

Selenium

Ders-01

Selenium Giriş

WebDriver Method'ları

WebElements

# Software Testing Nedir ?

Software testing var olan veya geliştirilmekte olan bir uygulamanın, tasarım asamasında planlanan özellikleri taşıyıp taşımadığının belirlenmesi için yapılan faaliyetlerin bütünüdür.

Software testi için tasarım asamasında 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 geliştirme sürecinde developer'lar bir feature için kod yazmaya başladıklarında, tester'larda o feature'i analiz ederek acceptance criteria çerçevesinde expected result'leri tespit etmeli, yazılımın bu ihtiyaçları karşıladığından emin olmak için positive ve negative test senaryoları oluşturmali ve bu testleri otomasyonla yapacak test case'leri oluşturmali.



# Software Testing Neden Onemlidir ?

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

Ayrica developer'larin user case'den anladiklari ile end – user'larin kullanim aliskanliklari her zaman uyusmayabilir.

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

Kullanicinin beklentilerini karsilamayan uygulamalar basarisiz olur.



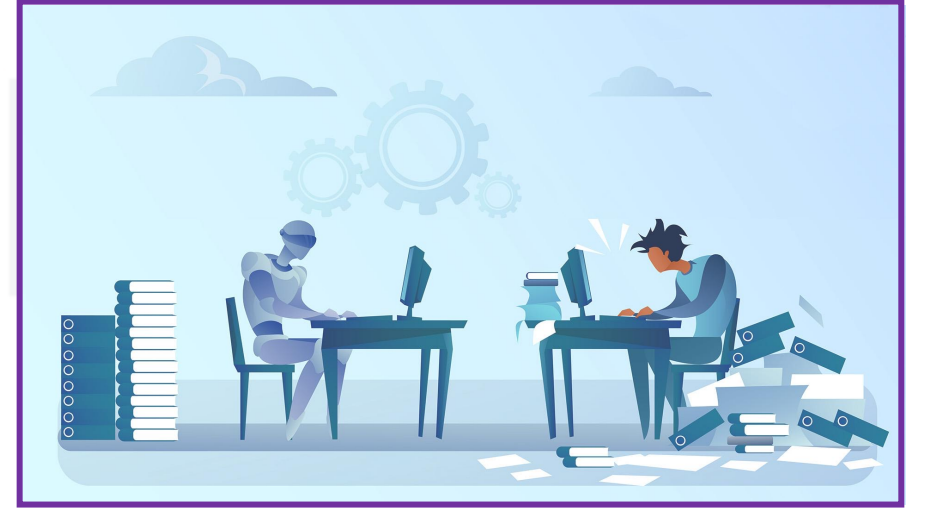
- Urunun end-user kullanimina hazir oldugundan emin olmak
- End-user tarafından karsilasilan sorunlarda duzeltme ve yeniden yapma maliyetlerini azaltmak
- Uygulamanin marketteki algisinin ust seviyede olmasini saglamak
- Sonradan eklenen islevlerin eski islevleri bozmadigindan emin olmak icin software testing yazilim gelistirme surecinin vazgelcilmez bir parcasi olmustur.

# Manual Software Testing Nedir ?

Manual testing, uygulamanin planlanan sonuclara uygun calisip calismadigini hic bir otomasyon araci kullanmadan, bir end-user gibi test edilmesidir.

Ancak insan gucuyle yapilan bu testler icin hem cok fazla insan gucune ve zamana ihtiyac duyulur hem de insani ozelliklerimizden dolayi testlerde yanlisliklar yapılabilir.

Zaman ve ihttiyac duyulan insan gucunu azaltmak, testler calistirilirken ortaya cikabilecek hatalari minimum'a indirmek icin test otomasyonu gereklidir.



Manual tester'lar uygulama uzerinde cok fazla zaman harcadiklari ve her adimi manual yaptiklari icin uygulama bilgileri daha iyidir.

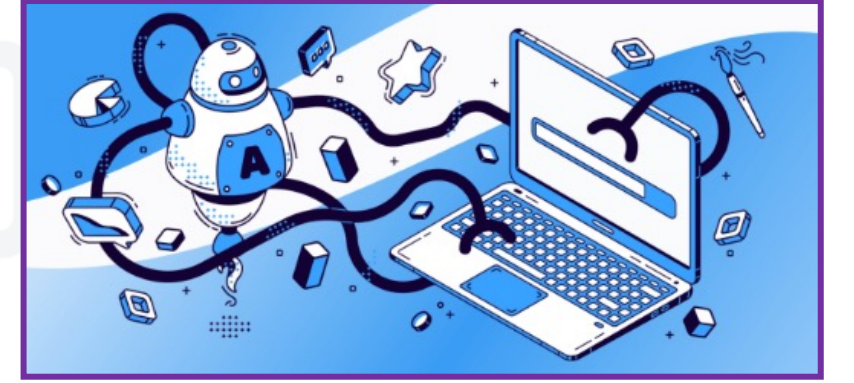
Automation tester'lar uygulamayi daha iyi anlamak ve sistem ihtiyaclarini gormek icin baslangicta bir kac kez manual test yapip sonra otomasyon yapmalidir.

# Test Otomasyonu Nedir ?

Test otomasyonu, insan gucu ile klavye ve mouse kullanilarak yapilabilecek bir yazilim testinin, bir otomasyon araci kullanilarak kodlar araciligi ile yapılmasıdır.

Test otomasyon sayesinde

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



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

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

# Automation & Manual Testing

Yandaki kod sizce nedir ?

A- Test Case

B- Manuel tester icin test adimlari

C- Otomasyon ile test yapan kodlar

Feature: US1010 herokuapp Delete testi

@heroku @sirali @pr1

Scenario: TC15 herokuapp'dan delete butonu calismali

Given kullanıcı "herokuappUrl" anasayfasında

And add element butonuna basar

And kullanıcı 3 sn bekler

Then Delete butonu görünür oluncaya kadar bekler

And Delete butonunun görünür olduğunu test eder

Then Delete butonuna basar

And Delete butonunun görünmediğini test eder

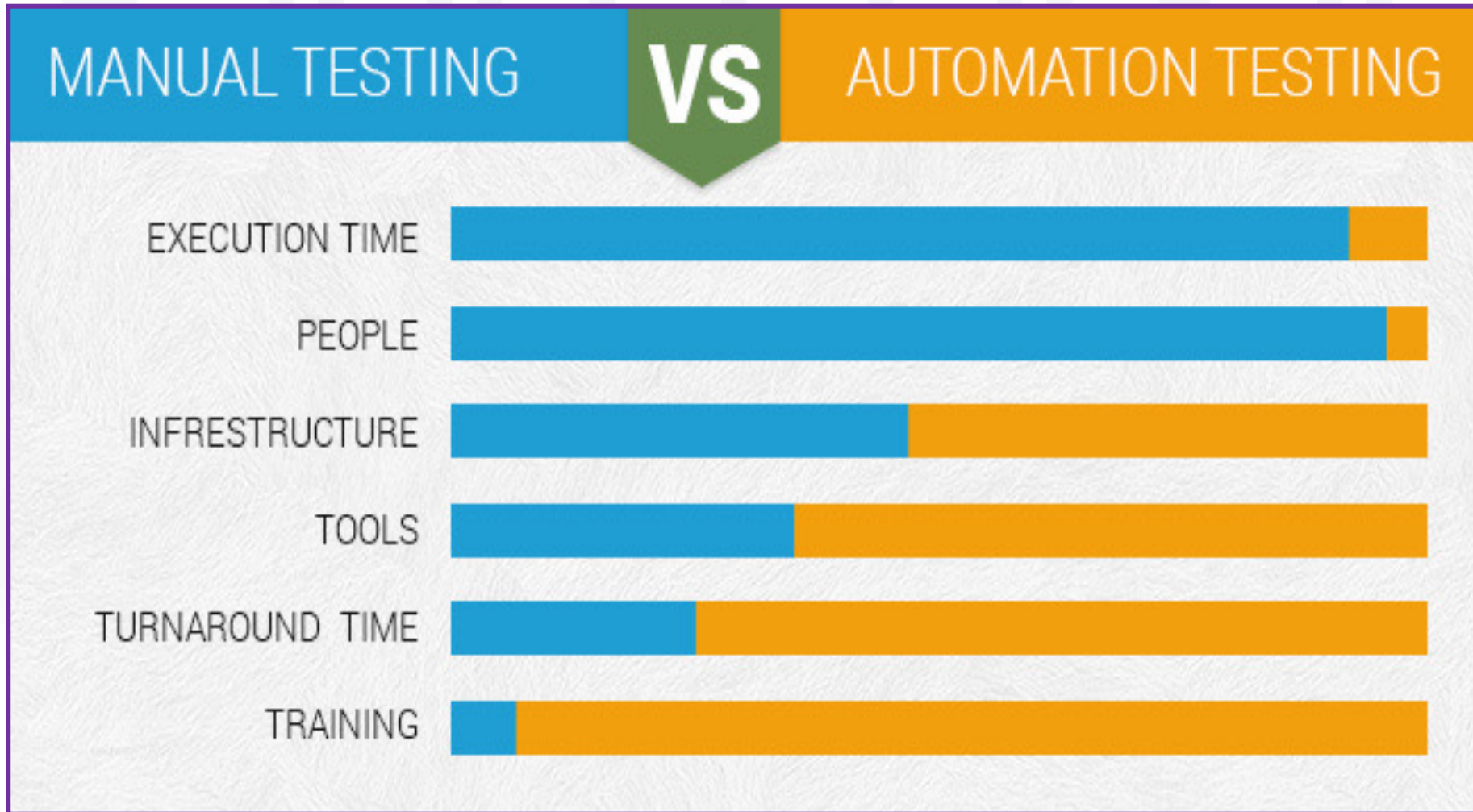
And sayfayı kapatır

Test otomasyonu sayesinde herkesin anlayacağı test case'ler oluşturabilir, daha kısa sürede, daha az insan kaynağı ile testlerinizi gerçekleştirebilir, istediğiniz raporları otomatik olarak oluşturabilirsiniz.

Manuel Test sayesinde kod bilgisi olmasa bile insanlara test yaptırabilir, temel test ihtiyaçlarınızı karşılayabilirsiniz.

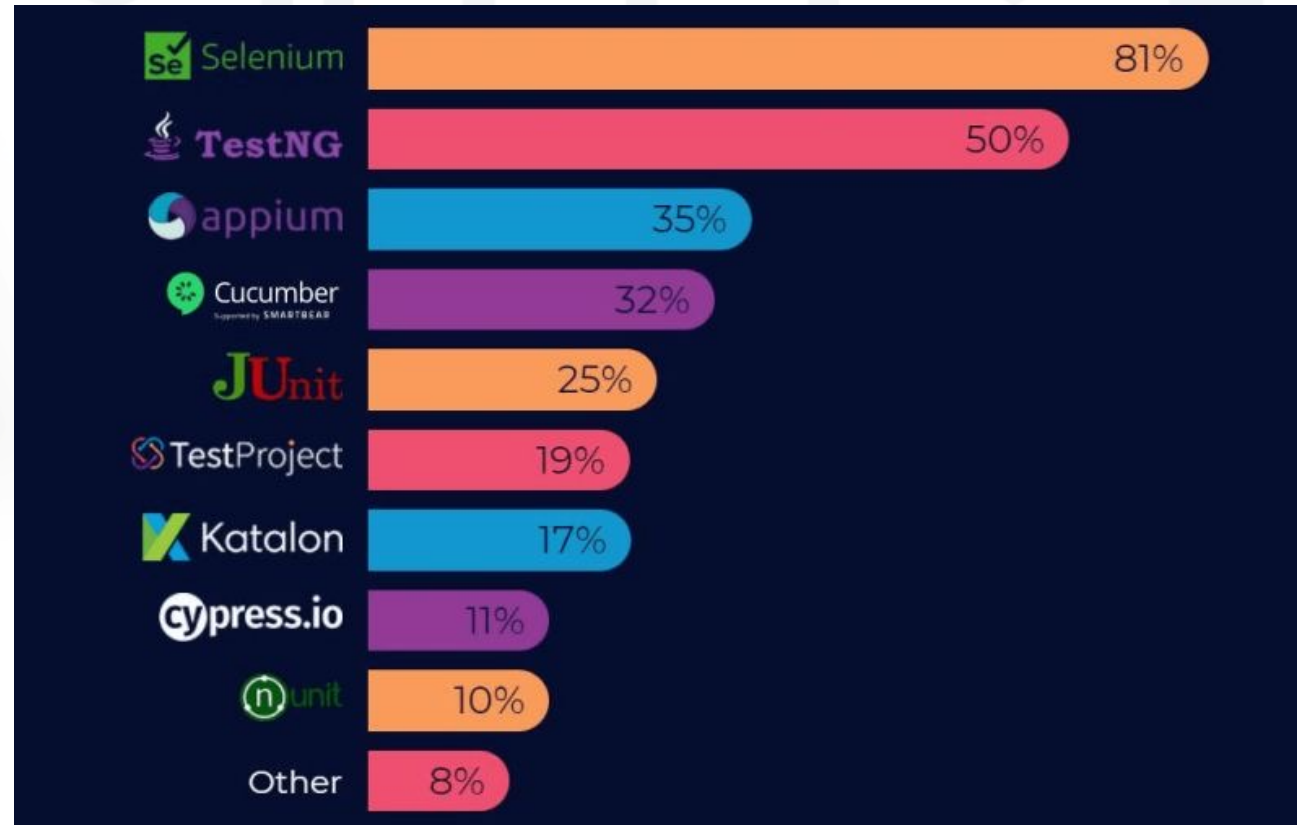


# Automation & Manual Testing



# En Cok Kullanilan Tool'lar

En cok kullanılan test otomasyon tool'lari





# Selenium Nedir ?

## About Selenium

Selenium is a suite of tools for automating web browsers.

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

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

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

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

# Selenium Nedir ?

**Selenium automates browsers. That's it!**  
What you do with that power is entirely up to you.

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

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

Selenium browser'lari otomasyonla calistirir, bu otomasyon gucu ile ne yapacaginiz tamamen size kalmistir.

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

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

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

# IntelliJ Nedir ?

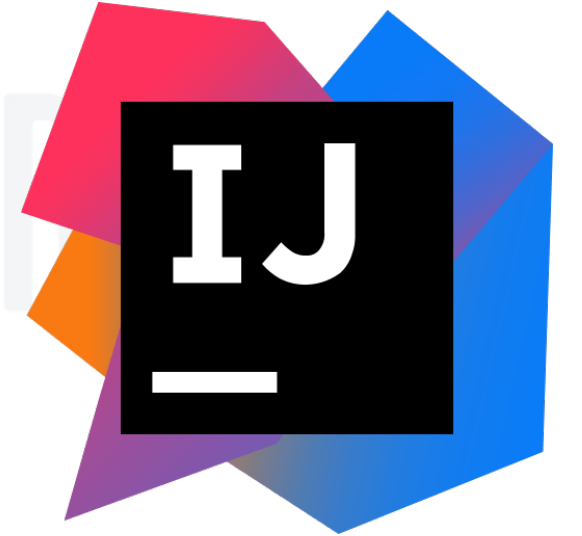
High Level programlama dilleri calisabilmek icin derleyicilere ihtiyac duyarlar.

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

IntelliJ üretkenliği en üst düzeye taşıyacak akıllı kodlama yardımı, kod tamamlama ve ergonomic tasarım gibi özelliklerle kod yazimini sadece verimli değil, aynı zamanda keyifli hale getirmiştir.

Bir çok framework ve plugin ile çalışma imkanı sağlar.

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



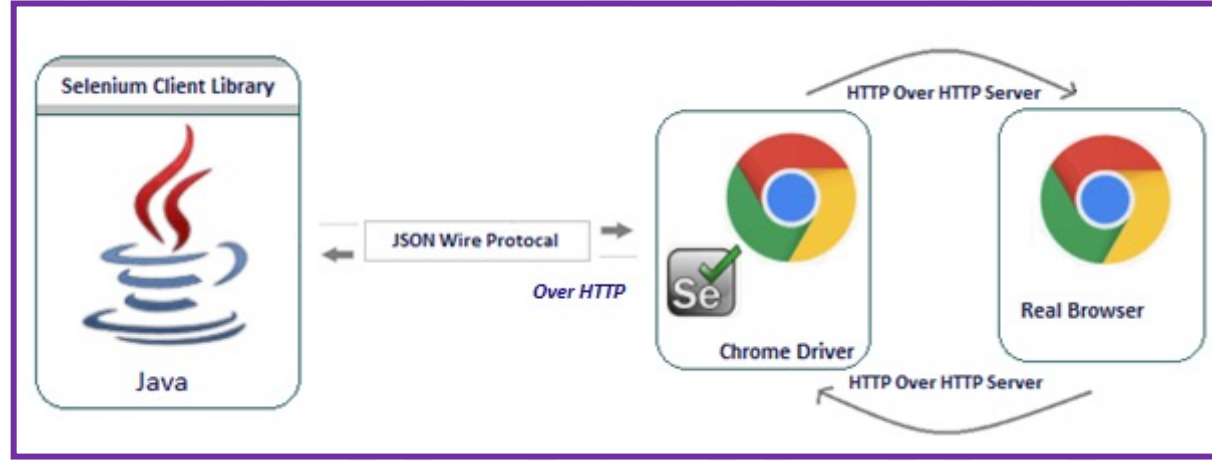
# Selenium Bileşenleri

**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 Nasıl Çalışır ?



Selenium test otomasyonunu WebDriver ile gerçekleştirir.

Java ile yazdığımız kodlar ile kullanılacak browser'a uygun bir webDriver objesi oluşturulur.

Selenium kullanarak WebDriver class'ından oluşturulan driver objesi bizim elimiz, gözümüz gibi çalışır. Gönderildiği web sayfasında klavye ve mouse ile yapılabilecek işlemleri yapar, elementlere tıklama, yazı gönderme, elementler üzerindeki yazıları alma gibi pek çok işlemi gerçekleştirir. Elde ettiği sonuçları Java kodlarının olduğu 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.)

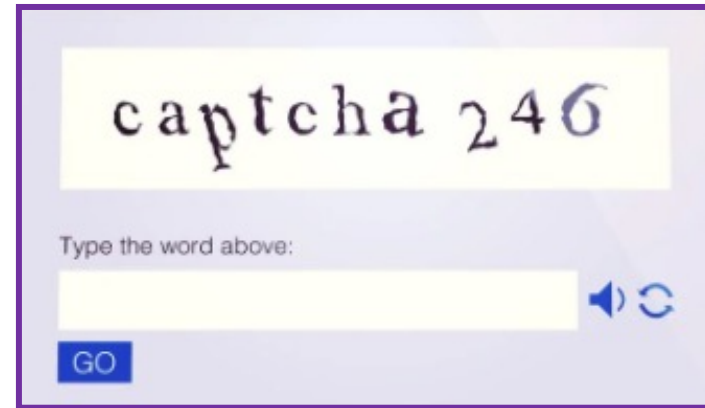
Programlama bilgisi gerektirir (Biz Java biliyoruz)

Yalnızca web tabanlı uygulamaları test eder

Profesyonel desteğe sahip değil

performans testleri yapamaz

Captcha'yi aşamaz (diğer tüm otomasyon araçları gibi)





# Framework Nedir ?

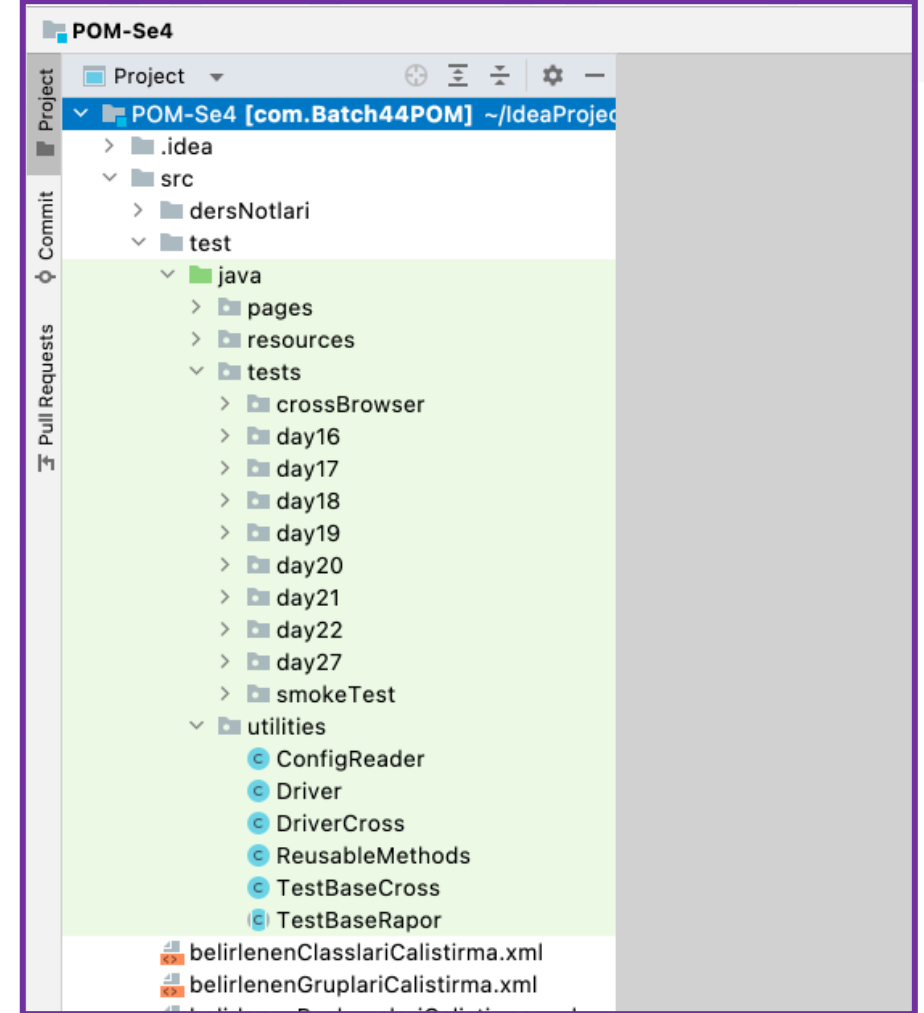
Framework, üzerine kodlarımızı yazarak projelerin oluşturulabileceği bir yapıdır.

Test otomasyonu yaparken herseye sıfırdan başlayıp, herseyi sıfırdan kurgulayıp oluşturmak yerine,

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

Framework, test yapmak için kullandığımız tüm enstrümanları birleştirir.

Framework ile UI, API ve Database testleri yapılandırılabilir.



# Jar Dosyaları ile Selenium Kurulumu

- 1) <https://www.selenium.dev/downloads/> adresine gidin
- 2) Selenium Client & WebDriver Language Bindings altında Java driver'ini download edin
- 3) Browsers altında Chrome documentation linkini tıklayalım  
  
Chrome'un kendi sayfasına gidip Current stable release'i tıklayıp size uygun olanı download edin  
  
İndirilen surum ile bilgisayarınızdaki Chrome browser surumunun aynı olduğundan emin olun
- 4) src altında resources director'si oluşturun
- 5) Bu klasör altında drivers ve libraries klasörleri oluşturun
- 6) İndirdiğimiz chromedriver'i drivers klasörüne, selenium-java dosyasını ise libraries klasörüne çıkartın
- 7) IntelliJ 'de yeni project / package / class oluşturalım ve class içinde main method oluşturun
- 8) File/Project Structure/Modules/Dependencies kısmından jar dosyalarını yükleyin

# WebDriver Objesi Olusturma

Selenium jar dosyalari ile projeye eklendiğinde, kullanmak istenen tum browser'larin driver'larinin da projeye eklenmesi gerekmektedir.

Kullanilacak browser'a ait driver projeye eklendikten sonra, her class'da bilgisayardaki browser'i yönetecek bir WebDriver objesi olusturulur ve o obje yardimiyla WebDriver Class'indaki hazır method'lar kullanılabilir.

WebDriver objesi olusturmak ve objeye kullanılacak browser'a uygun değeri atamak için main method icerisinde

1) Java'daki `setProperty("webdriver.chrome.driver", "driverPath");` ile sistem ayarlari yapilir.

`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 olusturulur ve istenen browser'a uygun değeri atamasi yapilir.

# WebDriver Objesi Kullanma

Selenium ile otomasyon yapabilmenin ilk adimi WebDriver Class'ından obje olusturmaktır.

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

public class C01_DriverMethods {

    public static void main(String[] args) throws InterruptedException {

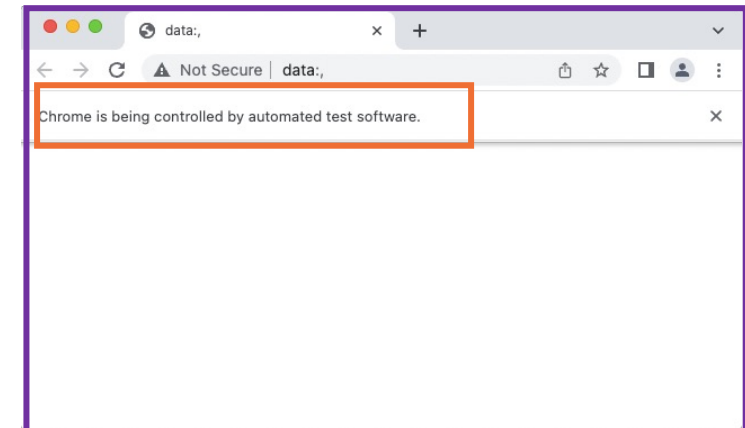
        System.setProperty("webdriver.chrome.driver", "src/resources/chromedriver");

        WebDriver driver= new ChromeDriver();
```

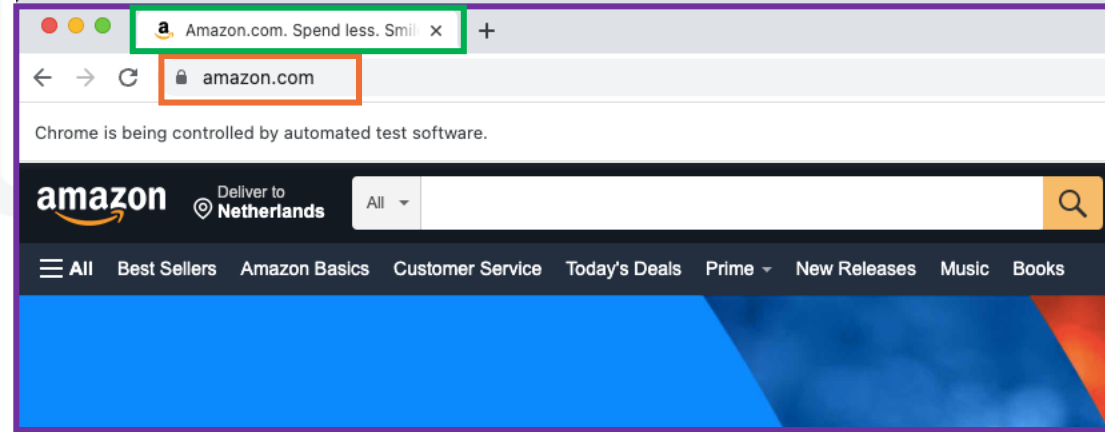
İlgili ayarlari yapip bir driver objesi olusturdugumuzda, Selenium bu driver objesi sayesinde otomasyon yapabilecegimiz bir browser acar.

Bu browser'un Selenium tarafından kontrol edildiği yazilidir.

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();`

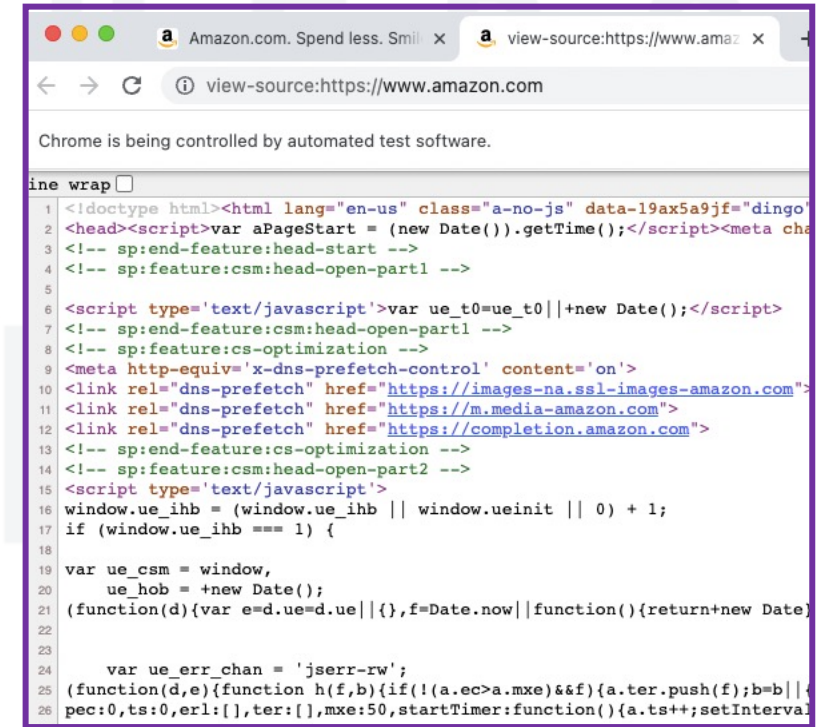
Test sirasinda birden fazla window acilmissa, tumunu kapatir.

# driver.get...( ) Method'ları

## 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.



## 7- driver.getWindowHandle()

`CDwindow-8C07925B8CBA4C8EF3039F660C30DDA1`

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

## 8- driver.getWindowHandles()

Test sırasında driver birden fazla window açıysa, 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 anlaşılabilir olması, testin kısa olmasından önemlidir.

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

// url test yapalım

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 içerdiği test edilmek isteniyorsa, expected ve actual değerler kaydedilip, bir if else bloğu içerisinde istenen test yapıp, sonuç yazdırılabilir.

# WebDriver Method'lari

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. Sayfayi 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 sağlayabilmesidir.

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();`

İçinde olunan web sayfasını yeniler.

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

1. Yeni bir Class olusturalim.CO5\_NavigationMethods
2. Youtube ana sayfasina gidelim . <https://www.youtube.com/>
3. Amazon soyfasina gidelim. <https://www.amazon.com/>
4. Tekrar YouTube'sayfasina donelim
5. Yeniden Amazon sayfasina gidelim
6. Sayfayi Refresh(yenile) yapalim
7. Sayfayi kapatalim / Tum sayfalari kapatalim

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

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

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

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

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

Acılan driver'i tam sayfa yapar.

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

# driver.manage( )... Method'ları

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

Acilan driver'i onceden belirlenmis standart buyukluklere getirir.

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

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

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

Acilan driver'in bulunduгу pozisyonu ve boyutlarini döndürür.



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

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 sayfayi maximize yapin
6. Sayfanin konumunu ve boyutlarini maximize durumunda yazdirin
7. Sayfayi fullscreen yapin
8. Sayfanin konumunu ve boyutlarini fullscreen durumunda yazdirin
9. Sayfayi kapatın

# driver.manage( )... Method'ları

1. Yeni bir Class olusturalim.CO7\_ManageWindowSet
2. Amazon soyfasina gidelim. <https://www.amazon.com/>
3. Sayfanin konumunu ve boyutlarini yazdirin
4. Sayfanin konumunu ve boyutunu istediginiz sekilde ayarlayin
5. Sayfanin sizin istediginiz konum ve boyuta geldigini test edin
8. Sayfayi kapatın

# WebDriver Method'ları

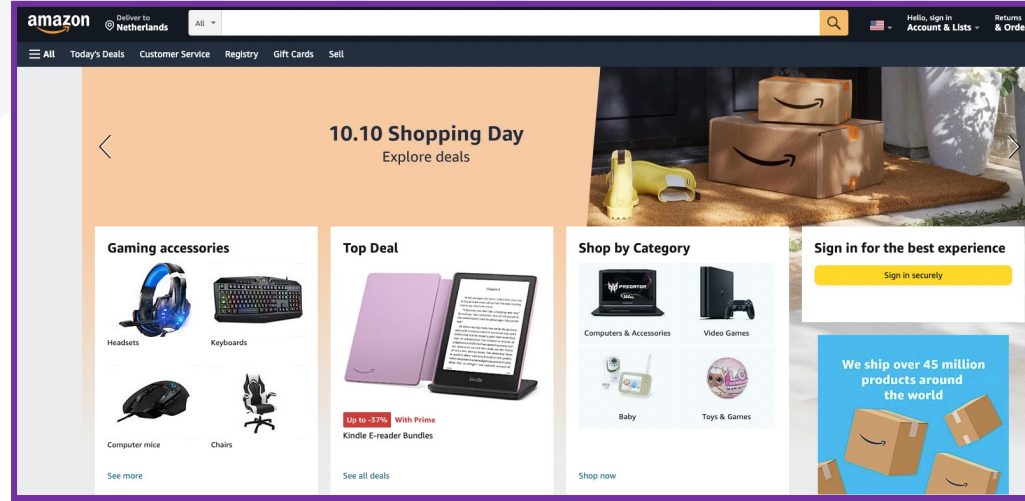
- 1.Yeni bir class olusturalim (Homework)
- 2.ChromeDriver kullanarak, facebook sayfasina gidin ve sayfa basliginin (title) “facebook” oldugunu dogrulayin (verify), degilse dogru basligi yazdirin.
- 3.Sayfa URL'inin “facebook” kelimesi icerdigini dogrulayin, icermiyorsa “actual” URL'i yazdirin.
- 4.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

# WebDriver Method'ları

1. Yeni bir class oluşturun (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, değilse doğru URL'yi yazdırın
- 11.Sayfayı kapatın

# WebElements

Bir web sayfasında kullanılan herseye web element denir.



Her web element farkli ozelliklerde olur. Link, acilir menu, button gibi etkilesimli web elementler oldugu gibi resim, background gibi etkilesimsiz web elementler de olur.

Gorunen her web element aslinda developer'lar tarafından yazilan bir HTML kodun gorsellestirilmis halidir.

Selenium webDriver gorsel elementleri degil, HTML kodlari kullanir.

Otomasyon sirasinda kullanilmak istenen web elementler HTML kodlari kullanilarak unique olarak webDriver'a tarif edilmelidir.

# WebElements



```
<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">
```

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

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



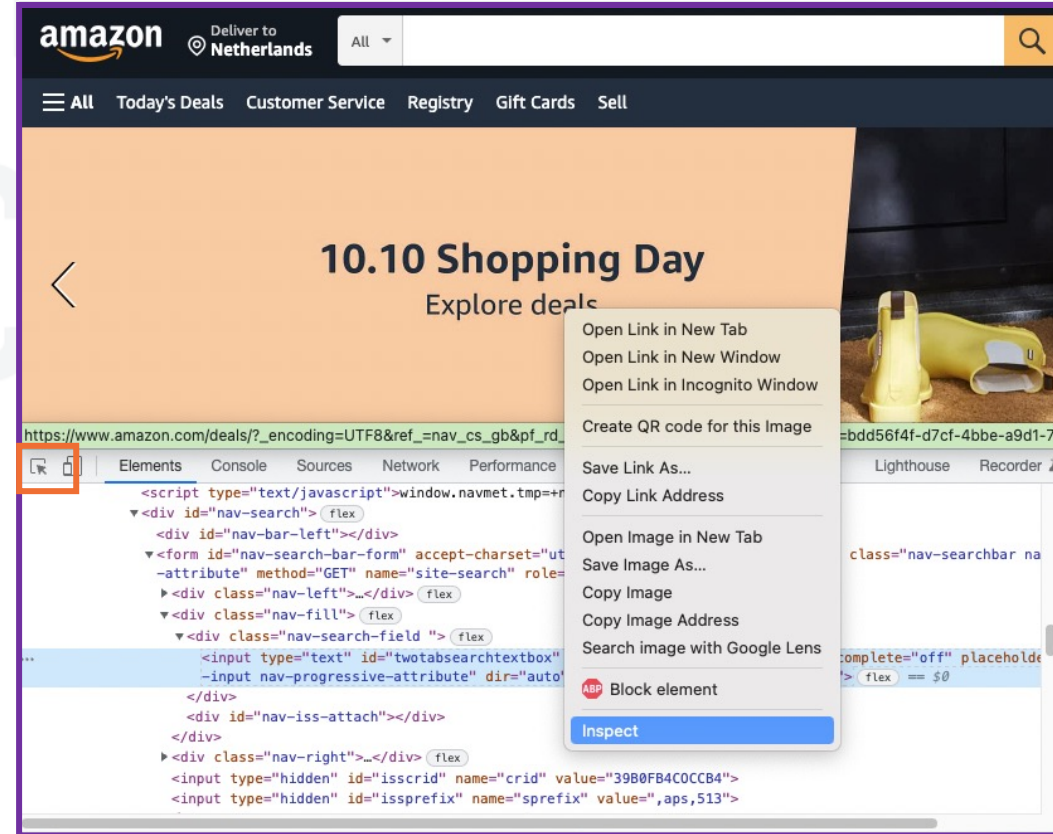
# WebElements

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

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

Veya menüdeki → isareti seçilip, mavi iken mouse ile istenen element seçilebilir.

HTML kodlar açık iken ctrl+f tuslarına basılınca açılan bölümde web element'in özellikleri aratılırsa, o özellikte kaç web element bulunduğu görülebilir.



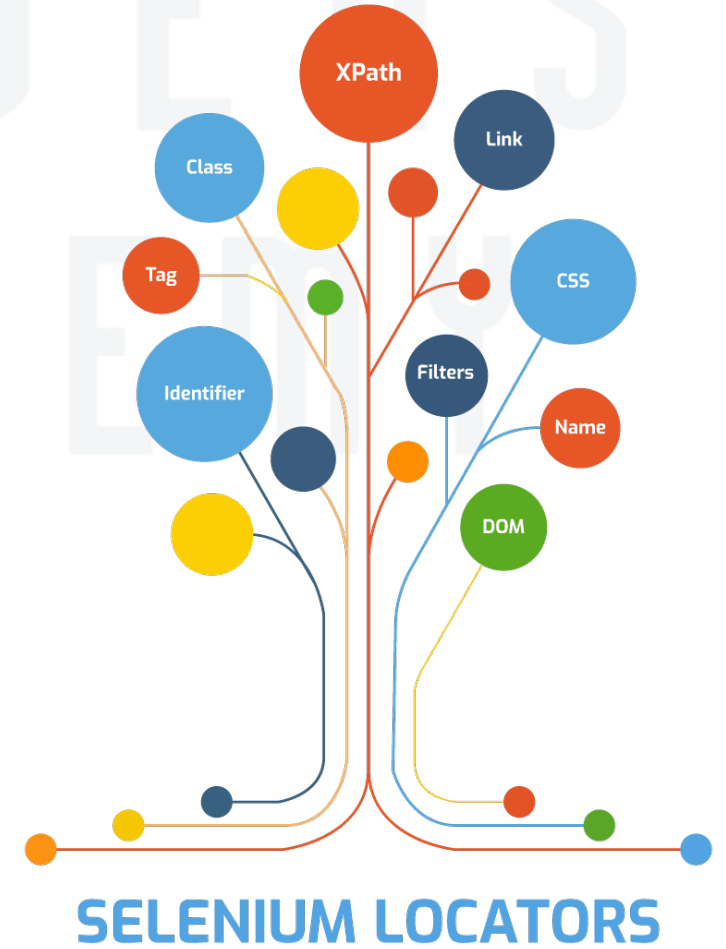
# Locators

Selenium LOCATORS, web sayfasındaki web öğelerini tanımlamak için kullanılır.

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

Konum belirleyiciler bize web elementleri tanımlamada yardımcı olur.

Web Elementlerine ulasmak icin tag veya bazi attribute'lerin kullanildigi 6 adet locators bulunur, bunlarla ulasilamayan webelementleri icin ozel olarak tanimlanan Xpath ve css locator'lari kullanilir.



# 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">
```

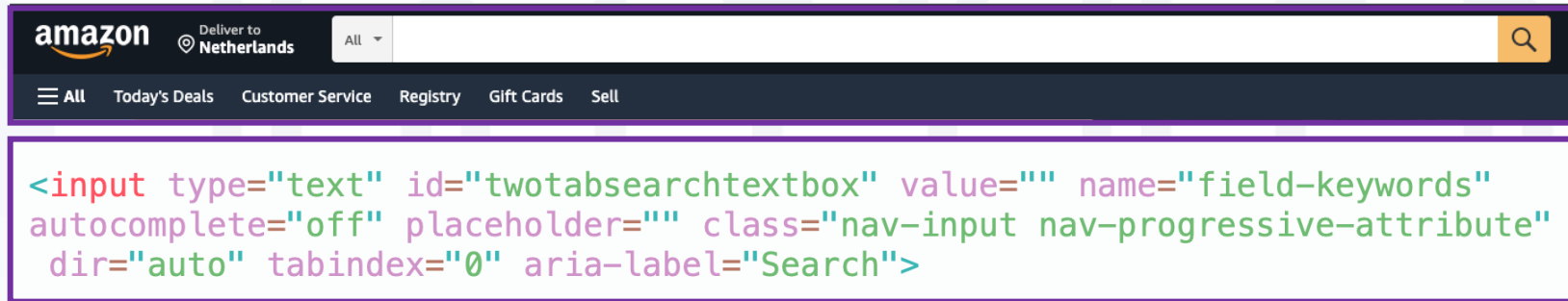
## 1- By.id("uniqueId")

Web elementi tanımlamak için ilk bakacağımız 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 aranarak, unique sonuca ulaşıldığı gözlemlenmelidir.

# driver.findElement( ) Method'u



Unique locator'i tespit edilen web element kullanılmak için, driver objesi ile LOCATE edilip, WebElement class'ından oluşturulan objeye atanmalıdır.

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

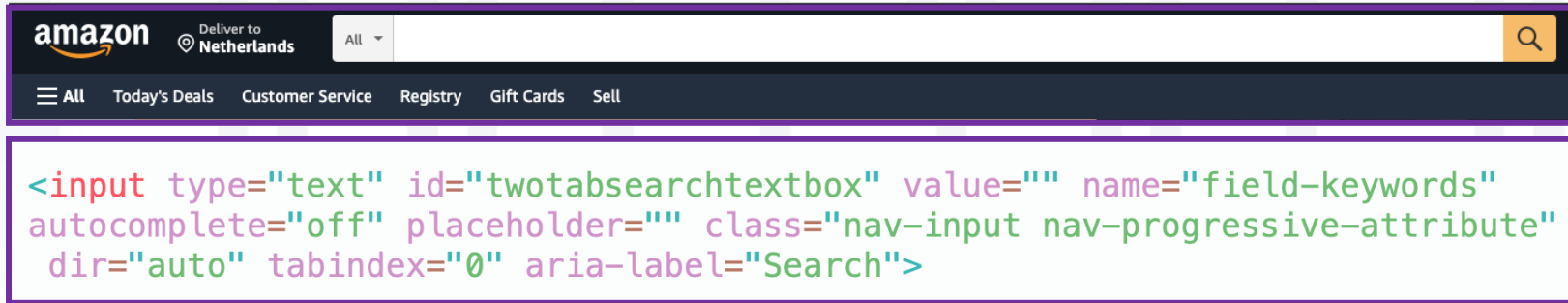
```
WebElement amazonAramaKutusu = driver.findElement(By.id("twotabsearchtextbox"));
```

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

findElement( ) ile locate edilip, objeye atanan webElement testler sırasında kullanılabilir.

webElement bir obje olduğu için direk yazdırılmaz, hazır method'lar kullanılarak manipüle 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



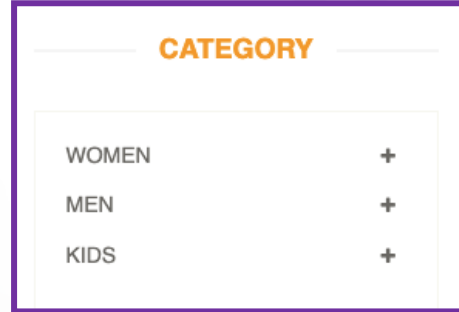
```
<input type="text" id="twotabsearchtextbox" value="" name="field-keywords"
autocomplete="off" placeholder="" class="nav-input nav-progressive-attribute"
dir="auto" tabindex="0" aria-label="Search">
```

## 2- By.name("uniqueName")

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

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

# Locators



```
▼<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>
▼<div class="brands_products">
```

### 3- By.classname("uniqueClassValue")

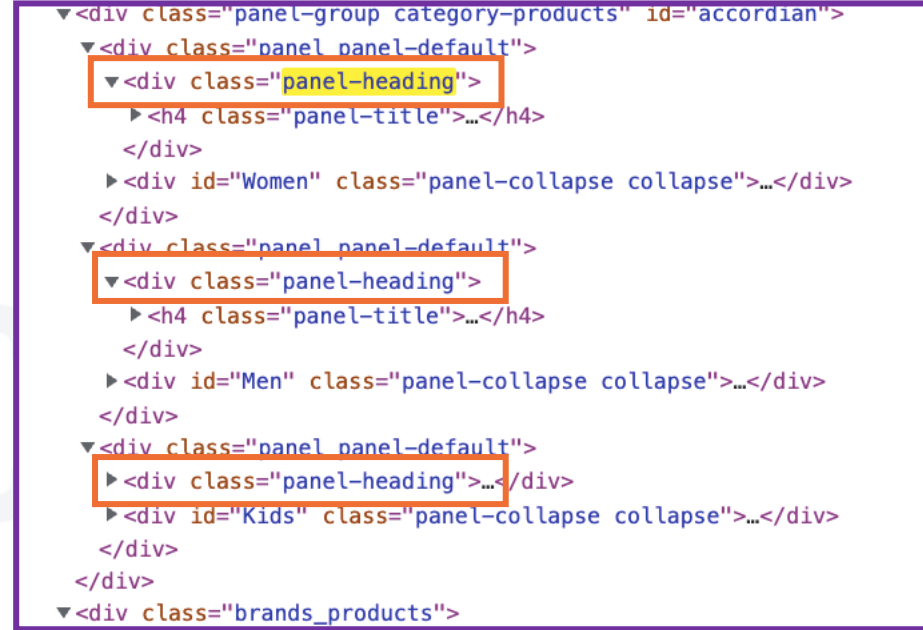
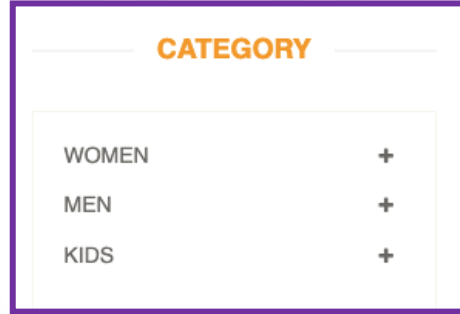
class attribute'u genellikle benzer özellikleri barındıran web elementleri gruplandırmak için kullanılır.

Bu sebeple class attribute'u ile yapacağımız locate işlemleri genellikle 1 web element değil, birden fazla element döndürür.

bu elementleri store edebilmek için bir web element değil, web elementlerden oluşan bir list gereklidir.

**NOT** : class value'sünde boşluk (space) varsa By.classname ile locate işlemlerinde sorunlar yaşanabilir.

# driver.findElements( ) Method'u



driver.findElements(.....)

Locator'a uygun tüm web elementlerini döndürür.

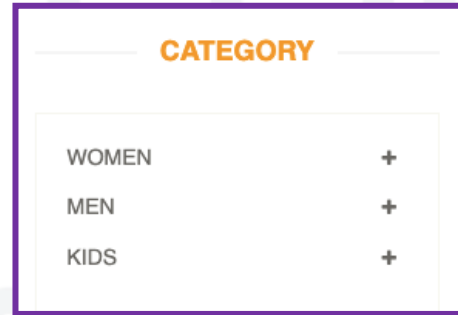
findElements( ) birden fazla web element döndürebileceği için dönen elementleri store etmek için bir list kullanılmalıdır.

Locator'a uyan hiçbir webelement olmasa da exception oluşmaz, bos bir list oluşur.

List'teki tüm elementler web element olduğu için direk 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- Sayfayi kapatın

# Locators

## 4- By.tagName("tagValue")

```
<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">
```

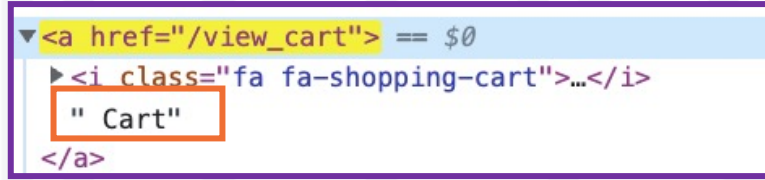
```
List<WebElement> inputTagList =driver.findElements(By.tagName("input"));
```

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

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

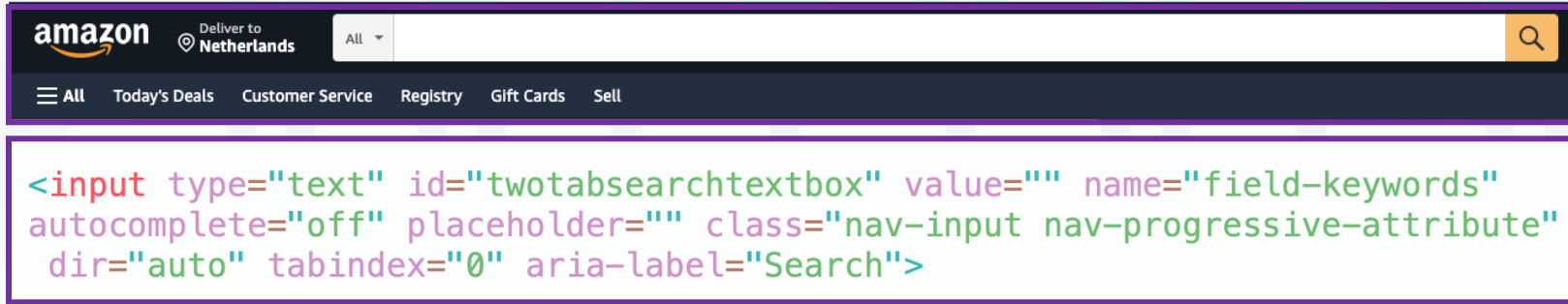
Her link üzerinde bulunan yazı kullanılarak locate yapmamızı sağlar.

Link üzerinde bulunan yazı String data türünde olduğundan case sensitive'dir.

`By.linkText ( )` için boşluklar da dikkate alınarak tüm metin yazılmalıdır.

Tüm metnin yazılamaması, yazının kısmi olarak kullanılması isteniyorsa `By.partialLinkText ( )` kullanılmalıdır.

# WebElement Method'ları



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

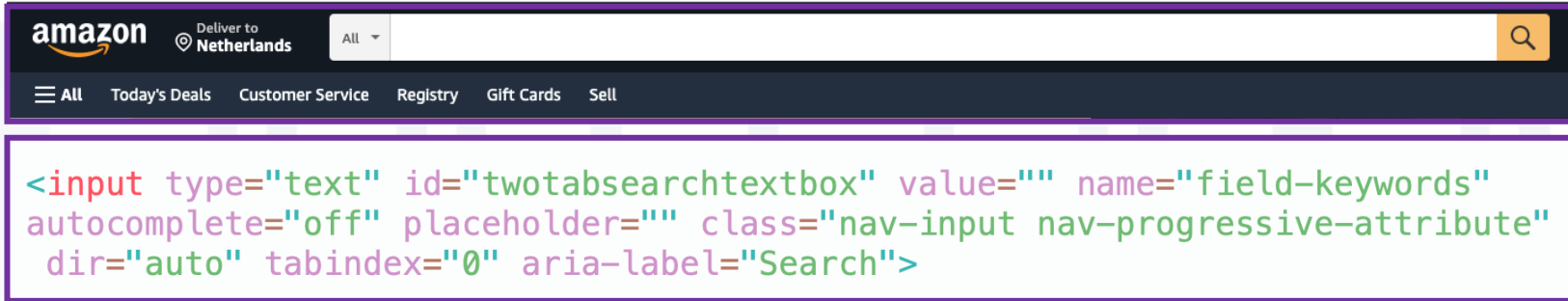
Webelement'i locate ettikten sonra variable atamak veya atamadan direk kullanmak test adımlarına ve belirlenen genel test stratejisine bağlıdır.

Webelement ile yapabileceğimiz işlemler için hazır method'ları kullanırız.

1- `webElement.click();` Web element'e click yapar.

2- `webElement.sendKeys( ...keysToSend: "İstenen Metin");` Web element'e istenen metni yollar

# WebElement Method'ları



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

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

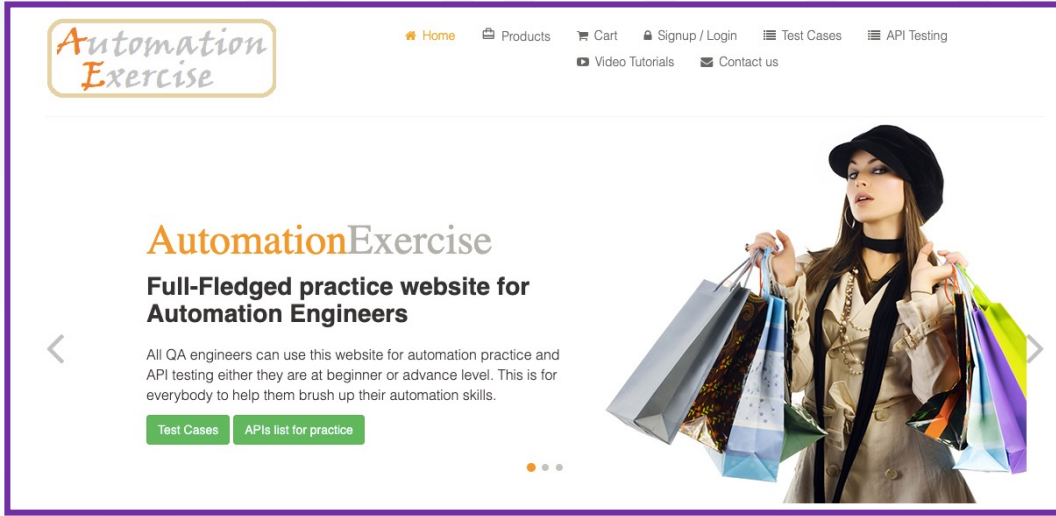
Web element'e istlenen metni yollayıp, sonra ENTER tusuna basar

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

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

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

# Locators



## Automation Exercise Link Testi

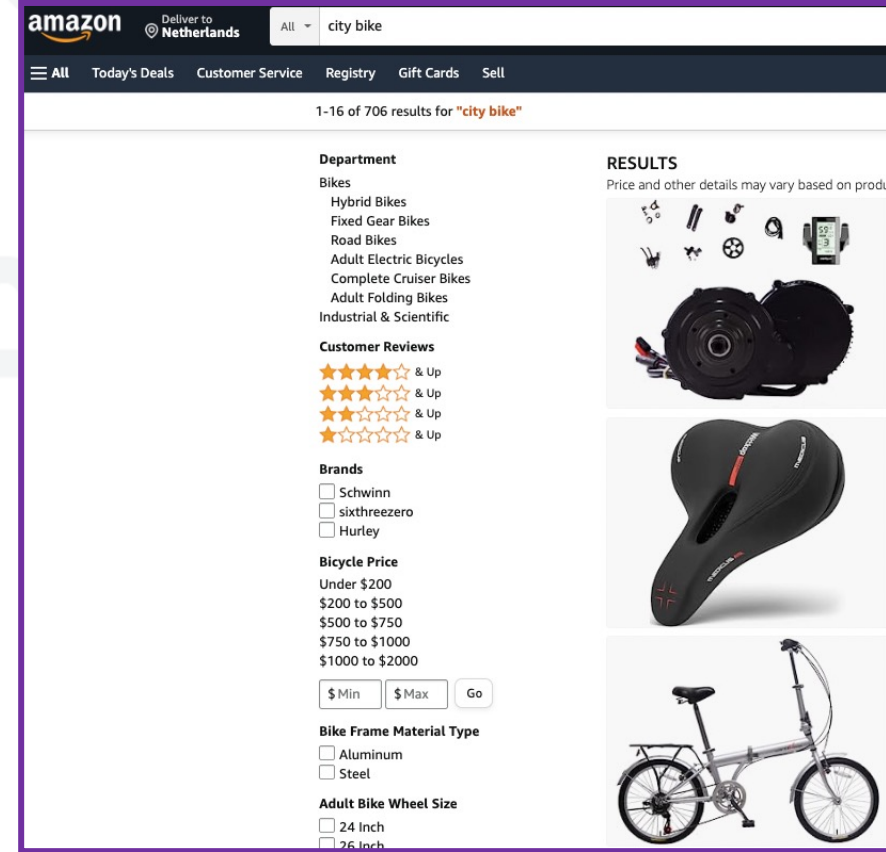
- 1- Bir test class'i olusturun ilgili ayarlari yapin
- 2- <https://www.automationexercise.com/> adresine gidin
- 3- Sayfada 147 adet link bulundugunu test edin.
- 4- Products linkine tiklayin
- 5- special offer yazisinin gorundugunu test edin
- 6- Sayfayi kapatın

# Locators

	findElement( )	findElements( )
websayfasinda birden fazla Web Element Locator ile uyusursa	Ilk elemani dondurur	Tum elemanlari ondurur
websayfasinda hicbir Web Element Locator ile uyuşmazsa	NoSuchElementException firlatir	Exception firlatmaz, bos bir liste dondurur
Return Type	WebElement	List<WebElement>
Elemana erisim	Direk ulasilabilir	Liste'den index veya iterator ile ulasilabilir

# Locators

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





# Locators - Xpath

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

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

Önceki 6 locator, HTML element oluşturulurken developer'ın yazdığı kodlara göre yapılır.

Örneğin; HTML element'de id attribute'u varsa By.id( ) method'u kullanılabilir ama developer id attribute'u koymadı ise kullanamayız.

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



Xpath de HTML kodu kullanır ancak farklı kombinasyonlar kullanabildiği için dinamik ve her web element için mutlaka bir xpath bulunabilir.

2 çeşit Xpath yazılabilir

1. **Absolute** xpath (mutlak)

2. **Relative** xpath (bağıl)

# 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>
      </tr>
    </tbody>
  </table>
</div>
```

Her HTML sayfası <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" cellpadding="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>
            // tbody / tr / td[3] / a / span
          </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>
```

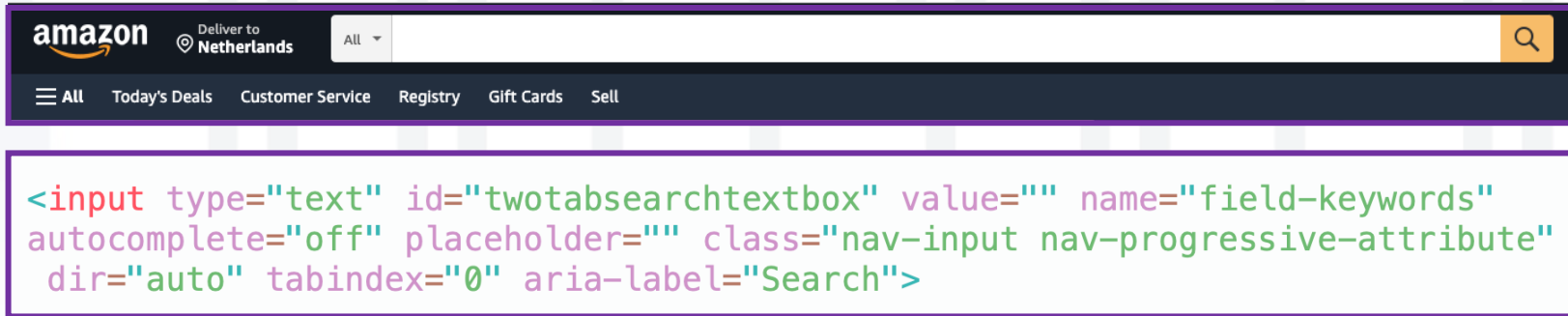
Absolute xpath yazmak için en basa // sonraki her adımda / yazarak hedef web element'e kadar tüm tag'lar yazılır.

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

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

# 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']`

# Locators

## 2.Relative Xpath



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

Unique deger icin 3 bilesenin tamamı kullanılmak zorunda degildir.

Hedef unique olarak webelement'i locate etmektir. Daha az bilesenle de unique degere ulasilabilen 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 bazı webelementler'inde de text bulunabilir.

Bu text'ler o webelement'e özel olduğu için unique bir xpath elde etmek için kullanışlı olabilirler.

Text ile locate yazmak için kullanılan genel syntax :

```
"/tagname[text()='yazinin tamamı']"
```

Genel xpath kullanımına uygun olarak tagname veya attribute ismi yazılmadan da text ile xpath yazılabilir.

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

```
"/[*[.='yazinin tamamı']] "
```

Metnin sadece bir kısmı kullanılacaksa

```
"/[*[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



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

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

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

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

cssSelector ozellikle class ve id attribute'leri ile kısa sekilde yazilabilir

```
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- Sayfayi “refresh” yapin
- 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-Sayfayi kapatın

# Relative Locators

Relative Locators nedir ?

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

Ornegin yandaki resimde Berlin icin bir cok relative locator tanimlayabiliriz.

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



Bu ozellik Selenium 4 ile gelen yeniliklerden biridir.

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

# Relative Locators

## Class Work: Relative Locators

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



# Relative Locators

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

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

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

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



# Maven



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

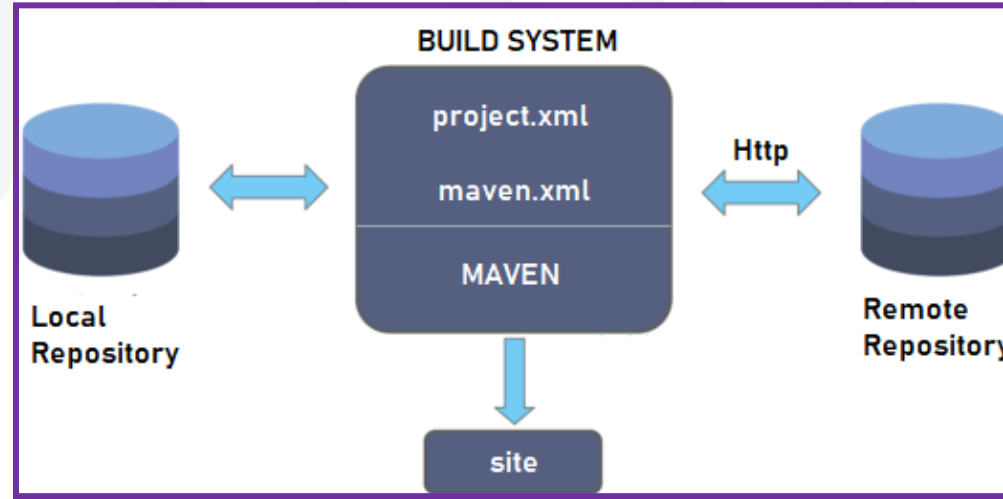
Proje nesne modeli (POM) konseptine dayalı olarak Maven, bir projenin inşasını, raporlamasını ve dokümantasyonunu merkezi bir bilgi parçasından(dependency) yönetebilir.

- 1- Projeleri oluşturma için standart belirleme,
- 2- projenin nelerden oluştuğunu net olarak tanımlama,
- 3- proje bilgilerini yayınlamanın 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 oluşturma aracıdır (**build tool**). Maven proje otomasyon ve yönetim aracıdır (**automation and management tool**).

Maven, konfigürasyon için pom.xml dosyasını kullanır.

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

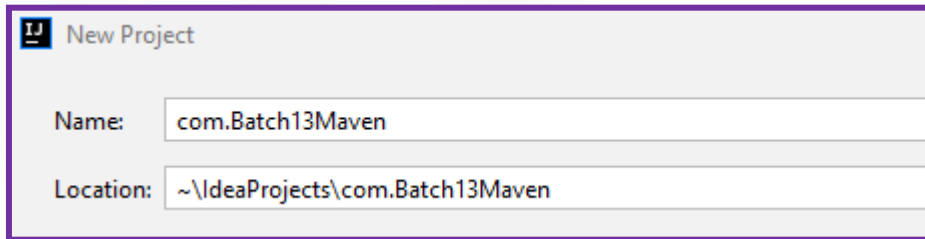
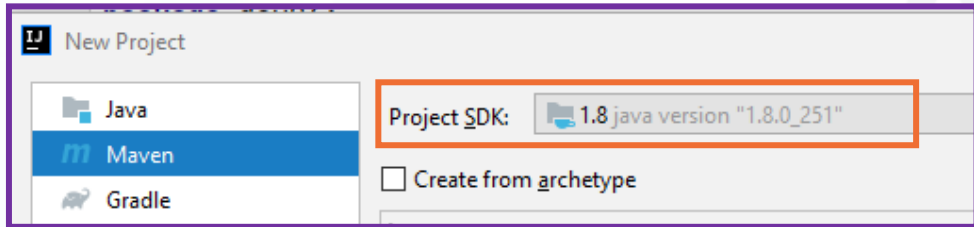
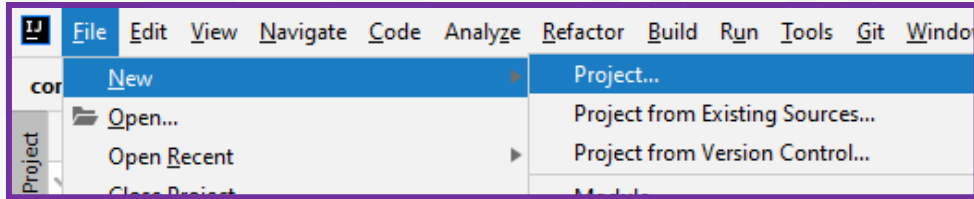
Maven ile çalışan tüm projeler aynı konfigürasyonu kullandığı için yeni bir projede çalışmaya başlandığında projenin anlaşılması çok kolay olacaktır.

# Neden Maven ?

- 1- Proje yönetiminde oluşturma, belgeleme, yayınlama 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 oluşturmalarını 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 kontrol edin

3. Name: com.projeAdi -> click finish

EnableAutoImport sorarsa click

4. Package olusturun, name : day04

5. Class olusturun, 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 bazı bilgiler ve Maven tarafından projeyi oluşturmak için kullanılan konfigürasyon detaylarını gösterir.

Bir projenin hangi framework'u kullandığını anlamak için 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.

İstenen dependency için version seçilirken güncellik, stabil versiyon olma ve kullanılma sayıları 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...>

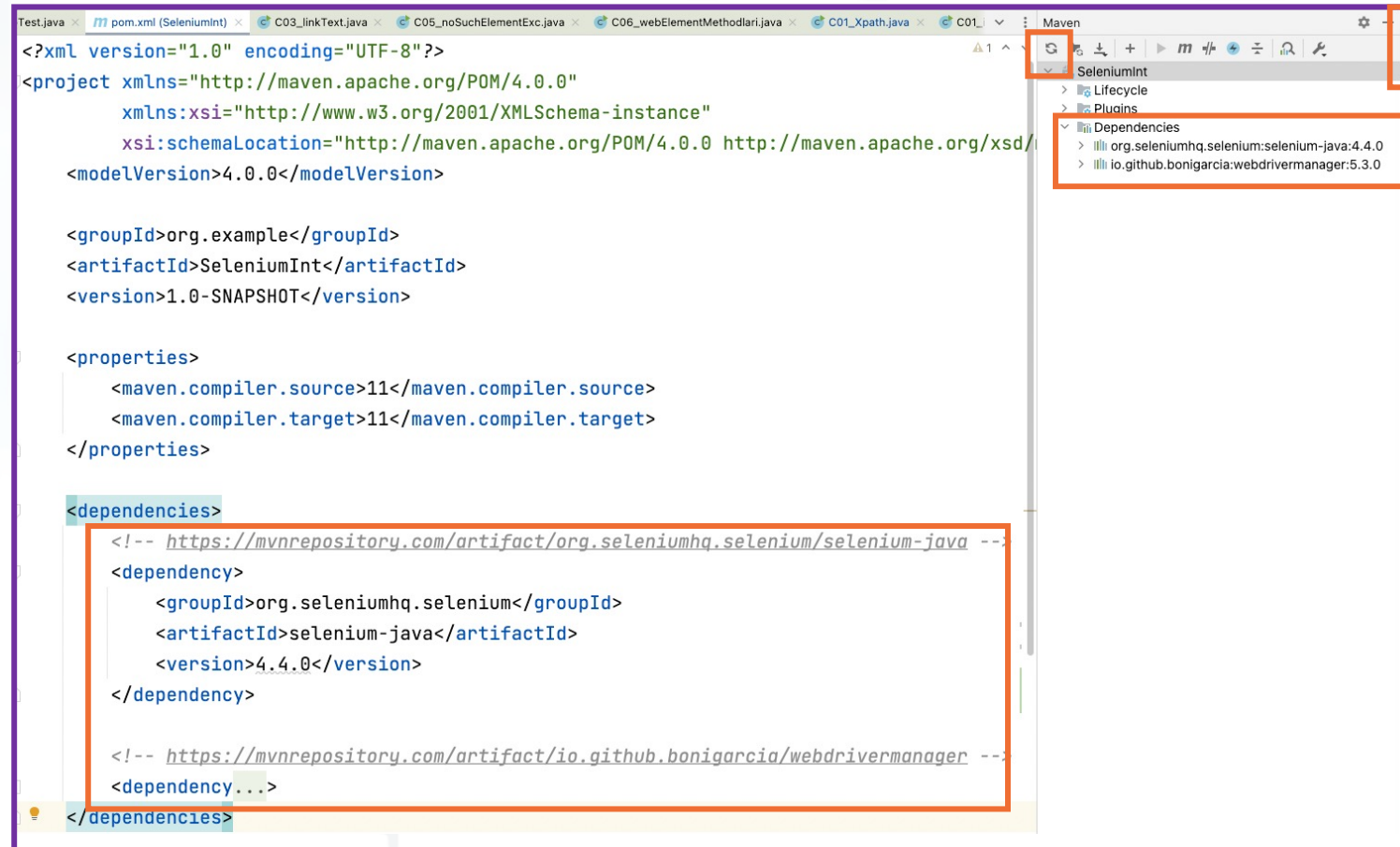
  <!-- https://mvnrepository.com/artifact/io.github.bonigarcia/webdrivermanager -->
  <dependency...>
</dependencies>
</project>
```

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

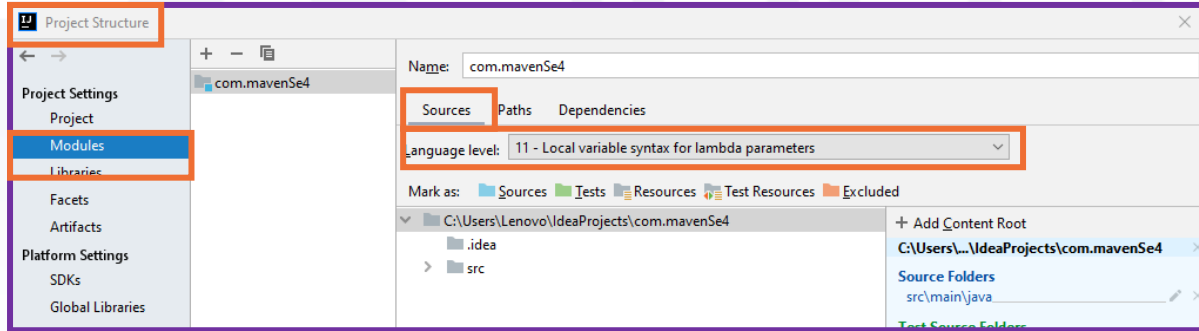
```
<version>4.5.0</version>
```

# 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.



# pom.xml'e Dependency Ekleme



File

Project Structure

Modules

Sources

Language level : min 8 yapin, bilgisayarınızda kurulu java 8 veya ustü ise java versiyonu ayni olmalı

File

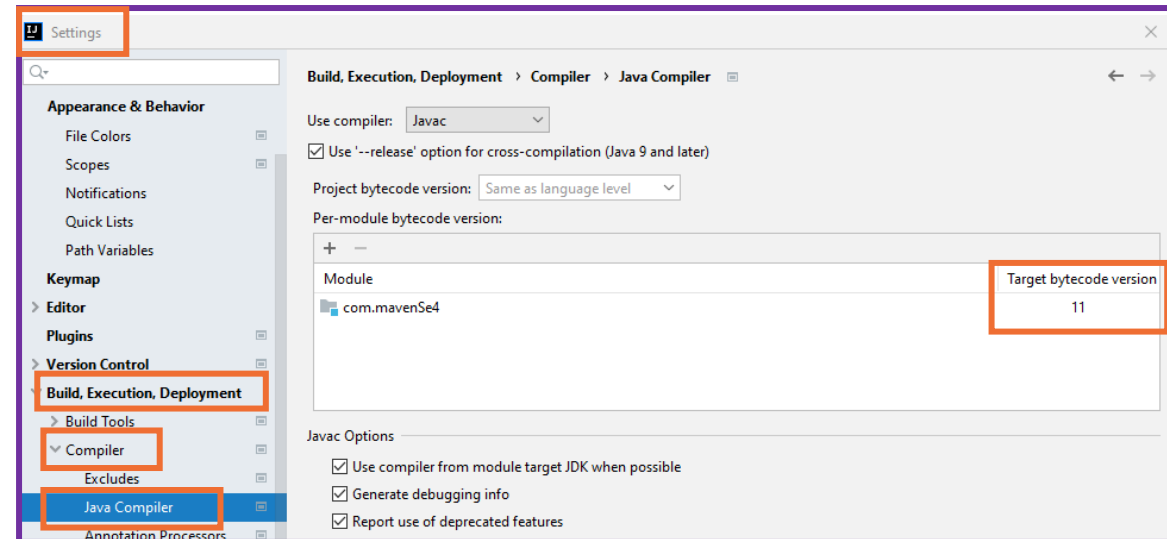
Settings

Build,Execution,Deployment

Compiler

Java Compiler

Target bytecode version : min 8 yapin, bilgisayarınızda kurulu java 8 veya ustü ise java versiyonu ayni olmalı



# Maven WebDriver Olusturma

Maven ile Selenium Java ve WebDriverManager dependency'lerini projemize eklendigi icin, her seferinde driver.exe dosyasini driver'a tanitma mecburiyeti kalmadi.

```
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 ayarlari yapilirken ilk satirdaki sistem ayarlarini bonigarcia WebDriverManager kullanarak yapacagiz.

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

Chrome disinda bir browser kullanilmek istenirse, 1. ve 2.satirda Chrome yerine o browser'in secilmesi yeterli olacaktır.

# 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 tiklayin
3. Login alanine “username” yazdirin
4. Password alanina “password” yazdirin
5. Sign in buttonuna tiklayin
6. Back tusu ile sayfaya donun
7. Online Banking menunden Pay Bills sayfasina gidin
8. amount kismina yatirmek istediginiz herhangi bir miktari yazin
9. tarih kismina “2020-09-10” yazdirin
10. Pay buttonuna tiklayin
11. “The payment was successfully submitted.” mesajinin ciktigini test edin

# Maven Tekrar Testi

- 1- CO1\_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- Sayfayi 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. Sayfayi kapatın

