfayı basit bir text dosyası gibi dizayn ederiz her test datasını key=value şeklinde key,value ile oluştururuz.

- **ConfigReader** : Bu class test clasımız ile configuration.properties dosyası arasında tercumanlık yapar. İçinde .properties uzantılı dosyaları okumak için gerekli bir static blok oluştururuz. Ayrıca Test classlarımızdan çağrılmak için getProperty() methodunu oluştururuz. Bu method test class'ından gönderdiğimiz key değerini static blok yardımı ile configuration.properties'de bulup karşısındaki value'yu bize dondurur.

- **Driver** : Test clasimizda ve page clasinda kullanacagimiz driver'i olusturdugumuz class'tir. Utilities Package'i altında olustururuz. Driver class'ini Singleton yapabiliriz.

Driver'i static olarak olusturur ve olusturdugumuz getDriver() method icinde driver'imiza deger atamasi yapariz.

Is hayatinda karsilasacagimiz farkli browser'lar (chrome,firefox,safari vb..) deger atama islemi yapmadan once kullanicinin tercihini aliriz.

Kullanici tercihini almak icin configuration.properties dosyasinda browser=chrome gibi bir key,value ikilisi olusturur buradaki tercihe gore driver'a deger atamak icin de switch case kullaniriz.

Ayrica her driver cagirdigimizda yeni driver olusturmamasi icin once if ile driver'in atamasi yapilmis mi control ederiz, atama yapilmissa ayni driver ile devam eder, atama yapilmamissa yeni bir driver olusturur ve deger atayip test sayfasina doneriz.

Bu Class'ta ayrica window.manage ayarlarini da yapar, en sonda da closeDriver method ile driver'i kapatma islemine de yardimci oluruz

TestNG

Smoke Test

Smoke Test: Kullandigimiz uygulamanin onemli/temel fonksiyonlarini test etmek icin yapilir. Genellikle sabah ilk ise baslama gorevimizdir. Login,logout,sepete ekle,odeme yap.. gibi temel islevleri test ederiz. Eger smoke test FAILED olursa zaman gecirmeksizin tum ekibi haberدار ederiz.

<https://www.qualitydemy.com> sayfasinda
genel testlerimizi yapacagiz

Sistemin saglikli calistigini anlamak icin 3 test yapacagiz

- 1- Pozitif Login Test
- 2- Negatif Login Test
- 3- E2E Testi (system testi)



End to End Test(E2E): Bir uygulamanin tum adimlarini bastan sona kadar test etmek demektir. Ornegin biz yonetici bir oda olusturabilir mi diye test yaptigimizda sisteme giristen oda olusturuldu yazisi gorulunceye kadar tum adimlari test etmis oluruz, dolayisiyla E2E testi yapmis oluruz. Bu testin diger adi da Sistem Testi'dir.

TestNG

Xml Files

XML, hem insanların hem de makinelerin okuyabileceği belgeleri kodlamak için bir sözdizimi tanımlamak üzere World Wide Web Consortium (W3C) tarafından oluşturulan bir biçimlendirme dilidir.

Veri saklamak ve farklı işletim sistemleri arasında veri transfer etmek için kullanılan metin işaretleme dili XML ile hazırlanmış dökümanlar .xml formatına sahip dosyalarda saklanır.

Biz de framework'umuzdeki belirli testleri veya tüm testleri otomatik olarak calistirmak icin xml dosyaları kullanırız

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

<suite name="Suite1" verbose="1" >
  <test name="Nopackage" >
    <classes>
      <class name="NoPackageTest" />
    </classes>
  </test>

  <test name="Regression1">
    <classes>
      <class name="test.sample.ParameterSample"/>
      <class name="test.sample.ParameterTest"/>
    </classes>
  </test>
</suite>
```

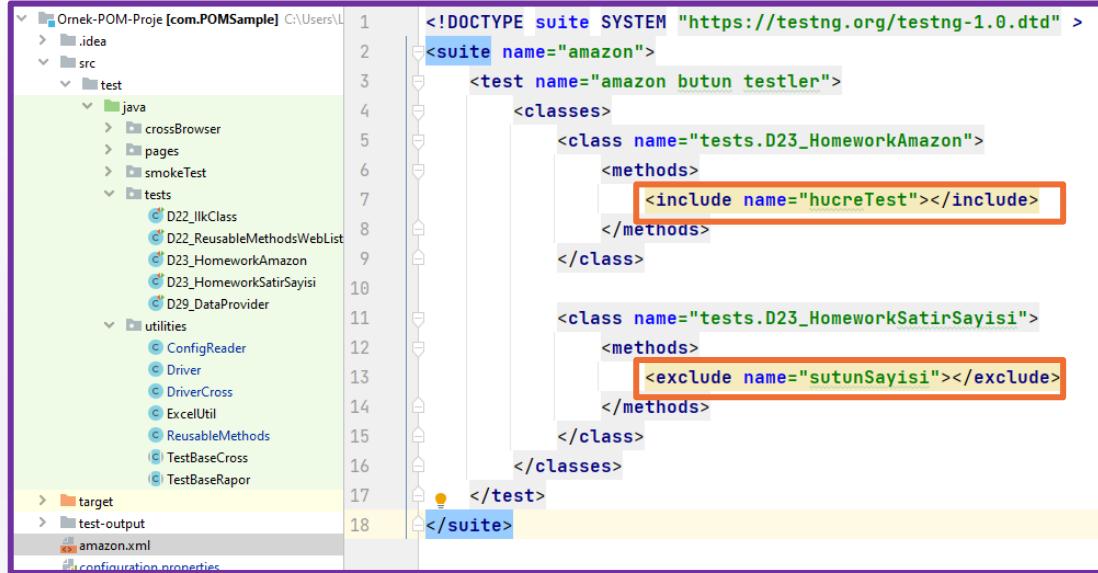
[TestNG xml ile ilgili tüm dokumantasyon için :](#)
<https://testng.org/doc/documentation-main.html#testng-xml>

TestNG

Xml Files

Testng framework'de xml dosyasi kullanma nedenlerinden biri, belirli suitleri, testleri, package lari, classları veya method lari çalıştırmaktadır.

Belirli testleri, package lari, classları veya method'lari dahil edebilir (include) veya hariç (exclude) tutabiliriz. Bu da bize esneklik (flexibility) kazandırır.



The screenshot shows a Java project structure in an IDE. The `src/test/java` directory contains several packages like `crossBrowser`, `pages`, `smokeTest`, and `tests`. The `tests` package contains multiple test classes such as `D22_IlkClass`, `D22_ReusableMethodsWebList`, `D23_HomeworkAmazon`, `D23_HomeworkSatirSayisi`, and `D29_DataProvider`. The `utilities` package contains utility classes like `ConfigReader`, `Driver`, `DriverCross`, `ExcelUtil`, `ReusableMethods`, `TestBaseCross`, and `TestBaseRapor`. The `target` and `test-output` directories are also visible.

The `amazon.xml` file is an XML configuration file for TestNG. It defines a suite named "amazon" which runs the test "amazon butun testler". It includes the class `tests.D23_HomeworkAmazon` and excludes the method `hucreTest`. It also includes the class `tests.D23_HomeworkSatirSayisi` and excludes the method `sutunSayisi`.

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="amazon">
  <test name="amazon butun testler">
    <classes>
      <class name="tests.D23_HomeworkAmazon">
        <methods>
          <include name="hucreTest"></include>
        </methods>
      </class>
      <class name="tests.D23_HomeworkSatirSayisi">
        <methods>
          <exclude name="sutunSayisi"></exclude>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```

Sadece birkaç basit yapılandırma ile TestNG.xml dosyasını kullanarak belirli test senaryolarını yürütebiliriz.

Daha fazlasi icin: <https://testng.org/doc/documentation-main.html>



TestNG

Xml Files

XML dosya olustururken hiyerarsi (buyukten kucuge siralama) onemlidir. Her zaman suite ile baslayip hangi seviyede test calistirmak istersek o seviyeye kadar sirali olarak kademeleri yazmaliyiz

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="coklu class calistirma">
  <test name="coklu calistirma testi">
    <classes>
      <class name="tests.D23_HomeworkSatirSayisi"></class>
      <class name="tests.D23_HomeworkAmazon"></class>
    </classes>
  </test>
</suite>
```

Eger calistiracagimiz class'lar farkli hiyerarsilere ait ise yine suite ile baslariz, sonra ayrisma kadamesinden itibaren farkli hiyerarsi kumeleri olustururuz.

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="sirali method calistirma">
  <test name="sirali method">
    <classes>
      <class name="tests.D23_HomeworkAmazon">
        <methods>
          <include name="AmazonYazisi"></include>
        </methods>
      </class>
      <class name="tests.D23_HomeworkSatirSayisi">
        <methods>
          <exclude name="sutunSayisi"></exclude>
        </methods>
      </class>
    </classes>
  </test>
</suite>
```



TestNG

Xml Files

Istenen Package'lari Calistirma

Yeni bir xml dosyasi olusturalim :

belirliPackageCalistirma

Smoke Test package'indaki tum testleri calistiralim

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="smoke package">
    <test name="smoke package">
        <packages>
            <package name="smokeTest"></package>
        </packages>
    </test>
</suite>
```

TestNG

Xml Files

Istenen Class'lari Calistirma

Yeni bir xml dosyasi olusturalim :

belirliClasslariCalistirma

<package> attribute yerine <classes> ve sonra <class> attribute kullanarak istediginiz class'lari calistirin

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="coklu class calistirma">
    <test name="coklu calistirma testi">
        <classes>
            <class name="tests.D23_HomeworkSatirSayisi"></class>
            <class name="tests.D23_HomeworkAmazon"></class>
        </classes>
    </test>
</suite>
```

TestNG

Xml Files

Istenen Method'lari Calistirma

Yeni bir xml dosyasi olusturalim : belirliMethodlariCalistirma

<classes> attribute altinda <class> ve <methods> attribute'lerini ve icinde <include>, <exclude> attribute'lerini kullanalim

```
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd" >
<suite name="sirali method calistirma">
    <test name="sirali method">
        <classes>
            <class name="tests.D23_HomeworkAmazon">
                <methods>
                    <include name="AmazonYazisi"></include>
                </methods>
            </class>
            <class name="tests.D23_HomeworkSatirSayisi">
                <methods>
                    <exclude name="sutunSayisi"></exclude>
                </methods>
            </class>
        </classes>
    </test>
</suite>
```

Istenen Gruplari Calistirma

Yeni bir xml dosyasi olusturalim : belirliGruplariCalistirma

Group calistirmak icin hem grup adini tanimlamak hem de nerede arayacagimizi belirtmek zorundayiz

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="minor regression" verbose="2">
<test name="Regression1">
<groups>
  <run>
    <include name="Regression1" />
  </run>
</groups>
<packages>
  <package name="com.abc.smokeTest"></package>
</packages>
</test>
</suite>
```



TestNG

Xml Files

Yeni bir xml dosyasi olusturalim : tumTestleriCalistirma

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
<suite name="minor regression" verbose="2">
    <test name="Regression1">
        <packages>
            <package name=".*"></package>
        </packages>
    </test>
</suite>
```



Wise Quarter
first class IT courses

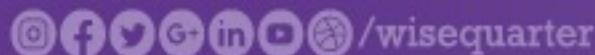
Selenium
Ders-13
Paralel Tests
Cross Browser Tests
Cucumber



The future at your fingertips

+1 912 888 1630

www.wisequarter.com



TestNG

Html Reports

TESTNG rapor hazırlamaz. Eğer testimiz ile ilgili rapor hazırlamak istersek, farklı kütüphanelerden yardım almamız gereklidir.

Pom.xml dosyamıza aeventstack dependency'sini ekliyoruz.

```
<!-- https://mvnrepository.com/artifact/com.aventstack/extentreports -->
<dependency>
    <groupId>com.aventstack</groupId>
    <artifactId>extentreports</artifactId>
    <version>4.0.9</version>
</dependency>
```



TestNG

Html Reports

Extent Reports :

HTML raporlama aracıdır. Bize Html raporları verir. Test adımlarını kaydetmemize yardımcı olur. Ayrıca ekran görüntüleri ekleyebiliriz.

3 tane obje oluşturup kullanırız

1. *ExtentReports extentReports;* Raporlamayı başlatmak için ExtentReports'a ihtiyacımız var.
flush() metodunu için ExtentReports kullanıyoruz.

2. *ExtentHtmlReporter extentHtmlReporter;* Bu, özel raporlara ve raporların yapılandırmasına sahip olmamıza yardımcı olur, html raporlarını oluşturur. Bunu raporun oluşturulacağı yolu ayarlamak için de kullanıyoruz.

3. *ExtentTest extentTest;* Bilgi eklemek için. Test adımlarını eklemek için (açıklama). Günlükleri(logs) ekliyoruz.

`extentTest.info ("URL'yi Açıma");` bilgi sadece adım eklemek içindir



TestNG

Html Reports

```
@BeforeTest      : burada rapor için nesne oluşturuyoruz, hazırlık yapıyoruz  
  
@Test           : raporu dolduruyoruz, içerisinde veriler ekliyoruz.  
@AfterMethod    : eğer @Test başarısızsa, rapora ekran görüntüsü ekliyoruz.  
  
@AfterTest       : raporlandırma işlemini sonlandırıyoruz.
```

Testimize rapor oluşturma adımları

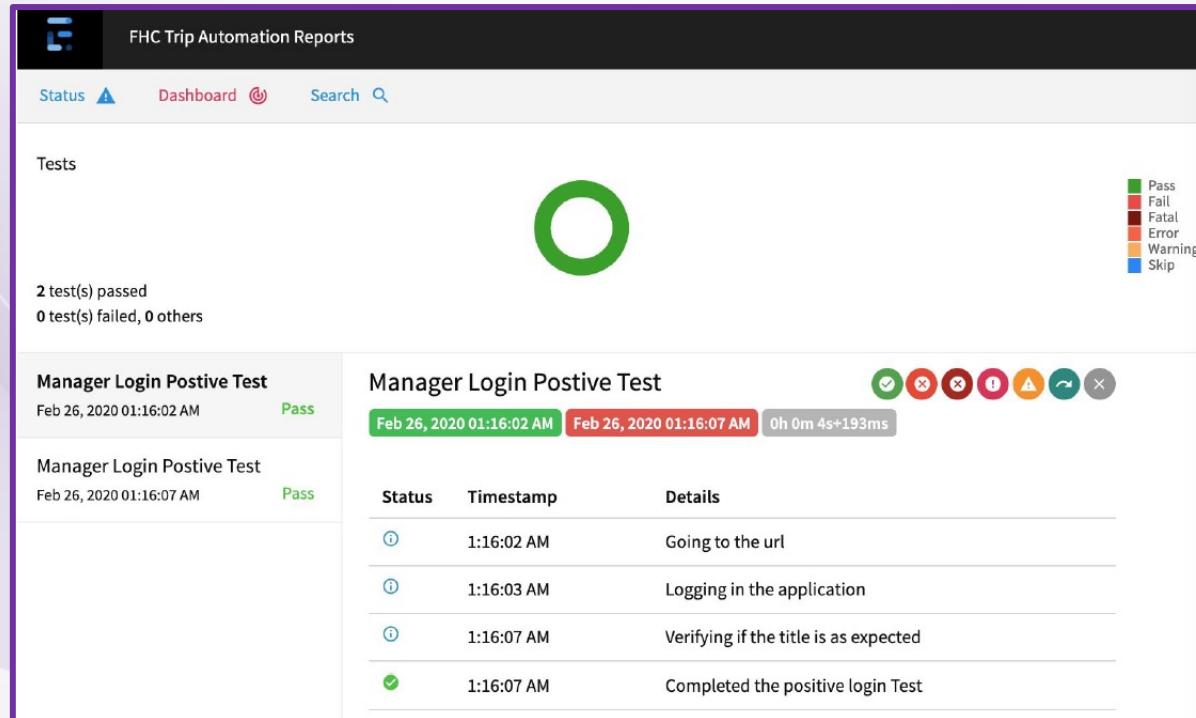
- 1- Test class'ini extends ile TestBaseRapor Class'ına child yapalım
- 2- *extentTest=extentReports.createTest("Test ismi", "Tanim");* rapor olusturalım
- 3- Gerekli/istedigimiz satirlara extentTest.info("Acıklama") ekleyelim
- 4- Assert olan satırda açıklamayı extentTest.pass ile yapabiliriz

TestNG

Html Reports

Testimiz bittikten sonra olusturulan raporu “open in browser” ile acabiliriz.

Eger test basarisiz ise Screenshots klasorunden resmine de ulasabiliriz



FHC Trip Automation Reports

Status ▲ Dashboard ↻ Search 🔍

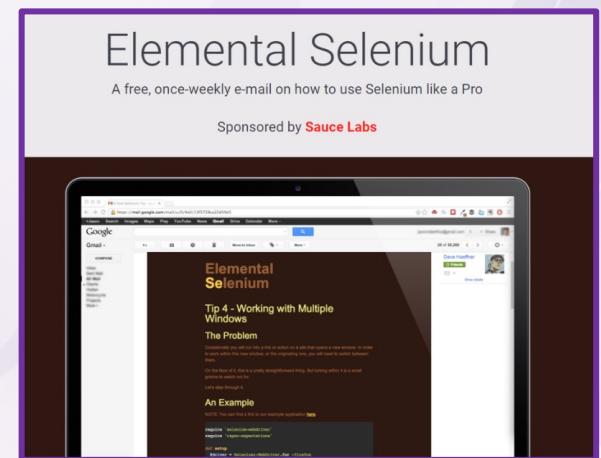
Tests

2 test(s) passed
0 test(s) failed, 0 others

Manager Login Positive Test
Feb 26, 2020 01:16:02 AM Pass

Manager Login Positive Test
Feb 26, 2020 01:16:02 AM Feb 26, 2020 01:16:07 AM 0h 0m 4s+193ms

Status	Timestamp	Details
ⓘ	1:16:02 AM	Going to the url
ⓘ	1:16:03 AM	Logging in the application
ⓘ	1:16:07 AM	Verifying if the title is as expected
✓	1:16:07 AM	Completed the positive login Test



Elemental Selenium

A free, once-weekly e-mail on how to use Selenium like a Pro

Sponsored by Sauce Labs

Tip 4 - Working with Multiple Windows

The Problem

Occasionally you will run into the situation where a file or action on a file that opens a new window. In order to interact with this file or action, or to register one, you will need to switch between windows.

One way to do this is to use a context switcher tool, but using selenium it is a much easier process.

Let's take a look at an example.

An Example

NOTE: You can find a link to our example application here.

TestNG Paralel Testing

TestNg'de paralel test xml dosyasi kullanilarak yapilir.

Paralel test calisma suresini azaltir, dolayisiyla zaman kazanmak icin parallel test kullanilir.

Paralel test ayni anda birden fazla test case'i eszamanli olarak calistirmak demektir.

Xml dosyasinda belirlenen testleri belirledigimiz level seviyesinde belirledigimiz thread-count sayisinda parallel calistirir

Classes,methods seviyesinde calistirirsak verilen tum gorevler bitene kadar baska class veya method varsa calismaya devam eder. Level olarak Tests secilirse testlerin tanimlanmasi gereklidir

Cross Browser (Multi Browser) test ise farkli browserlar ile test yapmak demektir. Sirali (sequential) veya paralel yapilabilir.

TestNG Paralel Testing

Coklu calistirma, Parallel calistirma ve Cross Browser calistirma farkli farkli islemlerdir.

Mesela 5 testi sirali olarak ama topluca calistirirsak → sirali coklu calistirma

5 testi ayni anda calismaya baslayan iki driver'la calistirirsam

→ parallel calistirma

5 testi sirali olarak calistirip, ilk ucunu chrome, son ikisini firefox'da calistirirsam

→ sirali cros browser

5 testin ucunu chrome, ikisini firefox ile calistirip, chrome ve firefox'u ayni zamanda calistirirsam

→ parallel cros browsing test olur



TestNG Paralel Testing

Classes

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Paralel Test 2" parallel="classes" thread-count="2">
  <test name="Class Paralel">
    <classes>
      <class name="com.abc.tests.D26_AmazonSatirSutunSayisi"></class>
      <class name="com.abc.tests.D26_AmazonHucreTesti"></class>
      <class name="com.abc.tests.D25_HtmlRapor1"></class>
      <class name="com.abc.tests.D25_WindowHandle"></class>
    </classes>
  </test>
</suite>
```



TestNG

TestNG Parallel Testing

Packages

```
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Parallel Test 1" parallel="methods" thread-count="2" >
    <test name="Smoketest Parallel" >
        <packages>
            <package name="com.abc.smokeTest"></package>
            <package name="com.abc.tests"></package>
        </packages>
    </test>
</suite>
```

TestNG

@DataProvider

@DataProvider bir TestNG annotation'ıdır, dolayısıyla sadece TestNG ile kullanılır. DDT (Data Driven Test) konseptinde teste veri sağlamak için kullanılır.

```
public class C01_DataProvider {  
    @DataProvider  
    public static Object[][] AranacakKelimeProvider() {  
  
        Object[][] arananKelimeler= {{"Nutella"}, {"Java"}, {"Apple"}, {"Samsung"}, {"TV"}};  
        return arananKelimeler;  
    }  
  
    @Test(dataProvider = "AranacakKelimeProvider")  
    public void aramaTesti(String aranacakKelime){  
        // Amazon anasayfaya gidin  
        Driver.getDriver().get(ConfigReader.getProperty("amazonUrl"));  
        // Nutella, Java, Apple, Samsung, TV için arama yapın  
        AmazonPage amazonPage=new AmazonPage();  
        amazonPage.aramaKutusu.sendKeys( ...keysToSend: aranacakKelime + Keys.ENTER);  
        // Arama sonuçlarının aranın kelime içerdigini test edin  
        String actualSonucYazisi=amazonPage.aramaSonucElementi.getText();  
        ReusableMethods.bekle( saniye: 3);  
        Assert.assertTrue(actualSonucYazisi.contains(aranacakKelime));  
    }  
}
```

Cucumber'daki Scenario Outline ile aynı işlev sahiptir

TestNG

Cross Browser Testing

1. Cross Browser test bir uygulamayı farklı browserlar ile test etmek demektir
2. Testleri sıralı (sequential) veya paralel olarak yapabiliriz
3. Cv'niz açısından Cross Browser test önemlidir cunku ileri seviyeyi gösterir
4. Cross Browser testi yapabilmek için framework'de gerekli düzenlemeleri yapmanız gereklidir. (Bu her tester'in sahip olacağı bir özellik değildir, dolayısıyla size 1 adım one çıkarır)
5. Herbir adımı ezbere bilmek zorunda değiliz ama mantığı anlamak ve bunu sözlü olarak ifade edebilmek zorundayız



Cross Browser Testing

1. Crossbrowser testi icin yeni bir driver class'i olusturacagiz : DriverCross
 - getDriver methoduna parametre ekleyecegiz WebDriver getDriver(String browser)
 - if blogundan once bir satir kod ekleyecegiz

```
browser = browser == null ? ConfigReader.getProperty("browser") : browser;
```
 - switch case'deki degeri degistirelim switch(browser)

Bu class'in gorevi xml dosyasindan parameter olarak gonderilen browser'i driver olarak tanimlamaktir. Xml dosyasindan parametre gelmezse .properties dosyasinda tanimli browser'i kullanir
2. Crossbrowser testi icin yeni bir TestBase class'i olusturacagiz
 - Basa gelen parametreyi kullanmak icin @Parameters("browser") ekleyecegiz
 - setup methodune parametre gonderecegiz setUp(@Optional String browser) burada optional yazma sebebimiz parameter gelmese de calismasini istememiz

TestNG

Cross Browser Testing

3. Farkli browser'lar ile calistirmak istedigimiz class'lari bir package altina toplayalim **crossBrowser** ve class'lari TestBaseCross clasina **extends** ile child olarak tanimlayalim
4. Xml dosyasi olusturalim ve cross browser icin <test> satirinin altina browser icin parametre gonderelim

```
<parameter name="browser" value="firefox"></parameter>
```

5. Paralel calistirmak istersek paralel calistirma kodlarini eklememiz yeterli

Ornek Proje Olusturma

1. File – New – Project e tikliyoruz
2. Maven'i seciyoruz
3. Name'e projemizin ismini yaziyoruz
4. Cikan Alert mesajinda New Window veya This Window secilebilir
5. Pom xml'imizi düzenleniyoruz

- a)

```
<properties>
<maven.compiler.source>1.8</maven.compiler.source>
<maven.compiler.target>1.8</maven.compiler.target>

</properties>
```

*** bu kod Javanin sürümüyle alakali sorunları halletmeye yarıyor
- b)

```
<dependencies>
    Kutuphanelerimizi bu tag'lar arasına yazıyoruz
</dependencies>
```

6) <https://mvnrepository.com/> a gidip kutuphanelerimizi tek tek aliyoruz

a) Selenium-Java Kutuphanesi

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.141.59</version>
</dependency>
```

b) WebDriverManager Kutuphanesi

```
<!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
<dependency>
    <groupId>io.github.bonigarcia</groupId>
    <artifactId>webdrivermanager</artifactId>
    <version>4.2.0</version>
</dependency>
```



c) Testng Kutuphanesi

```
<!-- https://mvnrepository.com/artifact/org.testng/testng -->
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
    <version>7.1.0</version>
    <scope>test</scope>
</dependency>
```

d) Java Faker Kutuphanesi

```
<!-- https://mvnrepository.com/artifact/com.github.javafaker/javafaker -->
<dependency>
    <groupId>com.github.javafaker</groupId>
    <artifactId>javafaker</artifactId>
    <version>1.0.2</version>
</dependency>
```

Ornek Proje Olusturma

7) Pom Dosyasini oluşturma işlemimiz bitti. Kutuphaneler kırmızı renkte olabilir. Sağ tarafta Maven yazan sekmede Reload oklarina tiklayip beklediğimiz de hata gitmiş oluyor

8) Kullanicinin gordugu arayuzde test ederiz. (UI)

Kullanicagimiz paketleri uygun isimlerde test-Java bolumun içerisinde oluşturuyoruz

9) Java ya sag tiklariz new package – com (paketin ismi) yazariz

10) com package'ina sag tiklariz – new package – abc (paketin ismi) yazariz

11) artik projemiz com. abc

Bu package'in altina frameworkumuzun package'larini yolosturuyoruz. Bunlar

A-pages

B-smokeTest

C-tests

D-utilities

Ornek Proje Olusturma

12) Resources paketi olusturma:

Java'ya sag tikliyoruz-new-package -->resources (yeni package)

Bu resources paketinin altina dokumanlarimizi copy-paste ederiz

13) configuration.properties dosyasi olusturma

En yukarda Projemize sag tikliyoruz new- File ' a tikliyoruz

Ismi önemli değil ama uzantisi MUTLAKA .properties olmalı

İsmi configuration.properties yaziyoruz

Bu dosyanin içine Data'larimizi key=value seklinde yaziyoruz

kr_url= <https://www.qualitydemy.com/>

kr_valid_username=user_1106147@login.com

kr_valid_password=31488081

14) ConfigReader Class'l olusturma

utilities package inin altında ConfigReader Classi oluşturuyoruz. Bu class configuration.properties deki dosyalarimizi okumak için bir aracı

Ornek Proje Olusturma

15) ConfigReader Classinda :

1-ilk yapacagimiz sey Instance olarak Properties objesi olusturmak. Bu objeyi static blok icinde kullanacagimdan static yapmam gerek

Bu objeyi sadece bu class ta kullanacagim icin private yapmamiz önerilir

2-Properties objesini kullanmak üzere bir static blok kurmalıyız. neden static? Cunku her zaman ilk static block çalışır

```
public class ConfigReader {  
  
    static Properties properties;  
  
    static {  
        String dosyaYolu="configuration.properties";  
  
        try {  
            FileInputStream fis = new FileInputStream(dosyaYolu);  
            properties=new Properties();  
            properties.load(fis);  
  
        } catch (IOException e) {  
            throw new RuntimeException(e);  
        }  
    }  
  
    public static String getProperty(String key){  
  
        return properties.getProperty(key);  
    }  
}
```

Ornek Proje Olusturma

16) Driver class'ini düzenleniyoruz

Singleton class : object olusturulmasi kontrol altina alınan (genelde izin verilmeyen) classdir. Bunun icin baska classlarda Driver clasindan obje uretmemizi saglayan default constructor'i gorunur sekilde yapip access modifier'i private yapariz

Bu class'da test class'larimizda kullanacagimiz driver'i olusturacak ve kapatacak getDriver() ve closeDriver() methodlarini olusturuyoruz

Bu methodlari static yaparak obje olusturmadan Class adi ile cagirmak icin kullanisli hale getiriyoruz

```
public class Driver {  
    private Driver(){  
    }  
    static WebDriver driver;  
    public static WebDriver getDriver(){  
        if(driver==null) { // method ilk cagrildiginda driver degeri atanma  
            // sonraki calistirmalarda degeri atanmis oldugu  
            String browser= ConfigReader.getProperty("browser");  
            switch (browser){  
                case "chrome" :  
                    WebDriverManager.chromedriver().setup();  
                    driver = new ChromeDriver();  
                    break;  
                case "firefox":  
                    WebDriverManager.firefoxdriver().setup();  
                    driver=new FirefoxDriver();  
                    break;  
                case "safari" :  
                    WebDriverManager.safaridriver().setup();  
                    driver=new SafariDriver();  
                    break;  
                default:  
                    WebDriverManager.chromedriver().setup();  
                    driver = new ChromeDriver();  
            } }  
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(15));  
        driver.manage().window().maximize();  
        return driver;  
    }
```

Ornek Proje Olusturma

18) pages package inin altinda kullanacagimiz her websayfasi icin bir page Class'i olustururuz

- a) Bu class'da ilk yapmamiz gereken test class'larinda bu class'dan obje uretebilmemiz icin gerekli olan Constructor'i olusturmaktir.

```
public class MyCoursedemyPage {  
    public MyCoursedemyPage(){  
  
        PageFactory.initElements(Driver.getDriver(), page: this);  
    }  
}
```

- b) Ardinda Locate islemlerimizin tamamini yaziyoruz bu sayfaya

```
@FindBy(xpath = "//a[text()='Log in']")  
public WebElement loginLinki;  
  
@FindBy(xpath = "//input[@id='login-email']")  
public WebElement emailKutusu;  
  
@FindBy(xpath = "//input[@id='login-password']")  
public WebElement passwordKutusu;  
  
@FindBy(xpath = "//button[text()='Login']")  
public WebElement loginButonu;  
  
@FindBy(linkText = "My courses")  
public WebElement coursesLinki;
```

TestNG Genel Tekrar Soru Cozumu

Soru 1 :

- Amazon anasayfaya gidebilecek sekilde bir page sayfasi olusturun : AmazonPage
- Amazon ana sayfasinda en alta bulunan Webtable'i inceleyebilmek icin AmazonPage clasinda en alta gitme isini yapacak bir method olusturun
- Tests paketi altinda yeni bir class olusturun: D26_AmazonSatirSutunSayisi
- Bu class'in altinda bir test method olusturun : satirSayisi() ve webtable'da 10 satir oldugunu test edin
- Yeni bir method olusturun : sutunSayisi() ve yazi olan sutun sayisinin 7oldugunu test edin

TestNG Genel Tekrar Soru Cozumu

Soru 2 :

- AmazonPage sayfasinda istedigim satir ve sutun sayisi ile cagirdigimda bana hucredeki yaziyi getirecek bir method olusturun
- Tests paketi altinda yeni bir class olusturun: D26_AmazonHucreTesti
- Bu class'in altinda bir test method olusturun : hucretesti() ve webtable'da 3. satir 2.sutundaki yazinin "Home Services" yazisi icerdigini test edin
- Yeni bir method olusturun : AmazonYazisi() ve tabloda 9 Hucrede "Amazon" yazisi bulundugunu test edin

TestNG Genel Tekrar Soru Cozumu

Soru 3 :

- Amazon uzerine yapılan 4 testi otomatik olarak calistiracak xml kodunu yazin ve calistirin

- D26_AmazonSatirSutunSayisi class'indan satirSayisi() testini ve D26_AmazonHucreTesti class'indan hucretesti() testini calistiracak xml kodunu yazin ve calistirin



Soru 4 :

- D26_AmazonSatirSutunSayisi class'indan satirSayisi() testini ve D26_AmazonHucreTesti class'indan hucretesti() testini rapor alacak sekilde hazirlayin ve 3.sorudaki xml dosyasi ile calistirip raporu olusturun



Wise Quarter
first class IT courses

Selenium Ders-14 Cucumber



The future at your fingertips

+1 912 888 1630

www.wisequarter.com

/wisequarter

Cucumber BDD (behaviour driven development / Davranış tabanlı geliştirme) yaklaşımı için kullanılmakta olan açık kaynak kodlu bir kütüphanedir.

Cucumber bir iş ararken önemli bir rol alacaktır.

Su ana kadar JUnit ve TestNG ile HTML elementleri otomasyon ile nasıl kullanabilecegimizi gorduk, Cucumber derslerinde framework'e odaklanacagiz.

```
Feature: US1006 Dogru kullanici adi ve password ile pozitif login testi

Scenario: TC12 Kullanici mycoursedemy sitesine giris yapabilmeli

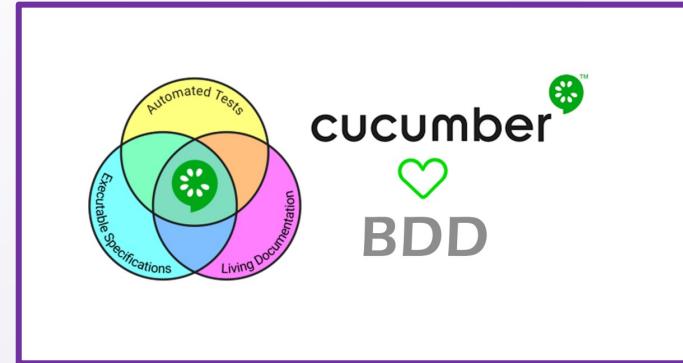
    Given Kullanici "myUrl" anasayfaya gider
    Then myCourse anasayfa login linkine tiklar
    And myCourse kullanici adi olarak "myGecerliEmail" girer
    And myCourse password olarak "myGecerliPassword" girer
    And myCourse login butonuna basar
    Then myCourse giris yapabildigini test eder
    And Sayfayı kapatır
```

Agile methodolojisinde, insanlar uygulamanın işlevsellliğini geliştirmek için birlikte çalışmak zorundadır. Cucumber development team'deki herkesin test case'leri anlayabilmesini saglar.

BDD (behaviour driven development) (Davranış güdümlü geliştirme)- ilk olarak behavior(davranis) veya functionalitileri yazıyorsunuz (Epic=Feature, Story, AC, etc), daha sonra development and testing baslıyor.

BDD'de behaviour'lar başarısız olduğunda kod başarısız olur..

Anlasılabilir Gherkin Language nedeniyle BDD development için Cucumber harika bir uygulamadır.



Gherkin: Projede her bir davranış için .feature uzantılı bir Gherkin dosyası oluşturulur. Bu feature dosyasına ilgili özelliğin farklı durumlardaki davranışları tanımlanır.

Given anahtar kelimesi ile ön koşul yani başlangıç durumu tanımlanır,

When, And anahtar kelimeleri ile olayı

Then anahtar kelimesi ile de sonuç tanımlanır.

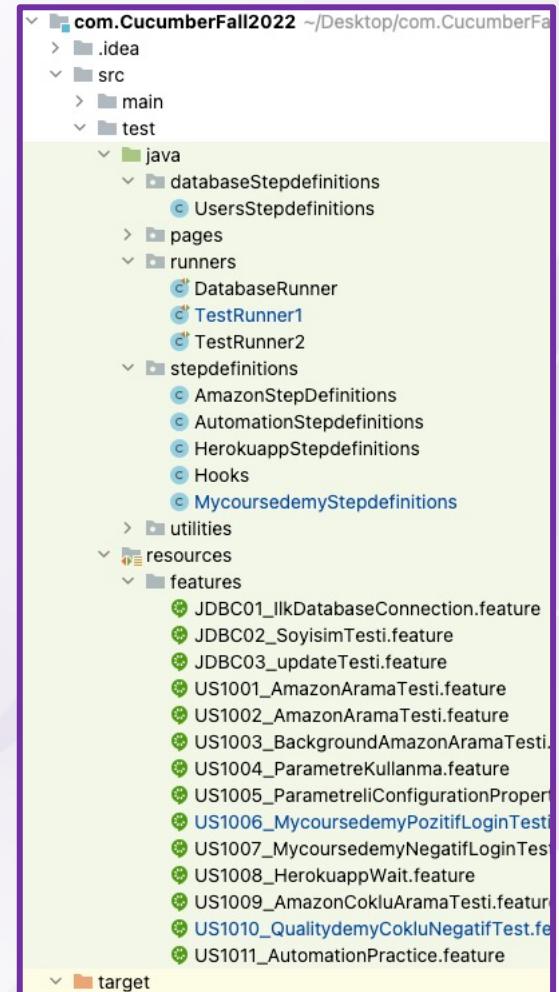
Given-When-And-Then adımları sadece okuyanlar için farkeder Java için hepsi birdir.

Cucumber (TDD)(Test Driven Development) test odaklı geliştirmeye izin verir, çünkü Cucumber ile Junit veya TestNG kullanabiliriz.

Cucumber iş için önemlidir, çünkü anlaşılabilir ve harika raporlara sahiptir.

Cucumber, teknik olmayan (none-technical) kişileri ve teknik kişileri birbirine bağlar.

Developer veya team lead gibi teknik elemanlar da bazen testerların yaptığını anlayamayabilir. Gherkin onların da testlerimizi anlamalarını kolaylaştırır





Proje Olusturma

1. Create Project: File => New => Project => Select maven => click next

2. Name: team105Cucumber => finish

3. Add Dependencies =>

Selenium-java,

webdrivermanager,

cucumber java,

cucumber junit

4. Click Maven => click “Enable auto-reload after any changes” (Reload)

Proje Olusturma

5. Java'ya sag click yapip asagidaki paketleri olusturalim

- a. utilities
- b. pages
- c. runners (test case'leri calistirmak ve control etmek icin kullanacagiz)
- d. stepdefinitions (kodlarimizi burada olusturacagiz)

6- Utilities paketi altinda Driver ve ConfigReader Class'larini olusturalim

7- Projeye sag click yapip configuration.properties dosyasi olusturalim

8- test paketi altinda yeni bir klasor olusturalim : resources

9- resources klasoru altinda yeni bir klasor olusturalim : features (Java kodu icermeyen dosyalari buraya koyacagiz)

10- features'a sag clik yapip dosya olusturalim amazonsearch.feature

11- cucumber for Java plugin'i intelliJ'e ekleyelim (settings/Plugins)

MAC => IntelliJ Idea->Preference->Plugins->Marketplace->Type Cucumber for Java->Install->Restart



İlk Cucumber Testi

Yeni bir feature file olusturalim : amazonsearch.feature

Given kullanici amazon sayfasina gider

And iPhone icin arama yapar

Then sonuclarin Iphone icerdigini test eder

Given kullanici amazon sayfasina gider

And tea pot icin arama yapar

Then sonuclarin tea pot icerdigini test eder

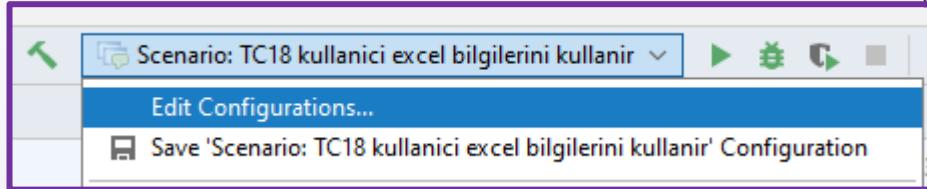
Given kullanici amazon sayfasina gider

And flower icin arama yapar

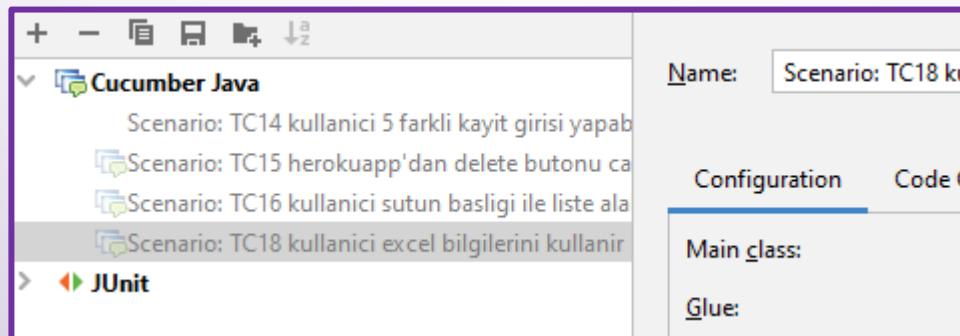
Then sonuclarin flower icerdigini test eder

Cucumber

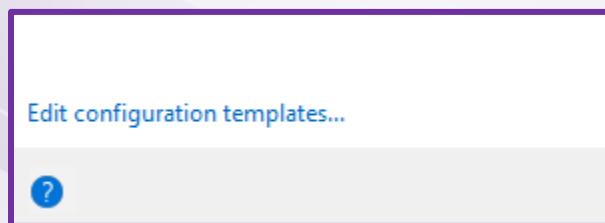
Build Ayarları



1- Edit Configurations secin



2- Sol bolumdeki CucumberJava ve altindakileri silin.



3- Alt kisimdan Edit configuration menusunu acin

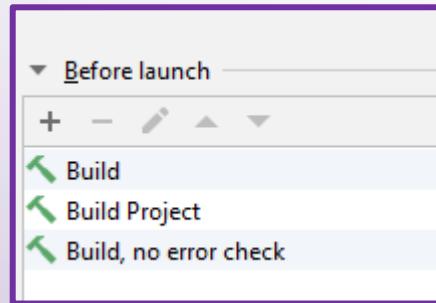


Cucumber

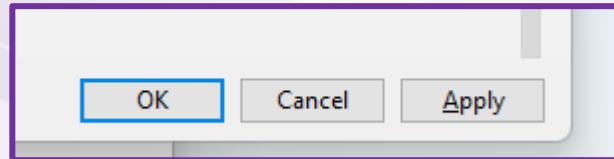
Build Ayarları



4- Listededen Cucumber Java'yı seçin



5- Altaki Before Launch bolumunde 3 maddenin oldugunu kontrol edin, yoksa ekleyin



6- degisiklikleri kayit etmek icin Apply'a basin

Cucumber

Runner Class

Cucumber'da Runner class'i bos bir class'tir. Runner class'ini farkli kilan ve TestNG'deki xml dosyalari gibi calismasini saglayan 2 adet notasyon mevcuttur

```
@RunWith(Cucumber.class)
@CucumberOptions(
    features = "src/test/resources/features" ,
    glue = "stepdefinitions" ,
    tags= "@wip",
    dryRun = false
)

public class Runner {
```

`@RunWith` notasyonu projemize cucumber junit dependency eklememizin sebebidir bu sayede runner class'larimiz cucumber ile calisir

`@CucumberOptions` notasyonu ile istedigimiz ozellikleri Runner class'ina ekleyebiliriz. Raporlama gibi ekstra option'lari da ileride ekleyecegiz

Ancak oncelikli gorevi features dosyalari ile stepdefinitions'da bulunan Java method'larini ilisklendirmektir

`dryRun` : iki deger alabilir

`true` : secilirse, verilen tag ile işaretli olan Feature veya Scenario'daki eksik stepdefinitions'lari bulup ilgili method'lari olusturur. Hic bir sekilde testimizi calistirmaz. Eksik adim yoksa test passed olarak işaretler

`false` : secilirse, verilen tag ile işaretlenen Feature veya Scenario'lari calistirir

`tags` : features classoru içerisinde yazilan tag(lari) aratip buldugu tum feature veya scenario'lari calistirir



Cucumber

Background

Feature: US1001 amazon arama

@amazon @nutella

Scenario: TC01 amazon nutella arama

```
When kullanıcı amazon sayfasına gider
And nutella için arama yapar
Then sonucun nutella icerdigini test eder
And sayfayı kapatır
```

@amazon @java

Scenario: TC02 amazon java arama

```
When kullanıcı amazon sayfasına gider
And java için arama yapar
Then sonucun java icerdigini test eder
And sayfayı kapatır
```

@amazon @ipad

Scenario: TC03 amazon ipad arama

```
When kullanıcı amazon sayfasına gider
And ipad için arama yapar
Then sonucun ipad icerdigini test eder
And sayfayı kapatır
```

Farklı senaryoların başında ortak adımlarımız varsa:

1. Feature file in basına Background oluşturun.
2. Bu ortak adımları Background altına yazın.

Feature: US1002 amazon background ile arama

Background: amazon sayfasına gitme

When kullanıcı amazon sayfasına gider

Scenario: TC04 amazon nutella arama

And nutella için arama yapar

Then sonucun nutella icerdigini test eder

And sayfayı kapatır

@wip

Scenario: TC05 amazon java arama

And java için arama yapar

Then sonucun java icerdigini test eder

And sayfayı kapatır

Scenario: TC06 amazon ipad arama

And ipad için arama yapar

Then sonucun ipad icerdigini test eder

And sayfayı kapatır

3. Background, Feature file'daki her Scenario'dan önce çalışır

4. Duplication olmadigindan emin olun.
Background un altındaki adımı yazdıktan sonra senaryolardan silin.



Cucumber

@tags

Tag'ları onceden belirledigimiz senaryoları (scenario) çalıştırılmak için kullanırız.

Tag'ları senaryolarımızı gruplandırmak için de kullanabiliriz (smoke test, regression test, vs.)

```
@RunWith(Cucumber.class)
@CucumberOptions (
    plugin={"html:target\\cucumber-reports.html",
            "json:target/json-reports/cucumber.json",
            "junit:target/xml-report/cucumber.xml"},
    features="src/test/resources/features",
    glue="stepdefinitions",
    tags="@toplu" ,

    dryRun= false           // dryRun=true dedigimizde test
```

Feature: US1001 amazon page search

@amazon @search @apple

Scenario: TC01 amazon arama testi

```
Given kullanıcı amazon sayfasına gider
And "apple" için arama yapar
Then sonuçların "apple" içerdigini test eder
Then sayfayı kapatır
```



Cucumber

Feature File'i Parametre ile Kullanma

```
Feature: US1000 Amazon search test
```

```
Scenario: TC01 iphone aramasi testi
Given kullanici amazon sayfasina gider
And iphone icin arama yapar
Then sonuclarin iphone icerdigini test eder
```

```
Scenario: TC02 tea pot aramasi testi
Given kullanici amazon sayfasina gider
And tea pot icin arama yapar
Then sonuclarin tea pot icerdigini test eder
```

```
Scenario: TC03 flower aramasi testi
Given kullanici amazon sayfasina gider
And flower icin arama yapar
Then sonuclarin flower icerdigini test eder
Then sayfayı kapatır
```

Kodlarımızı parametreli ve dinamik hale getirmek için feature file da degisen olarak kullanacagımız kelimeyi çift tırnak " " icine alırız.

```
@rapor2
```

```
Scenario: TC07 istenen kelimenin oldugunu test etme
Given kullanici "amazonUrl" sayfasina gider
And "iphone" icin arama yapar
Then sonucun "iphone" icerdigini test eder
And sayfayı kapatır
```

Kodlarımızı parametreli olarak yazdıktan sonra sadece " " içindeki değeri değiştirerek test datalarını feature file dan kontrol edebiliriz.

Kodlarımızı parametreli olarak yazmak framework'u daha dinamik hale getirir(kodumuz artık hard coded degildir diyebiliriz).



Cucumber

Parametre ile Pozitif login testi

Feature: US1006 Kullanici configuration dosyasındaki bilgilerle login olabilmeli

Scenario: TC09 Gecerli kullanici adi ve sifre ile Pozitif Login Testi

```
Given kullanici "qdUrl" anasayfaya gider
Then ilk sayfa login linkine click yapar
And kullanici kutusuna "qdGecerliUsername" yazar
And password kutusuna "qdGecerliPassword" yazar
Then login butonuna basar
And basarili giris yapildigini test eder
And 3 saniye bekler
Then sayfayı kapatır
```



Cucumber

Parametre ile Negatif login testi

Feature: Kullanici yanlis bilgilerle giris yapamaz

Scenario: TC10 Gecerli kullanici adi ve gecersiz sifre ile negatif Login Testi

```
Given kullanici "qdUrl" anasayfaya gider
Then ilk sayfa login linkine click yapar
And kullanici kutusuna "qdGecerliUsername" yazar
And password kutusuna "qdGecersizPassword" yazar
Then login butonuna basar
And giris yapislamadigini test eder
And 3 saniye bekler
Then sayfayı kapatır
```

Scenario: TC11 Gecersiz kullanici adi ve gecerli sifre ile negatif Login Testi

```
Given kullanici "qdUrl" anasayfaya gider
Then ilk sayfa login linkine click yapar
And kullanici kutusuna "qdGecersizUsername" yazar
And password kutusuna "qdGecerliPassword" yazar
Then login butonuna basar
And giris yapislamadigini test eder
And 3 saniye bekler
Then sayfayı kapatır
```



Wise Quarter
first class IT courses

Selenium Ders-15 Cucumber



The future at your fingertips

+1 912 888 1630

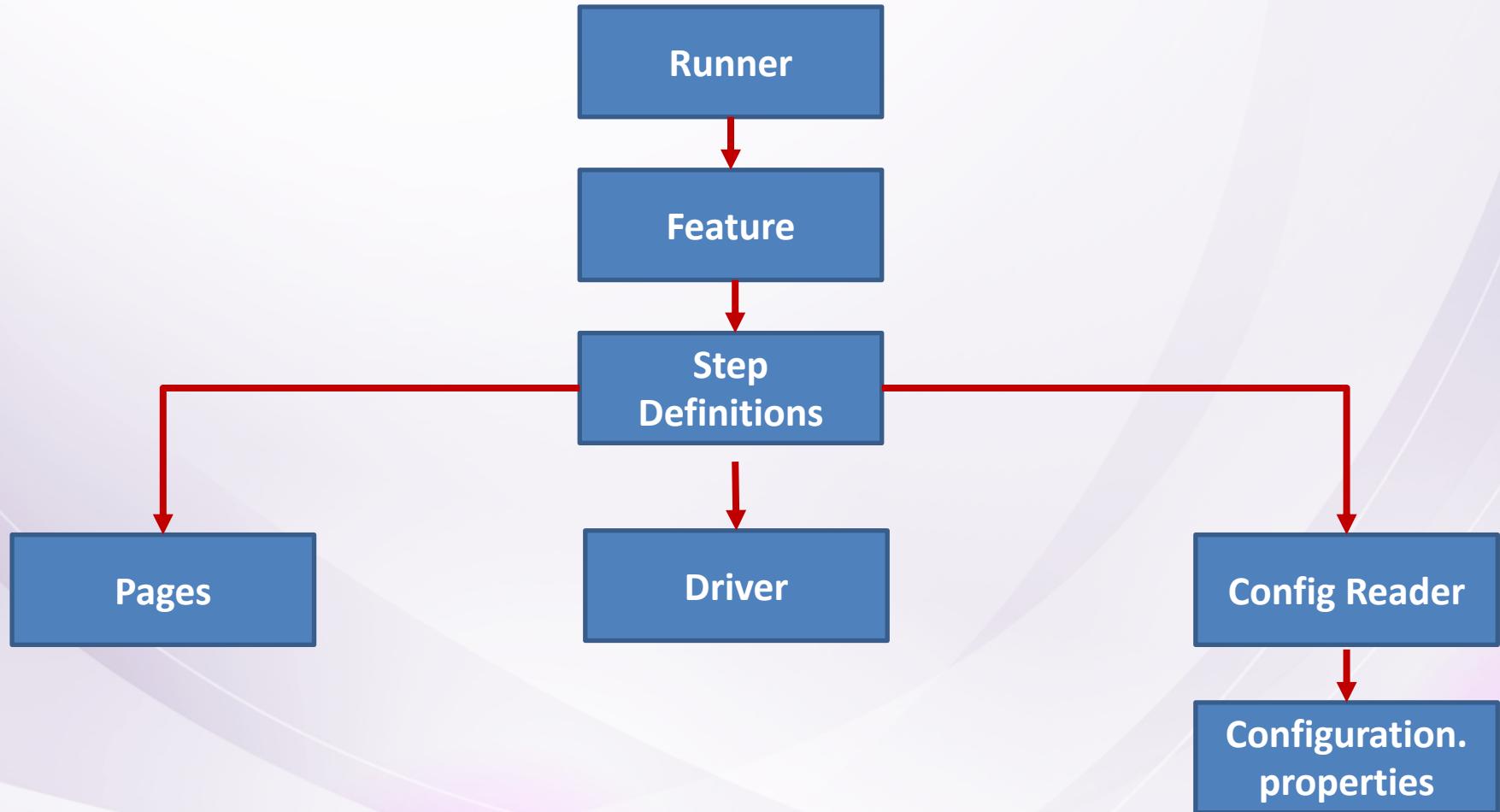
www.wisequarter.com

/wisequarter



Cucumber

Calisma Semasi



Cucumber

Scenario Outline

Scenario Outline: ayni teste birden fazla datayi kullanmamizi saglar

Bir liste kullanmak istedigimiz degeri "`<value>`" seklinde yazariz

Daha sonra testin sonuna Examples: yazip ilk satir olarak `| value |` yazariz ve altina kullanmak istedigimiz degerleri ekleriz.
(`| elma |`, `| armut |` ... gibi)

```
@amazon @search @apple
Scenario Outline: TC01 amazon arama testi

Given kullanici amazon sayfasina gider
And "<kelime>" icin arama yapar
Then sonuclarin "<kelime>" icerdigini test eder
Then sayfayı kapatır
```

Examples:
`|kelime|`
`|teapot|`
`|flower|`
`|avsfdfhvbj|`

Class Work : US1001 de kullandigimiz feature dosyasi altinda yeni bir Scenario olusturalim TCO4 ve orada yaptigimiz aramayı Scenario Outline kullanarak farkli urunler icin yapalim

Yeni bir feature file olusturalim: US1006_Dinamik_url_test.feature

Yeni bir Scenario olusturalim: TC08_yazilan_her_url'e_gitmeli

Scenario Outline: TC01 amazon arama testi

```
Given kullanici "<sayfa_url>" sayfasina gider
And url'in "<sayfa_url>" oldugunu test eder
Then sayfayı kapatır
```

Examples:

```
|sayfa_url|
|amazon_url|
|bestbuy_url|
|ebay_url|
```

Configuration.properties dosyasında tanimlanmis tum url'lerden key olarak yazdigimda ilgili sayfaya gidecek sekilde bir stepdefinition olusturun.

Bu stepdefinition'i amazon_url, bestbuy_url ve ebay_url ile test edin

Cucumber

Scenario Outline

Qualitydemy Login negative test case'i asagidaki 5 kullanici ismi ve sifresi icin calisacak sekilde duzenleyin

Kullanici adi	Password
Manager5	Manager5!
Manager6	Manager6!
Manager7	Manager7!
Manager8	Manager8!
Manager9	Manager9!

```
Given kullanici "qdUrl" anasayfaya gider
Then ilk sayfa login linkine click yapar
And kullanici kutusuna "kullanici adi" yazar
And password kutusuna "password" yazar
Then login butonuna basar
And giris yapilamadigini test eder
And 3 saniye bekler
Then sayfayı kapatır
```

Cucumber

Scenario Outline

Yeni bir feature file olusturun: `US1007_kullanici_data_ekleyebilmeli`

DataTableStepDefinition dosyasi ve gerekli step definition'lari olusturun ve 5 farkli kayit ekleyin

When kullanici <https://editor.datatables.net/> adresine gider

Then new butonuna basar

And tum bilgileri girer

And Create tusuna basar

When kullanici ilk isim ile arama yapar

Then isim bolumunde isminin oldugunu doğrular

Cucumber

Class Work

Yeni bir test methodu olusturalim

https://the-internet.herokuapp.com/add_remove_elements/ adresine gidin

- 1) "Add Element" butona basin
- 2) "Delete" butonu gorunur oluncaya kadar bekleyin
- 3) "Delete" butonunun gorunur oldugunu test edin
- 4) Delete butonuna basarak butonu silin
- 5) Delete butonunun gorunmedigini test edin

Cucumber

Class Work

- 1."<http://webdriveruniversity.com/>" adresine gidin
- 2."Login Portal" a kadar asagi inin
- 3."Login Portal" a tiklayın
- 4.Diger window'a gecin
- 5."username" ve "password" kutularina deger yazdirin
- 6."login" butonuna basin
- 7.Popup'ta cikan yazinin "validation failed" oldugunu test edin
- 8.Ok diyerek Popup'i kapatın
- 9.Ilk sayfaya geri donun
- 10.Ilk sayfaya donuldugunu test edin

1. <http://demo.guru99.com/test/web-table-element.php> sayfasina gidelim
2. Cucumber framework'de US1012_Guru_Web_Tables altinda

Scenario : TC_18_kullanici_sutun_basligi_ile_liste_alabilmeli olusturun ve asagidaki testi yapin

Given user web sayfasinda

And “Istelenen Baslik”, sutunundaki tum degerleri yazdirir

Cucumber

HTML Reports

Cucumber raporları, şirketlerin Cucumber kullanmasının ana nedenlerinden biridir.

```
@RunWith(Cucumber.class)
@CucumberOptions(
    plugin={"html:target/cucumber_rapor.html"},
    features = "src/test/resources/features",
    glue="stepdefinitions",
    tags="@amazon",
    dryRun = false
)
```

Html rapor almak için runner classına eklenti(plugin) eklememiz yeterlidir.

plugin={"html:target\\cucumber-reports.html"}



Wise Quarter
first class IT courses

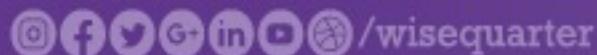
Selenium Ders-16 Cucumber Cucumber Reports Parallel Testing



The future at your fingertips

+1 912 888 1630

www.wisequarter.com



Cucumber

Hooks ve Screenshot Ekleme

Cucumber hooks, senaryolardan önce veya sonra çalışan kod bloklarına sahip olan bir classtır. (Daha once kullandigimiz TestBase gibi)

@Before ve @After annotation'ları kullanılarak kodları projemizde ve step definitionlarda kullanabiliriz.

Cucumber hooks, kod çalışma akışını daha iyi yönetmemizi kolaylaştırır ve kod fazlalığını azaltmamıza yardımcı olur.



@After

```
public void tearDown(Scenario scenario){  
    final byte[] screenshot=((TakesScreenshot)  
Driver.getDriver()).getScreenshotAs(OutputType.BYTES);  
    if (scenario.isFailed()) {  
        scenario.attach(screenshot, "image/png","screenshots");  
    }  
    Driver.closeDriver();  
}
```

Cucumber

Yeni Raporlar Ekleme

Plugin ekleyerek yeni raporlar da olusturabiliriz

Tester'lar icin onemli olan rapor Html olsa da json ve xml formatinda da rapor almak mumkundur.

Ayrica maven-cucumber-reporting plugin yuklemek istersek pom.xml'e plugin ekleyebiliriz

```
@RunWith(Cucumber.class)
@CucumberOptions(
    plugin={"html:target/cucumber_rapor.html"},
    features = "src/test/resources/features",
    glue="stepdefinitions",
    tags="@amazon",
    dryRun = false
)
```

Cucumber

Paralel Testing

Paralel testing: Birden fazla browser'in es zamanlı çalıştırılmasıdır.

Cucumber ile parallel test çalıştırmak testing'ye göre daha zordur.

Ancak raporlama ihtiyacı varsa TestNG'deki karmaşık raporlama prosesleri yerine cucumber'da parallel çalıştırmanın zorluguna katlanmak tercih edilebilir.

Paralel çalıştırılmak için birden fazla Runner Class'ına ihtiyacımız var

Cucumber'da, testleri paralel olarak çalıştırılabilmek için bazı eklentilere(plugin) ve yapılandırmalara da ihtiyacımız vardır.

Pom da yaptığımız ayarlamalardan sonra testleri Runner'dan değil Terminalden “mvn clean verify” kodunu yazarak çalıştıracağız

Cucumber

Paralel Testing

1. Birden fazla runner classı ekleyin. Aynı anda calistirmak istediginiz kadar Runner Class'ına sahip olmalisiniz. Class isimleri belirledigimiz ortak bir String icermelidir. Ornegin :TestRunner.
 - a. SmokeTestRunner
 - b. FirstTestRunner
 - c. SecondTestRunner
2. maven-failsafe-plugin eklentisi ekleyin.(Belirli testler başarısız olduktan sonra testleri çalışmaya devam için.)

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-failsafe-plugin</artifactId>
    <version>3.0.0-M1</version>
    <configuration>
        <testFailureIgnore>true</testFailureIgnore>
        <skipTests>false</skipTests>
        <includes>
            <include>**/runners/*TestRunner*.java</include>
        </includes>
    </configuration>
</plugin>
```



Cucumber

Paralel Testing

3. maven-surefire-plugin ekleyin ve yapılandırın. Paralel test için gerekli eklentidir

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <version>3.0.0-M1</version>
  <configuration>
    <parallel>classes</parallel>
    <forkMode>perthread</forkMode>
    <threadCount>4</threadCount>
    <reuseForks>false</reuseForks>
    <argLine>-Duser.language=en</argLine>
    <argLine>-Xmx1024m</argLine>
    <argLine>-XX:MaxPermSize=256m</argLine>
    <argLine>-Dfile.encoding=UTF-8</argLine>
    <useFile>false</useFile>
    <includes>
      <include>**/runners/*TestRunner*.java</include>
    </includes>
    <testFailureIgnore>true</testFailureIgnore>
  </configuration>
</plugin>
```



Cucumber

Paralel Testing

JDK sorunu yasayanlar icin opsiyonel plugin

```
<plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.1</version>
    <configuration>
        <source>1.7</source>
        <target>1.7</target>
        <fork>true</fork>
        <executable>C:\Program
Files\Java\jdk1.8.0_251\bin\javac</executable>
    </configuration>
</plugin>
```



4. maven-cucumber-reporting plugin ekle.
Gelismis rapor icin
gereklidir

Cucumber

Paralel Testing

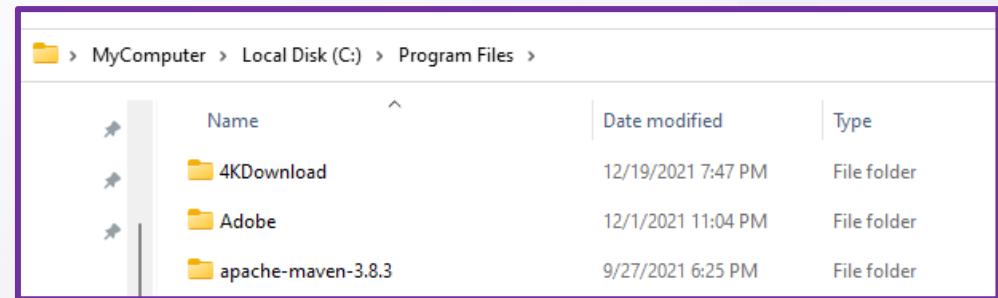
```
<plugin>
  <groupId>net.masterthought</groupId>
  <artifactId>maven-cucumber-reporting</artifactId>
  <version>5.0.0</version>
  <executions>
    <execution>
      <id>execution</id>
      <phase>verify</phase>
      <goals>
        <goal>generate</goal>
      </goals>
      <configuration>
        <projectName>cucumber-jvm-example</projectName>
        <outputDirectory>${project.build.directory}</outputDirectory>
        <inputDirectory>${project.build.directory}</inputDirectory>
        <jsonFiles>
          <param>**/json-reports/*.json</param>
        </jsonFiles><classificationFiles>->
        <param>sample.properties</param>
        <param>other.properties</param>
      </classificationFiles>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Cucumber

Maven Environment Ayarı

Bilgisayarına maven indirmeyenler terminal'den mvn clean verify ile calistiramazlar.

1- Bilgisayarınızda program Files clasorunde apache-maven olup olmadığını kontrol edin



2- maven.apache.org sitesinden bilgisayarınıza Binary zip archive dosyalarını indirin



Link	Checksums	Signature	
Binary tar.gz archive	apache-maven-3.8.6-bin.tar.gz	apache-maven-3.8.6-bin.tar.gz.sha512	apache-maven-3.8.6-bin.tar.gz.asc
Binary zip archive	apache-maven-3.8.6-bin.zip	apache-maven-3.8.6-bin.zip.sha512	apache-maven-3.8.6-bin.zip.asc
Source tar.gz archive	apache-maven-3.8.6-src.tar.gz	apache-maven-3.8.6-src.tar.gz.sha512	apache-maven-3.8.6-src.tar.gz.asc
Source zip archive	apache-maven-3.8.6-src.zip	apache-maven-3.8.6-src.zip.sha512	apache-maven-3.8.6-src.zip.asc

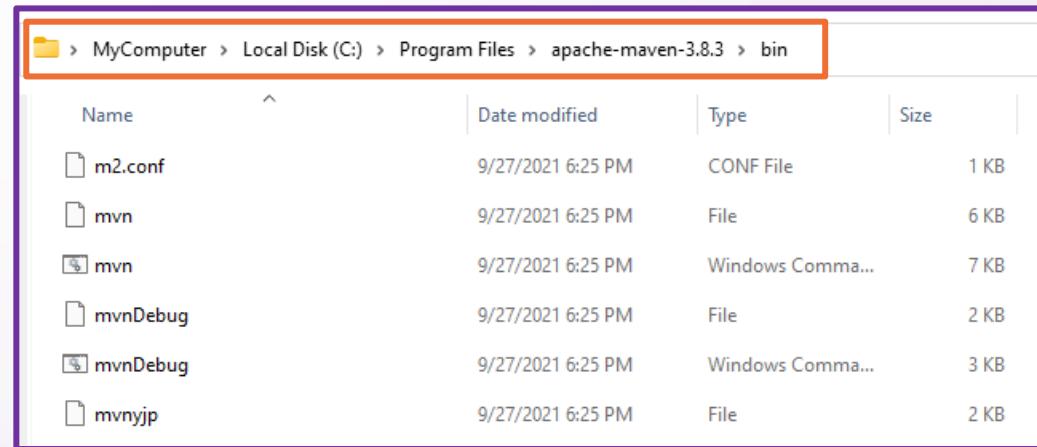
Cucumber

Maven Environment Ayarı

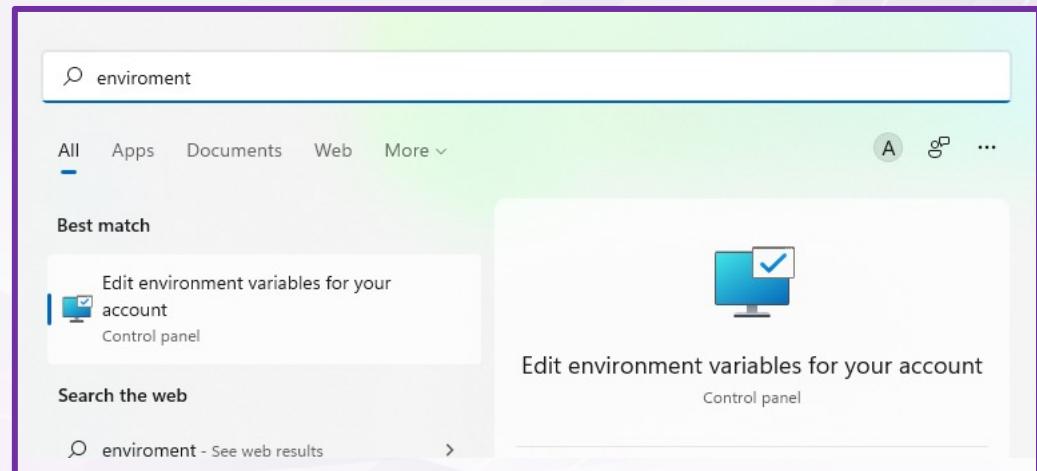
3- zip seklinde gelen dosyayı bir klasöre çıkartıp, bilgisayarınızda Program Files klasörüne kopyalayın

4- Maven klasörünün içerisinde bin klasörüne girip klasörün yolunu kopyalayın

5- Bilgisayarınızın arama kısmında environment yazıp Edit environment variables for your account menusune girin



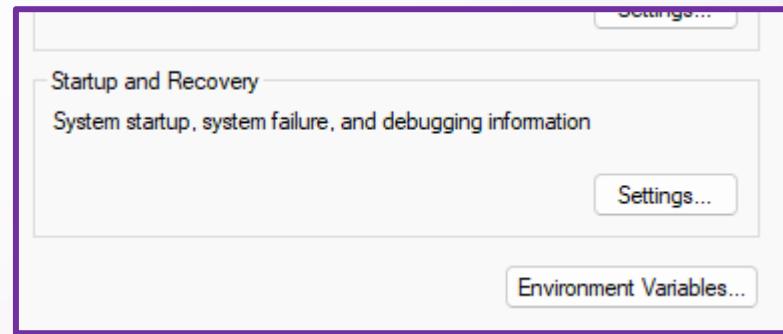
Name	Date modified	Type	Size
m2.conf	9/27/2021 6:25 PM	CONF File	1 KB
mvn	9/27/2021 6:25 PM	File	6 KB
mvn	9/27/2021 6:25 PM	Windows Comma...	7 KB
mvnDebug	9/27/2021 6:25 PM	File	2 KB
mvnDebug	9/27/2021 6:25 PM	Windows Comma...	3 KB
mvnyjp	9/27/2021 6:25 PM	File	2 KB



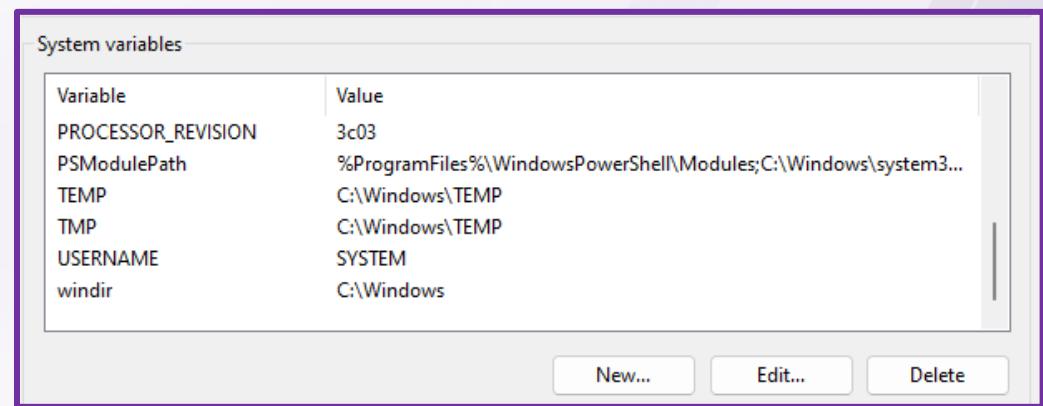
Cucumber

Maven Environment Ayarı

6- Acilan menuden Environment Variables'i tiklayin



7- Sistem variables bolumunde New butonuna basin ve kopyaladiginiz dosya yolunu buraya ekleyin.

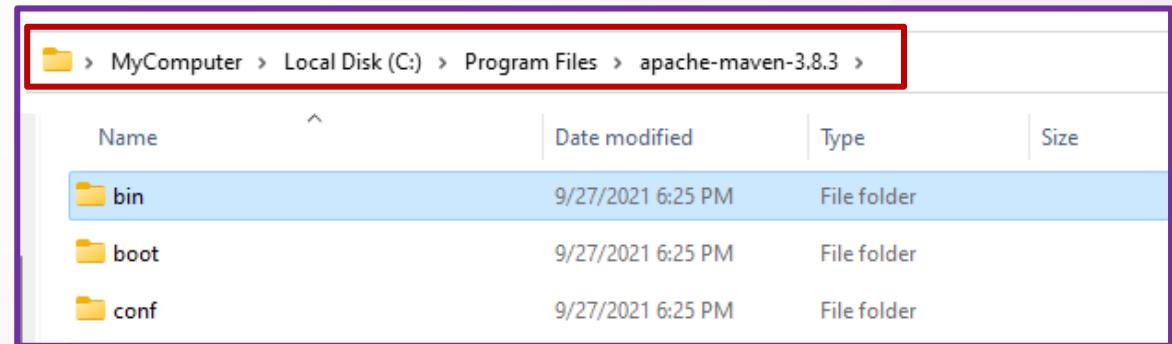


8- Ok diyerek menuleri kapatın

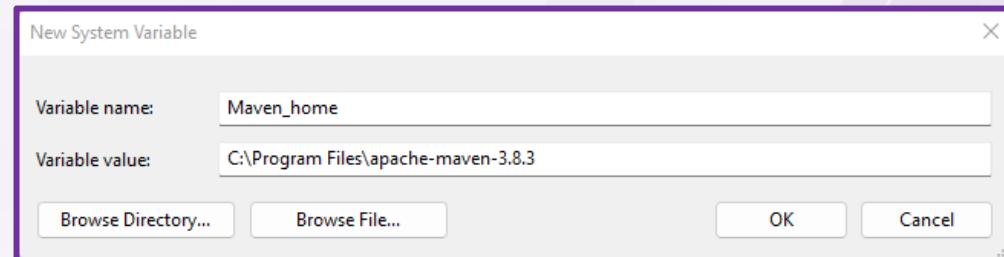
Cucumber

Maven Environment Ayari

9- Program Files klasorunde maven klasorune girip dosya yolunu kopyalayin



10- Yeniden Edit envoirement variables for your account menusune girip Environment Variables'i acin New butonuna basip kopyaladiginiz maven klasorunun yolunu yapistirin



11- Ok diyerek menuleri kapatın

12- Ayarlari yaptiktan sonra IntelliJ'yi kapatip acin



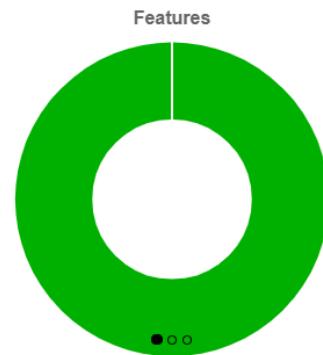
Cucumber

Paralel Testing

Project	Number	Date
cucumber-jvm-example	1	01 Jan 2021, 15:21

Features Statistics

The following graphs show passing and failing statistics for features



Feature	Steps						Scenarios			Features	
	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
US1001 amazon page search	10	0	0	0	0	10	1	0	1	30.902	Passed
US1002_amazon_search_background	12	0	0	0	0	12	3	0	3	41.928	Passed
US1004_amazon_search_scenario_outline	16	0	0	0	0	16	4	0	4	53.219	Passed
	7	0	0	0	0	7	1	0	1	28.033	Passed
US1009 Ck Hotels Log	7	0	0	0	0	7	1	0	1	15.471	Passed
	52	0	0	0	0	52	10	0	10	2:49.553	5
	100.00%	0.00%	0.00%	0.00%	0.00%		100.00%	0.00%			100.00%



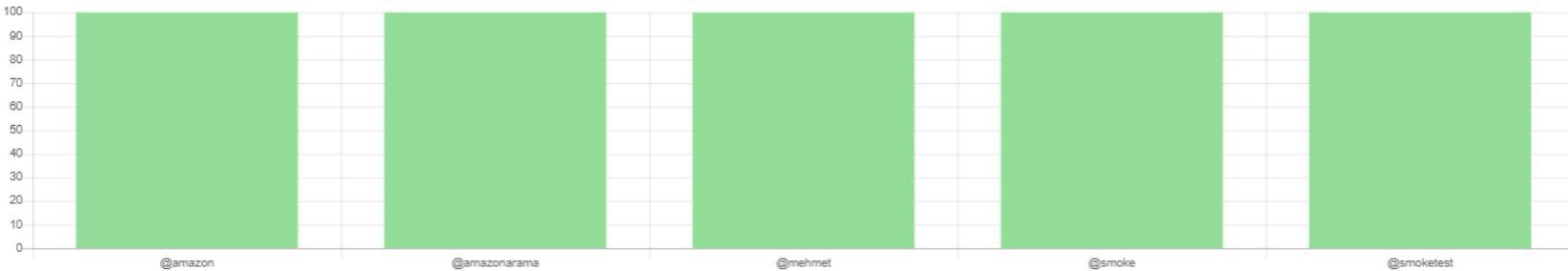
Cucumber

Paralel Testing

Project	Number	Date
cucumber-jvm-example	1	01 Jan 2021, 15:21

Tags Statistics

The following graph shows passing and failing statistics for tags



Tag	Steps						Scenarios			Features	
	Passed	Failed	Skipped	Pending	Undefined	Total	Passed	Failed	Total	Duration	Status
@amazon	35	0	0	0	0	35	8	0	8	1:35.482	Passed
@amazonarama	3	0	0	0	0	3	1	0	1	3.187	Passed
@mehmet	3	0	0	0	0	3	1	0	1	3.865	Passed
@smoke	14	0	0	0	0	14	2	0	2	43.504	Passed
@smoketest	6	0	0	0	0	6	2	0	2	7.052	Passed
	61	0	0	0	0	61	14	0	14	2:33.090	5
	100.00%	0.00%	0.00%	0.00%	0.00%		100.00%	0.00%			100.00%

Cucumber

Paralel Testing

Project	Number	Date
cucumber-jvm-example	1	01 Jan 2021, 15:21

Steps Statistics

The following graph shows step statistics for this build. Below list is based on results. step does not provide information about result then is not listed below. Additionally @Before and @After are not counted because they are part of the scenarios, not steps.

Implementation	Occurrences	Average duration	Max duration	Total durations	Ratio
stepdefinitions.AmazonStepDefinitions.flower_icin_arama_yapar()	2	3.892	4.117	7.784	100.00%
stepdefinitions.AmazonStepDefinitions.icinArاماYapar(java.lang.String)	4	3.460	4.144	13.842	100.00%
stepdefinitions.AmazonStepDefinitions.iphone_icin_arama_yapar()	2	3.216	3.448	6.432	100.00%
stepdefinitions.AmazonStepDefinitions.kullaniciSayfayiKapatir()	10	0.088	0.119	0.884	100.00%
stepdefinitions.AmazonStepDefinitions.kullanici_amazon_anasayfaya_gider()	10	8.741	13.112	1:27.416	100.00%
stepdefinitions.AmazonStepDefinitions.sonucunIcerdiginiTestEder(java.lang.String)	4	0.118	0.177	0.474	100.00%
stepdefinitions.AmazonStepDefinitions.sonucun_flower_icerdigini_test_eder()	2	0.165	0.206	0.331	100.00%
stepdefinitions.AmazonStepDefinitions.sonucun_iphone_icerdigini_test_eder()	2	0.122	0.142	0.244	100.00%
stepdefinitions.AmazonStepDefinitions.sonucun_tea_pot_icerdigini_test_eder()	2	0.112	0.152	0.225	100.00%
stepdefinitions.AmazonStepDefinitions.tea_pot_icin_arama_yapar()	2	4.324	4.504	8.649	100.00%
stepdefinitions.BestbuyStepDefinitions.kullanici_anasayfaya_gider(java.lang.String)	2	13.412	14.311	26.825	100.00%
stepdefinitions.CKHotelsStepDefinitions.gecerli_password_girer()	1	0.188	0.188	0.188	100.00%
stepdefinitions.CKHotelsStepDefinitions.gecerli_username_girer()	1	0.230	0.230	0.230	100.00%
stepdefinitions.CKHotelsStepDefinitions.gecersizPasswordGirer()	1	0.190	0.190	0.190	100.00%
stepdefinitions.CKHotelsStepDefinitions.gecersizUsernameGirer()	1	0.221	0.221	0.221	100.00%
stepdefinitions.CKHotelsStepDefinitions.log_in_yazisina_tiklar()	2	1.145	1.200	2.291	100.00%
stepdefinitions.CKHotelsStepDefinitions.login_butonuna_basar()	2	6.593	12.063	13.186	100.00%
stepdefinitions.CKHotelsStepDefinitions.sayfayaGirisYapilamadiginikontrolEder()	1	0.104	0.104	0.104	100.00%
stepdefinitions.CKHotelsStepDefinitions.sayfaya_giris_yaptigini_kontrol_eder()	1	0.037	0.037	0.037	100.00%
19		52	3.260	1:27.416	2:49.553
					Totals

Cucumber

Genel Tekrar

1. <https://automationexercise.com/> sayfasina gidelim
2. Cucumber ile asagidaki testi yapalim

Given user web sayfasinda

And user sign up linkine tiklar

And user Create an account bölümüne email adresi girer

And signUp butonuna basar

And user kisisel bilgilerini ve iletisim bilgilerini girer

And user Create Account butonuna basar

Then hesap olustugunu test edin



Cucumber

Genel Tekrar

1. <http://automationpractice.com/index.php> sayfasina gidelim
2. Cucumber ile asagidaki testi yapalim
 - Given user web sayfasinda
 - And user sign in linkine tiklar
 - And email kutusuna @isareti olmayan email adresi yazar ve enter'a tiklar
 - Then error mesajinin “Invalid email address” oldugunu dogrulayin



Cucumber

Genel Tekrar

1. <http://demo.guru99.com/test/web-table-element.php> sayfasina gidelim
2. Cucumber framework'de US1012_Guru_Web_Tables olusturun
3. Scenario : TC_16_kullanici_liste_alabilmeli asagidaki testi yapin

Given user web sayfasinda

Then Company listesini consola yazdirir

And DCB Bank'in listede oldugunu test eder

Cucumber

Genel Tekrar

1. <http://demo.guru99.com/test/web-table-element.php> sayfasina gidelim
2. Cucumber framework'de US1012_Guru_Web_Tables altinda

Scenario : TC_17_kullanici_sirket_Prev_Close_alabilmeli olusturun ve asagidaki testi yapin

Given user web sayfasinda

And “Istenen Sirket” Prev.Close degerini yazdirir



Cucumber

Genel Tekrar

1. <http://demo.guru99.com/test/web-table-element.php> sayfasina gidelim
2. Cucumber framework'de US1012_Guru_Web_Tables altinda

Scenario : TC_18_kullanici_satir_sutun_degeri_ile_yazi_alabilmeli olusturun
ve asagidaki testi yapin

Given user web sayfasinda

And “Istenen Satir”, “Istenen Sutun” daki yaziyi yazdirir

Cucumber

Genel Tekrar

1. <http://demo.guru99.com/test/web-table-element.php> sayfasina gidelim
2. Cucumber framework'de US1012_Guru_Web_Tables altinda

Scenario : TC_18_kullanici_sutun_basligi_ile_liste_alabilmeli olusturun
ve asagidaki testi yapin

Given user web sayfasinda

And “Istelenen Baslik”, sutunundaki tum degerleri yazdirir



Cucumber

Genel Tekrar

- 1) Yeni bir class olusturalim D34_readExcel
- 2) Baskentler excelini framework'e ekleyelim ve excelle ilgili islemleri yaparak dosyayı kullanabilir hale getirelim
 - 1.satirdaki 2.hucreye gidelim ve yazdiralim
 - 1.satirdaki 2.hucreyi bir string degiskene atayalim ve yazdiralim
 - baskenti Jakarta olan ulke ismini yazdiralim
 - Ulke sayisini bulalim
 - Fiziki olarak kullanılan satır sayısını bulun
 - Tüm bilgileri map olarak kaydedelim
 - baskenti Jakarta olan ulkenin tüm bilgilerini yazdiralim
 - Satır ve sutun bilgisi ile hucre yazdiralim



Cucumber

Genel Tekrar

Yeni bir Class olusturalim D34_WriteExcel

1) Yeni bir sutun olusturalim

- baslik satirinda ilk bos hucreye yeni bir cell olusturalim
- Olusturdugumuz hucreye “ulke nufusu” yazdiralim
- 2.satir ulke nufusu kolonuna “1,5 milyar” yazdiralim
- 8.satir nufus kolonuna 250 milyon yazdiralim
- Dosyayı kaydedelim
- Dosyayı kapatalım

Cucumber

Genel Tekrar

Yeni bir Class olusturalim Day36_ExplicitlyWait

- 1) <https://demoqa.com/browser-windows> adresine gidin
- 2) Alerts'e tiklayin
- 3) On button click, alert will appear after 5 seconds karsisindaki click me butonuna basin
- 4) Allert'in gorunur olmasini bekleyin
- 5) Allert uzerindeki yazinin "This alert appeared after 5 seconds" oldugunu test edin
- 6) Ok diyerek alerti kapatin



Cucumber

Genel Tekrar

Yeni bir test methodu olusturun

- 1) <https://demoqa.com/dynamic-properties> adresine gidin
- 2) “Will enable 5 seconds” butonunun enable olmasini bekleyin
- 3) “Will enable 5 seconds” butonunun enable oldugunu test edin



Cucumber

Genel Tekrar

Yeni bir test methodu olusturun

<https://demoqa.com/dynamic-properties> adresine gidin

- 1) “Visible After 5 seconds” butonunun gorunur olmasini bekleyin
- 2) “Visible After 5 seconds” butonunun gorunur oldugunu test edin



Cucumber

Genel Tekrar

Yeni bir test methodu olusturalim

https://the-internet.herokuapp.com/add_remove_elements/ adresine gidin

- 1) "Add Element" butona basin
- 2) "Delete" butonu gorunur oluncaya kadar bekleyin
- 3) "Delete" butonunun gorunur oldugunu test edin
- 4) Delete butonuna basarak butonu silin
- 5) Delete butonunun gorunmedigini test edin