

Q U A I T Y U E R S E  
A C A D E M Y

Selenium

Ders-01

Selenium Giriş

Webdriver Method'lari

WebElements

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# Software Testing Nedir ?

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

Software testi icin tasrim 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 karşıladığından emin olmak icin positive ve negative test senaryolari olusturmali ve bu testleri otomasyonla yapacak test case'leri olusturmalıdır.



# Software Testing Neden Önemlidir ?

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

Ayrıca developer'larin user case'den anladiklari ile end – user'larin kullanım aliskanliklari her zaman uyusmayabilir.

Uygulamaya sonradan eklenen bazi feature'lar calisan uygulamada bazi islevleri negatif etkileyebilir.

Kullanicinin beklentilerini karsilamayan uygulamalar basarisiz olur.



- Urunun end-user kullanımına hazır olduğundan emin olmak
- End-user tarafından karşılanan 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 ?

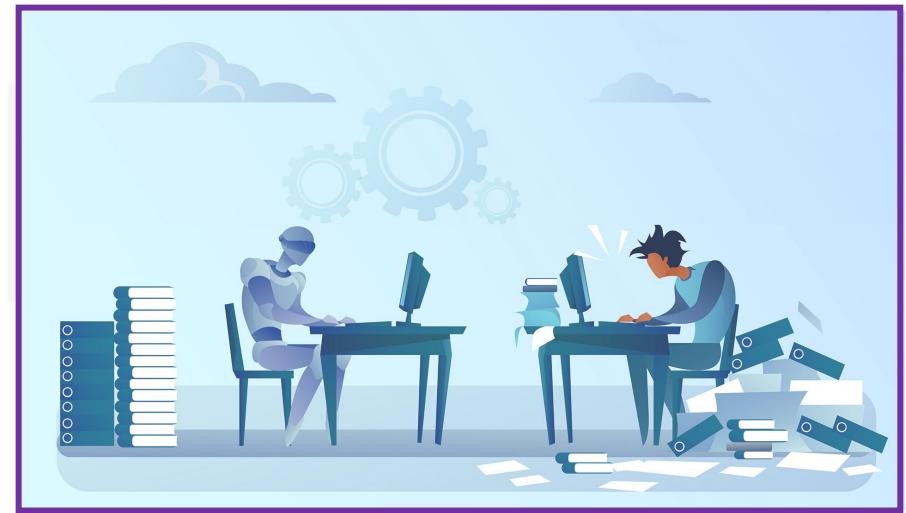
Manual testing, uygulamanın planlanan sonuçlara uygun çalışıp çalışmadığını hiç bir otomasyon aracı kullanmadan, bir end-user gibi test edilmesidir.

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 ihtiyac 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 aracı kullanilarak kodlar aracılıgi ile yapılmasidir.

Test otomasyon sayesinde

- klavye ve mouse kullanilarak yapilabilecek islemlerin cogu yapilabilir,
- yapılan islemler sonucunda gereklesen 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 çok talep gormektedir.

# Automation & Manual Testing

Yandaki kod sizce nedir ?

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

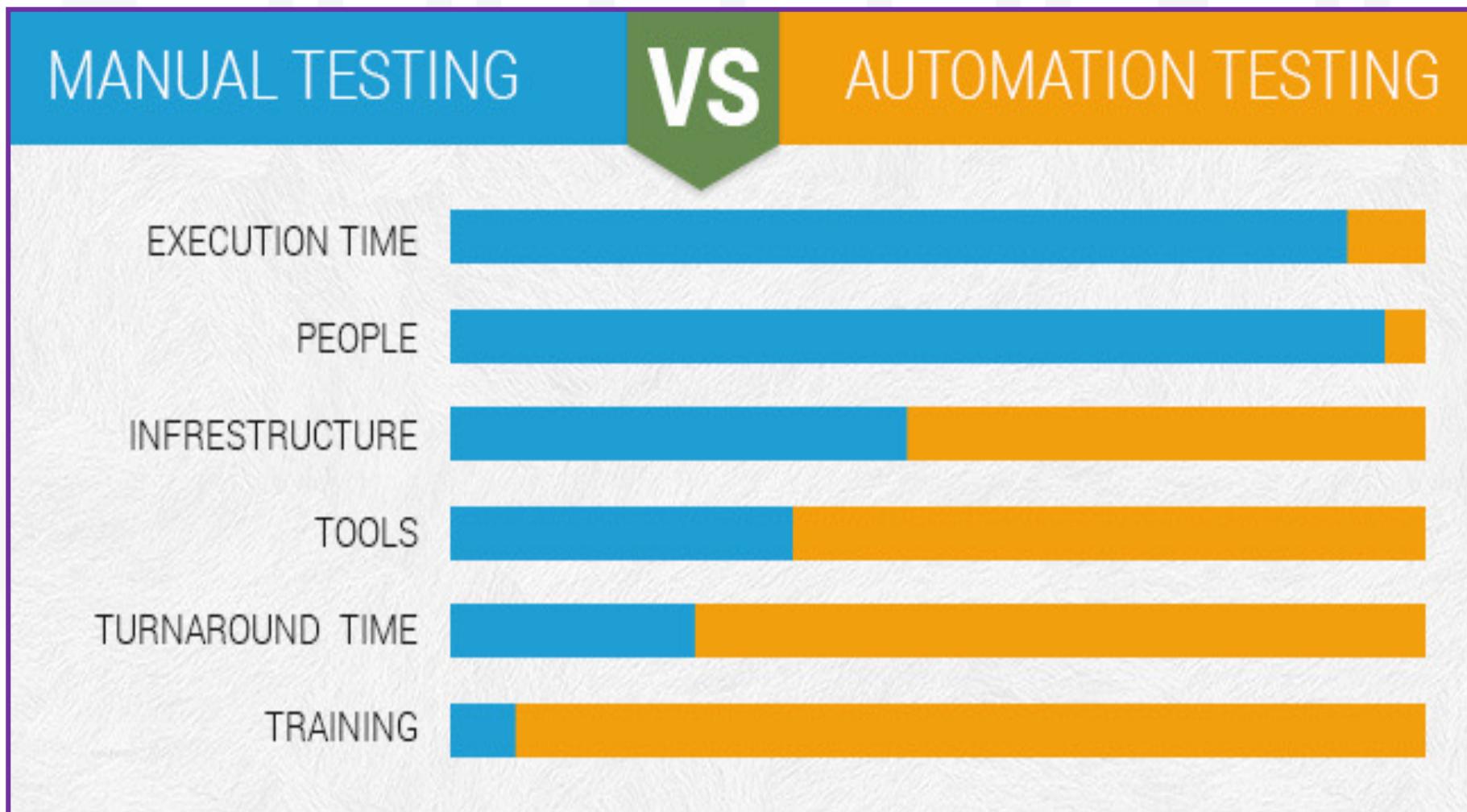
**Feature:** US1010 herokuapp Delete testi

```
@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 kapatır
```

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

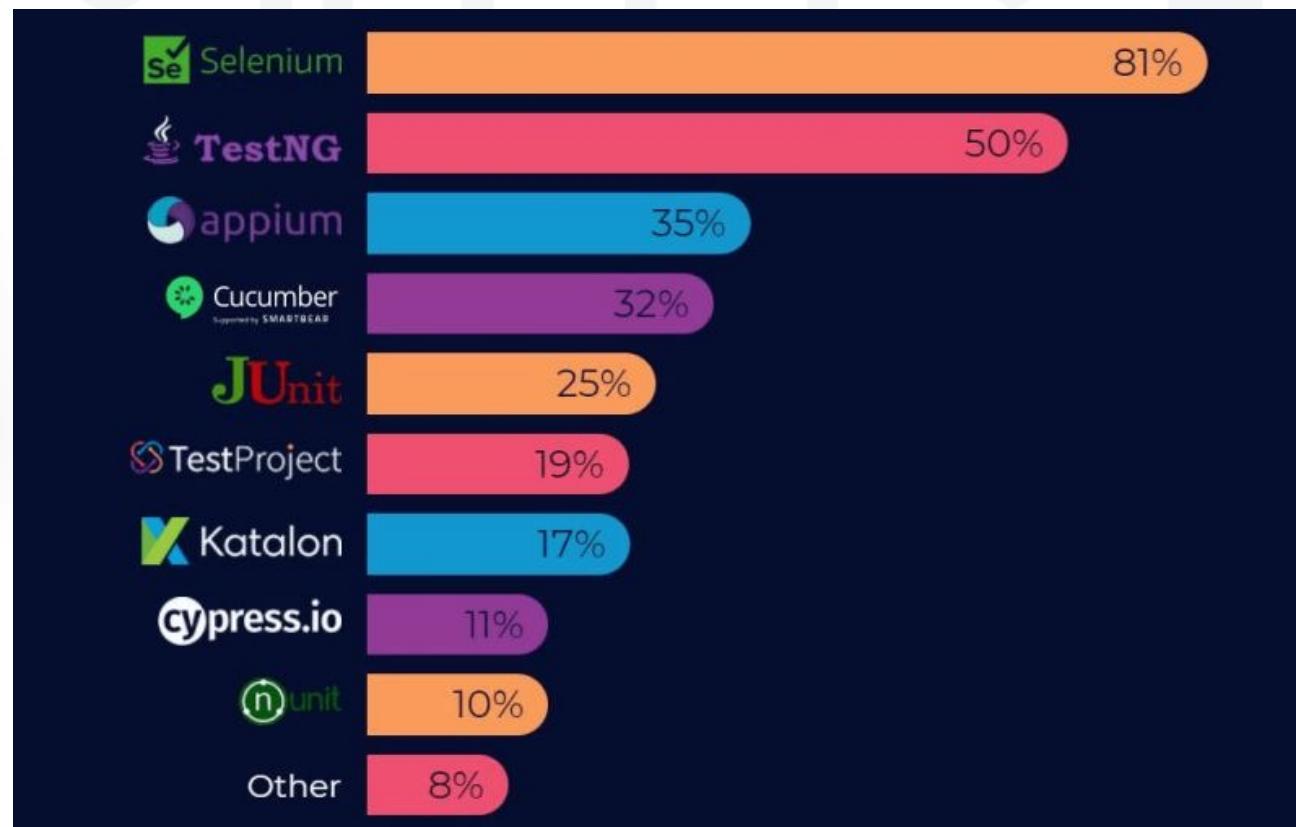
Manuel Test sayesinde kod bilgisi olmasa bile insanlara test yapabilir, temel test ihtiyaclarinizi karsilayabilirsiniz.

# Automation & Manual Testing



# 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'ları otomasyon ile çalıştıracak tool'ların çalışma için oluşturulan bir paketdir.

Selenium farklı programlama dilleri ile çalışarak günümüzde kullanılan browser'ların tamamını otomasyon ile çalıştırabilme için oluşturulan class ve method'lara sahiptir.

Selenium'u kullanabilmek için bu class'lar çalışılan projeye eklenmelidir.

Selenium'un class'larını, kendi sitesinden indireceğimiz jar dosyalarını projeye ekleyerek projemize dahil edebilir veya bu işi bizim adımıza yapacak maven gibi tool'lari kullanarak class'lari direkt 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'lari 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 ?

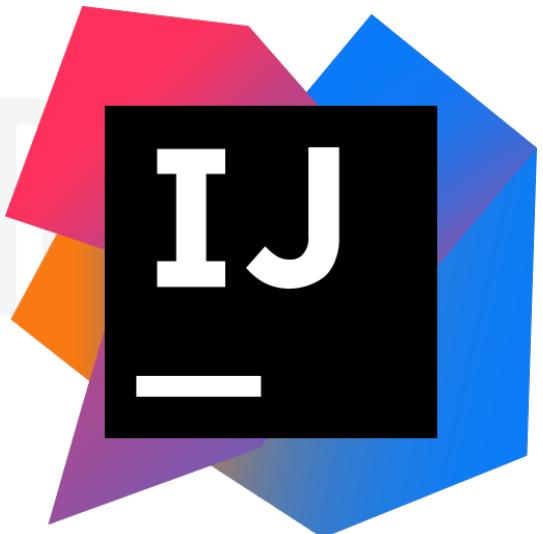
High Level programlama dilleri calisabilmek icin derleyicilere ihtiyac duyarlar.

IntelliJ IDEA 2000 yılında kurulmuş olan JetBrains firmasına ait olan, popüler bir kod gelistirme ortamı (**I**ntegrated **D**evelopment **E**nvironment) dir.

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

Bir çok framework ve plugin ile calisma imkani saglar.

Kısaca, IntelliJ IDE ihtiyaçlarınızı tahmin eder ve sıkıcı ve tekrarlayan geliştirme görevlerini otomatikleştirir, böylece büyük resme odaklanabilirsiniz.



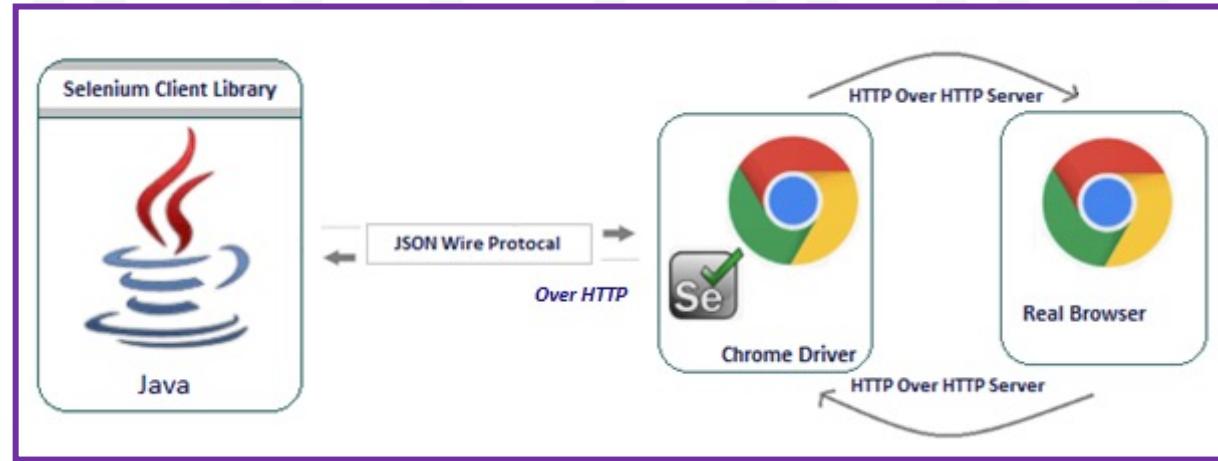
# Selenium Bileşenleri

**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 tiklama, yazi gonderme, elementler uzerindeki yazilari alma gibi pekcoek 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.)

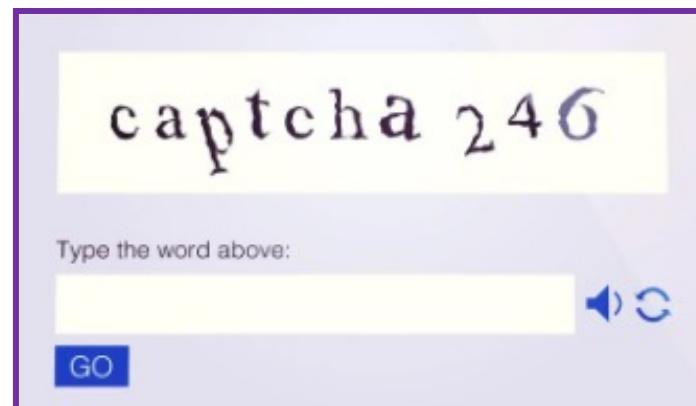
Programlama bilgisi gerektirir (Biz Java biliyoruz)

Yalnızca web tabanlı uygulamaları test eder

Profesyonel desteği sahip değil

performans testleri yapamaz

Captcha'yı asamaz(düger tüm otomasyon araçları gibi)



# Framework Nedir ?

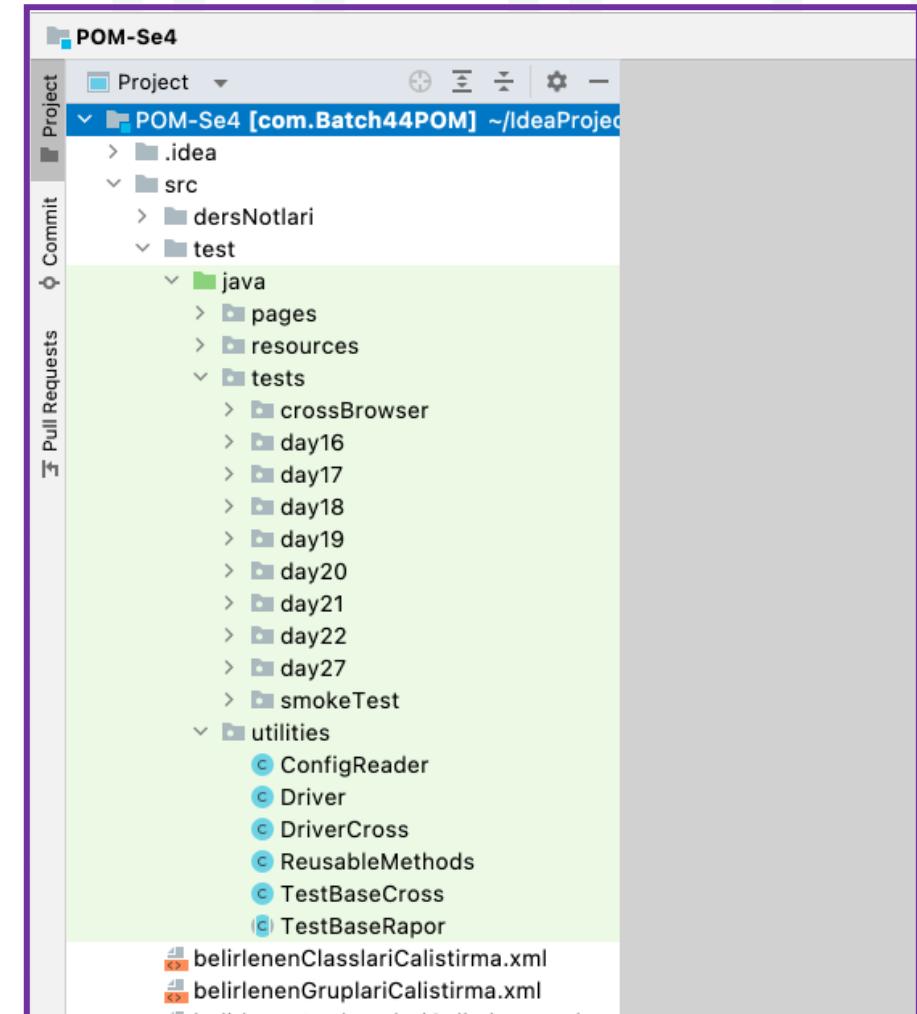
Framework, uzerine kodlarimizi yazarak projelerin olusturulabilecegi bir yapidir.

Test otomasyonu yaparken herseye sifirdan baslayip, herseyi sifirdan kurgulayip olusturmak yerine,

Herkesin anlayabilecegi, test olusturma ve gozden gecirme sureclerini kolaylastirmak, tum ekibin ortak calisabilecegi bir yapı olusturmak icin test framework'lari olusturulmustur.

Framework, test yapmak icin kullandigimiz tum enstrumanlari birlestirir.

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 olani download edin  
Indirilen surum ile bilgisayarınızdaki Chrome browser surumunun aynı olduğundan emin olun
- 4) src altında resources director'si olusturun
- 5) Bu klasor altında drivers ve libraries klasorleri olusturun
- 6) Indirdigimiz chromedriver'i drivers klasorune, selenium-java dosyasını ise libraries klasorune cikartın
- 7) intelliJ 'de yeni project / package / class olusturalım ve class icinde main method olusturun
- 8) File/Project Structure/Modules/Dependencies kismindan jar dosyalarini yukleyin

# 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.

# WebDriver Objesi Kullanma

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

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

public class C01_DriverMethods {

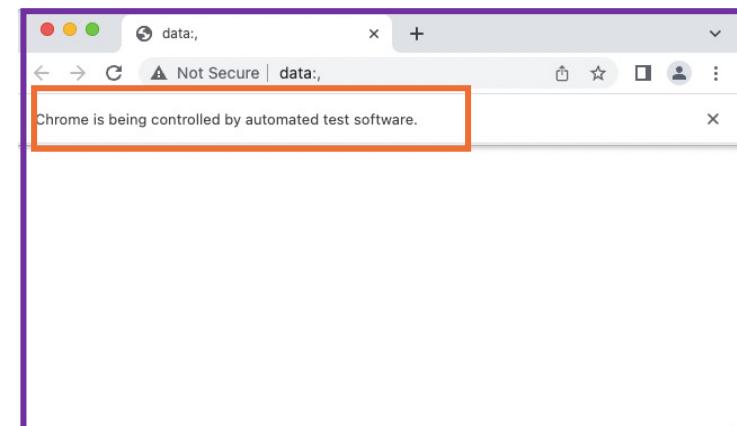
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver", "src/resources/chromedriver");

        WebDriver driver= new ChromeDriver();
```

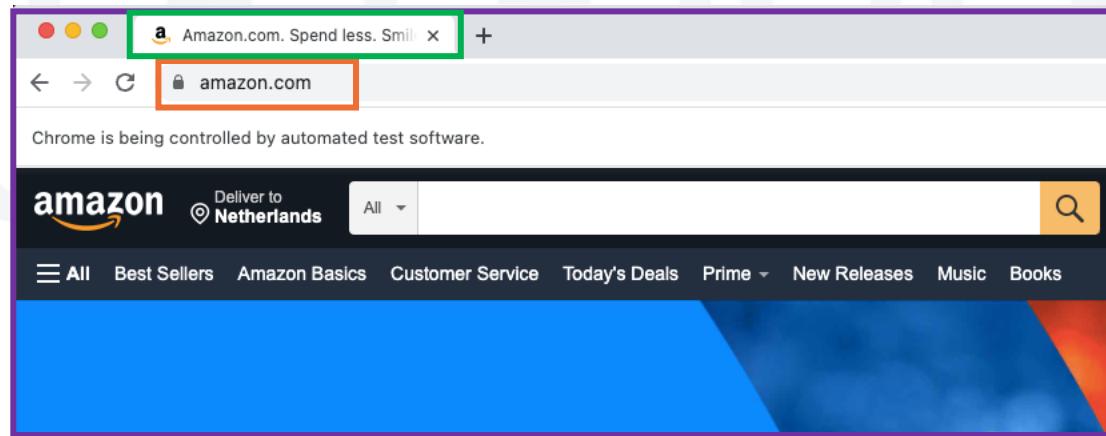
Ilgili ayarları yapıp bir driver objesi olusturdugumuzda, Selenium bu driver objesi sayesinde otomasyon yapabilecegimiz bir browser acar.

Bu browser'un Selenium tarafından kontrol edildigi yazılıdır.

Chrome disinda bir browser kullanilacaksa, o browser'in driver'ini da projeye eklenmeli ve class icindeki ayarlarda o driver'in dosya yolu driver'a gosterilmelidir.



# driver.get...() Method'lari



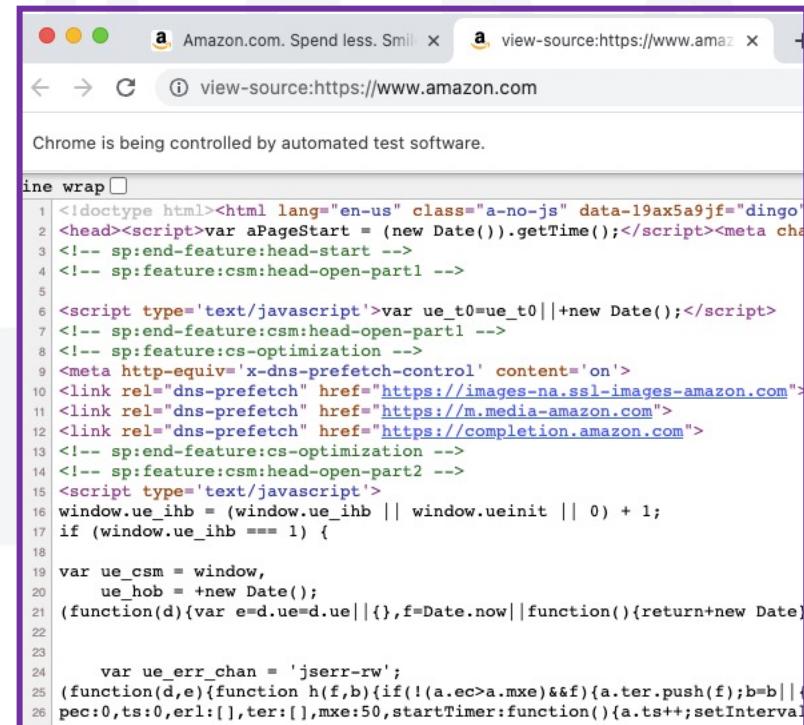
- 1- `driver.get("https://www.amazon.com");` driver'i istenen url'e goturur.
- 2- `driver.getCurrentUrl();` Gidilen Web sayfasinin URL bilgisini döndürür.
- 3- `driver.getTitle();` Gidilen Web sayfasinin title (baslik) bilgisini döndürür.
- 4- `driver.close();` Acilmis olan driver'i kapatir
- 5- `driver.quit();`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

# 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.



A screenshot of a Google Chrome browser window. The address bar shows 'view-source:https://www.amazon.com'. The page content is the raw HTML source code of the Amazon homepage. At the top, it says 'Chrome is being controlled by automated test software.' Below that, the source code starts with a doctype declaration and various meta tags and script elements.

```
<!DOCTYPE html><html lang="en-us" class="a-no-js" data-19ax5a9jf="dingo'>
<head><script>var aPageStart = (new Date()).getTime();</script><meta cha
<!-- sp:end-feature:head-start -->
<!-- sp:feature:csm:head-open-part1 -->
<!-- sp:feature:csm:head-open-part2 -->
<script type='text/javascript'>var ue_t0=ue_t0||+new Date();</script>
<!-- sp:end-feature:csm:head-open-part1 -->
<!-- sp:feature:cs-optimization -->
<meta http-equiv='x-dns-prefetch-control' content='on'>
<link rel="dns-prefetch" href="https://images-na.ssl-images-amazon.com">
<link rel="dns-prefetch" href="https://m.media-amazon.com">
<link rel="dns-prefetch" href="https://completion.amazon.com">
<!-- sp:end-feature:cs-optimization -->
<!-- sp:feature:csm:head-open-part2 -->
<script type='text/javascript'>
window.ue_ihb = (window.ue_ihb || window.ueinit || 0) + 1;
if (window.ue_ihb === 1) {
    var ue_csm = window,
        ue_hob = +new Date();
    (function(d){var e=d.ue=d.ue||{},f=Date.now||function(){return+new Date()
        var ue_err_chan = 'jserr-rw';
        (function(d,e){function h(f,b){if(!((a.ec>a.mxe)&&f)){a.ter.push(f);b=b||{pec:0,ts:0,erl:[],ter:[],mxe:50,startTimer:function(){a.ts++;setInterval
            ,0,1000)};b.ter.push(h);b.ter.push(f);b.ter.push(b)}}
        h(e,d);
    })(d,e);
}
```

## 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");
}
```

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

# WebDriver Method'ları

1. Yeni bir package olusturalim : day01
2. Yeni bir class olusturalim : CO3\_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 donulmusse yeniden ilk sayfaya götürür.

12- `driver.navigate().refresh();`

Içinde olunan web sayfasını yeniler.

# driver.navigate...( ) Method'lari

1. Yeni bir Class olusturalim.C05\_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 sayfalari 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 olmasının her testin basında 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 basında `maximize()` method'u kullanmasında fayda vardır.

# driver.manage( )... Method'lari

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 kullanıcının istedigi ozel olculere getirir ve istenen noktaya tasir.

17- 

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

Acilan driver'in bulunduğu 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

Q U A I T Y U E R S E  
A C A D E M Y

# Selenium

## Ders-02

### WebElements

### Locators

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# WebDriver Method'lari

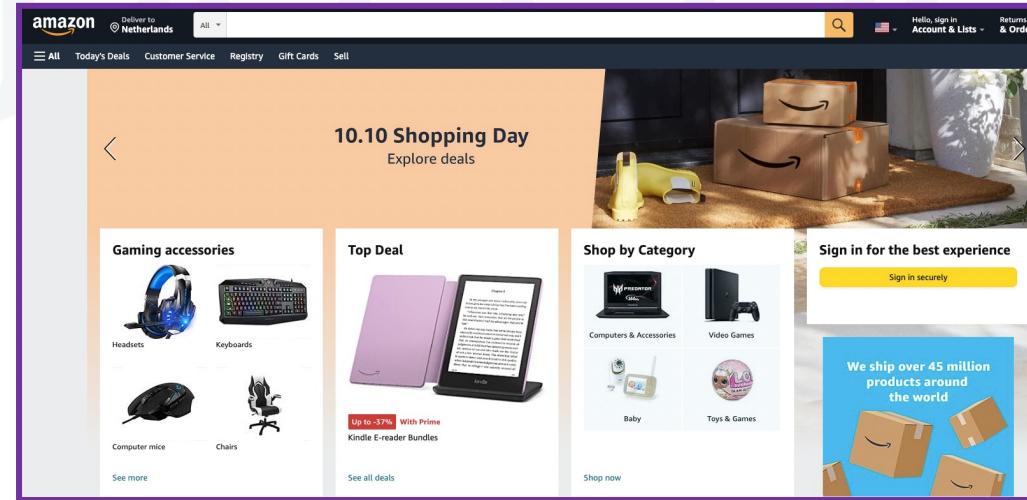
- 1.Yeni bir class olusturalim (Homework)
- 2.ChromeDriver kullanarak, facebook sayfasina gidin ve sayfa basliginin (title) “facebook” oldugunu doğrulayin (verify), degilse dogru basligi yazdirin.
- 3.Sayfa URLinin “facebook” kelimesi icerdigini doğrulayin, icermiyorsa “actual” URL’i yazdirin.
- 4.<https://www.walmart.com/> sayfasina gidin.
5. Sayfa basliginin “Walmart.com” icerdigini doğrulayin.
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.



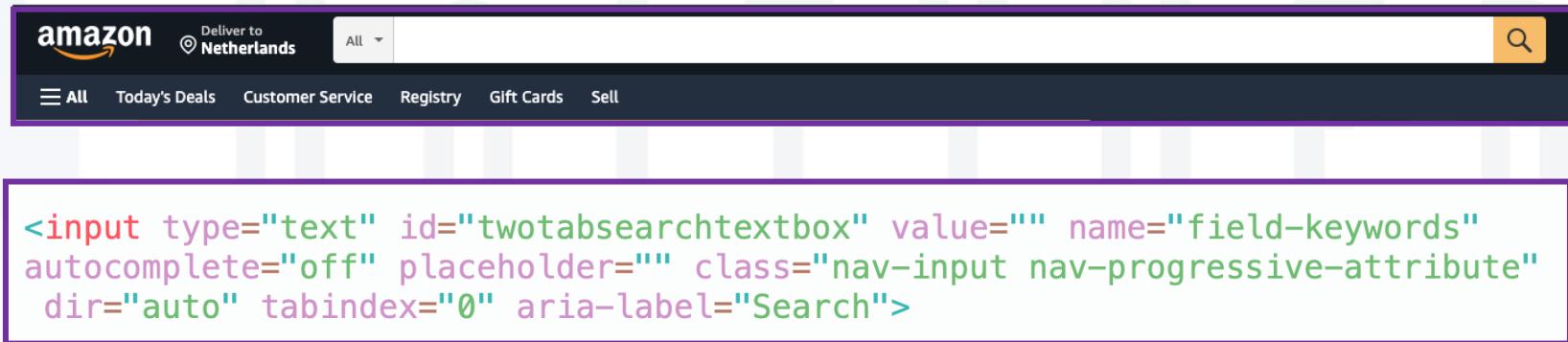
Her web element farklı özelliklerde olur. Link, açılır menu, button gibi etkilesimli web elementler olduğu gibi resim, background gibi etkilesimsiz web elementler de olur.

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

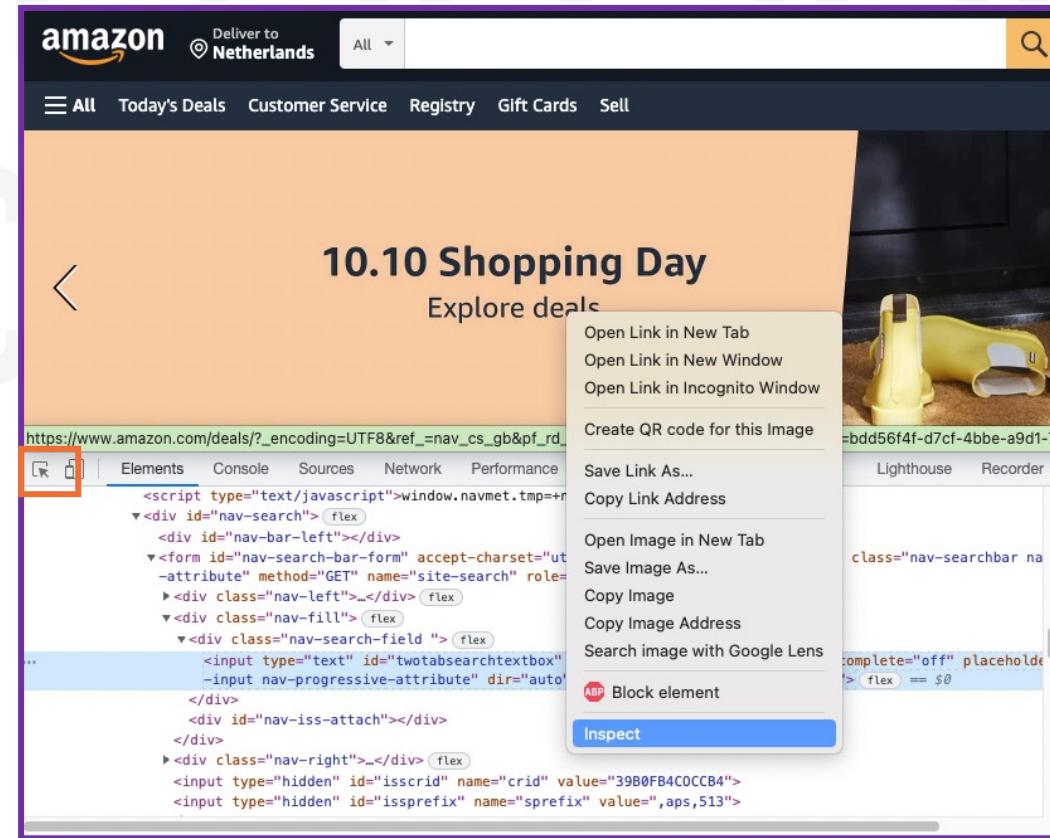
# WebElements

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 istenen element seçilebilir.

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



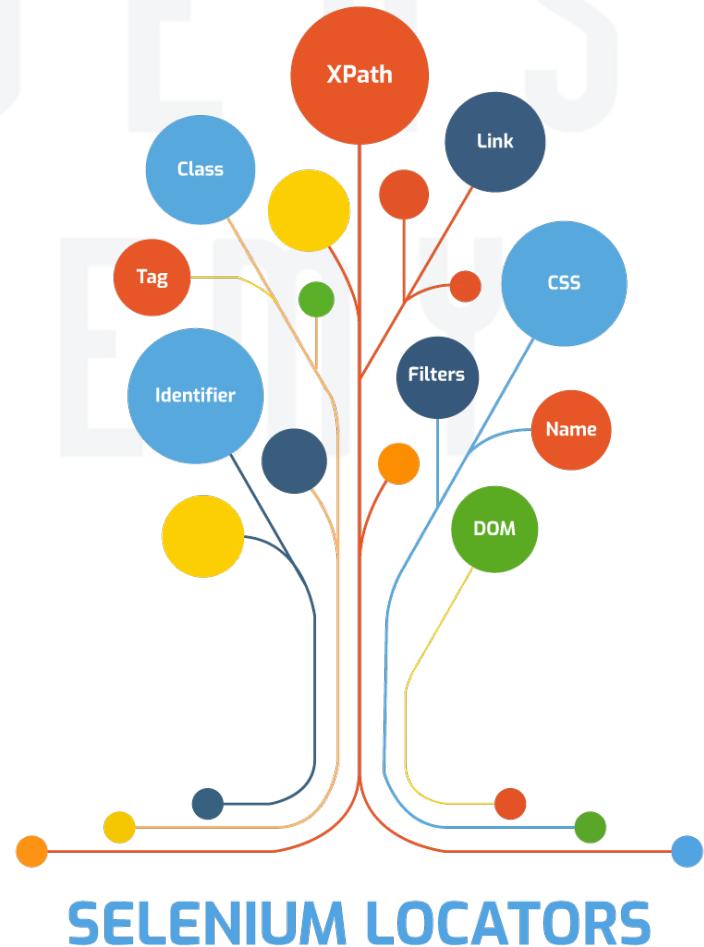
# Locators

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.



# 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">
```

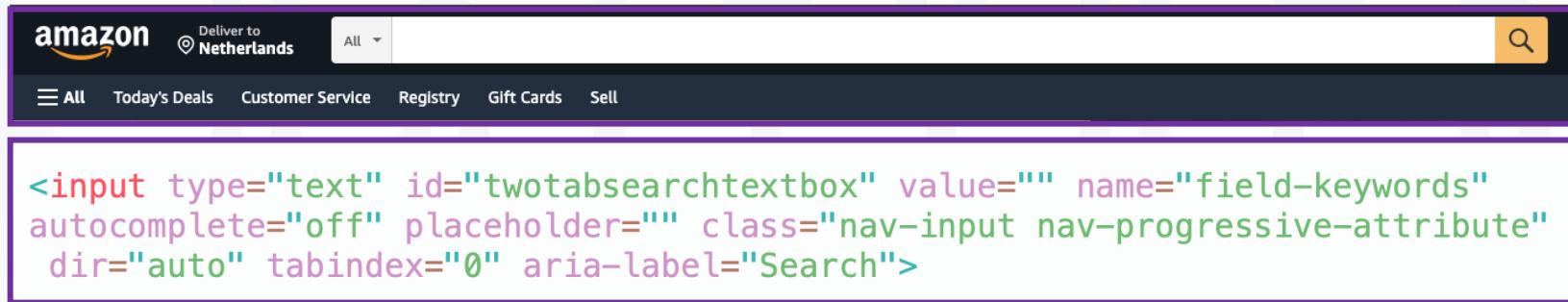
## 1- By.id("uniqueld")

Web elementi tanımlamak için ilk bakacagımız locator id olabilir.

Id genellikle unique olduğu için 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



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

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

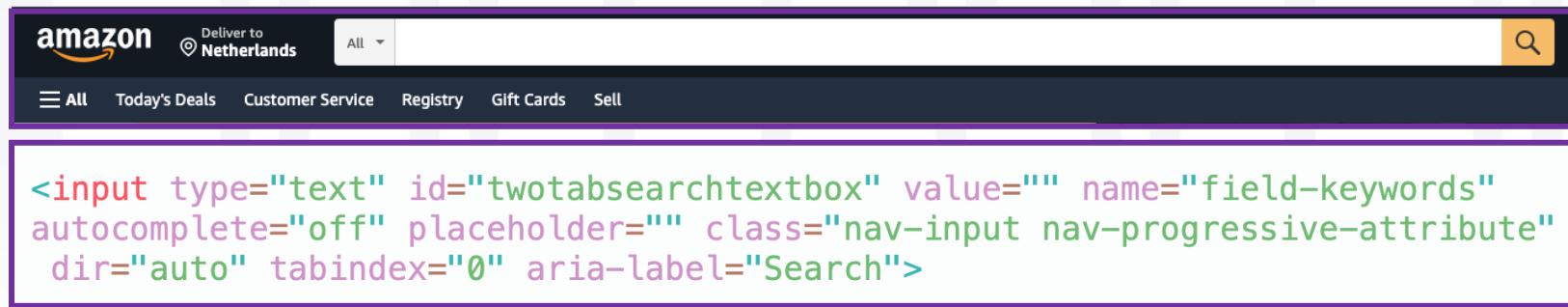
```
WebElement amazonAramaKutusu = driver.findElement(By.id("twotabsearchtextbox"));
```

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

findElement( ) ile locate edilip, objeye atanen webElement testler sirasinda kullanilabilir.

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

# WebElement Objesi Olusturma



## 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



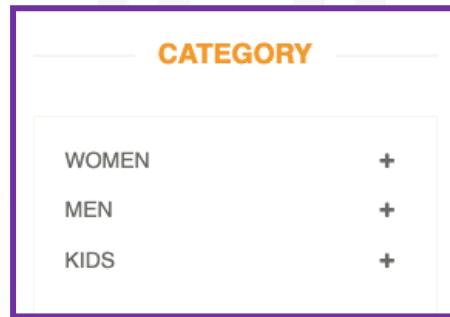
```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords" autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute" dir="auto" tabindex="0" aria-label="Search">
```

## 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



## 3- By.className("uniqueClassName")

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

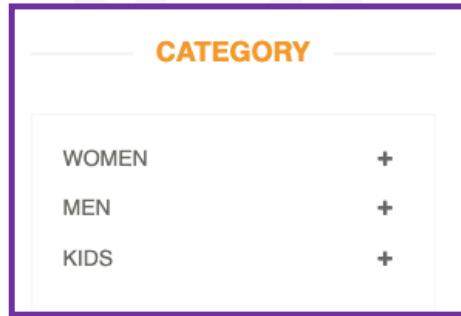
Bu sebeple class attribute'u ile yapacagimiz locate islemleri genellikle 1 web element degil, birden fazla element döndürür.

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

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

```
<div class="panel-group category-products" id="accordian">
  <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">
```

# driver.findElements( ) Method'u



driver.findElements(.....)

Locator'a uygun tüm web elementlerini döndürür.

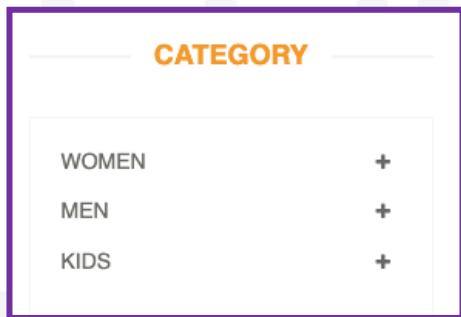
```
<div class="panel-group category-products" id="accordian">
  <div class="panel panel-default">
    <div class="panel-heading">
      <h4 class="panel-title">...</h4>
    </div>
    <div id="Women" class="panel-collapse collapse">...</div>
  </div>
  <div class="panel panel-default">
    <div class="panel-heading">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

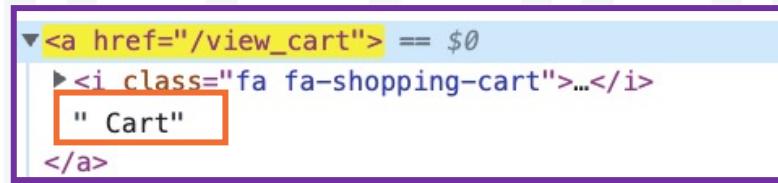
```
List<WebElement> inputTagList =driver.findElements(By.tagName("input"));
```

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

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

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

# Locators



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

5- By.linkText("linkYazisininTamami")

6- By.partialLinkText("linkYazisininBirBolumu")

Sadece link'ler icin kullanilabilirler.

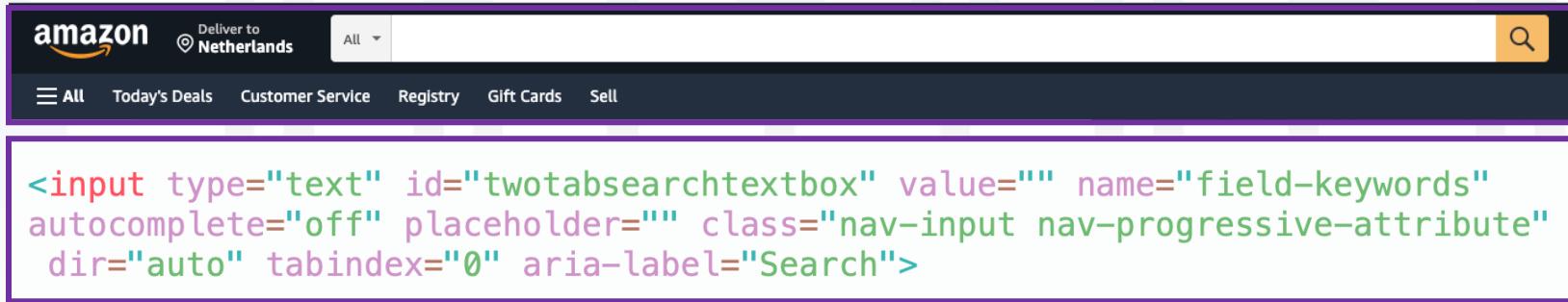
Her link uzerinde bulunan yazi kullanilarak locate yapmamizi saglar.

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

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

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

# WebElement Method'lari



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

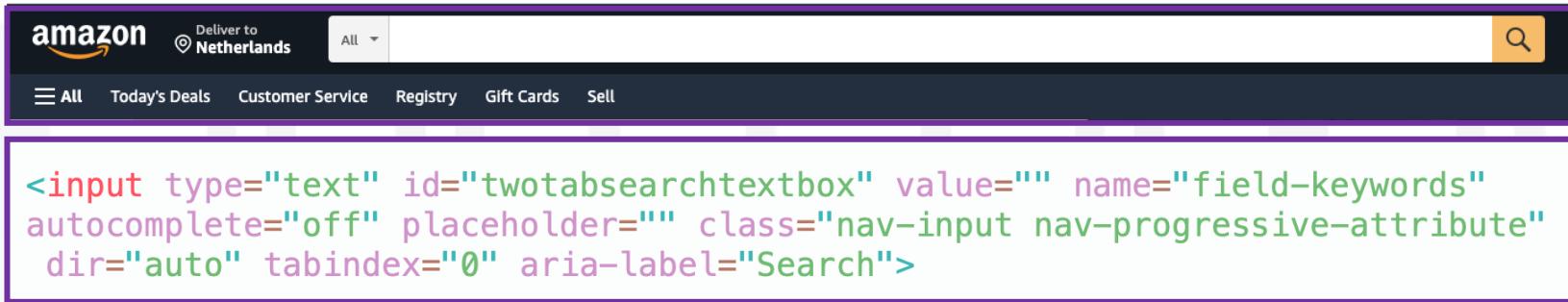
Webelement'i locate ettikten sonra variable atamak veya atamadan direk kullanmak test adimlarina ve belirlenen genel test stratejisine baglidir.

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



3- `webElement.submit();` Web element ile işlem yaparken ENTER tusuna basma işlemini yapar.

4- `webElement.sendKeys( ...keysToSend: "Istenen Metin" + Keys.ENTER);`

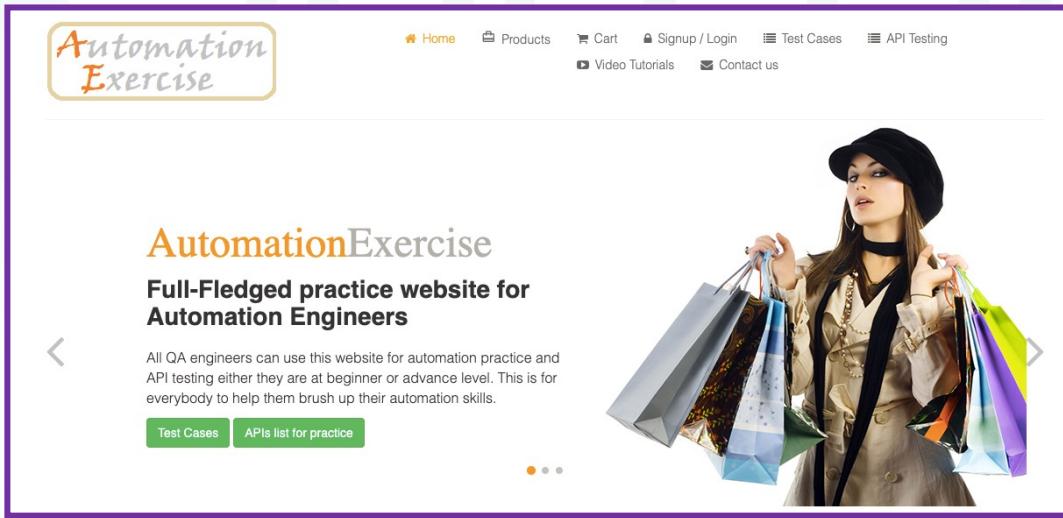
Web element'e istenen metni yollayıp, sonra ENTER tusuna basar

5- `webElement.isEnabled();` Web element erişilebilir ise true, yoksa false döner.

6- `webElement.isDisplayed();` Web element gorunuyor ise true, yoksa false döner.

7- `webElement.isSelected();` Web element secili ise true, yoksa false döner.

# Locators



## Automation Exercise Link Testi

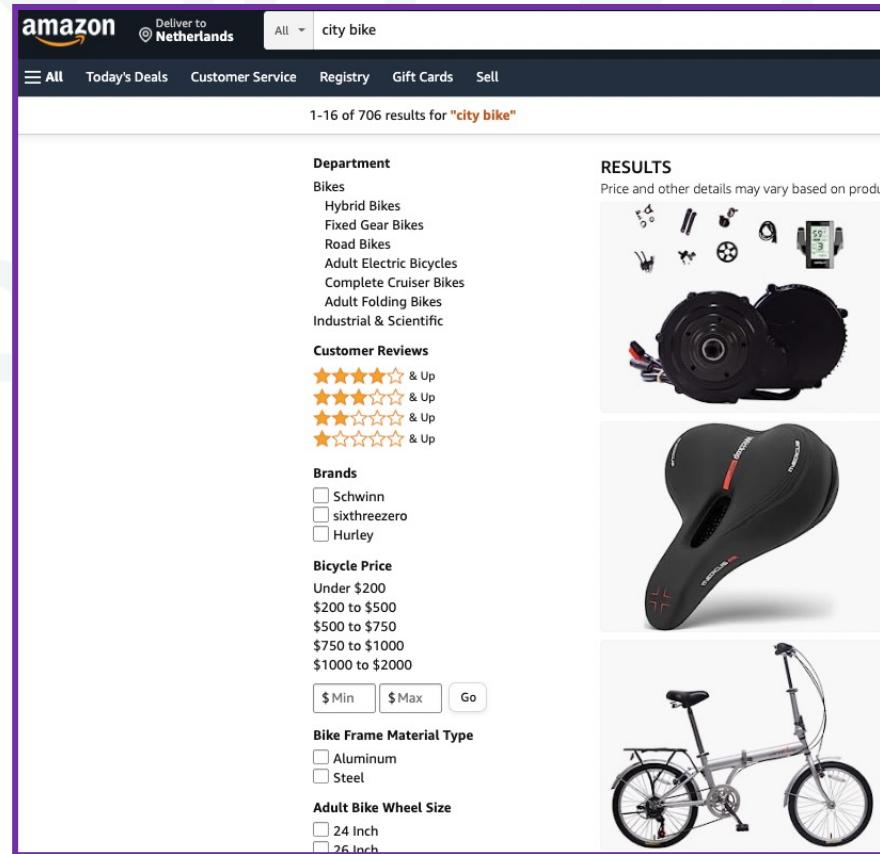
- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.automationexercise.com/> adresine gidin
- 3- Sayfada 147 adet link bulundugunu test edin.
- 4- Products linkine tiklayin
- 5- special offer yazisinin gorundugunu test edin
- 6- Sayfayi kapatin

# 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	<code>WebElement</code>	<code>List&lt;WebElement&gt;</code>
Elemana erişim	Direk ulaşılabilir	Liste'den index veya iterator ile ulaşılabilir

# Locators

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



# Locators - Xpath

Bir WebElement'i locate etmek icin kullanabilecegimiz en etkin yöntemdir.

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

Onceki 6 locator, HTML element olusturulurken developer'in yazdigı 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 tag'inin arasina yeni bir tag acildiginda konum olarak bir tab icerden baslar.

HTML tag'inin 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>
      </tr>
    </tbody>
  </table>
</div>
```

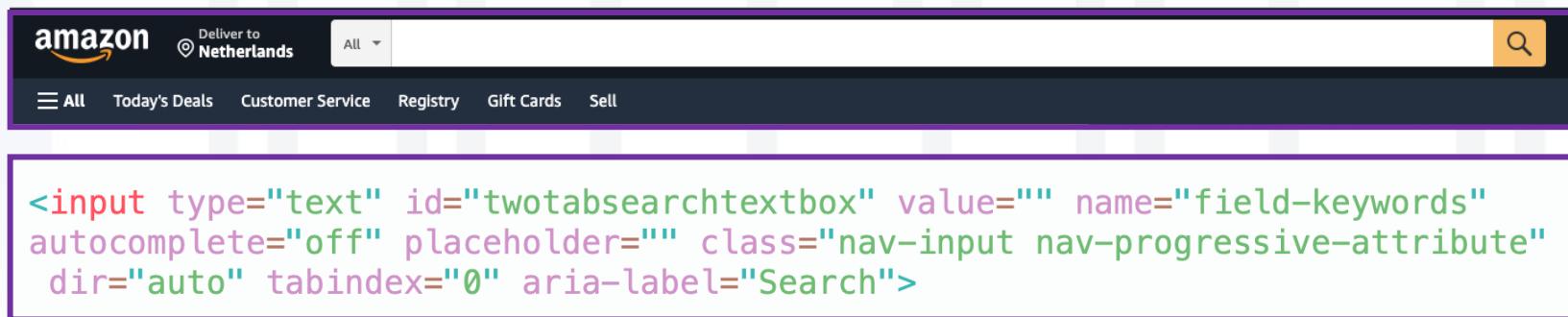
Absolute xpath yazmak icin en basa // sonraki her adimda / yazarak hedef web element'e kadar tum tag'lar yazilir.

Eger ayni path'e sahip birden fazla element varsa index kullanilabilir. [2] gibi

Eger bir parent'in grand child'lari icinde unique bir tag varsa parent // grand child yazilabilir

# Locators

## 2.Relative Xpath



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']

Q U A I T Y U E R S E  
A C A D E M Y

Selenium

Ders-03

Locators

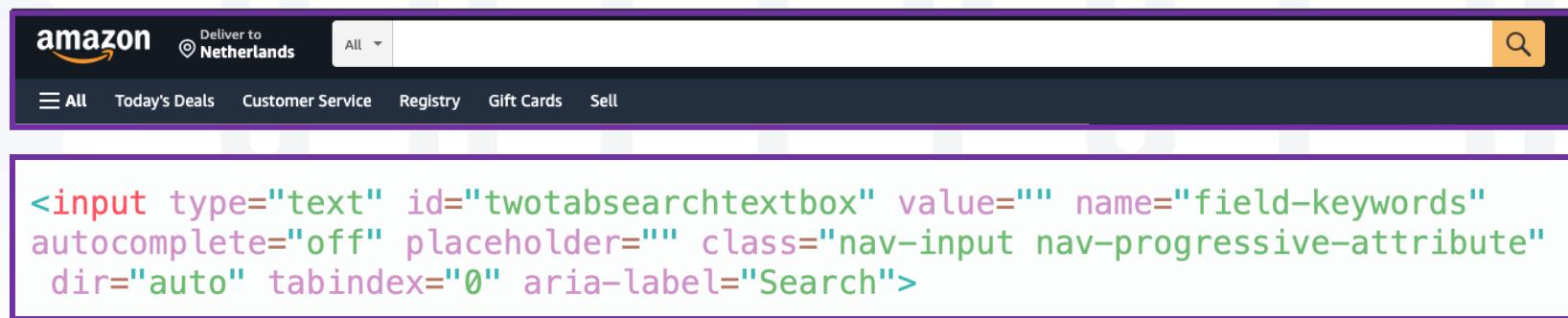
JUnit Framework

unityverseacademy.com

Egitmen : Ahmet BULUTLUOZ

# Locators

## 2.Relative Xpath



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 kullanılan 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')] "
```

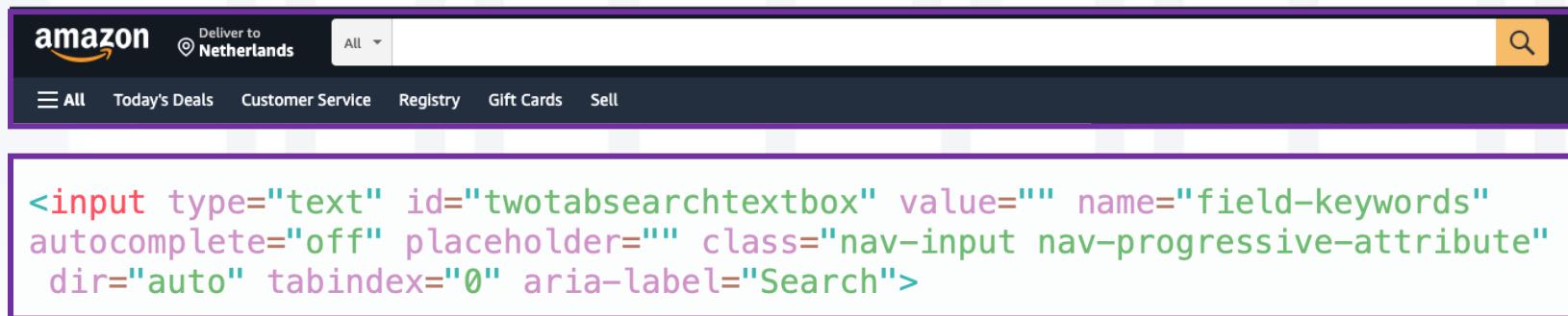
# Locators

## Relative Xpath Soru

- 1- [https://the-internet.herokuapp.com/add\\_remove\\_elements/](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



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 ulaşılabilen 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
- 3- Browseri tam sayfa yapin
- 4- Sayfayı "refresh" yapın
- 5- Sayfa basliginin "Spend less" ifadesi icerdigini 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 referansı 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 yakinalarinda 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.

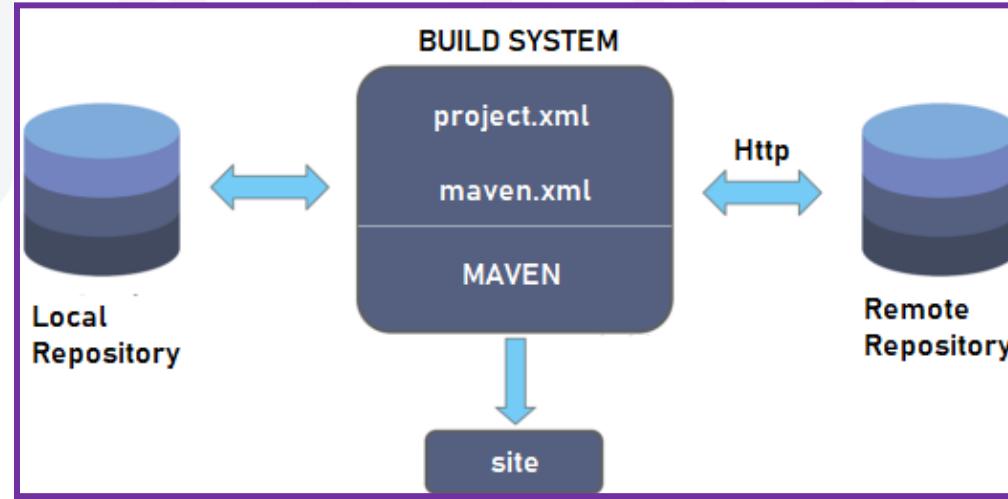
Proje nesne modeli (POM) konseptine dayalı olarak Maven, bir projenin inşasını, raporlamasını ve dokümantasyonunu merkezi bir bilgi parçasından(dependency) yönetebilir.

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

Amaçları çerçevesinde başlayan bir çalışma sonucu ortaya çıkmıştır.

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

# Maven



Maven bir Java olusturma 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)

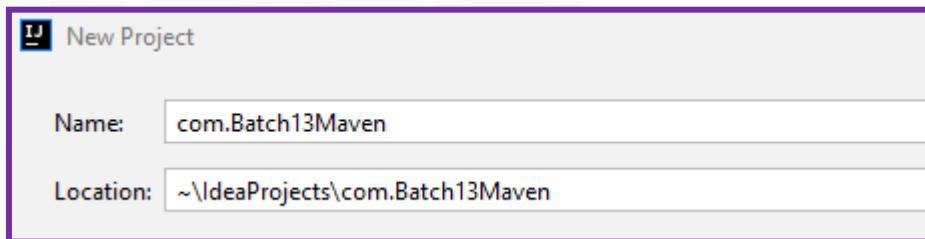
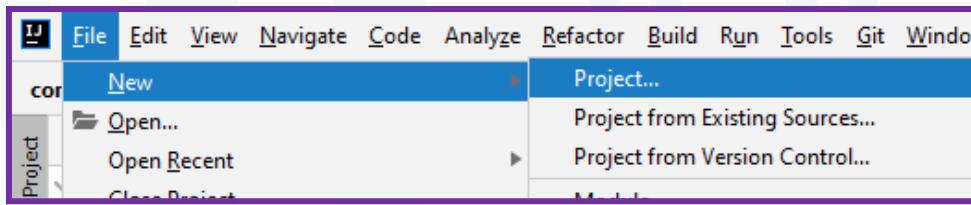
Maven ile çalışan tüm projeler aynı konfigürasyonu kullandığı için yeni bir projede çalışmaya başlandığında projenin anlaşılması çok kolay olacaktır.

# Neden Maven ?

- 1- Proje yönetiminde oluşturma, belgeleme, yayınılama ve dağıtım gibi tüm süreçlerin yönetilmesine yardımcı olur.
- 2- Projenin ve yapım 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 karşılaşılan sorunlara çözüm bulmakta zorluk yaşanmaz.



# Maven Proje Olusturma



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.

# pom.xml'e Dependency Ekleme

```
<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 arasında <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. Kirmiziligi 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, and C01\_... . The pom.xml tab is active. On the right side, there is a 'Maven' tool window. The 'Dependencies' section is expanded, showing two entries: org.seleniumhq.selenium:selenium-java:4.4.0 and io.github.bonigarcia:webdrivermanager:5.3.0. A red box highlights the 'Dependencies' section in the tool window and the corresponding XML code in the pom.xml file. Another red box highlights the 'SeleniumInt' project name in the tool window and its corresponding XML code in the pom.xml file. The XML code in the pom.xml file is as follows:

```
<?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/pom-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

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

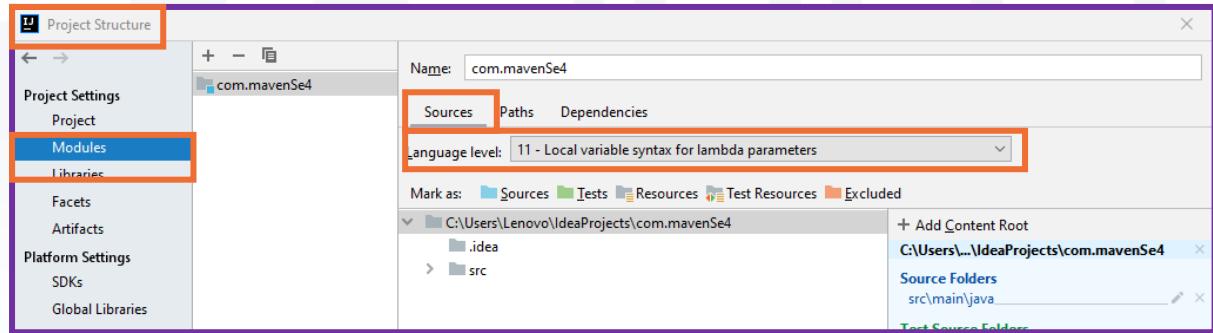
    <properties>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
        <dependency>
            <groupId>org.seleniumhq.selenium</groupId>
            <artifactId>selenium-java</artifactId>
            <version>4.4.0</version>
        </dependency>

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

```

# pom.xml'e Dependency Ekleme



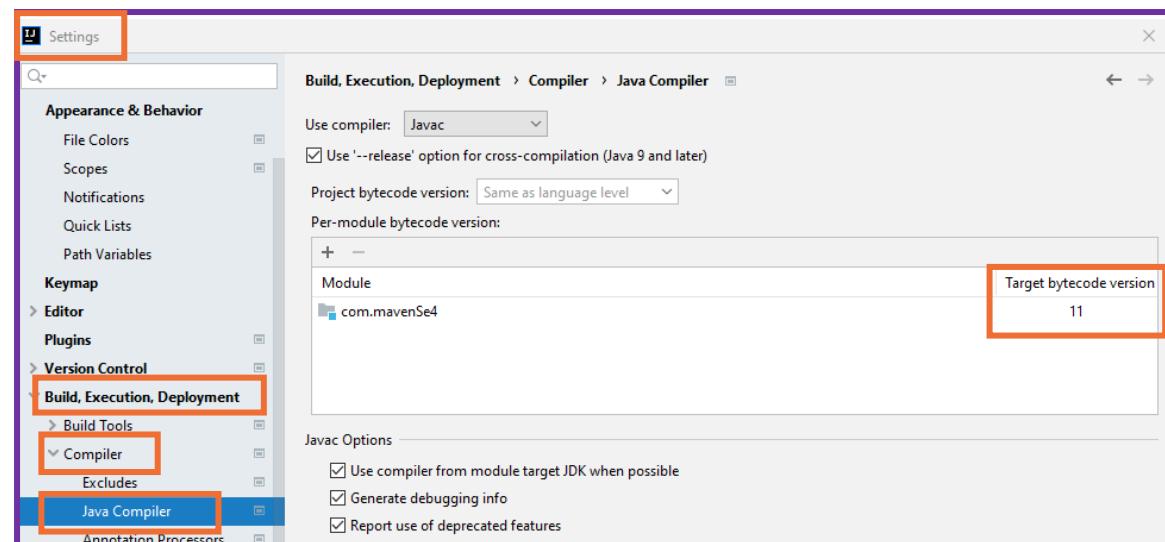
File  
Project Structure  
Modules  
Sources

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 Olusturma

Maven ile Selenium Java ve WebDriverManager dependency'lerini projemize eklendiği için, her seferinde driver.exe dosyasını driver'a tanıtmaya 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 yapılmırken 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. および 2. 行で Chrome 代わりにそのブラウザの選択が有効になります。

# 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

# Maven ClassWork

1. <http://zero.webappsecurity.com> sayfasina gidin
2. Signin buttonuna tiklayın
3. Login alanine “username” yazdırın
4. Password alanina “password” yazdırın
5. Sign in buttonuna tiklayın
6. Back tusu ile sayfaya donun
7. Online Banking menusunden Pay Bills sayfasina gidin
8. amount kismina yatırmak istediğiniz herhangi bir miktarı yazın
9. tarih kismina “2020-09-10” yazdırın
10. Pay buttonuna tiklayın
11. “The payment was successfully submitted.” mesajının çıktığını 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

# Selenium

## Ders-04

**JUnit Framework**

**Test Base Class**

**Dropdown Menu**

# JUnit

Java ile olusturulabilecek en temel Test Framework'udur.

Open-source bir kutuphanedir.

Developer'lar tarafından yapılan unit test'ler için de kullanılır.

Testlerimizi yaparken bize kolaylık sağlamak amacıyla oluşturulan Assert, Test, Before, After gibi bir çok class'a sahiptir.

JUnit, Maven altyapısını kullanır. JUnit framework için ihtiyacımız olan kutuphaneleri mvn.repository.com adresinden bulabileceğimiz dependency'ler ile projemize ekleyebiliriz.

JUnit'de testler için ihtiyaç duyulan pek çok olsa da TestNG next generation olarak daha fazla özellik ile piyasaya çıkmış ve piyasayı domine etmiştir.



# JUnit

## 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.

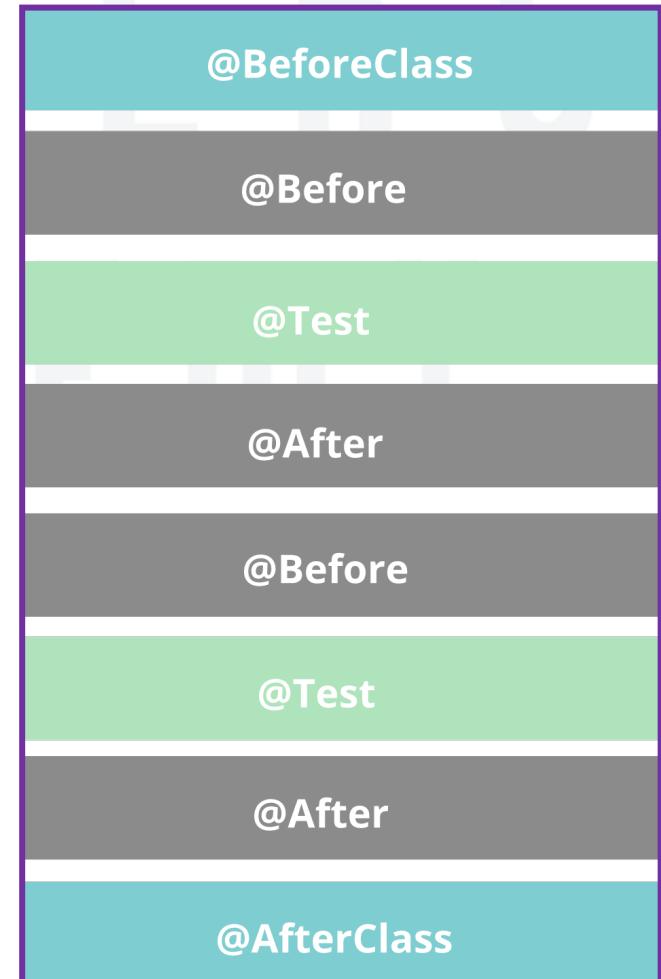
Testler sirasinda en çok kullanılan Junit notasyonları

@Test

@Ignore

@Before , @After

@BeforeClass , @AfterClass



# JUnit

## @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 olmalıdır.

@Ignore notasyonun tanimli olduğu metotlar test sirasinda calistirilmayacaktir. 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.

```
JUnit Test Report - C01_JUnitlikTest (ders05_notations)
```

Method	Status	Time (ms)
test01	Passed	1ms
test02	Ignored	0ms
test03	Failed	4ms

Tests failed: 1, passed: 1, ignored: 1 of 3 tests - 5ms

/Users/ahmetbulutluoz/Library/test01

Test ignored.

java.lang.AssertionError:  
Expected :5  
Actual :7  
[Click to see difference](#)

```
JUnit Test Report - C01_JUnitlikTest (ders05_notations)
```

Method	Status	Time (ms)
test01	Passed	1ms
test02	Ignored	0ms
test03	Failed	4ms

Tests failed: 1, passed: 1, ignored: 1 of 3 tests - 5ms

/Users/ahmetbulutluoz/Library/test01

Test ignored.

java.lang.AssertionError:  
Expected :5  
Actual :7  
[Click to see difference](#)

# JUnit

## @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();
}
```

# JUnit

## 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 bütündür.

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();
}
```

# JUnit

## @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 kalinmaz.

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

```
// amazon sayfasina giden
// 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();
}
```

# JUnit

## Annotations

Bir class'da hem @Before - @After notasyonlari, hem de @BeforeClass-@AfterClass notasyonlari birlikte kullanilabilir.

Birlikte kullanimda method'larin yapacaklari islemler ve yapilari, calisma sirasi ve notasyon ozelliklerine gore düzenlenmelidir.

Ornek : 2 Test method'u

@Before - @After notasyonlari

@BeforeClass-@AfterClass notasyonlari olan bir class calistirilirsa, yandaki gibi bir calisma olacak ve toplam 8 method calistirilmis olacaktir.



# JUnit

## Assertions

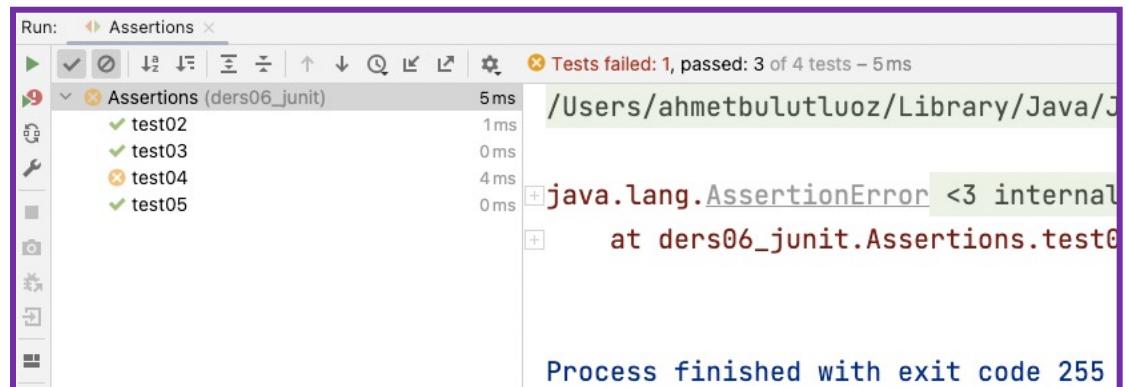
JUnit standart olarak, calistirilan her test method'unun sorunsuz olarak calisip calismadigini raporlar.

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, actual: sayi1+sayi2) // PASSED  
    }  
    @Test  
    public void test03(){  
        Assert.assertNotEquals(sayi3, sayi2) // PASSED  
    }  
  
    @Test  
    public void test04(){  
        Assert.assertTrue(condition: sayi3==sayi2) // FAILED  
    }  
  
    @Test  
    public void test05(){  
        Assert.assertFalse(condition: sayi3==sayi2); // Passed  
    }  
}
```



# JUnit

## 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); → Failed

Not equals

Assert.assertNotEquals(str1,str2); → Passed

Not equals

Assert.assertTrue(str1.equalsIgnoreCase(str2)); → Passed

True

Assert.assertFalse(str1.equals(str2)); → Passed

False

Assert.assertFalse(condition: a>b); → Passed

False

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

equals

Assert.assertNotEquals(b,c); → Passed

Not equals

Assert.assertTrue(condition: c>b); → Passed

true

# JUnit

## Assertions

- 1) Bir class oluşturun: BestBuyAssertions
- 2) <https://www.bestbuy.com/> Adresine gidin farkli test method'lari olusturarak asagidaki testleri yapin
  - o Sayfa URL'inin <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

# JUnit

## 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

# JUnit

## Check Box

Gerekli yapıyi olusturun ve aşağıdaki görevi tamamlayın.

- Verilen web sayfasına gidin.

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

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

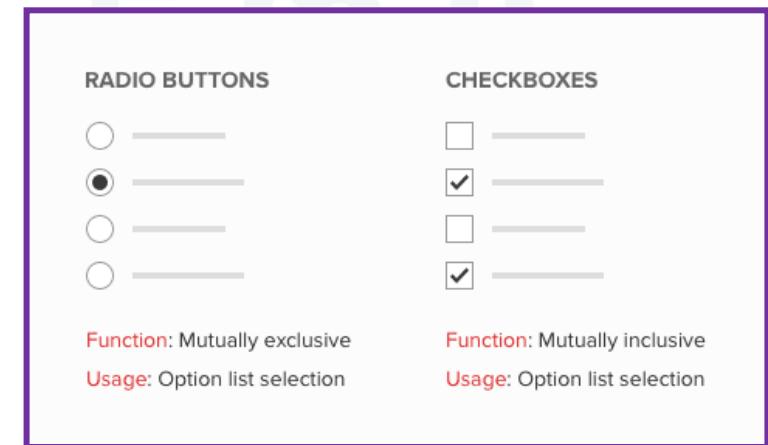
<b>Specialty Food Type</b>
<input type="checkbox"/> GMO-Free
<input type="checkbox"/> Organic
<input type="checkbox"/> USDA Organic
<b>Calories Per Serving</b>
<input type="checkbox"/> 0 Calories
<input type="checkbox"/> Up to 40 Calories
<input type="checkbox"/> 40 to 100 Calories
<input type="checkbox"/> 100 to 200 Calories
<input type="checkbox"/> 200 to 300 Calories
<b>Fat Calories Per Serving</b>
<input type="checkbox"/> 40-100 Calories
<b>Nutrition Facts Per Serving</b>
<input type="checkbox"/> Fat Free (<0.5g)
<input type="checkbox"/> Low Fat (<3g)
<input type="checkbox"/> Free of Saturated Fat (<0.5g)
<input type="checkbox"/> Free of Trans Fat (0g)
<input type="checkbox"/> Cholesterol Free (<2mg)
<input type="checkbox"/> Sodium Free (<5mg)
<input type="checkbox"/> Low Sodium (<140mg)
<input type="checkbox"/> Carbohydrate Free (<0.5g)
<input type="checkbox"/> Sugar Free (<0.5g)
<input type="checkbox"/> Dietary Fiber (>10g)
<input type="checkbox"/> Protein (>10g)
<b>Food Diet Type</b>
<input type="checkbox"/> Vegan
<b>Availability</b>
<input type="checkbox"/> Include Out of Stock

# JUnit

## Radio Buttons

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 olani secin
- e. Sectiginiz radio button'un secili, ötekilerin secili olmadigini test edin



# JUnit

```
@Test  
public void test02(){  
    int a= 10, b=20, c=30;  
    String str1="Ali", str2="ALI";  
  
    Assert.assertEquals(str1,str2); → Failed  
    Not equals  
    Assert.assertNotEquals(str1,str2); → Passed  
    Not equals  
    Assert.assertTrue(str1.equalsIgnoreCase(str2)); → Passed  
    True  
    Assert.assertFalse(str1.equals(str2)); → Passed  
    False  
    Assert.assertFalse(condition: a>b); → Passed  
    False  
    Assert.assertEquals(c, actual: a+b); → Passed  
    equals  
    Assert.assertNotEquals(b,c); → Passed  
    Not equals  
    Assert.assertTrue(condition: c>b); → Passed  
    true
```

# 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

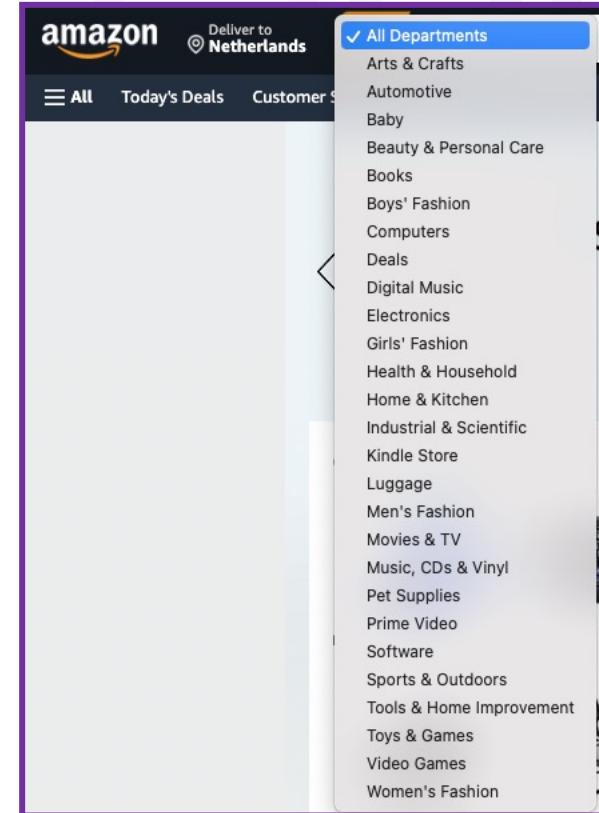
# JUnit

## 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.



# JUnit

## Handle Dropdown

Dropdown'daki opsiyonlardan birini secmek icin 3 islem yapilir.

1- Dropdown webelement'i locate edin

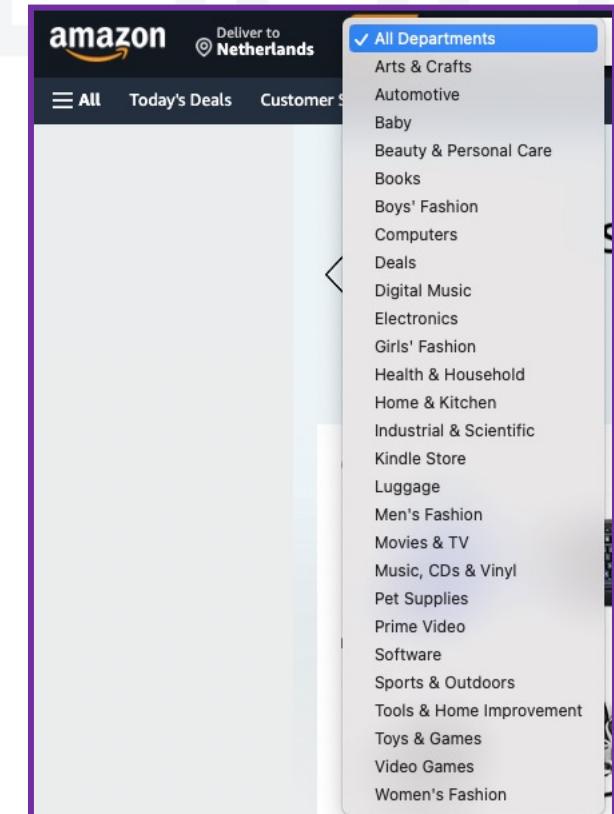
```
WebElement ddm= driver.findElement(By.id("searchDropdownBox"));
```

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");
```



# JUnit

## 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

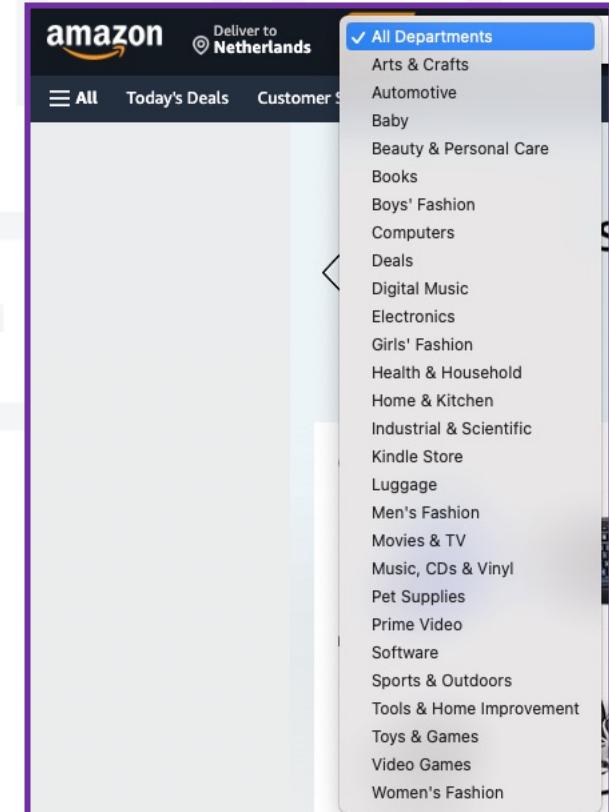
```
select.getFirstSelectedOption()
```

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();
```



# Selenium

## Ders-05

Dropdown Menu

Js Alerts/Iframe/Coklu Window Kullanma

Actions Class

# JUnit

## 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

# JUnit

## Handle Dropdown

- Bir class olusturun: 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

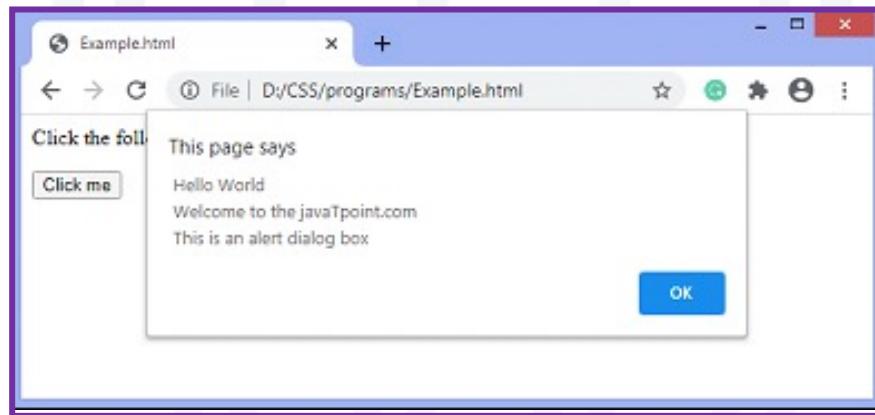
# JUnit

## 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 menusünden 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 çıktığını kontrol edin.

# JUnit

## JS Alerts



### Alert Nedir?

Alert kullanıcıya bir tür bilgi vermek veya belirli bir işlemi gerçekleştirmek 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 işleme gerek yoktur.

#### 2- Js Alerts

Js alerts inspect yapılamaz, ekstra işleme ihtiyaç vardır.

# JUnit

## 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 onaylanıyorsa 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](https://the-internet.herokuapp.com/javascript_alerts)

# JUnit

## 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

# JUnit

3 test method'u olusturup asagidaki gorevi tamamlayin

## 1. Test

- [https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts) adresine gidin
- 1.alert'e tiklayin
- Alert'deki yazinin "I am a JS Alert" oldugunu test edin
- OK tusuna basip alert'i kapatin

## 2.Test

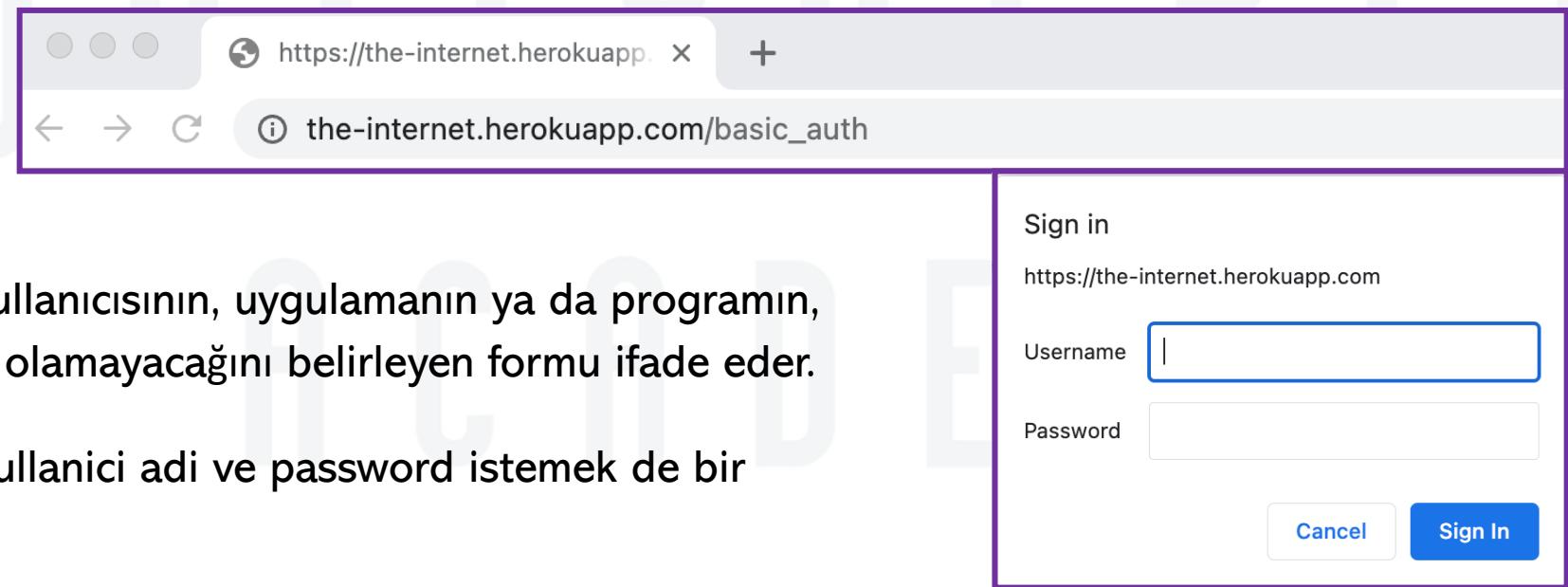
- [https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts) adresine gidin
- 2.alert'e tiklayalim
- Cancel'a basip, cikan sonuc yazisinin "You clicked: Cancel" oldugunu test edin

## 3.Test

- [https://the-internet.herokuapp.com/javascript\\_alerts](https://the-internet.herokuapp.com/javascript_alerts) adresine gidin
- 3.alert'e tiklayalim
- Cikan prompt ekranina "Abdullah" yazdiralim
- OK tusuna basarak alert'i kapatyalim
- Cikan sonuc yazisinin Abdullah icerdigini test edelim

# JUnit

## Basic Authentication



The screenshot shows a web browser window with the URL [https://the-internet.herokuapp.com/basic\\_auth](https://the-internet.herokuapp.com/basic_auth). A modal dialog box is displayed, titled "Sign in". It contains two input fields: "Username" and "Password". Below the fields are two buttons: "Cancel" and "Sign In".

### 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>

# JUnit

## Basic Authentication

The screenshot shows a browser window with the URL [https://the-internet.herokuapp.com/basic\\_auth](https://the-internet.herokuapp.com/basic_auth). A purple box highlights the browser's address bar and the page content. To the right of the browser window, a separate purple-bordered box contains a 'Sign in' form with the URL <https://the-internet.herokuapp.com>. The form has two text input fields: 'Username' and 'Password', both of which are empty. At the bottom of the form are two buttons: 'Cancel' and 'Sign In'.

- 1- Bir class olusturun : BasicAuthentication
- 2- [https://the-internet.herokuapp.com/basic\\_auth](https://the-internet.herokuapp.com/basic_auth) sayfasina gidin
- 3- asagidaki yontem ve test datalarini kullanarak authentication'i yapin

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

Username : admin  
password : admin

- 4- Basarili sekilde sayfaya girildigini dogrulayin

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

# JUnit

## Handle IFrame

HTML kodlarda kullanılan `<iframe>` tag'i bir HTML sayfasinin icerisine baska bir HTML sayfasi gömmek(embed) icin kullanilir.

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

`<iframe>` tag'i icerisinde header ve body bulunur. Bir HTML sayfasinda iframe varsa, HTML kodlar icerisinde birden fazla header veya body olacaktir.



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.

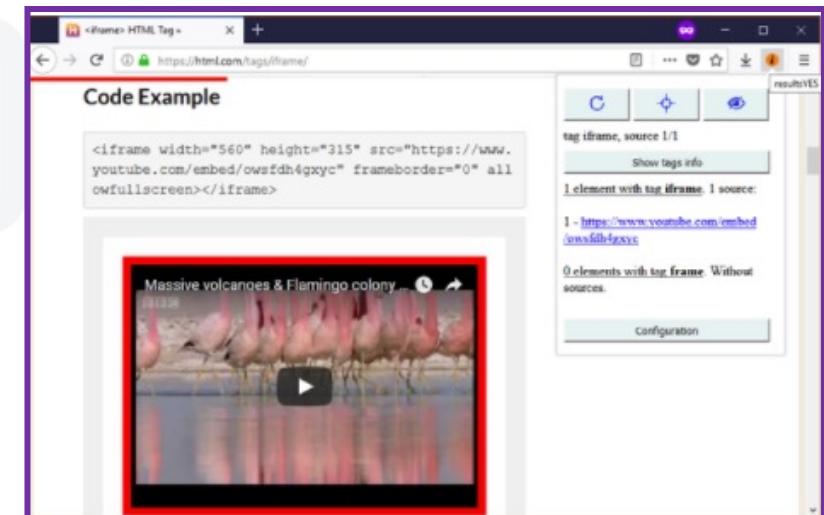
Webdriver'i istenen iframe'e switch yapabilmek için iframe'i driver'a tanıtmak gereklidir. Bu tanıma 3 farklı yolla yapılabilir.

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

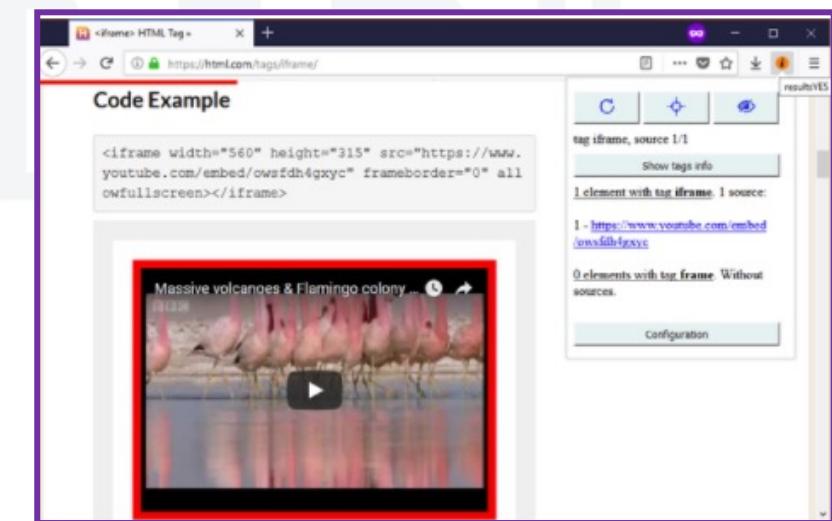
Iframe içerisindeyken oradan cikmak icin 2 yontem kullanilabilir.

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.

# JUnit

## Handle IFrame

- 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



## 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 geçiş yapması gereklidir.

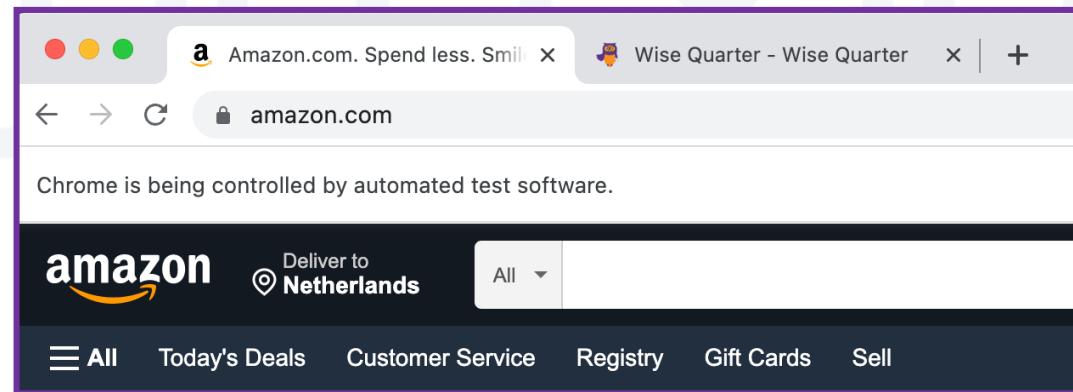
Selenium4 ile yeni gelen bir özellik olarak, test sırasında yeni bir tab veya window açılabilir.

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

Bu method kullanıldığında yeni sayfa/tab driver ile açıldıgından 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);
```

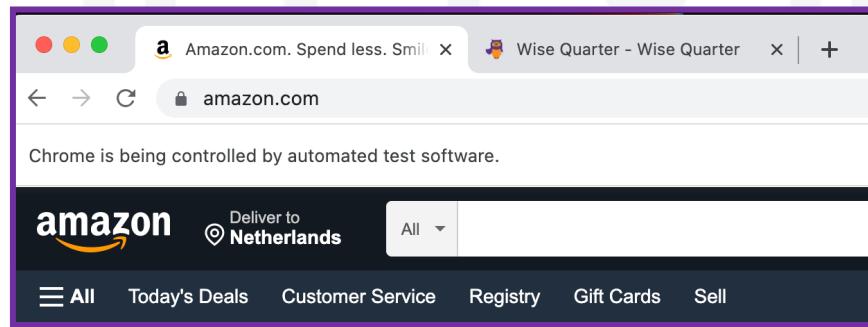
Bir window'a geçiş yapabilmek için o window'un sayfa handle değerine ihtiyaç vardır.

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

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

# JUnit

## Handle Windows



Ikinci sayfa acildiginda o sayfaya gecmek icin window handle degerini driver ilk sayfada iken bulmamiz gereklidir. Bunun icin hazır 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.

# JUnit

## Handle Windows

- 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 dondugunuzu test edin

# JUnit

## Handle Windows

- 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.

# Selenium

## Ders-06

### Actions Class

### File Testleri

Egitmen : Ahmet BULUTLUOZ

# JUnit

## Actions Class

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

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

- Çift tıklama (double click),
  - sürükleme ve bırakma (drag and drop)
  - mouse'u objeye götürme (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 1 :** Action Class'ını 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](https://the-internet.herokuapp.com/context_menu) sitesine gidin
- 3- Cizili alan uzerinde 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 “*Make sure your code lands*” 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

# 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 islev 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'larini 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- Cookies kabul edin
- 3- Yeni hesap olustur butonuna basalim
- 4- Ad, soyad, mail ve sifre kutularina deger yazalim ve kaydol tusuna basalim
- 5- Kaydol tusuna basalim

# JUnit

## Faker Kutuphanesi

Faker class'i testlerimizi yaparken ihtiyac duyduğumuz isim, soyisim, adres vb bilgiler için fake değerler üretmemize imkan tanır.

Faker değerler üretmek için Faker class'ından bir obje üretir ve var olan method'lari kullanırız.

1. "https://facebook.com" Adresine gidin
2. "create new account" butonuna basın
3. "firstName" giriş kutusuna bir isim yazın
4. "surname" giriş kutusuna bir soyisim yazın
5. "email" giriş kutusuna bir email yazın
6. "email" onay kutusuna emaili tekrar yazın
7. Bir şifre girin
8. Tarih için gün seçin
9. Tarih için ay seçin
10. Tarih için yıl seçin
11. Cinsiyeti seçin
12. Isaretlediğiniz cinsiyetin seçili, diğer cinsiyet kutusunun seçili olmadığını 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

# JUnit

## 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

# JUnit

## Genel Tekrar Homework

### Test01 :

- 1- amazon gidin
- 2- Arama kutusunun solundaki dropdown menuyu 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

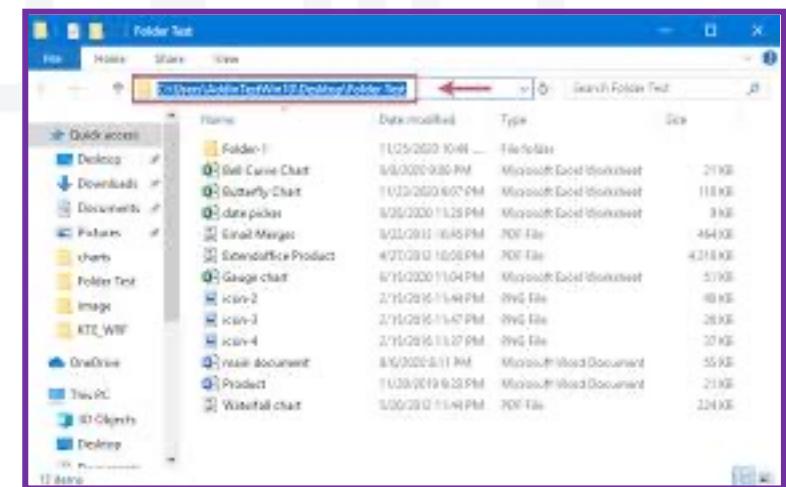
## File Exist

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 olacağından, bir bilgisayarda yazılan dosya yolu başka bilgisayarda çalışmaz.

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



# JUnit

## File Exist

getProperty( ) method'u iki farklı parametre ile çalışabilir.

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

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

Şeklinde oluşturulup kaydedilen dosya yolu dinamik olduğu için kodların çalışacağı tüm 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.

# JUnit

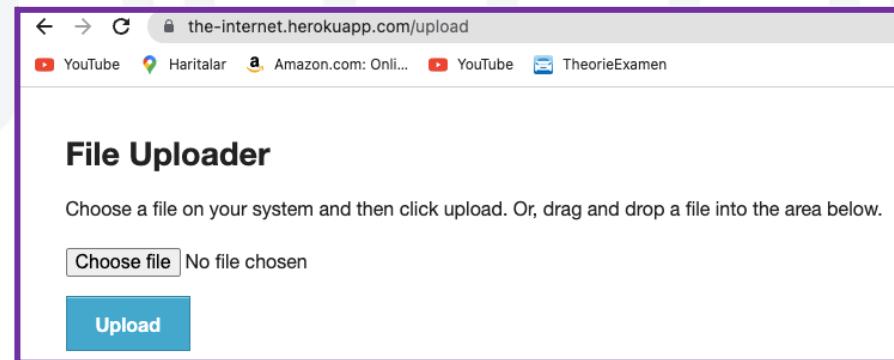
## File Exist

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 gönderin

3- Upload butonunu locate edip tıklayın

# JUnit

## 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.

# Selenium Ders-07

Waits

Cookies

Web Tables

Excel Automation

Egitmen : Ahmet BULUTLUOZ

# 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 yasanın 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 işlem 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.

# JUnit

## Implicitly Waits

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

Selenium tabanlı wait'lerde verilen sure max. beklenme 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.

# JUnit

## Implicitly Wait

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

Webelement 10 saniyeden kısa surede 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.



# JUnit

## Explicitly Waits

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

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

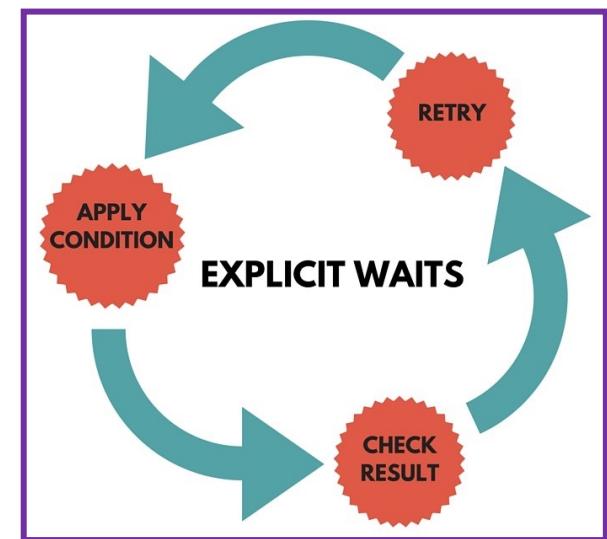
Nadiren karşılaşılan ve daha uzun bekleme süresi gerektiren işlem uygulanan webelementler için explicitly wait kullanılır.

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 beklenecek condition kullanılır. Cunku olmayan bir webelement'in locate edilmesi mümkün olamayabilir.

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





## Explicitly Waits / Expected Conditions

- |                                      |  |
|--------------------------------------|--|
| 1.alertIsPresent( )                  | 10.textToBePresentInElement( )         |
| 2.elementSelectionStateToBe( )       | 11.textToBePresentInElementLocated( )  |
| 3.elementToBeClickable( )            | 12.textToBePresentInElementValue( )    |
| 4.elementToBeSelected( )             | 13.titleIs( )                          |
| 5.frameToBeAvailableAndSwitchToIt( ) | 14.titleContains( )                    |
| 6.invisibilityOfTheElementLocated( ) | 15.visibilityOf( )                     |
| 7.invisibilityOfElementWithText( )   | 16.visibilityOfAllElements( )          |
| 8.presenceOfAllElementsLocatedBy( )  | 17.visibilityOfAllElementsLocatedBy( ) |
| 9.presenceOfElementLocated( )        | 18.visibilityOfElementLocated( )       |

# JUnit

## Explicitly Waits

1. Bir class olusturun : EnableTest
2. Bir metod olusturun : isEnabled()
3. [https://the-internet.herokuapp.com/dynamic\\_controls](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

# JUnit

## 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 tiklayin
- 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 kapatin
- 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 bilgisayarınızdaki diğer dosyaları görüntüleyemez.

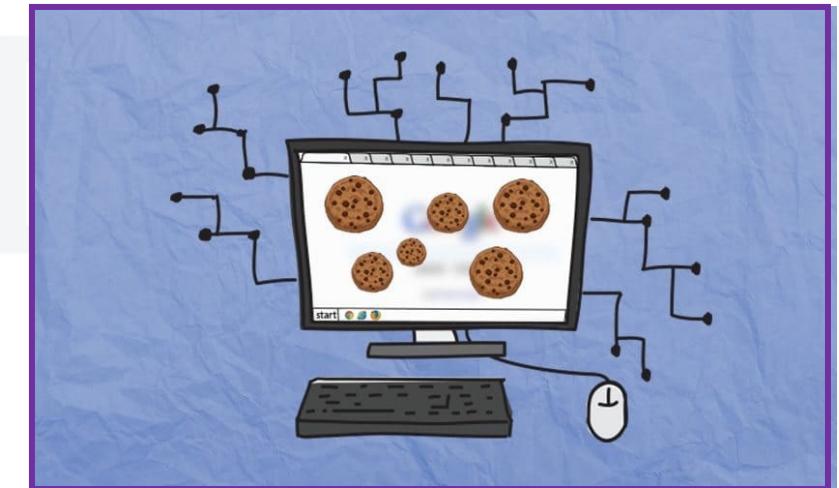
# JUnit

## 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 yazılınır. Oturum sona erdiğinde oturum çerezleri otomatik olarak silinir.

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

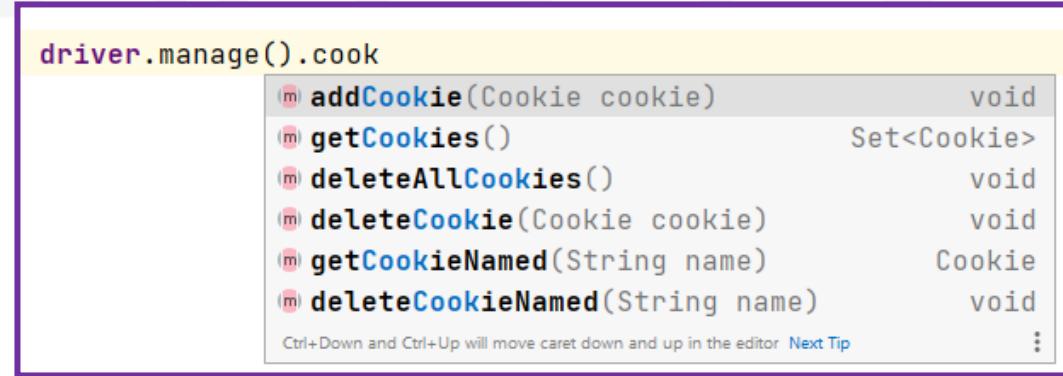


[Üçüncü taraf çerezler](#) daha sıkıntılıdır. Bunlar, genellikle kullanıcıların halihazırda gezindiği web sayfalarındaki reklamlarla bağlantılı olduklarından bu sayfalardan farklı web siteleri tarafından oluşturulur.

# JUnit

## Cookies

Selenium ile cookies otomasyonu yapabiliyoruz.



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

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

# JUnit

## Cookies

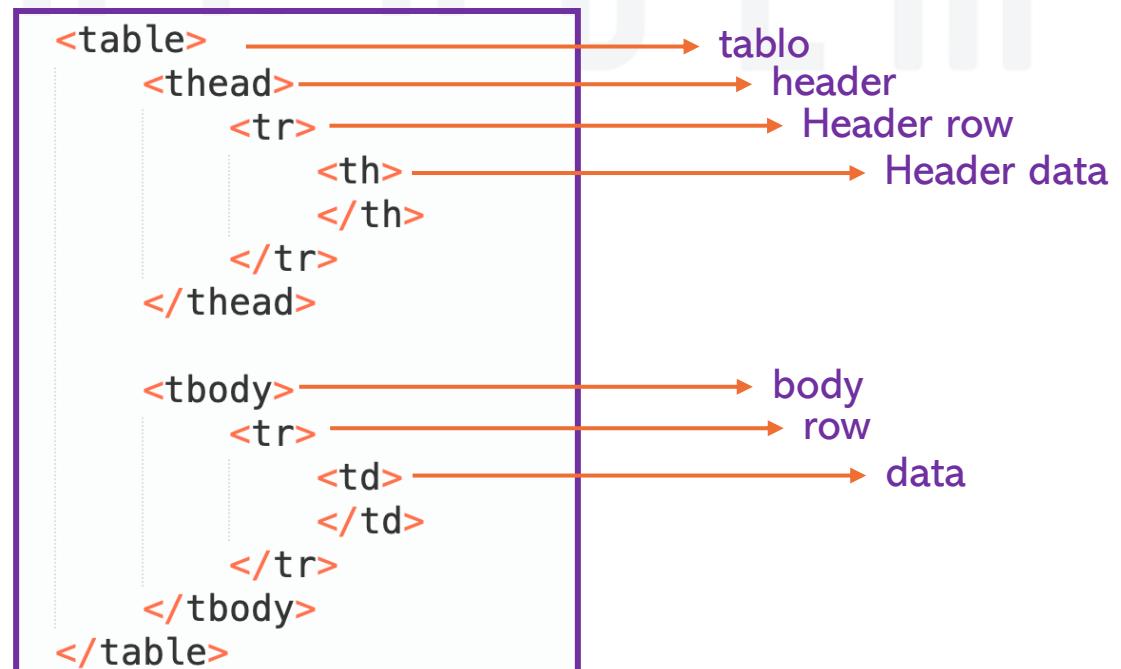
Yeni bir class olusturun : cookiesAutomation

- 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	 



# JUnit

## Web Tables

- 1."<https://www.amazon.com>" adresine gidin
- 2.Sayfanin en altina inin
- 3.Web table tum body'sini yazdirin
- 4.Web table'daki satir sayisinin 7 oldugunu test edin
- 5.Tum satirlari yazdirin
6. Web table'daki sutun sayisinin 13 olduğunu test edin
7. 5.sutunu yazdirin
- 8.Satir ve sutun sayisini parametre olarak alip, hucredeki bilgiyi döndüren bir method olusturun

# **Selenium**

## **Ders-08**

### **Web Tables**

### **Excel Automation**

**Egitmen : Ahmet BULUTLUOZ**

# JUnit

## 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 yazdirlisin

# JUnit

## Excel Automation

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

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 içerisinde veri varsa satırları sayar. Default olarak, Java perspektifinden satır sayısı 0'dır

**Cells (hucre)** 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	Algiers	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	Australya	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	Belçika	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



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>
```

# JUnit

## 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 inputStream objesini girelim
10. WorkbookFactory.create(inputStream)
11. Worksheet objesi olusturun workbook.getSheetAt(index)
12. Row objesi olusturun sheet.getRow(index)
13. Cell objesi olusturun row.getCell(index)

# JUnit

## 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

# JUnit

## Write Excel

- 1) Yeni bir Class olusturalim WriteExcel
- 2) Yeni bir test method olusturalim writeExcelTest()
- 3) Adimlari takip ederek Sayfa1'de 1.satira kadar gidelim
- 4) 5.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 kapatelim

# JUnit

## Get Screenshot / Tum 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);
```

# JUnit

## 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 calistiralim 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);
```

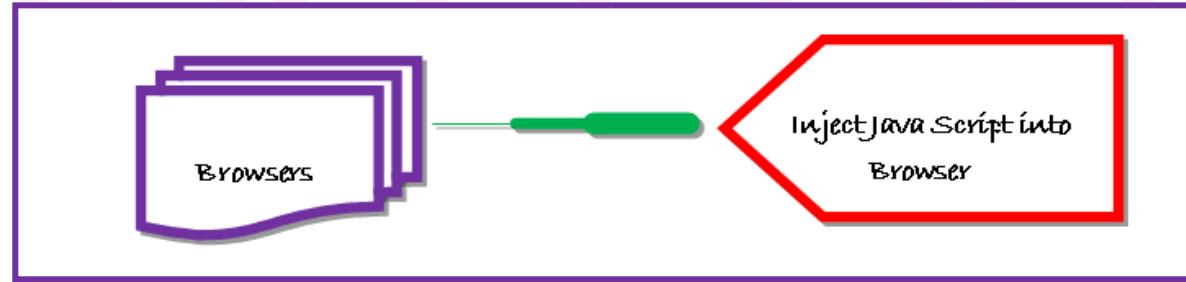


Yeni bir class olusturun : amazonNutellaSearch

- 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

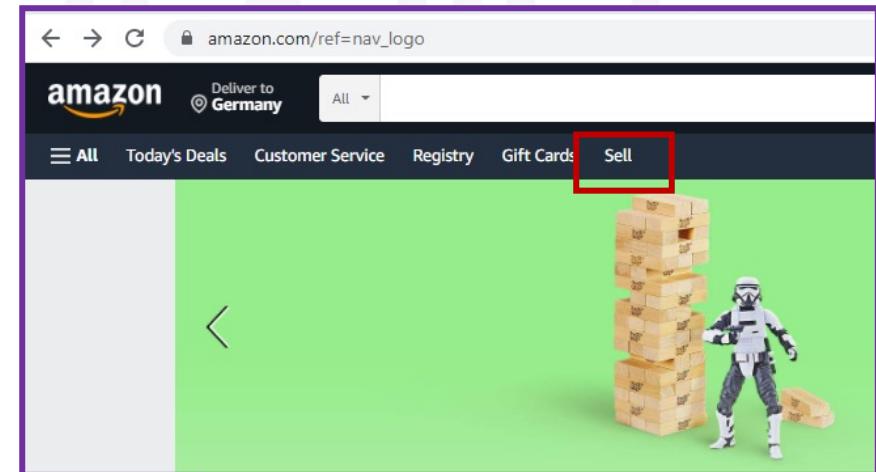
1.Adım : JavascriptExecutor kullanmak için ilk adım olarak driver'imizi JavascriptExecutor'a cast edelim

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

2.Adım : Kullanicigimiz 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 icin

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

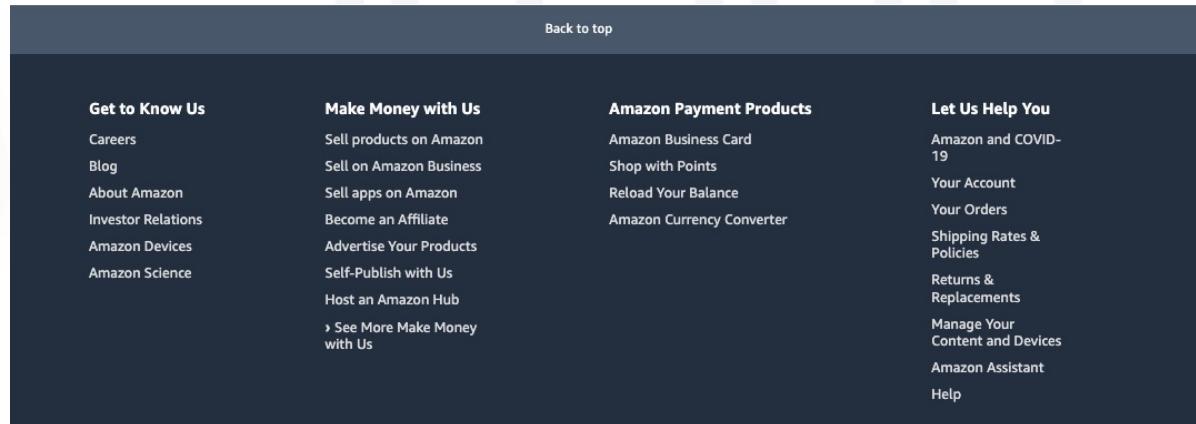
# JUnit

## JS Executors

Istelenen Webelement'e kadar asagi inmek icin

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

Ornek : amazon anasayfasina gidin alt kisimda bulunan “Back to top” butonuna kadar asagi inin ve bu butona click yapin



JS alert kullanarak uygulama'dan kullaniciya mesaj vermek icin

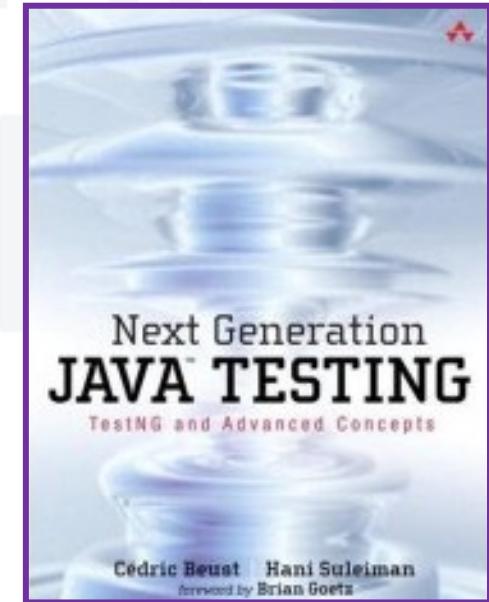
```
jse.executeScript("alert('yasasinnnn');");
```

# 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 calistirabilme
- 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.

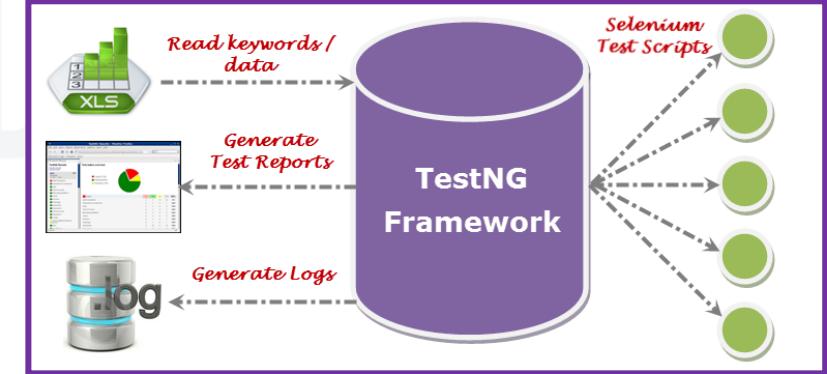
Test caseleri siralama ozelligi (**priority**) ve test caselerin birbirine bagimlilik (**dependsOnMethod**) bize testleri organize etmekte yardım eder.

Yazılan test case'lerin paralel olarak çalıştırılmasına, birden fazla browser kullanmasına imkan tanır.

Error mesajlarının daha detaylı bir şekilde gösterilmesini sağlar.

Paralel ve Cross-Browser Test yapmamiza imkan tanır

Kullanisli HTML veya xml raporlari olusturma imkani vardir



# 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

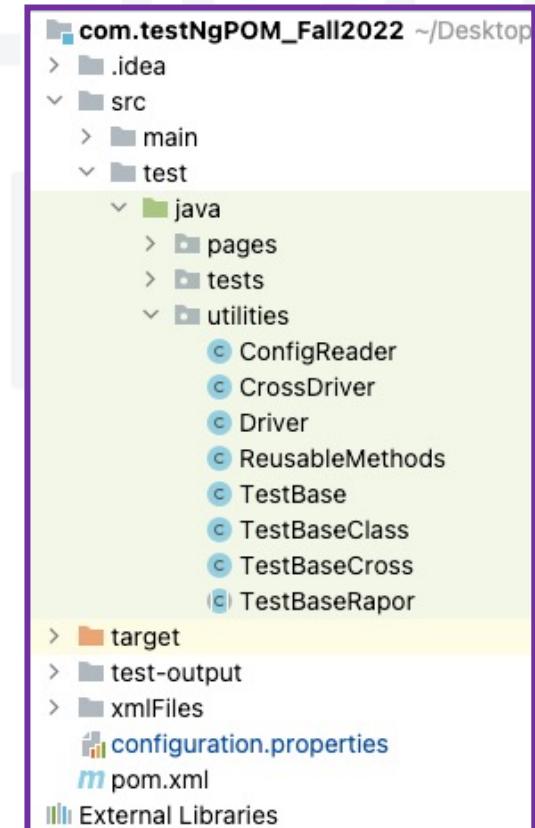
bir yapı olusturulacaktır.

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.

# TestNG

## @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

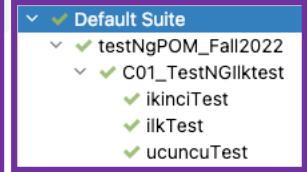
TestNG test method'larini alfabetik siraya gore calistirir.

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

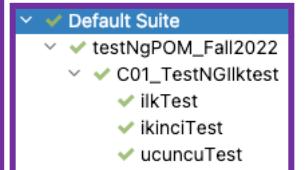
Kurallar:

- 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");  
}
```



# TestNG

## Priority

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 görüntünlendiğini (isDisplayed()) test edin
  - Search Box 'in erisilebilir olduğunu test edin (isEnabled())
  - wrongTitleTest => Sayfa basliginin “youtube” olmadığını doğrulayın

# TestNG

## dependsOnMethods

dependsOnMethods test method'larinin siralamasi ile ilgili degildir. Siralamayı priority ile tanımlayabiliriz.

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 yapildiysa ilk urunu tiklayin ve fiyatinin \$16.83 oldugunu test edin

# TestNG

## Assertions

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

TestNG testleri yaparken bize daha fazla tercih imkani verir.

- Junit'deki gibi ilk failed olan assertion'da çalışmayı durdurmak
- Biz raporla diyene kadar tüm testleri gerçekleştirip, raporla deyince kaç testin FAILED olduğunu raporlayıp, sonra çalışmayı 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();  
    }  
}
```

# TestNG

## Hard Assert

JUnit'te Öğrendiğimiz Assertion ile ayndır. TestNG'de soft assertion da olduğundan, ayristirmak için bu isim kullanılmamıştır.

JUnit'ten bildigimiz üzere kullanabileceğimiz 3 çeşit hard assertion turu vardır

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

Hard assertion herhangi bir assertion FAILED olursa, test method'nun çalışmasını durdurur ve kalan kodları yürütmez (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 istediğiniz bolumde yapılan tüm testleri raporlar

Eğer assertion'lardan FAILED olan varsa raporlama yapılan satırdan sonrasını calistirmaz.



SoftAssert class'indaki method'lari kullanmak icin kullanabilmemiz için object olusturmamız 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();
```

# TestNG

## 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.

# TestNG

## Soft Assert (Verification)

Yeni bir Class Olusturun : CO3\_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 menusünden Eurozone'u secin
9. soft assert kullanarak "Eurozone (euro)" secildigini test edin
10. soft assert kullanarak DropDownList listesinin su seçenekleri 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)"

Test**NG**

Unityverse  
Academy

[unityverseacademy.com](http://unityverseacademy.com)

Test**NG**

Unityverse  
Academy

[unityverseacademy.com](http://unityverseacademy.com)

Test**NG**

Unityverse  
Academy

[unityverseacademy.com](http://unityverseacademy.com)

Test**NG**

Unityverse  
Academy

[unityverseacademy.com](http://unityverseacademy.com)

Test**NG**

Unityverse  
Academy

[unityverseacademy.com](http://unityverseacademy.com)