**Selenium'da Wait(Bekle) Komutları Nelerdir?**

Bekleme komutları, Selenium testlerinin yürütülmesi söz konusu olduğunda çok önemlidir. Zaman gecikmesindeki değişiklik nedeniyle oluşabilecek sorunları gözlemlemeye ve gidermeye yardımcı olurlar.

Selenium testlerini çalıştırırken, test kullanıcılarının “ Element Not Visible Exception ” mesajını alması yaygındır. Bu, WebDriver'ın etkileşime girmesi gereken belirli bir web öğesi, yüklenmesinde geciktiğinde ortaya çıkar. Bu İstisnayı önlemek için Selenium Bekleme Komutları kullanılmalıdır.

Otomasyon testinde, bir sonraki adıma geçmeden önce belirli bir süre duraklatmak için doğrudan test yürütmesini bekleyin komutları. Bu, WebDriver'ın bir veya daha fazla web öğesinin mevcut/görünür/zenginleştirilmiş/tıklanabilir vb. olup olmadığını kontrol etmesini sağlar.

**Kullanıcılar neden Selenium Wait komutlarına ihtiyaç duyar?**

Bir web sayfası bir tarayıcıya yüklendiğinde, birinin etkileşimde bulunmak istediği çeşitli web öğeleri (düğmeler, bağlantılar, resimler) çeşitli aralıklarla yüklenebilir.

Otomatik Selenium testinde bu , belirli öğeleri tanımlarken bazı sorunlara neden olur. Bir öğe bulunamazsa “ ElementNotVisibleException ” belirir. Selenium Wait komutları bu sorunun çözülmesine yardımcı olur.

* Selenium WebDriver, testlerde beklemeleri uygulamak için üç komut sağlar.

1. Implicit Wait
2. Explicit Wait
3. Fluent Wait

**Selenyumda Örtülü Bekleme (Implicit Wait in Selenium)**

Örtük Bekleme, Selenium Web Sürücüsünü bir istisna oluşturmadan önce belirli bir süre beklemeye yönlendirir. Bu süre ayarlandıktan sonra, WebDriver istisna oluşmadan önce öğeyi bekleyecektir.

Komut yerine getirildikten sonra, Örtülü Bekleme tarayıcının açık olduğu tüm süre boyunca yerinde kalır. Varsayılan ayarı 0'dır ve belirli bekleme süresinin aşağıdaki protokol tarafından ayarlanması gerekir.

Test komut dosyalarına örtük beklemeler eklemek için aşağıdaki paketi içe aktarın.

import java.util.concurrent.TimeUnit;

Syntax

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

Yukarıdaki kodu test komut dosyasına ekleyin. WebDriver örnek değişkeninin somutlaştırılmasından sonra örtük bir bekleme ayarlar.

**Örnek:**

Package waitExample;  
  
import java.util.concurrent.TimeUnit;  
import org.openqa.selenium.\*;  
import org.openqa.selenium.firefox.FirefoxDriver;  
import org.testng.annotations.AfterMethod;  
import org.testng.annotations.BeforeMethod;  
import org.testng.annotations.Test;  
  
public class WaitTest {

private WebDriver driver;  
private String baseUrl;  
private WebElement element;  
  
@BeforeMethod  
public void setUp() throws Exception {  
driver = new FirefoxDriver();  
baseUrl = "http://www.google.com";  
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);  
}  
  
@Test  
public void testUntitled() throws Exception {  
driver.get(baseUrl);  
element = driver.findElement(By.id("lst-ib"));  
element.sendKeys("Selenium WebDriver Interview questions");  
element.sendKeys(Keys.RETURN);  
List<WebElement> list = driver.findElements(By.className("\_Rm"));  
System.out.println(list.size());  
  
}  
  
@AfterMethod  
public void tearDown() throws Exception {  
driver.quit();  
}  
}

Ancak, örtük bekleme, test komut dosyası yürütme süresini artırır. Test yürütmeye devam etmeden önce her komutu tanımlanan süre boyunca bekletir. Uygulama normal şekilde yanıt verirse, örtük bekleme, test komut dosyalarının yürütülmesini yavaşlatabilir.

**Selenyumda Açık Bekleme(Explicit Wait in Selenium)**

Açık Bekle komutu kullanılarak, Web Sürücüsü, kodu yürütmeye devam etmeden önce belirli bir koşul oluşana kadar beklemeye yönlendirilir.

Açık Bekleme ayarı, yüklenmesi doğal olarak daha fazla zaman alan belirli öğelerin olduğu durumlarda önemlidir. Örtük bir bekleme komutu ayarlanırsa, tarayıcı her web öğesini yüklemeden önce aynı zaman çerçevesini bekleyecektir. Bu, test komut dosyasının yürütülmesinde gereksiz bir gecikmeye neden olur.

Açık bekleme daha akıllıdır, ancak yalnızca belirtilen öğeler için uygulanabilir. Ancak, programın dinamik olarak yüklenen Ajax öğeleri için duraklamasına izin verdiği için örtük beklemede bir gelişmedir.

Açık bir bekleme ilan etmek için “ExpectedConditions” kullanmak gerekir. Aşağıdaki Beklenen Koşullar Açık Beklemede kullanılabilir.

* alertIsPresent() uyarı Mevcut
* elementSelectionStateToBe() eleman Seçim Durumunda Olacak
* elementToBeClickable() eleman Tıklanabilir
* elementToBeSelected() Seçilecek eleman
* frameToBeAvaliableAndSwitchToIt() Mevcut Olmak ve Geçiş Yapmak İçin Çerçeve
* invisibilityOfTheElementLocated() Bulunan Öğenin Görünmezliği
* invisibilityOfElementWithText() Metinle Öğenin Görünmezliği
* presenceOfAllElementsLocatedBy() Tarafından Bulunan Tüm Öğelerin Varlığı
* presenceOfElementLocated() Bulunan Öğenin varlığı
* textToBePresentInElement() Metin Öğede Bulunacak
* textToBePresentInElementLocated() Metin Bulunacak Öğede incelenmiş olacak
* textToBePresentInElementValue() textToBePresentInElementValue
* titleIs() başlık
* titleContains() başlık İçerir
* visibilityOf() görünürlüğü
* visibilityOfAllElements() Tüm Elemanların Görünürlüğü
* visibilityOfAllElementsLocatedBy() Tarafından Bulunan Tüm Öğelerin Görünürlüğü
* visibilityOfElementLocated() Bulunan Öğenin Görünürlüğü

1. Test komut dosyalarında Explicit Wait'i kullanmak için aşağıdaki paketleri komut dosyasına içe aktarın.

import org.openqa.selenium.support.ui.ExpectedConditions  
import org.openqa.selenium.support.ui.WebDriverWait

1. Ardından, WebDriverWait Sınıfını kullanarak Bir Bekleme Nesnesi Başlatın.

WebDriverWait wait = new WebDriverWait(driver,30);

Burada, <WebDriverWait> sınıfı için başvuru değişkeni <wait> olarak adlandırılır. WebDriver örneği kullanılarak başlatılır. Yürütmenin işten çıkarılması için maksimum bekleme süresi ayarlanmalıdır. Bekleme süresinin saniye cinsinden ölçüldüğünü unutmayın.

**Örnek:**

Aşağıdaki örnekte, test komut dosyası bir kullanıcı adı ve parola ile "gmail.com"da oturum açmak içindir. Başarılı bir giriş yaptıktan sonra kod, ana sayfada “oluştur” düğmesinin bulunmasını bekler. Sonunda butona tıklar.

package waitExample;  
import java.util.concurrent.TimeUnit;  
import org.openqa.selenium.By;  
import org.openqa.selenium.Keys;  
import org.openqa.selenium.WebDriver;  
import org.openqa.selenium.WebElement;  
import org.openqa.selenium.firefox.FirefoxDriver;  
import org.openqa.selenium.support.ui.ExpectedConditions;  
import org.openqa.selenium.support.ui.WebDriverWait;  
import org.testng.annotations.AfterMethod;  
import org.testng.annotations.BeforeMethod;  
import org.testng.annotations.Test;  
public class ExpectedConditionExample {  
// created reference variable for WebDriver  
WebDriver driver;  
  
@BeforeMethod  
public void setup() throws InterruptedException {  
// initializing driver variable using FirefoxDriver  
driver=new FirefoxDriver();  
// launching gmail.com on the browser  
driver.get("https://gmail.com");  
// maximized the browser window  
driver.manage().window().maximize();  
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);}

@Test  
public void test() throws InterruptedException {  
// saving the GUI element reference into a "element" variable of WebElement type  
WebElement element = driver.findElement(By.id("Email"));  
// entering username  
element.sendKeys("dummy@gmail.com");  
element.sendKeys(Keys.RETURN);  
// entering password  
driver.findElement(By.id("Passwd")).sendKeys("password");  
// clicking signin button  
driver.findElement(By.id("signIn")).click();  
// explicit wait - to wait for the compose button to be click-able  
WebDriverWait wait = new WebDriverWait(driver,30);  
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[contains(text(),'COMPOSE')]")));  
// click on the compose button as soon as the "compose" button is visible  
driver.findElement(By.xpath("//div[contains(text(),'COMPOSE')]")).click();  
}  
  
@AfterMethod  
public void teardown() {  
// closes all the browser windows opened by web driver  
driver.quit();  
}  
  
}

Yukarıdaki kod, Selenium WebDriver'a bir TimeoutException oluşturmadan önce 30 saniye beklemesini söyler. Öğeyi 30 saniyeden önce bulursa hemen geri döner. Bundan sonra, “Oluştur” düğmesine tıklayacaktır. Bu durumda, program 30 saniyenin tamamını beklemez, böylece zamandan tasarruf sağlar ve betiği daha hızlı çalıştırır.

**Selenyumda Akıcı Bekleme(Fluent Wait in Selenium)**

Selenium'da Akıcı Bekleme, Selenium WebDriver'ın belirli bir koşul (web öğesi) görünür hale gelmesi için beklemesi için maksimum süreyi işaretler. Ayrıca, WebDriver'ın " ElementNotVisibleException " atmadan önce koşulun görünüp görünmediğini ne sıklıkta kontrol edeceğini de tanımlar.

Basitçe söylemek gerekirse, Fluent Wait, zaman aşımı gerçekleşene veya nesne bulunana kadar düzenli aralıklarla bir web öğesini tekrar tekrar arar.

Akıcı Bekleme komutları en çok, yüklenmesi daha uzun süreler alabilen web öğeleriyle etkileşim kurarken kullanışlıdır. Bu, Ajax uygulamalarında sıklıkla meydana gelen bir şeydir.

Fluent Wait'i kullanırken, gerektiği gibi varsayılan bir yoklama süresi ayarlamak mümkündür. Kullanıcı, yoklama süresi boyunca herhangi bir istisnayı yoksaymak için beklemeyi yapılandırabilir.

Akıcı beklemelere bazen akıllı beklemeler de denir, çünkü bunlar kodda tanımlanan sürenin tamamını beklemezler. Bunun yerine, test, öğe algılanır algılanmaz içinde belirtilen koşul belirlenir belirlenmez, yürütülmeye devam eder. till(YourCondition) yöntemi doğru olur.

Syntax

Wait wait = new FluentWait(WebDriver reference)  
.withTimeout(timeout, SECONDS)  
.pollingEvery(timeout, SECONDS)  
.ignoring(Exception.class);  
  
WebElement foo=wait.until(new Function<WebDriver, WebElement>() {  
public WebElement applyy(WebDriver driver) {  
return driver.findElement(By.id("foo"));  
}  
});

**Örnek:**

//Declare and initialise a fluent wait  
FluentWait wait = new FluentWait(driver);  
//Specify the timout of the wait  
wait.withTimeout(5000, TimeUnit.MILLISECONDS);  
//Sepcify polling time  
wait.pollingEvery(250, TimeUnit.MILLISECONDS);  
//Specify what exceptions to ignore  
wait.ignoring(NoSuchElementException.class)  
  
//This is how we specify the condition to wait on.  
//This is what we will explore more in this chapter  
wait.until(ExpectedConditions.alertIsPresent());

Source

Bu komut iki ana parametre ile çalışır: zaman aşımı değeri ve yoklama frekansı . Yukarıdaki kod, zaman aşımı değerini 5 saniye ve yoklama sıklığını 0,25 saniye olarak tanımlar. Belirli bir durumu doğrulamak için WebDriver'ı en fazla 5 saniye beklemeye yönlendirir. Koşul bu 5 saniye içinde ortaya çıkarsa, test komut dosyasındaki bir sonraki adımı yürütür. Değilse, “ ElementNotVisibleException ” döndürür.

**Birkaç diğer ilişkili komutlar şunlardır:**

* **PageLoadTimeout Komutu**

Bu komut, WebDriver'ın bir hatayı tetiklemeden önce bir sayfanın tamamen yüklenmesini beklemesi gereken süreyi belirler. Timeout setinin negatif olması durumunda sayfa yükleme süresi belirsiz olabilir.

driver.manage().timeouts().pageLoadTimeout(100, SECONDS);

* **SetScriptTimeout Komutu**

Bu komut, WebDriver'ın bir hatayı tetiklemeden önce eşzamansız bir komut dosyasının yürütmeyi bitirmesini bekleyeceği süreyi belirler. Önceki komut gibi, zaman aşımı negatif bir değere ayarlanırsa komut dosyası süresiz olarak çalışır.

driver.manage().timeouts().setScriptTimeout(100,SECONDS);

* **Sleep (Uyku) Komutu**

Uyku komutu, oldukça etkisiz olduğu için nadiren kullanılır. WebDriver'ın belirli bir süre beklemesine neden olur (ve belirtilen koşul karşılansa bile daha hızlı çalışmasına izin vermez). Aslında, Selenium bekleme komutları, Uyku komutuna göre daha akıllı ve daha etkili bir alternatif olarak kabul edilir.

thread.sleep(1000);

Selenium'da Örtülü(Implicit Wait) ve Açık(Explicit Wait) Bekleme Komutları Arasındaki Fark

|  |  |
| --- | --- |
| **Örtük bekle** (Implicit Wait) | **Açık Bekleme** (Explicit Wait) |
| Bir test komut dosyasındaki tüm öğeler için geçerlidir. | Kullanıcının amaçladığı şekilde yalnızca belirli öğeler için geçerlidir. |
| Bulunacak eleman üzerinde "ExpectedConditions" belirtmeye gerek yok | Bulunacak öğede her zaman “ExpectedConditions” belirtilmelidir |
| Örtük beklemede belirtilen zaman çerçevesi ile öğelerin bulunduğu bir test durumunda kullanıldığında en etkilidir. | Öğelerin yüklenmesi uzun zaman aldığında kullanıldığında en etkilidir. *Ayrıca, componentOfElementLocated, elementToBeClickable,elementToBeSelected* gibi öğenin özelliklerini doğrulamak için de kullanışlıdır. |

**Hap Bilgi:**

1. Selenium waits, (senkronizasyon): Kodlarımızla, internet ve bilgisayarın çalışmasını uyumlu hale getirmek için bazen kodlarımızı bekletmemiz gerekir.

2. Seleniumda kullandığımız 3 çesit wait vardır.

**1. Thread.sleep()**: Java’dan gelir ve kodları durdurur. Kendisine verilen surenin tamamını bekler. Dinamik değil statiktir.

**2. implicitlyWait:** Selenium’dan gelir. Sayfanın yüklenmesi ve her bir webElementin locate edilmesi için beklenecek max. süreyi belirler. Esnektir. Esas olan beklemek değil, kodun çalışmasını sağlamaktır. Kod çalışırsa beklemeden devam eder, kod çalışmazsa max.süre doluncaya bekler ve sonra hata mesajı verir(Exception).

**3. ExplicitlyWait :** Selenium’dan gelir, spesifik bir işlem için beklenecek max. süreyi belirler. ExplicitlyWait kullanabilmek için wait objesi oluşturmamız gerekir. wait.until(exoectedConditions….. istediğimiz fonk.) şeklinde kullanılır. Genellikle wait.until() method’u locate ile birlikte kullanılır.

**Kaynak:**

* <https://www.browserstack.com/guide/wait-commands-in-selenium-webdriver>
* <https://kadertarlan.wordpress.com/2020/03/25/selenium-waits-implicit-explicit/>
* <https://techproeducation.com/>

***Hazırlayan: Zeliha Öznük***