**Selenium Actions Sınıfını Kullanarak Klavye ve Fare İşlemleri**

Karmaşık kullanıcı hareketlerini (sağ tıklama, iki kere tıklama, sürükleme, vb.) taklit etmek için kullanılan API. Doğrudan Klavye veya Fare kullanmak yerine bu sınıf kullanılır.  
  
**Build patternini uygular:** yöntem çağrıları tarafından belirtilen tüm eylemleri içeren bir CompositeAction oluşturur.

* Kullanıcı hareketlerini uygulayan fonksiyonları kullanmak için Actions ve Action sınıfını import etmek gerekir:

Actions sınıfı ve Action sınıfı, WebDriver API'sinin org.openqa.selenium.Interactions paketinde bulunur.  
  
import org.openqa.selenium.interactions.Actions;  
  
import org.openqa.selenium.interactions.Action;

* Yöntemlerini kullanmaya çağırmak için Actions sınıfı nesnesine ihtiyaç vardır.

Actions actions = new Actions(WebDriver nesnesi);

Üzerinde işlem yapılacak elementi bulmak gerekiyor.

WebElement webElement = driver.findElement(...);

actions.contextClick(webElement).perform(); //contextClick, element üzerinde sağ tıklar.

* Bu sınıfının bazı fonksiyonları:

clickAndHold () Geçerli fare konumuna (serbest bırakmadan) tıklar.

contextClick () Geçerli fare konumunda sağ tıklar. (Sağ Tıklama İşlemi)

doubleClick () Geçerli fare konumunda çift tıklama gerçekleştirir.

dragAndDrop (kaynak, hedef) Kaynak öğenin konumunda tıklayıp bekletir, hedef öğenin konumuna gider, sonra fareyi serbest bırakır.

* Parametreler:

kaynak-elementin tutulduğu yer.

hedef-elementin götürüleceği ve farenin serbest bırakılacağı yer

dragAndDropBy (kaynak, x-offset, y-offset) Kaynak öğenin konumunda tıklayıp bekletir, belirli bir ofset tarafından hareket eder ve ardından fareyi serbest bırakır.

* Parametreler:

kaynak-elementin tutulduğu yer

xOffset-yatay hareket ofseti.

yOffset - dikey hareket ofseti.

keyDown (modifier\_key) Bir değiştirici tuşa basma işlemini gerçekleştirir. Değiştirici tuşu serbest bırakmaz - sonraki etkileşimler basılı tutulduğunu varsayabilir.

* Parametreler:

modifier\_key - değiştirici tuşlardan herhangi biri (Keys.ALT, Keys.SHIFT veya Keys.CONTROL)

keyUp (değiştirici \_key) Bir anahtar bırakma işlemi gerçekleştirir.

* Parametreler:

modifier\_key - değiştirici tuşlardan herhangi biri (Keys.ALT, Keys.SHIFT veya Keys.CONTROL)

moveByOffset (x-offset, y-offset) Fareyi mevcut konumundan (veya 0,0) belirtilen ofsete hareket ettirir.

* Parametreler:

x-uzaklık-yatay sapma. Negatif bir değer, fareyi sola hareket ettirmek anlamına gelir.

y-sapma - dikey sapma. Negatif bir değer, fareyi aşağı hareket ettirmek anlamına gelir.

moveToElement (toElement) Fareyi öğenin ortasına taşır.

* Parametreler:

toElement-taşınacak element.

release () Geçerli fare konumundaki basılan sol fare düğmesini serbest bırakır

sendKeys (onElement, charsequence) Öğeye bir dizi tuş vuruşu gönderir.

* Parametreler:

onElement - tuş vuruşlarını alacak olan eleman, genellikle bir metin alanı

charsequence - gönderilecek tuş vuruşlarının sırasını temsil eden herhangi bir dize değeri

\*build fonsiyonunu son yazarak birden fazla fonksiyonu peşpeşe kullanabilirsiniz.  
\*Thread.sleep in amacı sadece yazılanı gözle görmek, kullanmayabiliriz.

**Daha ayrıntılı incelemek istersek:**

**Selenyumda sürükleyip bırakın**

Selenium'da Sürükle ve Bırak, Selenium'daki Actions sınıfı ile üç farklı yaklaşımla yapılabilir:

* **Yapı yöntemiyle Selenium'da Action Class ile Selenium'da Sürükle ve Bırak:**

Selenium'da Sürükle ve Bırak işlemini gerçekleştirmek için Source WebElement (sürüklenecek olan) ve Destination WebElement (Source WebElement'in bırakılacağı yer, yani Destination WebElement gibi) gibi iki webelementimiz olması gerekir. Aşağıdaki yöntem özelleştirilmiş bir yöntemdir ( Selenium'da Sürükle ve Bırak işlemini gerçekleştiren Framework WebUtils'inizi oluşturmak için kullanabileceğiniz) ve bu yöntem iki argüman alır, biri Source WebElement, diğeri daha önce belirtildiği gibi Destination WebElement:

public void dragAndDrop(Kaynağa göre, Hedefe göre) { try { WebElement sourceElement = driver.findElement(source); WebElement targetElement = driver.findElement(hedef); Eylem oluşturucu = yeni Eylemler(sürücü); Eylem dragAndDrop = builder.clickAndHold(sourceElement).moveToElement(destinationElement) .release(destinationElement).build(); dragAndDrop.perform(); BASE\_LOGGER.info("Sürükle ve Bırak işlemi başarıyla gerçekleştirildi "); } catch (İstisna e) { Dize istisnaData = e.getCause().getMessage(); BASE\_LOGGER.error("DragAndDrop yapılırken hatayla karşılaşıldı, yani : " + istisnaData + " "); } }

* **Selenyum'da ClickAndHold**

ClickAndHold, Selenium'daki Actions sınıfından (org.openqa.selenium.interactions paketinden) başka bir önemli yöntemdir.  
Önce herhangi bir webElement üzerinde sol tıklama işlemini gerçekleştirin ve fareyi bırakmadan basılı tutun.

* **Selenyum'da MoveToElement**

MoveToElement, Selenium'daki Actions sınıfından başka bir WebElement'e (Destination Webelement) hareketi gerçekleştirmek için kullanılan ve bu yöntemin ilgili bir argüman olarak aldığı bir yöntemdir.

* **Selenyumda Çift Tıklama**

Çift tıklama işlemini fare ile çoğaltmak için Selenium'da Actions sınıfı aracılığıyla çift tıklama gerçekleştirmemiz gerekiyor ve bunu aşağıdaki yaklaşımla yapabiliriz:

public void doubleClick(Yer bulucuya göre) { try { WebElement öğesi = driver.findElement(yer bulucu); Eylemler eylemler = yeni Eylemler(sürücü); eylemler.doubleClick(element).perform(); BASE\_LOGGER.info("Elemana çift tıklama yapıldı: " + konumlandırıcı); } catch (StaleElementReferenceException e) { BASE\_LOGGER.error("Öğe sayfa belgesine eklenmemiş " + e.getCause().getMessage()); } catch (NoSuchElementException e) { BASE\_LOGGER.error("Element " + konumlandırıcı + " DOM'da bulunamadı " + e.getCause().getMessage()); } catch (İstisna e) { BASE\_LOGGER.error("Element " + konumlandırıcı + " tıklanabilir değildi " + e.getCause().getMessage());}}

Yukarıdaki kod parçacığı, temel olarak Locator, yani üzerinde çift tıklamanın gerçekleştirilmesi gereken WebElement olarak bir argüman alan bir yöntemdir.

* **Bağlam tıklaması veya Selenyum sağ tıklaması**

Bağlam tıklama veya sağ tıklama işlemini fare ile çoğaltmak için Selenium'daki Actions sınıfı aracılığıyla bağlam tıklama yöntemini gerçekleştirmemiz gerekiyor ve bunu aşağıdaki yaklaşımda yapabiliriz:

public void rightClick(Locator tarafından) { try { WebElement element = driver.findElement(locator); Eylemler eylemler = yeni Eylemler(sürücü); eylemler.contextClick(element).perform(); BASE\_LOGGER.info("Bağlam Gerçekleştirildi Öğeye tıklayın: " + konumlandırıcı); } catch (StaleElementReferenceException e) { BASE\_LOGGER.error("Öğe sayfa belgesine eklenmemiş " + e.getCause().getMessage()); } catch (NoSuchElementException e) { BASE\_LOGGER.error("Element " + konumlandırıcı + " DOM'da bulunamadı " + e.getCause().getMessage()); } catch (İstisna e) { BASE\_LOGGER.error("Element " + konumlandırıcı + " tıklanabilir değildi " + e.getCause().getMessage()); } }

Yukarıdaki kod parçacığı, temel olarak Locator, yani üzerinde çift tıklamanın gerçekleştirilmesi gereken WebElement olarak bir argüman alan bir yöntemdir.

* **Selenyumda duraklama**

Farklı Eylemler arasında herhangi bir zaman gecikmesi eklemek istediğimizde, Eylemler işlemleri arasında duraklatma yöntemini kullanabiliriz, örneğin, sürükle ve bırak işlemi arasında birkaç saniye gecikme olmasını istiyorsak, duraklatmayı çağırabiliriz ( ) aşağıdaki yaklaşımda Selenium'daki Eylemler Sınıfı yöntemi:

public void duraklamaBetweenActions(Kaynağa göre,Hedefe göre, int timeUnit) { try { WebElement sourceElement = driver.findElement(source); WebElement targetElement = driver.findElement(hedef); Eylem oluşturucu = yeni Eylemler(sürücü); builder.clickAndHold(sourceElement).moveToElement(destinationElement).pause(timeUnit).release(destinationElement).build().perform(); BASE\_LOGGER.info("Sürükle ve Bırak işlemi başarıyla gerçekleştirildi "); } catch (İstisna e) { Dize istisnaData = e.getCause().getMessage(); BASE\_LOGGER.error("DragAndDrop yapılırken hatayla karşılaşıldı, yani : " + istisnaData + " "); } }

Yukarıdaki yöntemde üç argüman alır: bir kaynak WebElement, başka bir argüman hedef öğedir ve sonuncusu timeUnit içindir, yani Selenium'da duraklama() kullanarak tanıtmak istediğimiz zaman gecikmesi.

* **Selenyum içinde fare vurgusu**

Selenium işleminde Mouse Hover, Selenium'daki Actions sınıfı kullanılarak elde edilebilir, temel olarak, herhangi bir webElement'e odaklandığımızda ve oradan, bu Mouse Hover işlemi ve bu WebElement ile fare imlecinizi başka bir öğeye getirmek istiyoruz. Mouse imlecimizi almak istediğimizde Target WebElement olarak adlandırılıyor.

Aynısı için kod parçacığı aşağıdadır:

public void mouseHover(TargetedLocator tarafından) { try { WebElement targetElement = driver.findElement(targetedLocator); Eylem oluşturucu = yeni Eylemler(sürücü); builder.moveToElement(targetedElement).build().perform(); BASE\_LOGGER.info("Selenyum işleminde Fare vurgusu başarıyla gerçekleştirildi "); } catch (İstisna e) { Dize istisnaData = e.getCause().getMessage(); BASE\_LOGGER.error("Selenium'da Fare üzerine gelinirken hatayla karşılaşıldı, yani : " + istisnaData + "); } }

* **Klavye Eylemleri - Selenium'da Eylemler Sınıfı**

**Klavye eylemleri nedir ve nerede gerektiği:**

Klavye eylemleri, WebElements ile etkileşim halindeyken Contrl+A'yı veya Control+C'yi birlikte gerçekleştirmeye çalıştığımız veya klavyeden Key up veya Keydown butonunu gerçekleştirdiğimiz gibi, işlemlerin doğrudan bir klavyeden gerçekleştirildiği anlamına gelir.

Selenium'da Actions sınıfı yardımıyla birkaç Klavye etkileşimi gerçekleştirebiliriz.

* **Selenyum'da sendKey'ler :**

Selenium'da Actions sınıfı yardımıyla sendKeys işlemini aşağıdaki şekilde gerçekleştirebiliriz:

public void sendKeysWithActionClass(Yer bulucuya göre) { try { WebElement öğesi = driver.findElement(yer bulucu); Eylemler eylemler = yeni Eylemler(sürücü); action.sendKeys("KeyBoard Veri Girişi"); // Control ve c tuşlarına birlikte basarak Klavye Eylemlerini gerçekleştirin action.sendKeys(Keys.CONTROL); action.sendKeys("c"); eylemler.build().perform(); } catch (İstisna e) { Dize istisnaData = e.getCause().getMessage(); } }

* **Selenyum'da KeyUp :**

Benzer şekilde Selenium'da Actions sınıfı ile KeyUp işlemlerini aşağıdaki şekilde gerçekleştirebiliriz:

public void keyUpWithActionClass(Yer bulucuya göre) { try { WebElement öğesi = driver.findElement(yer bulucu); Eylemler eylemler = yeni Eylemler(sürücü); action.sendKeys("KeyBoard Veri Girişi"); // Klavye Eylemlerini Control ve c tuşlarına basarak, keyUp action.keyUp(Keys.CONTROL) ile birlikte gerçekleştirin; action.sendKeys("c"); eylemler.build().perform(); } catch (İstisna e) { Dize istisnaData = e.getCause().getMessage(); } }

* **Selenyum'da KeyDown**

KeyDown işlemini Selenium'da yine Actions sınıfı yardımıyla aşağıdaki yaklaşımla gerçekleştirebiliriz:

public void keyDownWithActionClass(Yer bulucuya göre) { try { WebElement öğesi = driver.findElement(yer bulucu); Eylemler eylemler = yeni Eylemler(sürücü); action.sendKeys("KeyBoard Veri Girişi"); // Control ve V tuşlarına basarak, keyDown action.keyDown(Keys.CONTROL) ile birlikte Klavye Eylemlerini gerçekleştirin; action.sendKeys("v"); eylemler.build().perform(); } catch (İstisna e) { Dize istisnaData = e.getCause().getMessage(); } }

* **Hap Bilgiler:**

➢ Actions class’ını kullanarak mouse ve klavye ile yapabileceğimiz tüm işlevleri gerçekleştirebiliriz.

➢ Actions Class birçok kullanışlı mouse ve klavye işlevine sahiptir.

➢ Çift tıklama (double click), sürükleme ve bırakma(drag and drop), mouse’u hareket ettirme (mouse actions) gibi karmaşık mouse eylemleri için veya Keyboard ile yapabileceğimiz pageUp, pageDown, shift, arrowDown gibi işlemleri Actions class’ından object üreterek driver ile yapabiliriz.

**Actions’u kullanmak için:**

➢ **1.Adım:** Actions class’ta bir object oluşturulur.

Actions actions= new Actions(driver);

➢ **2. Adım:** Üzerinde çalışmak istediğiniz WebElement öğesini bulunur.

WebElement element = driver.findElement(By.id("ID"));

➢ **3.Adım:** Ardından bu webelement üzerinde action gerçekleştirilir. Örneğin tıklamak için.

actions.click(element).perform( );



**NOT:**

* Action Class’ını her kullanmak istediğimizde yeniden obje olusturmamız gerekmez.
* action objesi’ni bir kere oluşturunca, istediğiniz kadar action . ile başlayan komut yazar ve çalışması için sonuna perform( ) yazarız.
* action objesi kullanılarak başlayan her komut, çalışmak icin perform( ) bekler.

**Mouse Base Actions (Mouse Aksiyonları):**

➢ **doubleClick ( ):** WebElement’e çift tıklama yapar

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

➢ **dragAndDrop ( ):** WebElement’i bir noktadan diğerine sürükler ve bırakır.

➢ **moveToElement ( ):** Mouse’u istediğimiz WebElement’in üzerinde tutar.

➢ **contextClick ( ):** Mouse ile istediğimiz WebElement’e sağ tıklama yapar.

**Keyboard Base Actions (Klavye Aksiyonlar):**

➢ Action Class’indan kullanacağımız method’lar ile klavyedeki tuşları kontrol edebiliriz.

➢ Bunun için 3 method kullanırız.

■ **sendKeys ( ):** Öğeye bir dizi anahtar gönderir.

■ **keyDown ( ):** Klavyede tuşa basma işlemi gerçekleştirir.

■ **keyUp ( ):** Klavyede tuşu serbest bırakma işlemi gerçekleştirir.

* **Son Not:**

Actions Class: Selenium’daki Actions class’i mouse ve klavye ile yapabildiğimiz eylemleri driver ile yapabilmemize imkan tanır.

1- Mouse Base Actions :

click(), contextClick(), doubleClick(), clickAndHold(), dragAndDrop(), moveToElement()

2- Keyboard Base Actions :

keyDown(), keyUp(), sendKeys()

Actions class’ını kullanabilmek için actians objesi oluşturup, parametre olarak driver’ımızı eklemeli, en sona da perform() demeyi unutmamalıyız.

**Kaynak:**

* <http://sinemayk.blogspot.com/2019/09/selenium-actions-snfn-kullanarak-klavye.html>
* <https://tr.lambdageeks.com/critical-actions-class-in-selenium/>
* <https://techproeducation.com/>