EBook Gratuito

APPRENDIMENTO selenium-webdriver

Free unaffiliated eBook created from
Stack Overflow contributors.

#selenium-webdriver

Sommario

Di

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: selenium-webdriver

It is an unofficial and free selenium-webdriver ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official selenium-webdriver.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capitolo 1: Iniziare con selenio-webdriver

Osservazioni

Questa sezione fornisce una panoramica su cosa sia il selenio-webdriver e perché uno sviluppatore potrebbe volerlo utilizzare.

Dovrebbe anche menzionare eventuali soggetti di grandi dimensioni all'interno del selenio-webdriver e collegarsi agli argomenti correlati. Poiché la Documentazione per selenio-webdriver è nuova, potrebbe essere necessario creare versioni iniziali di tali argomenti correlati.

Versioni

Versione	Data di rilascio
0.0.1	2016/08/03

Examples

Installazione o configurazione

Per iniziare a utilizzare WebDriver è necessario ottenere il driver pertinente dal sito Selenium : Selenium HQ Downloads . Da qui è necessario scaricare il driver relativo ai browser e / o alle piattaforme su cui si sta tentando di eseguire WebDriver, ad esempio se si eseguiva il test in Chrome, il sito Selenium ti indirizzerà a:

https://sites.google.com/a/chromium.org/chromedriver/

Per scaricare `chromedriver.exe` .

Infine, prima di poter utilizzare WebDriver, è necessario scaricare i collegamenti linguistici pertinenti, ad esempio se si utilizza C # è possibile accedere al download dalla pagina Download Selenium HQ per ottenere i file .dll richiesti o, in alternativa, scaricarli come pacchetti in Visual Studio tramite il gestore pacchetti NuGet.

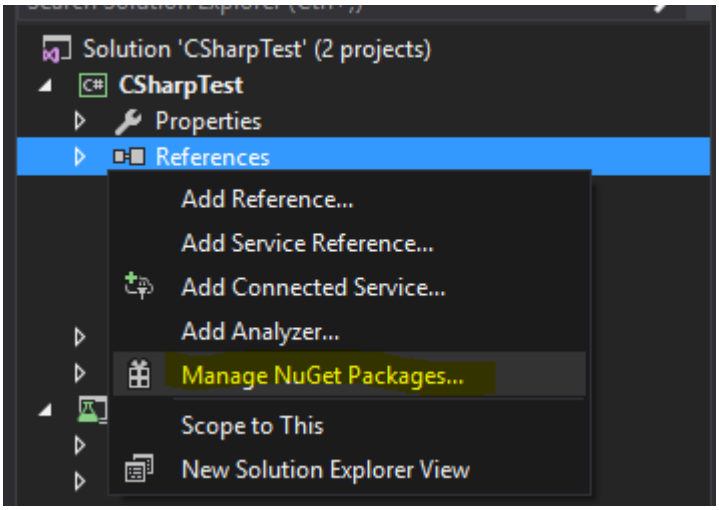
I file richiesti ora dovrebbero essere scaricati, per informazioni su come iniziare a usare WebDriver, fare riferimento all'altra documentazione sul `selenium-webdriver` .

Per Visual Studio [NuGet]

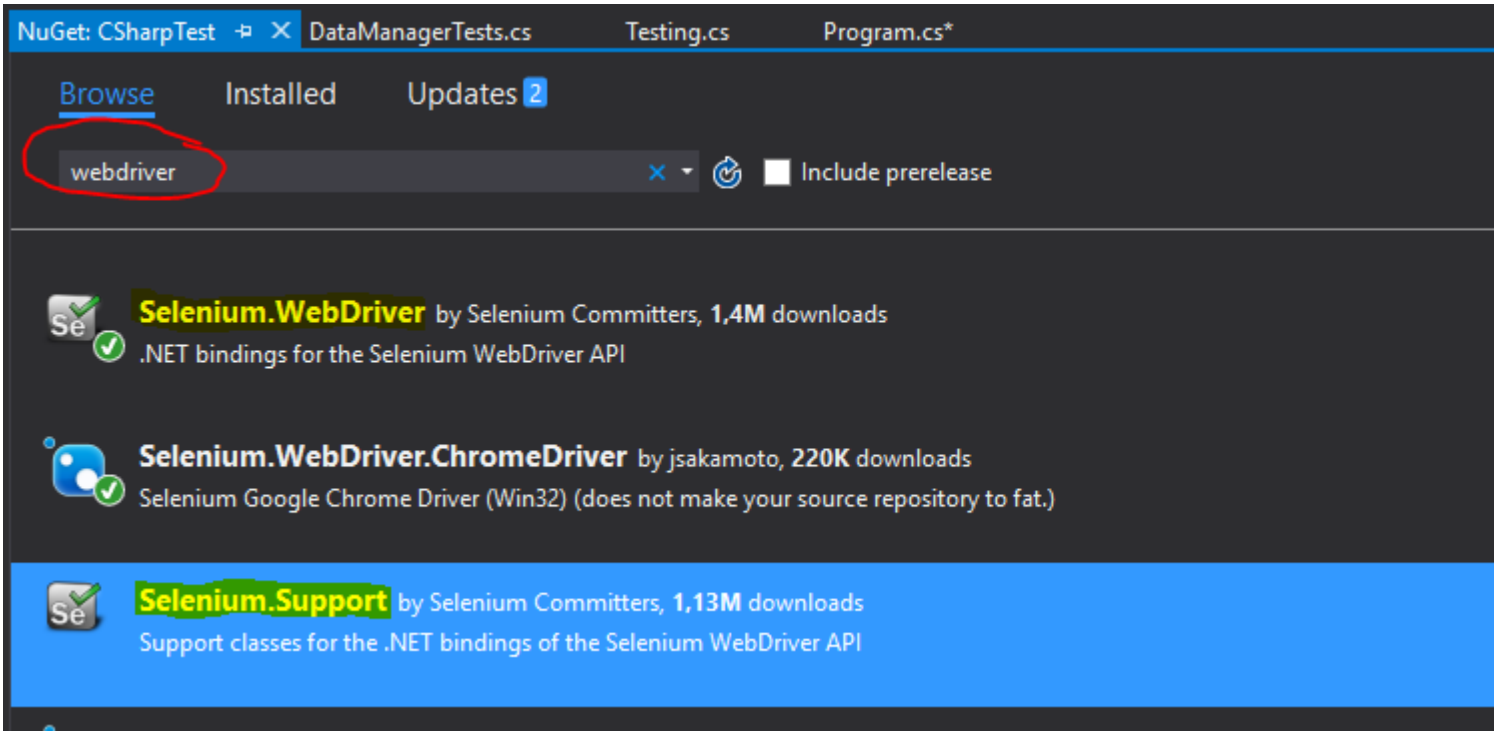
Il modo più semplice per installare Selenium WebDriver è utilizzare un gestore di pacchetti NuGet.

Nel tuo progetto, fai clic con il pulsante destro del mouse su "Riferimenti" e fai clic su "Gestisci

pacchetti NuGet" come mostrato:



Quindi, digita nella casella di ricerca " *webdriver* ". Dovresti quindi vedere qualcosa di simile a questo:



Installa " **Selenium.WebDriver** " e " **Selenium.Support** " (il pacchetto di supporto include risorse aggiuntive, come Wait) facendo clic sul pulsante Installa sul lato destro.

Quindi è possibile installare i WebDriver che si desidera utilizzare, ad esempio uno di questi:

- Selenium.WebDriver.ChromeDriver (Google Chrome)
- PhantomJS (senza testa)
-

Cos'è Selenium WebDriver?

Il selenio è un insieme di strumenti progettati per automatizzare i browser. Viene comunemente utilizzato per i test delle applicazioni Web su più piattaforme. Ci sono alcuni strumenti disponibili sotto l'ombrello Selenium, come Selenium WebDriver (ex Selenium RC), Selenium IDE e Selenium Grid.

WebDriver è *un'interfaccia di* controllo remoto che consente di manipolare elementi DOM nelle pagine Web e di comandare il comportamento dei programmi utente. Questa interfaccia fornisce un protocollo wire-less che è stato implementato per varie piattaforme come:

- GeckoDriver (Mozilla Firefox)
- ChromeDriver (Google Chrome)
- SafariDriver (Apple Safari)
- InternetExplorerDriver (MS InternetExplorer)
- MicrosoftWebDriver o EdgeDriver (MS Edge)
- OperaChromiumDriver (browser Opera)

così come altre implementazioni:

- EventFiringWebDriver
- HtmlUnitDriver
- PhantomJSDriver
- RemoteWebDriver

Selenium WebDriver è uno degli strumenti Selenium che fornisce API orientate agli oggetti in una varietà di lingue per consentire un maggiore controllo e l'applicazione di pratiche di sviluppo software standard. Per simulare con precisione il modo in cui un utente interagirà con un'applicazione web, utilizza "Eventi di livello del sistema operativo nativo" come opposto a "Eventi JavaScript sintetizzati".

Collegamenti da riferire:

- http://www.seleniumhq.org/
- http://www.aosabook.org/en/selenium.html
- https://www.w3.org/TR/webdriver/

Installazione o configurazione per Java

Per scrivere test utilizzando Selenium Webdriver e Java come linguaggio di programmazione, è necessario scaricare i file JAR di Selenium Webdriver dal sito Web Selenium.

Esistono diversi modi per configurare un progetto Java per il web driver Selenium, uno dei più facili tra tutti utilizza Maven. Maven scarica i binding Java richiesti per il web driver Selenium incluse tutte le dipendenze. L'altro modo è scaricare i file JAR e importarli nel progetto.

Procedura per impostare il progetto Selenium Webdriver utilizzando Maven:

1. Installa Maven su Windows Box seguendo questo documento: https://maven.apache.org/install.html
2. Crea una cartella con il nome `selenium-learing`

3. Crea un file nella cartella sopra usando qualsiasi editor di testo con nome `pom.xml`
4. Copia sotto il contenuto di `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>SeleniumLearning</groupId>
            <artifactId>SeleniumLearning</artifactId>
            <version>1.0</version>
            <dependencies>
                <dependency>
                    <groupId>org.seleniumhq.selenium</groupId>
                    <artifactId>selenium-learning</artifactId>
                    <version>3.0.0-beta1</version>
                </dependency>
            </dependencies>
    </project>
```

Nota : assicurati che la versione che hai specificato sopra sia la più recente. Puoi controllare l'ultima versione da qui: http://docs.seleniumhq.org/download/maven.jsp

5. Usando la riga di comando, esegui sotto il comando nella directory del progetto.

   ```
   mvn clean install
   ```

Il comando precedente scaricherà tutte le dipendenze richieste e lo aggiungerà al progetto.

6. Scrivi sotto il comando per generare un progetto eclipse che puoi importare nell'IDE di Eclipse.

   ```
   mvn eclipse:eclipse
   ```

7. Per importare il progetto in ide di eclissi, puoi seguire i passaggi seguenti

   Apri Elipse -> File -> Importa -> Generale -> Progetto esistente nell'area di lavoro -> Avanti -> Sfoglia -> Trova la cartella contiene pom.xml -> Ok -> Fine

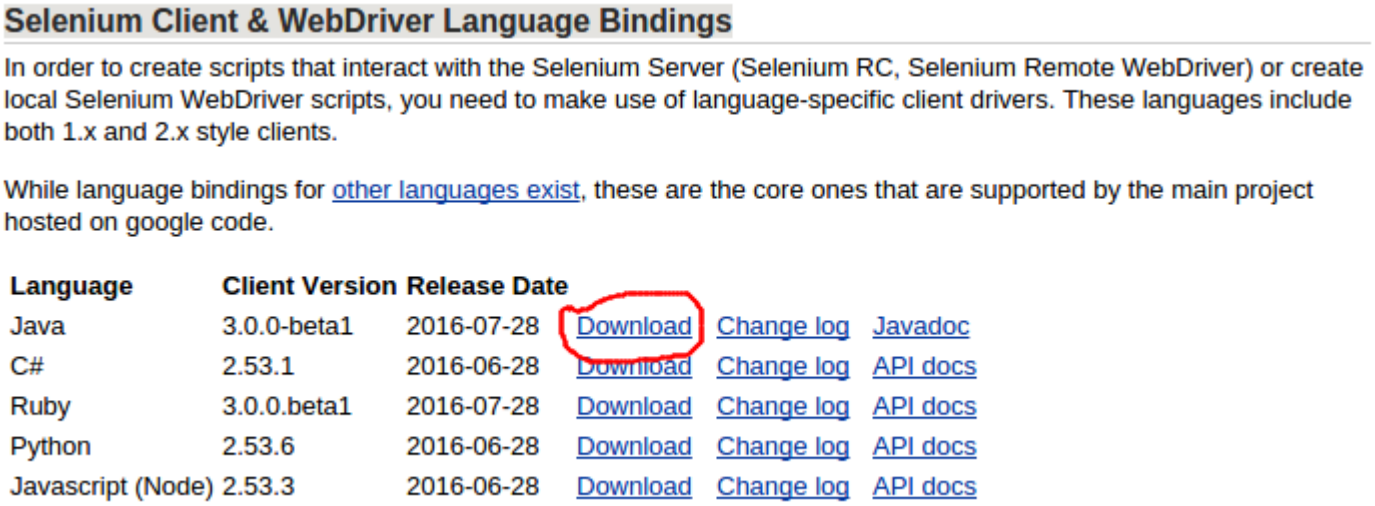
Installa il plug-in m2eclipse facendo clic con il pulsante destro del mouse sul progetto e seleziona Maven -> Abilita gestione dipendenze.

Passi per impostare il progetto Selenium Webdriver usando i file Jar

1. Crea un nuovo progetto in Eclipse seguendo i passaggi seguenti.

   Apri Elipse -> File -> Nuovo -> Progetto Java -> Fornisci un nome (selenio-apprendimento) -> Fine

2. Scarica i file jar da http://www.seleniumhq.org/download/ . È necessario scaricare sia **Selenium Standalone Server** che **Selenium Client e WebDriver Bindings** . Poiché questo documento parla di Java, è necessario scaricare solo jar dalla sezione Java. Dai uno sguardo allo screenshot allegato.



Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date			
Java	3.0.0-beta1	2016-07-28	Download	Change log	Javadoc
C#	2.53.1	2016-06-28	Download	Change log	API docs
Ruby	3.0.0.beta1	2016-07-28	Download	Change log	API docs
Python	2.53.6	2016-06-28	Download	Change log	API docs
Javascript (Node)	2.53.3	2016-06-28	Download	Change log	API docs

Nota: il server autonomo Selenium è richiesto solo se si desidera utilizzare il server remoto per eseguire i test. Dal momento che questo documento è tutto sopra l'impostazione del progetto, è meglio avere tutto a posto.

3. I vasi verranno scaricati nel file zip, decomprimili. Dovresti essere in grado di vedere `.jar` direttamente.

4. In eclissi, fai clic con il pulsante destro del mouse sul progetto che hai creato nel passaggio 1 e segui i passaggi seguenti.

   Proprietà -> Percorso build Java -> scheda Librerie Seleziona -> Fai clic su Aggiungi jar esterni -> Individua la cartella jar decompressa che hai scaricato in precedenza -> Seleziona tutti i jar dalla cartella `lib` -> Fai clic su OK -> Fai di nuovo clic su Aggiungi jar esterni - > Trova la stessa cartella decompressa -> Seleziona il jar che si trova all'esterno della cartella lib (`client-combined-3.0.0-beta1-nodeps.jar`) -> Ok

Allo stesso modo aggiungere il `Selenium Standalone Server` seguendo il passaggio precedente.

5. Ora puoi iniziare a scrivere il codice del selenio nel tuo progetto.

PS : la documentazione di cui sopra si basa sulla versione beta 3.0 di selenio, pertanto i nomi dei file jar specificati potrebbero cambiare con la versione.

Leggi Iniziare con selenio-webdriver online: https://riptutorial.com/it/selenium-webdriver/topic/878/iniziare-con-selenio-webdriver

Capitolo 2: Aspettare

Examples

Tipi di attesa in Selenium WebDriver

Durante l'esecuzione di qualsiasi applicazione Web è necessario prendere in considerazione il tempo di caricamento. Se il tuo codice tenta di accedere a qualsiasi elemento che non è ancora stato caricato, WebDriver genererà un'eccezione e il tuo script si fermerà.

Esistono tre tipi di attese:

- **Attese implicite**
- **Attese esplicite**
- **Attese fluide**

Le attese implicite vengono utilizzate per impostare il tempo di attesa in tutto il programma, mentre le attese esplicite vengono utilizzate solo su parti specifiche.

Attesa implicita

Un'attesa implicita è di dire a WebDriver di interrogare il DOM per una certa quantità di tempo quando si cerca di trovare un elemento o elementi se non sono immediatamente disponibili. Le attese implicite sono fondamentalmente il tuo modo di dire a WebDriver la latenza che vuoi vedere se l'elemento web specificato non è presente che WebDriver sta cercando. L'impostazione predefinita è 0. Una volta impostata, l'attesa implicita viene impostata per la durata dell'istanza dell'oggetto WebDriver. L'attesa implicita viene dichiarata nella parte di istanziazione del codice utilizzando il seguente snippet.

Esempio in **Java** :

```
driver.manage().timeouts().implicitlyWait(15, TimeUnit.SECONDS);
// You need to import the following class - import java.util.concurrent.TimeUnit;
```

Esempio in **C #** :

```
driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(15));
```

Quindi in questo caso, stai dicendo a WebDriver che dovrebbe aspettare 15 secondi nel caso in cui l'elemento specificato non sia disponibile sull'interfaccia utente (DOM).

Attesa esplicita

Potresti incontrare istanze quando alcuni elementi richiedono più tempo per essere caricati. L'impostazione di attesa implicita per questi casi non ha senso poiché il browser attenderà inutilmente lo stesso tempo per ogni elemento, aumentando il tempo di automazione. L'attesa esplicita aiuta qui aggirando l'attesa implicita del tutto per alcuni elementi specifici.

Le attese esplicite sono attese intelligenti che sono confinate a un particolare elemento web. Usando le attese esplicite stai praticamente dicendo a WebDriver al massimo è di aspettare X unità di tempo prima che si arrenda.

Le attese esplicite vengono eseguite utilizzando le classi WebDriverWait e ExpectedConditions. Nell'esempio seguente, dovremo attendere fino a 10 secondi per un elemento il cui id è username per diventare visibile prima di passare al comando successivo. Ecco i passaggi.

Esempio in **Java** :

```
//Import these two packages:
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;

//Declare a WebDriverWait variable. In this example, we will use myWaitVar as the name of the
variable.
WebDriverWait myWaitVar = new WebDriverWait(driver, 30);

//Use myWaitVar with ExpectedConditions on portions where you need the explicit wait to occur.
In this case, we will use explicit wait on the username input before we type the text tutorial
onto it.
myWaitVar.until(ExpectedConditions.visibilityOfElementLocated(By.id(“username”)));
driver.findElement(By.id(“username”)).sendKeys(“tutorial”);
```

La classe ExpectedConditions ha alcune condizioni comuni predefinite per attendere un elemento. Fare clic qui per visualizzare l'elenco di queste condizioni nell'associazione Java.

Esempio in **C #** :

```
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.PhantomJS;

// You can use any other WebDriver you want, such as ChromeDriver.
using (var driver = new PhantomJSDriver())
{
    driver.Navigate().GoToUrl("http://somedomain/url_that_delays_loading");

    // We aren't going to use it more than once, so no need to declare this a variable.
    new WebDriverWait(driver, TimeSpan.FromSeconds(10))
        .Until(ExpectedConditions.ElementIsVisible(By.Id("element-id")));

    // After the element is detected by the previous Wait,
    // it will display the element's text
    Console.WriteLine(driver.FindElement(By.Id("element-id")).Text);
}
```

In questo esempio, il sistema attenderà per 10 secondi finché l'elemento non sarà visibile. Se l'elemento non sarà visibile dopo il timeout, il WebDriver genererà un'eccezione `WebDriverTimeoutException`

.

Nota: se l'elemento è visibile prima del timeout di 10 secondi, il sistema procederà immediatamente per ulteriori processi.

Aspetta fluente

Diversamente dall'attesa implicita ed esplicita, l'attesa fluente utilizza due parametri. Valore di timeout e frequenza di polling. Diciamo che abbiamo il valore di timeout come 30 secondi e la frequenza di polling di 2 secondi. WebDriver controllerà l'elemento ogni 2 secondi fino al valore di timeout (30 secondi). Dopo che il valore di timeout è stato superato senza alcun risultato, viene generata un'eccezione. Di seguito è riportato un codice di esempio che mostra l'implementazione dell'attesa fluente.

Esempio in **Java** :

```
Wait wait = new FluentWait(driver).withTimeout(30, SECONDS).pollingEvery(2,
SECONDS).ignoring(NoSuchElementException.class);


WebElement testElement = wait.until(new Function() {
    public WebElement apply(WebDriver driver) {
        return driver.findElement(By.id("testId"));
    }
});
```

Un altro vantaggio dell'utilizzo dell'attesa fluente è che possiamo ignorare determinati tipi di eccezioni (ad esempio NoSuchElementExceptions) durante l'attesa. A causa di tutte queste disposizioni, l'attesa fluente è utile nelle applicazioni AJAX e negli scenari in cui il tempo di carico degli elementi oscilla spesso. L'uso strategico dell'attesa fluente migliora in modo significativo gli sforzi di automazione.

Diversi tipi di condizioni di attesa esplicite

Nell'attesa esplicita, ci si aspetta che una condizione si verifichi. Ad esempio, si desidera attendere fino a quando un elemento è cliccabile.

Ecco una dimostrazione di alcuni problemi comuni.

Si prega di notare: In tutti questi esempi è possibile utilizzare qualsiasi `By` *come un localizzatore, come il* `classname` , `xpath` , `link text` , `tag name` o `cssSelector`

Attendi fino a quando l'elemento è visibile

Ad esempio, se il tuo sito web richiede un po 'di tempo per essere caricato, puoi aspettare fino a quando la pagina completa il caricamento e il tuo elemento è visibile sul WebDriver.

C #

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.ElementIsVisible(By.Id("element-id")));
```

Giava

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("element-id")));
```

Attendi fino a quando l'elemento non è più visibile

Come prima, ma invertito.

C #

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.InvisibilityOfElementLocated(By.Id("element-id")));
```

Giava

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.invisibilityOfElementLocated(By.id("element-id")));
```

Attendere fino a quando il testo è presente nell'elemento specificato

C #

```
IWebElement element = driver.FindElement(By.Id("element-id"));

WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait.Until(ExpectedConditions.TextToBePresentInElement(element, "text"));
```

Giava

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

WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.textToBePresentInElement(element, "text"));
```

Se vai al link indicato sopra, vedrai tutte le condizioni di attesa lì.

La differenza tra l'utilizzo di queste condizioni di attesa è nel loro parametro di input.

Ciò significa che devi passare il WebElement se il suo parametro di input è WebElement, devi

passare il locator degli elementi se prende il locator By come parametro di input.

Scegli con saggezza il tipo di condizione di attesa che desideri utilizzare.

In attesa di richieste Ajax da completare

C

```
using OpenQA.Selenium
using OpenQA.Selenium.Chrome;
using System.Threading;

namespace WebDriver Tests
{
    class WebDriverWaits
    {
        static void Main()
        {
            IWebDriver driver = new ChromeDriver(@"C:\WebDriver");

            driver.Navigate().GoToUrl("page with ajax requests");
            CheckPageIsLoaded(driver);

            // Now the page is fully loaded, you can continue with further tests.
        }

        private void CheckPageIsLoaded(IWebDriver driver)
        {
            while (true)
            {
                bool ajaxIsComplete = (bool)(driver as
IJavaScriptExecutor).ExecuteScript("return jQuery.active == 0");
                if (ajaxIsComplete)
                    return;
                Thread.Sleep(100);
            }
        }
    }
}
```

Questo esempio è utile per le pagine in cui vengono fatte richieste ajax, qui usiamo `IJavaScriptExecutor` per eseguire il nostro codice JavaScript. Poiché è all'interno di un ciclo `while` , continuerà a funzionare fino a `ajaxIsComplete == true` e quindi l'istruzione return viene eseguita.

Controlliamo che tutte le richieste ajax siano complete confermando che `jQuery.active` è uguale a `0` . Questo funziona perché ogni volta che viene fatta una nuova richiesta `jQuery.active` viene incrementato e ogni volta che una richiesta viene completata viene decrementato, da questo possiamo dedurre che quando `jQuery.active == 0` tutte le richieste `jQuery.active == 0` devono essere completate.

Aspetta fluente

L'attesa fluente è una superclasse di attesa **esplicita** (`WebDriverWait`) che è più configurabile

poiché può accettare un argomento per la funzione di attesa. Trascorrerò un'attesa **implicita** , dal momento che è una buona pratica evitarla.

Utilizzo (Java):

```
Wait wait = new FluentWait<>(this.driver)
        .withTimeout(driverTimeoutSeconds, TimeUnit.SECONDS)
        .pollingEvery(500, TimeUnit.MILLISECONDS)
        .ignoring(StaleElementReferenceException.class)
        .ignoring(NoSuchElementException.class)
        .ignoring(ElementNotVisibleException.class);

WebElement foo = wait.until(ExpectedConditions.presenceOfElementLocated(By.yourBy));

// or use your own predicate:
WebElement foo = wait.until(new Function() {
  public WebElement apply(WebDriver driver) {
    return element.getText().length() > 0;
  }
});
```

Quando usi Explicit wait con i suoi valori predefiniti è semplicemente un `FluentWait<WebDriver>` con valori predefiniti di: `DEFAULT_SLEEP_TIMEOUT = 500;` e ignorando `NotFoundException` .

Aspetta fluente

Ogni istanza di FluentWait definisce la quantità massima di tempo di attesa per una condizione, nonché la frequenza con cui controllare la condizione. Inoltre, l'utente può configurare l'attesa per ignorare determinati tipi di eccezioni durante l'attesa, come NoSuchElementExceptions durante la ricerca di un elemento nella pagina. È associato al driver.

```
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
.withTimeout(30, SECONDS) //actuall wait for the element to pe present
.pollingEvery(5, SECONDS) //selenium will keep looking for the element after every 5seconds
.ignoring(NoSuchElementException.class); //while ignoring this condition
wait.until(ExpectedConditions.visibilityOf(driver.findElement(By.id("username"));
```

Leggi Aspettare online: https://riptutorial.com/it/selenium-webdriver/topic/4435/aspettare

Capitolo 3: Azioni (Emulazione di gesti complessi dell'utente)

introduzione

La classe `Actions` ci offre un modo di emulare esattamente come un utente interagirebbe con una pagina web / elementi. Usando un'istanza di questa classe puoi descrivere una serie di azioni, come fare clic, fare doppio clic, trascinare, premere i tasti, ecc. Una volta descritte queste azioni, per eseguire le azioni, devi chiamare deve costruire le azioni (`.Build()`) e quindi istruirli affinché vengano eseguiti (`.Perform()`). Quindi dobbiamo descrivere, costruire, eseguire. Gli esempi seguenti si espanderanno su questo.

Sintassi

- dragAndDrop (origine WebElement, destinazione WebElement)
- dragAndDropBy (origine WebElement, int xOffset, int yOffset)
- eseguire()

Parametri

parametri	Dettagli
fonte	Elemento per emulare il pulsante in basso a.
bersaglio	Elemento in cui spostarsi e rilasciare il mouse su.
xOffset	x coordinata per spostarsi a.
YOffset	y coordinare per passare a.

Osservazioni

Questa sezione contiene informazioni sulla classe Actions di Selenium WebDriver. La classe Actions ti offre metodi convenienti per eseguire gesti utente complessi come trascina e rilascia, tieni premuto e fai clic su altro.

Examples

Trascinare e rilasciare

C

```
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Interactions;

namespace WebDriverActions
{
    class WebDriverTest
    {
        static void Main()
        {
            IWebDriver driver = new FirefoxDriver();

            driver.Navigate().GoToUrl("");
            IWebElement source = driver.FindElement(By.CssSelector(""));
            IWebElement target = driver.FindElement(By.CssSelector(""));
            Actions action = new Actions(driver);
            action.DragAndDrop(source, target).Perform();
        }
    }
}
```

Quanto sopra troverà un `IWebElement` , `source` , e lo trascinerà a, e lo `IWebElement` nel secondo `IWebElement` , `target` .

Giava

Trascina e rilascia utilizzando il webelement di origine e di destinazione.

Un metodo pratico che esegue il click-and-hold nella posizione dell'elemento sorgente, si sposta nella posizione dell'elemento di destinazione, quindi rilascia il mouse.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.interactions.Actions;

/**
 * Drag and Drop test using source and target webelement
 */
public class DragAndDropClass {
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.get("");
        WebElement source = driver.findElement(By.cssSelector(""));
        WebElement target = driver.findElement(By.cssSelector(""));
        Actions action = new Actions(driver);
        action.build();
        action.dragAndDrop(source, target).perform();
    }
}
```

Trascina un elemento e rilasciarlo a un dato offset.

Un metodo pratico che esegue il click-and-hold nella posizione dell'elemento sorgente, si sposta di

un dato offset (xey, entrambi gli interi), quindi rilascia il mouse.

```
WebElement source = driver.findElement(By.cssSelector(""));
Actions action = new Actions(driver);
action.build()
action.dragAndDropBy(source, x, y).perform(); // x and y are integers value
```

Sposta in elemento

C

Supponiamo di voler verificare che quando si passa con il mouse su un elemento, viene visualizzato un elenco a discesa. Potresti voler controllare il contenuto di questo elenco, o forse selezionare un'opzione dall'elenco.

Per prima cosa crea un'azione, passa il mouse sopra l'elemento *(ad es. Il mio elemento ha il testo del link "Admin")* :

```
Actions mouseHover = new Actions(driver);
mouseHover.MoveToElement(driver.FindElement(By.LinkText("Admin"))).Perform();
```

Nell'esempio sopra:

- Hai creato l'azione `mouseHover`
- Hai detto al `driver` di passare a un elemento specifico
- Da qui puoi eseguire altre `Actions` con l'oggetto `mouseHover` o continuare a testare con l'oggetto `driver`

Questo approccio è di particolare utilità quando si fa clic su un elemento che esegue una funzione diversa rispetto a passare sopra di esso.

Un esempio completo:

```
Actions mouseHover = new Actions(driver);
mouseHover.MoveToElement(driver.FindElement(By.LinkText("Admin"))).Perform();

Assert.IsTrue(driver.FindElement(By.LinkText("Edit Record")).Displayed);
Assert.IsTrue(driver.FindElement(By.LinkText("Delete Record")).Displayed);
```

Leggi Azioni (Emulazione di gesti complessi dell'utente) online: https://riptutorial.com/it/selenium-webdriver/topic/4849/azioni--emulazione-di-gesti-complessi-dell-utente-

Capitolo 4: Browser senza testa

Examples

PhantomJS [C #]

`PhantomJS` è un browser Web headless con funzionalità complete **con** supporto JavaScript.

Prima di iniziare, devi scaricare un driver PhantomJS e assicurati di inserirlo all'inizio del codice:

```
using OpenQA.Selenium;
using OpenQA.Selenium.PhantomJS;
```

Ottimo, ora sull'inizializzazione:

```
var driver = new PhantomJSDriver();
```

Questo semplicemente creerà una nuova istanza della classe PhantomJSDriver. È quindi possibile utilizzarlo allo stesso modo di ogni WebDriver come:

```
using (var driver = new PhantomJSDriver())
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Funziona bene Tuttavia, il problema che probabilmente hai riscontrato è che, quando si lavora con l'interfaccia utente, `PhantomJS` apre una nuova finestra della console, che nella maggior parte dei casi non è richiesta. Fortunatamente, possiamo nascondere la finestra e persino migliorare leggermente le prestazioni usando `PhantomJSOptions` e `PhantomJSDriverService` . Esempio di lavoro completo di seguito:

```
// Options are used for setting "browser capabilities", such as setting a User-Agent
// property as shown below:
var options = new PhantomJSOptions();
options.AddAdditionalCapability("phantomjs.page.settings.userAgent",
"Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:25.0) Gecko/20100101 Firefox/25.0");

// Services are used for setting up the WebDriver to your likings, such as
// hiding the console window and restricting image loading as shown below:
var service = PhantomJSDriverService.CreateDefaultService();
service.HideCommandPromptWindow = true;
service.LoadImages = false;
```

```
// The same code as in the example above:
using (var driver = new PhantomJSDriver(service, options))
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

`PhantomJSDriverService` : fai clic su una classe (ad es. `PhantomJSDriverService`) e premi F12 per vedere esattamente cosa contengono e una breve descrizione di ciò che fanno.

SimpleBrowser [C #]

`SimpleBrowser` è un WebDriver leggero **senza** supporto JavaScript.

È considerevolmente più veloce di un `PhantomJS` sopra, tuttavia quando si tratta di funzionalità, è limitato a semplici compiti senza caratteristiche di fantasia.

In primo luogo, è necessario scaricare il pacchetto SimpleBrowser.WebDriver e quindi inserire questo codice all'inizio:

```
using OpenQA.Selenium;
using SimpleBrowser.WebDriver;
```

Ora, ecco un breve esempio su come usarlo:

```
using (var driver = new SimpleBrowserDriver())
{
    driver.Navigate().GoToUrl("http://stackoverflow.com/");

    var questions = driver.FindElements(By.ClassName("question-hyperlink"));

    foreach (var question in questions)
    {
        // This will display every question header on StackOverflow homepage.
        Console.WriteLine(question.Text);
    }
}
```

Browser senza testa in Java

HTMLUnitDriver

HTMLUnitDriver è l'implementazione più leggera del browser headless (senza GUI) per Webdriver basato su HtmlUnit. Modella i documenti HTML e fornisce un'API che consente di richiamare

pagine, compilare moduli, fare clic su collegamenti, ecc. Proprio come fai nel tuo normale browser. Supporta JavaScript e funziona con le librerie AJAX. Viene utilizzato per testare e recuperare dati dal sito web.

Esempio: utilizzo di HTMLUnitDriver per scaricare l'elenco di domande da `http://stackoverflow.com/` .

```
import java.util.List;
import java.util.concurrent.TimeUnit;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.htmlunit.HtmlUnitDriver;

class testHeadlessDriver{
            private void getQuestions() {
                    WebDriver driver = new HtmlUnitDriver();
                    driver.get("http://stackoverflow.com/");
                    driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);
                    List<WebElement> questions = driver.findElements(By.className("question-
hyperlink"));
                    questions.forEach((question) -> {
                        System.out.println(question.getText());
                    });
                   driver.close();
                }
    }
```

È uguale a qualsiasi altro browser (Mozilla Firefox, Google Chrome, IE), ma non ha GUI, l'esecuzione non è visibile sullo schermo.

Leggi Browser senza testa online: https://riptutorial.com/it/selenium-webdriver/topic/3931/browser-senza-testa

Capitolo 5: Cambio di frame

Sintassi

- **Giava**
- driver.switchTo (). frame (nome stringa);
- driver.switchTo (). frame (String id);
- driver.switchTo (). frame (int index);
- driver.switchTo (). frame (WebElement frameElement);
- . Driver.switchTo () defaultContent ();
- **C #**
- driver.SwitchTo (). Frame (int frameIndex);
- driver.SwitchTo (). Frame (IWebElement frameElement);
- driver.SwitchTo (). Frame (string frameName);
- . Driver.SwitchTo () DefaultContent ();
- **Pitone**
- driver.switch_to_frame (nameOrId)
- driver.switch_to.frame (nameOrId)
- driver.switch_to_frame (indice)
- driver.switch_to.frame (indice)
- driver.switch_to_frame (frameElement)
- driver.switch_to.frame (frameElement)
- driver.switch_to_default_content ()
- driver.switch_to.default_content ()
- **JavaScript**
- driver.switchTo (). telaio (nameOrId)
- driver.switchTo (). telaio (index)
- driver.switchTo (). defaultContent ()

Parametri

parametro	dettagli
nameOrId	Seleziona un frame con il suo nome di id.
indice	Seleziona un frame per il suo indice a base zero.
frameElement	Seleziona una cornice usando il suo WebElement precedentemente posizionato

Examples

Per passare a una cornice utilizzando Java

Per un'istanza, se il codice sorgente HTML di una vista o di un elemento html è racchiuso da un iframe come questo:

```
<iframe src="../images/eightball.gif" name="imgboxName" id="imgboxId">
   <p>iframes example</p>
   <a href="../images/redball.gif" target="imgbox">Red Ball</a>
</iframe><br />
```

... quindi per eseguire qualsiasi azione sugli elementi web dell'iframe, devi prima spostare lo stato attivo sull'iframe, utilizzando uno dei seguenti metodi:

Usando frame ID (dovrebbe essere usato solo se conosci l'id dell'iframe).

```
driver.switchTo().frame("imgboxId"); //imgboxId - Id of the frame
```

Usando il nome del frame (dovrebbe essere usato solo se conosci il nome dell'iframe).

```
driver.switchTo().frame("imgboxName"); //imgboxName - Name of the frame
```

L'uso di frame index (dovrebbe essere usato solo se non si ha l'id o il nome dell'iframe), dove l'indice definisce la posizione dell'iframe tra tutti i frame.

```
driver.switchTo().frame(0); //0 - Index of the frame
```

Nota: se nella pagina sono presenti tre frame, il primo frame sarà all'indice 0, il secondo all'indice 1 e il terzo all'indice 2.

Usando il webelement precedentemente posizionato (dovrebbe essere usato solo se hai già localizzato il frame e lo hai restituito come `WebElement`).

```
driver.switchTo().frame(frameElement); //frameElement - webelement that is the frame
```

Quindi, per fare clic sull'ancora `Red Ball` :

```
driver.switchTo().frame("imgboxId");
driver.findElement(By.linkText("Red Ball")).Click();
```

Per uscire da una cornice usando Java

Per spostare lo stato attivo su un documento principale o il primo fotogramma della pagina. Devi usare la sintassi qui sotto.

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

Passa a una cornice usando C

1. Passa a una cornice per Indice.

Qui stiamo passando all'indice 1. L'indice si riferisce all'ordine dei frame sulla pagina. Questo dovrebbe essere usato come ultima risorsa, in quanto id frame o nomi sono molto più affidabili.

```
driver.SwitchTo().Frame(1);
```

2. Passa a un frame per nome

```
driver.SwitchTo().Frame("Name_Of_Frame");
```

3. Passa a una cornice per Titolo, Id o altri passando IWebElement

Se vuoi passare a un frame per ID o titolo devi passare in un elemento web come parametro:

```
driver.SwitchTo().Frame(driver.FindElement(By.Id("ID_OF_FRAME")));
driver.SwitchTo().Frame(driver.FindElement(By.CssSelector("iframe[title='Title_of_Frame']")));
```

Tieni inoltre presente che se la cornice impiega alcuni secondi per venire visualizzata, potresti dover utilizzare un'attesa :

```
new WebDriverWait(driver, TimeSpan.FromSeconds(10))
    .Until(ExpectedConditions.ElementIsVisible(By.Id("Id_Of_Frame")));
```

Esci da una cornice:

```
driver.SwitchTo().DefaultContent()
```

Per uscire da una cornice usando C

Per spostare lo stato attivo su un documento principale o il primo fotogramma della pagina. Devi usare la sintassi qui sotto.

. WebDriver.SwitchTo () DefaultContent ();

Passa tra i frame secondari di un frame principale.

Considera di avere un frame padre (frame-padre). e 2 frame figlio (Frame_Son, Frame_Daughter). Vediamo le varie condizioni e come gestirle.

1. Da genitore a figlio o figlia:

```
 driver.switchTo().frame("Frame_Son");
 driver.switchTo().frame("Frame_Daughter");
```

2. Da Figlio a padre: se genitore è frame predefinito, passa al frame predefinito, altrimenti dal frame switch predefinito al frame principale. Ma non puoi passare direttamente da figlio a genitore.

```
driver.switchTo().defaultContent();
driver.switchTo().frame("Frame_Parent");
```

3. Da figlio a figlia: se tua sorella commette qualche errore non urlare contro di lei, rivolgiti semplicemente al tuo genitore. Allo stesso modo, dai il controllo al frame principale e poi al frame figlia.

```
driver.switchTo().defaultContent();
driver.switchTo().frame("Frame_Parent");
driver.switchTo().frame("Frame_Daughter");
```

Attendi il caricamento dei frame

In alcuni casi il tuo frame potrebbe non essere visualizzato immediatamente e probabilmente dovrai attendere fino al momento del caricamento. Oppure avrai NoSuchFrameException.

Quindi è sempre una buona scelta aspettare prima di cambiare. Di seguito è un modo ideale per attendere fino a quando viene caricato un frame.

```
try{
        new WebDriverWait(driver, 300).ignoring(StaleElementReferenceException.class).
                        ignoring(WebDriverException.class).

until(ExpectedConditions.visibilityOf((driver.findElement(By.id("cpmInteractionDivFrame"))));}
    catch{
```

// genera un'eccezione solo se il frame non è visibile con il tempo di attesa di 300 secondi}

Leggi Cambio di frame online: https://riptutorial.com/it/selenium-webdriver/topic/4589/cambio-di-frame

Capitolo 6: Configurazione della griglia di selenio

introduzione

Selenium Grid è un framework per eseguire test distribuiti su una gamma di dispositivi di test. È usato per testare le applicazioni web. È possibile scrivere test in diversi linguaggi di programmazione popolari, tra cui C #, Groovy, Java, Perl, PHP, Python e Ruby. I test possono essere eseguiti su una gamma di browser su piattaforme come Windows, Linux e OS X.

È un software open source, rilasciato con la licenza Apache 2.0: gli sviluppatori web possono scaricarlo e utilizzarlo gratuitamente.

Sintassi

- per eseguire il file jar, la seguente è la sintassi per ogni file jar
- `java -jar <jar-file-full-name>.jar -<your parameters if any>`

Parametri

parametri	Dettagli
ruolo	È ciò che dice al selenio quale era l' `hub` o il `node`
porta	Questo serve a specificare quale porta deve essere in ascolto l' `hub` o il `node` .
mozzo	Questo parametro viene utilizzato nel `node` per specificare l'url dell'hub
browserName	È stato utilizzato nel `node` per specificare il nome del browser come firefox, chrome, internet explorer
MaxInstances	È dove viene specificata l'istanza del browser, ad es. 5 significa che ci saranno 5 istanze del browser che l'utente specificato sarà presente.
nodeConfig	Un file di configurazione Json per il nodo. Puoi specificare il ruolo, la porta ecc. Qui
hubConfig	Un file di configurazione Json per il nodo. Qui puoi specificare il ruolo, la porta, le istanze massime ecc

Examples

Codice Java per Selenium Grid

```
String hubUrl = "http://localhost:4444/wd/hub"
DesiredCapabilities capability = DesiredCapabilities.firefox(); //or which browser you want
RemoteWebDriver driver = new RemoteWebDriver(hubUrl, capability);
```

Creazione di un hub e nodo di selenio

Creare un hub

Una configurazione rapida per l'installazione di hub e nodi nella griglia di selenio. Per ulteriori informazioni consultare: Grid 2 documenti

Requisiti

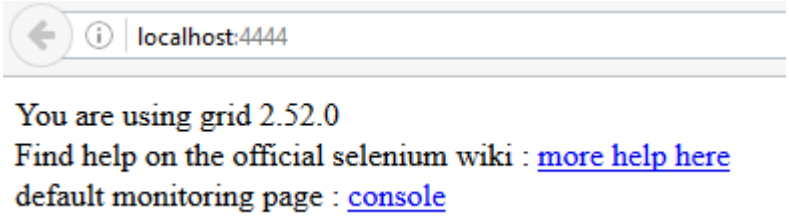
Per configurare un hub della rete è necessario il flusso:

- Selenio-server-standalone-.jar

Creazione dell'hub

Per creare un hub è necessario eseguire il server di selenio.

1. Scarica Selenium-server-standalone-.jar
2. Apri il tuo terminale e vai alla cartella in cui si trova Selenium-server-standalone-.jar
3. Esegui il comando seguente:
    1. Per la configurazione predefinita `java -jar selenium-server-standalone-<Version>.jar -role hub`
    2. Per la configurazione Json `java -jar selenium-server-standalone-<Version>.jar -role hub -hubConfig hubConfig.json`
4. Aprire http: // localhost: 4444 / vedrete un messaggio seguito



Facendo clic su `console` -> `View config` per visualizzare la configurazione per i dettagli dell'hub.

Creare un nodo

Requisiti

Per configurare un hub della rete è necessario il flusso:

- Selenio-server-standalone-.jar
- Webdrivers
    - Driver Chrome
    - Driver FireFox
    - Driver Microsoft Edge
- browser
    - Cromo
    - FireFox
    - Microsoft Edge (Windows 10)

Creazione del nodo

Ora Per creare i nodi per l'hub

1. Scarica Selenium-server-standalone-.jar
2. Scarica i browser su cui desideri eseguire il test
3. Scarica i driver per i browser su cui desideri eseguire il test
4. Aprire un nuovo terminale e navigare fino alla posizione del file jar del server di selenio
5. Esegui il comando seguente:
    1. per la configurazione predefinita `java -jar selenium-server-standalone-<VERSION NUMBER>.jar -role node`
    2. Per la configurazione Json `java -jar selenium-server-standalone-<Version>.jar -role node -nodeConfig nodeConfig.json`
6. Ora vai a http: // localhost: 4444 / grid / console per visualizzare i dettagli del nodo

Configurazione tramite Json

Una configurazione di esempio per un hub:

```
java -jar selenium-server-standalone-<version>.jar -role hub -hubConfig hubConfig.json
```

```
{
    "_comment" : "Configuration for Hub - hubConfig.json",
    "host": ip,
    "maxSessions": 5,
    "port": 4444,
    "cleanupCycle": 5000,
    "timeout": 300000,
    "newSessionWaitTimeout": -1,
    "servlets": [],
    "prioritizer": null,
    "capabilityMatcher": "org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
    "throwOnCapabilityNotPresent": true,
    "nodePolling": 180000,
    "platform": "WINDOWS"
}
```

Una configurazione di esempio per un nodo

```
java -jar selenium-server-standalone-<version>.jar -role node -nodeConfig nodeConfig.json
```

```
{
  "capabilities":
  [
    {
      "browserName": "opera",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.opera.driver": "C:/Selenium/drivers/operadriver.exe",
      "binary":"C:/Program Files/Opera/44.0.2510.1159/opera.exe"
    },
    {
      "browserName": "chrome",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.chrome.driver": "C:/Selenium/drivers/chromedriver.exe",
      "binary":"C:/Program Files/Google/Chrome/Application/chrome.exe"
    },
    {
      "browserName": "firefox",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "seleniumProtocol": "WebDriver",
      "webdriver.gecko.driver": "C:/Selenium/drivers/geckodriver.exe",
      "binary":"C:/Program Files/Mozilla Firefox/firefox.exe"
    }
  ],
  "proxy": "org.openqa.grid.selenium.proxy.DefaultRemoteProxy",
  "maxSession": 5,
  "port": 5555,
  "register": true,
  "registerCycle": 5000,
  "hub": "http://localhost:4444",
  "nodeStatusCheckTimeout": 5000,
  "nodePolling": 5000,
  "role": "node",
  "unregisterIfStillDownAfter": 60000,
  "downPollingLimit": 2,
  "debug": false,
  "servlets" : [],
  "withoutServlets": [],
  "custom": {}
}
```

Leggi Configurazione della griglia di selenio online: https://riptutorial.com/it/selenium-webdriver/topic/2504/configurazione-della-griglia-di-selenio

Capitolo 7: Eccezioni in Selenium-WebDriver

introduzione

Esistono numerose eccezioni che possono essere generate durante l'utilizzo di un webdriver. Gli esempi di seguito hanno lo scopo di dare un'idea di cosa significano.

Examples

Python Exceptions

Documentazione di eccezione al selenio

ElementNotInteractableException: generata quando un elemento è presente nel DOM ma le interazioni con quell'elemento colpiranno un altro elemento a causa dell'ordine di vernice

- **ElementNotSelectableException:** generata quando si tenta di selezionare un elemento non selezionabile. Esempi di elementi non selezionabili:
  - copione
- **ElementNotVisibleException:** generata quando un elemento è presente sul DOM, ma non è visibile e quindi non è in grado di interagire con esso. Più comunemente riscontrato quando si tenta di fare clic o leggere il testo di un elemento nascosto alla vista.
- **ErrorInResponseException:** generata quando si verifica un errore sul lato server. Ciò può accadere quando si comunica con l'estensione firefox o il server driver remoto.
- **ImeActivationFailedException:** generata quando si attiva un motore IME non riuscita.
- **ImeNotAvailableException: generato** quando il supporto IME non è disponibile. Questa eccezione viene generata per ogni chiamata di metodo relativa a IME se il supporto IME non è disponibile sulla macchina.
- **InvalidArgumentException:** gli argomenti passati a un comando non sono validi o non sono validi.
- **InvalidCookieDomainException:** generata quando si tenta di aggiungere un cookie in un dominio diverso rispetto all'URL corrente.
- **InvalidElementStateException:** generata quando un'azione comporta uno stato non valido per un elemento. sottoclassi:
  - ElementNotInteractableException
  - ElementNotSelectableException
  - ElementNotVisibleException
- **InvalidSelectorException:** generata quando il selettore utilizzato per trovare un elemento non restituisce un WebElement. Attualmente ciò accade solo quando il selettore è un'espressione xpath ed è sintatticamente non valido (ovvero non è un'espressione xpath) o l'espressione non seleziona WebElements (ad esempio "count (// input)").
- **InvalidSwitchToTargetException:** generata quando la destinazione della trama o della finestra da commutare non esiste.
- **MoveTargetOutOfBoundsException: generato** quando il target fornito al metodo

moveChains move () non è valido, cioè fuori documento.

- **NoAlertPresentException:** generata quando si passa a nessun avviso presentato. Ciò può essere causato chiamando un'operazione sulla classe Alert () quando un avviso non è ancora visualizzato sullo schermo.
- **NoSuchAttributeException:** generata quando non è stato possibile trovare l'attributo dell'elemento. Si consiglia di verificare se l'attributo esiste nel browser specifico che si sta testando. Alcuni browser potrebbero avere nomi di proprietà diversi per la stessa proprietà. (IE8 .innerText vs. Firefox .textContent)
- **NoSuchElementException:** generata quando non è stato possibile trovare l'elemento. Se si verifica questa eccezione, è possibile controllare quanto segue:
  - Controlla il tuo selettore usato nel tuo find_by ...
  - L'elemento potrebbe non essere ancora visualizzato sullo schermo al momento dell'operazione di ricerca, (la pagina web è ancora in fase di caricamento) vedere selenium.webdriver.support.wait.WebDriverWait () per come scrivere un wrapper di attesa per attendere la comparsa di un elemento.
- **NoSuchFrameException:** generata quando il target del frame da cambiare non esiste.
- **NoSuchWindowException:** generata quando l'obiettivo della finestra da commutare non esiste. Per trovare il set corrente di handle di finestra attivi, è possibile ottenere un elenco degli handle di finestra attivi nel modo seguente:
  ```
  print driver.window_handles
  ```
- **RemoteDriverServerException:**
- **StaleElementReferenceException:** generata quando un riferimento a un elemento è "stantio". Stantio significa che l'elemento non appare più sul DOM della pagina. Le possibili cause di StaleElementReferenceException includono, ma non sono limitate a:
  - Non ci si trova più nella stessa pagina o la pagina potrebbe essere stata aggiornata dall'ubicazione dell'elemento.
  - L'elemento potrebbe essere stato rimosso e riaggiunto allo schermo, dal momento che era posizionato. Come un elemento che viene spostato. Questo può accadere in genere con un framework javascript quando i valori vengono aggiornati e il nodo viene ricostruito.
  - L'elemento potrebbe essere stato all'interno di un iframe o di un altro contesto che è stato aggiornato.
- **TimeoutException:** generata quando un comando non viene completato in un tempo sufficiente.
- **UnableToSetCookieException:** generata quando un driver non riesce a impostare un cookie.
- **UnexpectedAlertPresentException:** generata quando viene visualizzato un avviso inatteso. Solitamente generato quando una modale attesta sta bloccando la forma del webdriver eseguendo altri comandi.
- **UnexpectedTagNameException:** generato quando una classe di supporto non ha ottenuto un elemento web previsto.
- **WebDriverException:** eccezione del webdriver di base. Tutte le eccezioni del webdriver utilizzano WebDriverException o InvalidStateException come classe padre.

Leggi Eccezioni in Selenium-WebDriver online: https://riptutorial.com/it/selenium-webdriver/topic/10546/eccezioni-in-selenium-webdriver

Capitolo 8: Esecuzione di Javascript nella pagina

Sintassi

- oggetto ExecuteAsyncScript (script stringa, oggetto params [] args);
- oggetto ExecuteScript (script stringa, oggetto params [] args);

Examples

C

Per eseguire JavaScript in un'istanza di `IWebDriver` è necessario eseguire il cast di `IWebDriver` su una nuova interfaccia, `IJavaScriptExecutor`

```
IWebDriver driver;
IJavaScriptExecutor jsDriver = driver as IJavaScriptExecutor;
```

È ora possibile accedere a tutti i metodi disponibili nell'istanza `IJavaScriptExecutor` che consentono di eseguire Javascript, ad esempio:

```
jsDriver.ExecuteScript("alert('running javascript');");
```

Pitone

Per eseguire Javascript in python, usa `execute_script("javascript script here")` . execute_script viene chiamato su un'istanza di webdriver e può essere qualsiasi javascript valido.

```
from selenium import webdriver
driver = webdriver.Chrome()
driver.execute_script("alert('running javascript');")
```

Giava

Per eseguire Javascript in Java, crea un nuovo webdriver che supporti Javascript. Per utilizzare la funzione `executeScript()` , è necessario eseguire il cast del driver su `JavascriptExecutor` oppure impostare una nuova variabile sul valore del driver casted: `((JavascriptExecutor)driver)` . `driver.executeScript()` accetta una stringa che è Javascript valido.

```
WebDriver driver = new ChromeDriver();
JavascriptExecutor JavascriptExecutor = ((JavascriptExecutor)driver);
JavascriptExecutor.executeScript("alert('running javascript');");
```

Rubino

```
require "selenium-webdriver"

driver = Selenium::WebDriver.for :chrome
driver.execute_script("alert('running javascript');")
```

Leggi Esecuzione di Javascript nella pagina online: https://riptutorial.com/it/selenium-webdriver/topic/6986/esecuzione-di-javascript-nella-pagina

Capitolo 9: Gestione degli errori nell'automazione con selenio

Examples

Pitone

`WebDriverException` è `WebDriverException` `Selenium-WebDriver` base che può essere utilizzata per catturare tutte le altre eccezioni di `Selenium-WebDriver`

Per essere in grado di catturare l'eccezione dovrebbe essere importato prima:

```
from selenium.common.exceptions import WebDriverException as WDE
```

e poi:

```
try:
    element = driver.find_element_by_id('ID')
except WDE:
    print("Not able to find element")
```

Allo stesso modo puoi importare altre eccezioni più specifiche:

```
from selenium.common.exceptions import ElementNotVisibleException
from selenium.common.exceptions import NoAlertPresentException
...
```

Se si desidera estrarre solo il messaggio di eccezione:

```
from selenium.common.exceptions import UnexpectedAlertPresentException

try:
    driver.find_element_by_tag_name('a').click()
except UnexpectedAlertPresentException as e:
    print(e.__dict__["msg"])
```

Leggi Gestione degli errori nell'automazione con selenio online: https://riptutorial.com/it/selenium-webdriver/topic/9548/gestione-degli-errori-nell-automazione-con-selenio

Capitolo 10: Gestisci un avviso

Examples

Selenio con Java

Ecco come gestire un avviso popup in Java con selenio:

Ci sono 3 tipi di popup.

1. **Avviso semplice** : avviso ("Questo è un avviso semplice");
2. **Avviso di conferma** : var popuResult = confirm ("Conferma pop-up con OK e pulsante Annulla");
3. **Avviso rapido** : var person = prompt ("Ti piace StackOverflow?", "Sì / No");

È fino all'utente quale tipo di popup deve essere gestito nel proprio caso di test.

O puoi

1. accept () Per accettare l'avviso
2. respingere () Per chiudere l'avviso
3. getText () Per ottenere il testo dell'avviso
4. sendKeys () per scrivere del testo all'avviso

Per semplice avviso:

```
Alert simpleAlert = driver.switchTo().alert();
String alertText = simpleAlert.getText();
System.out.println("Alert text is " + alertText);
simpleAlert.accept();
```

Per avviso di conferma:

```
Alert confirmationAlert = driver.switchTo().alert();
String alertText = confirmationAlert.getText();
System.out.println("Alert text is " + alertText);
confirmationAlert.dismiss();
```

Per avviso rapido:

```
Alert promptAlert  = driver.switchTo().alert();
String alertText = promptAlert .getText();
System.out.println("Alert text is " + alertText);
//Send some text to the alert
promptAlert .sendKeys("Accepting the alert");
Thread.sleep(4000); //This sleep is not necessary, just for demonstration
promptAlert .accept();
```

in base alle tue esigenze.

Un altro modo per farlo è avvolgere il codice all'interno di un try-catch:

```
try{
   // Your logic here.
} catch(UnhandledAlertException e){
  Alert alert = driver.switchTo().alert();
  alert.accept();
}
// Continue.
```

C

Ecco come chiudere un avviso popup in C # con selenio:

```
IAlert alert = driver.SwitchTo().Alert();
// Prints text and closes alert
System.out.println(alert.Text);
alert.Accept();
or
alert.Dismiss();
```

in base alle tue esigenze.

Un altro modo per farlo è avvolgere il codice all'interno di un try-catch:

```
try{
   // Your logic here.
} catch(UnhandledAlertException e){
  var alert = driver.SwitchTo().Alert();
  alert.Accept();
}
// Continue.
```

Pitone

Ci sono molti modi per passare al pop-up di avviso in `Python` :

1. *Obsoleto* :

```
alert = driver.switch_to_alert()
```

2. *Utilizzando* *`switch_to`* :

```
alert = driver.switch_to.alert
```

3. *Utilizzando* *`ExplicitWait`* :

```
 from selenium.webdriver.common.by import By
 from selenium.webdriver.support.ui import WebDriverWait
```

```
from selenium.webdriver.support import expected_conditions as EC

alert = WebDriverWait(driver, TIMEOUT_IN_SECONDS).until(EC.alert_is_present())
```

4. *Dichiarando l'istanza della classe* `Alert` :

```
from selenium.webdriver.common.alert import Alert

alert = Alert(driver)
```

Per riempire il campo di input nel pop-up attivato dal `prompt()` `JavaScript` `prompt()` :

```
alert.send_keys('Some text to send')
```

Per confermare il pop-up della finestra di dialogo *:

```
alert.accept()
```

Dismettere:

```
alert.dismiss()
```

Per ottenere il testo dal pop-up:

```
alert.text
```

* *PS* `alert.dismiss()` *può essere utilizzato per confermare i popup attivati da* `JavaScript` `alert()` *e* `alert.confirm()`

Leggi Gestisci un avviso online: https://riptutorial.com/it/selenium-webdriver/topic/6048/gestisci-un-avviso

Capitolo 11: Gli ascoltatori

Examples

JUnit

Se si sta utilizzando JUnit per l'esecuzione, è possibile estendere la classe `TestWatcher` :

```
public class TestRules extends TestWatcher {

    @Override
    protected void failed(Throwable e, Description description) {
        // This will be called whenever a test fails.
    }
```

Quindi nella tua classe di test puoi semplicemente chiamarlo:

```
public class testClass{

@Rule
public TestRules testRules = new TestRules();

@Test
public void doTestSomething() throws Exception{
    // If the test fails for any reason, it will be caught be testrules.
}
```

EventFiringWebDriver

Utilizzo di EventFiringWebDriver . È possibile collegare WebDriverEventListener ad esso e sovrascrivere i metodi, ad esempio il metodo onException:

```
EventFiringWebDriver driver = new EventFiringWebDriver(new FirefoxDriver());
WebDriverEventListener listener = new AbstractWebDriverEventListener() {
    @Override
    public void onException(Throwable t, WebDriver driver) {
        // Take action
    }
};
driver.register(listener);
```

Leggi Gli ascoltatori online: https://riptutorial.com/it/selenium-webdriver/topic/8226/gli-ascoltatori

Capitolo 12: Griglia di selenio

Examples

Configurazione del nodo

La configurazione del nodo griglia selenio risiede sul nodo stesso e contiene le informazioni sulla configurazione di rete e sulle funzionalità del nodo. La configurazione può essere applicata in vari modi:

- Configurazione predefinita
- Configurazione JSON
- Configurazione della riga di comando

Configurazione JSON

La configurazione del nodo nel file JSON è divisa in 2 sezioni:

- funzionalità
- Configurazione

Le funzionalità definiscono aree quali i tipi e le versioni del browser supportati, le posizioni dei binari del browser, il numero di istanze massime di ciascun tipo di browser.

La configurazione si occupa di impostazioni come hub e indirizzi e porte dei nodi.

Di seguito è riportato un esempio di un file di configurazione JSON:

```
{
  "capabilities": [
    {
      "browserName": "firefox",
      "acceptSslCerts": true,
      "javascriptEnabled": true,
      "takesScreenshot": false,
      "firefox_profile": "",
      "browser-version": "27",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "firefox_binary": "",
      "cleanSession": true
    },
    {
      "browserName": "chrome",
      "maxInstances": 5,
      "platform": "WINDOWS",
      "webdriver.chrome.driver": "C:/Program Files (x86)/Google/Chrome/Application/chrome.exe"
    },
    {
      "browserName": "internet explorer",
      "maxInstances": 1,
      "platform": "WINDOWS",
```

```
      "webdriver.ie.driver": "C:/Program Files (x86)/Internet Explorer/iexplore.exe"
    }
  ],
  "configuration": {
    "_comment" : "Configuration for Node",
    "cleanUpCycle": 2000,
    "timeout": 30000,
    "proxy": "org.openqa.grid.selenium.proxy.WebDriverRemoteProxy",
    "port": 5555,
    "host": ip,
    "register": true,
    "hubPort": 4444,
    "maxSessions": 5
  }
}
```

Come creare un nodo

Per creare un nodo, devi prima avere un hub. Se non hai un hub, puoi crearlo in questo modo:

```
java -jar selenium-server-standalone-<version>.jar -role hub
```

Quindi sei in grado di creare un nodo:

```
java -jar selenium-server-standalone-<version>.jar -role node -hub
http://localhost:4444/grid/register // default port is 4444
```

Maggiori informazioni qui: https://github.com/SeleniumHQ/selenium/wiki/Grid2

Leggi Griglia di selenio online: https://riptutorial.com/it/selenium-webdriver/topic/1359/griglia-di-selenio

Capitolo 13: Impostazione / acquisizione della dimensione della finestra del browser

introduzione

Impostazione o acquisizione delle dimensioni della finestra di qualsiasi browser durante l'automazione

Sintassi

- . Driver.manage () finestra () massimizzare ().;
- driver.manage (). window (). setSize (*DimensionObject*);
- driver.manage (). finestra (). getSize ()

Examples

GIAVA

Imposta la dimensione massima della finestra del browser:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Set Browser window size
driver.manage().window().maximize();
```

Imposta dimensioni specifiche della finestra:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Initialize Dimension class object and set Browser window size
org.openqa.selenium.Dimension d = new org.openqa.selenium.Dimension(400, 500);
driver.manage().window().setSize(d);
```

Ottieni dimensioni della finestra del browser:

```
//Initialize Browser
System.setProperty("webdriver.gecko.driver", "E:\\path\\to\\geckodriver.exe");
WebDriver driver = new FirefoxDriver();
driver.get("https://www.google.com/");

//Get Browser window size and print on console
```

```
System.out.println(driver.manage().window().getSize());
```

Leggi Impostazione / acquisizione della dimensione della finestra del browser online: https://riptutorial.com/it/selenium-webdriver/topic/10093/impostazione---acquisizione-della-dimensione-della-finestra-del-browser

Capitolo 14: Impostazione del selenio e2e

introduzione

Questo argomento copre la configurazione end-to-end di Selenium ie Selenium Webdriver + TestNG + Maven + Jenkins.

Per l'aggiunta del report, fare riferimento all'argomento Rapporti HTML

Examples

Setup TestNG

TestNG è il tuo framework di test aggiornato per junit. **Utilizzeremo testng.xml** per invocare le suite di test. Questo è utile quando useremo CI in anticipo.

testng.xml

Nella cartella radice del tuo progetto crea un file xml con il nome testng.xml. Nota che il nome può essere diverso, ma per comodità è usato come "testng" ovunque.

Di seguito è riportato il codice semplice per il file testng.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
<suite name="Smoke"> //name of the suite
    <test name="Test1"> //name of the test
        <classes>
            <class name="test.SearchTest">
            <methods>
                <include name="searchTest"/>
            </methods>
            </class>
        </classes>
    </test>
</suite>
```

Installazione di Maven

TBD. Come configurare pom.xml per chiamare testng.xml

Installazione di Jenkins

TBD. Coprirà l'installazione di Jenkins per estrarre codice da git / bitbucket ecc.

Leggi Impostazione del selenio e2e online: https://riptutorial.com/it/selenium-webdriver/topic/10724/impostazione-del-selenio-e2e

Capitolo 15: Individuazione degli elementi Web

Sintassi

- ByChained (parametri di [] bys)

Osservazioni

Gli oggetti si trovano nel Selenio attraverso l'uso di *localizzatori* e la classe `By` . Per realizzare un solido progetto di automazione con Selenium, è necessario utilizzare i localizzatori per Web Elements in modo intelligente. I localizzatori dovrebbero essere **descrittivi, unici e improbabili da modificare, in** modo da non ottenere falsi positivi nei test, ad esempio. La priorità è usare:

1. **ID** : dal momento che è unico e ottieni esattamente l'elemento che desideri.
2. **Nome classe** : è descrittivo e può essere unico in un determinato contesto.
3. **CSS** (prestazioni migliori di xpath) - Per selettori più complicati.
4. **XPATH** - Dove non è possibile utilizzare il CSS (asse XPATH), ad es. `div::parent` .

Il resto dei locatori sono inclini a modifiche o rendering, ed è preferibilmente evitato.

Regola pratica: se il codice non è in grado di individuare un particolare elemento, una ragione potrebbe essere che il codice non ha aspettato il download di tutti gli elementi DOM. Considera di dire al tuo programma di "aspettare" per un breve periodo di tempo (prova 3-5 secondi, e poi aumenta lentamente se necessario) prima di cercare il suddetto elemento. Ecco un esempio in Python, tratto da questa domanda :

```
from selenium import webdriver
import time

browser = webdriver.Firefox()
browser.get("https://app.website.com")

reports_element = browser.find_element_by_xpath("//button[contains(text(), 'Reports')]")

# Element not found! Try giving time for the browser to download all DOM elements:
time.sleep(10)

reports_element = browser.find_element_by_xpath("//button[contains(text(), 'Reports')]")
# This returns correct value!
```

Examples

Individuazione degli elementi della pagina tramite WebDriver

Per interagire con WebElements in una pagina Web, innanzitutto è necessario identificare la

posizione dell'elemento.

Di è la parola chiave disponibile nel selenio.

Puoi localizzare gli elementi di ..

1. **Per** *ID*
2. **Per** *nome della classe*
3. **Con** *TagName*
4. **Per** *nome*
5. **Tramite il** *testo del collegamento*
6. **Con il** *testo di collegamento parziale*
7. **Dal** *selettore CSS*
8. **Con** *XPath*
9. **Utilizzando** *JavaScript*

Considera l'esempio di script seguente

```
<form name="loginForm">
Login Username: <input id="username" name="login" type="text" />
Password: <input id="password" name="password" type="password" />
<input name="login" type="submit" value="Login" />
```

Nel codice sopra il nome utente e la password sono impostati usando id. Ora identificherete gli elementi con id.

```
driver.findElement(By.id(username));

driver.findElement(By.id(password));
```

Poiché il selenio supporta 7 lingue diverse, questo documento ti dà un'idea per individuare gli elementi in tutte le lingue.

Per ID

Esempio di come trovare un elemento usando l'ID:

```
<div id="coolestWidgetEvah">...</div>

Java       -  WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
C#         -  IWebElement element = driver.FindElement(By.Id("coolestWidgetEvah"));
Python     -  element = driver.find_element_by_id("coolestWidgetEvah")
Ruby       -  element = driver.find_element(:id, "coolestWidgetEvah")
JavaScript/Protractor -  var elm = element(by.id("coolestWidgetEvah"));
```

Per nome della classe

Esempio di come trovare un elemento usando il nome della classe:

```
<div class="cheese"><span>Cheddar</span></div>

Java       -  WebElement element = driver.findElement(By.className("cheese"));
C#         -  IWebElement element = driver.FindElement(By.ClassName("cheese"));
Python     -  element = driver.find_element_by_class_name("cheese")
Ruby       -  cheeses = driver.find_elements(:class, "cheese")
JavaScript/Protractor -  var elm = element(by.className("cheese"));
```

Per nome tag

Esempio di come trovare un elemento usando il nome del tag:

```
<iframe src="..."></iframe>

Java       -  WebElement element = driver.findElement(By.tagName("iframe"));
C#         -  IWebElement element = driver.FindElement(By.TagName("iframe"));
Python     -  element = driver.find_element_by_tag_name("iframe")
Ruby       -  frame = driver.find_element(:tag_name, "iframe")
JavaScript/Protractor -  var elm = element(by.tagName("iframe"));
```

Per nome

Esempio di come trovare un elemento usando il nome:

```
<input name="cheese" type="text"/>

Java       -  WebElement element = driver.findElement(By.name("cheese"));
C#         -  IWebElement element = driver.FindElement(By.Name("cheese"));
Python     -  element = driver.find_element_by_name("cheese")
Ruby       -  cheese = driver.find_element(:name, "cheese")
JavaScript/Protractor -  var elm = element(by.name("cheese"));
```

Tramite il testo del collegamento

Esempio di come trovare un elemento usando il testo del link:

```
<a href="http://www.google.com/search?q=cheese">cheese</a>>

Java       -  WebElement element = driver.findElement(By.linkText("cheese"));
C#         -  IWebElement element = driver.FindElement(By.LinkText("cheese"));
Python     -  element = driver.find_element_by_link_text("cheese")
Ruby       -  cheese = driver.find_element(:link, "cheese")
JavaScript/Protractor -  var elm = element(by.linkText("cheese"));
```

Con il testo di collegamento parziale

Esempio di come trovare un elemento usando il testo del collegamento parziale:

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>>

Java       -  WebElement element = driver.findElement(By.partialLinkText("cheese"));
C#         -  IWebElement element = driver.FindElement(By.PartialLinkText("cheese"));
Python     -  element = driver.find_element_by_partial_link_text("cheese")
Ruby       -  cheese = driver.find_element(:partial_link_text, "cheese")
JavaScript/Protractor -  var elm = element(by.partialLinkText("cheese"));
```

Da selettori CSS

Esempio di come trovare un elemento usando i selettori CSS:

```
<div id="food" class="dairy">milk</span>

Java       -  WebElement element = driver.findElement(By.cssSelector("#food.dairy")); //# is
used to indicate id and . is used for classname.
C#         -  IWebElement element = driver.FindElement(By.CssSelector("#food.dairy"));
Python     -  element = driver.find_element_by_css_selector("#food.dairy")
Ruby       -  cheese = driver.find_element(:css, "#food span.dairy.aged")
JavaScript/Protractor -  var elm = element(by.css("#food.dairy"));
```

Ecco un articolo sulla creazione di selettori CSS:
http://www.w3schools.com/cssref/css_selectors.asp

Con XPath

Esempio di come trovare un elemento usando XPath:

```
<input type="text" name="example" />

Java       -  WebElement element = driver.findElement(By.xpath("//input"));
C#         -  IWebElement element = driver.FindElement(By.XPath("//input"));
Python     -  element = driver.find_element_by_xpath("//input")
Ruby       -  inputs = driver.find_elements(:xpath, "//input")
JavaScript/Protractor -  var elm = element(by.xpath("//input"));
```

Ecco un articolo su XPath: http://www.w3schools.com/xsl/xpath_intro.asp

---

Utilizzando JavaScript

È possibile eseguire un javascript arbitrario per trovare un elemento e fino a quando si restituisce un elemento DOM, verrà automaticamente convertito in un oggetto WebElement.

Semplice esempio su una pagina che ha caricato jQuery:

```
Java       -  WebElement element = (WebElement)
((JavascriptExecutor)driver).executeScript("return $('.cheese')[0]");

C#         -  IWebElement element = (IWebElement)
((IJavaScriptExecutor)driver).ExecuteScript("return $('.cheese')[0]");

Python     -  element = driver.execute_script("return $('.cheese')[0]");
Ruby       -  element = driver.execute_script("return $('.cheese')[0]")
JavaScript/Protractor -
```

Nota: questo metodo non funzionerà se il tuo specifico WebDriver non supporta JavaScript, come SimpleBrowser .

Selezione in base a più criteri [C #]

È anche possibile utilizzare i selettori insieme. Questo viene fatto usando l'oggetto `OpenQA.Selenium.Support.PageObjects.ByChained` :

```
element  = driver.FindElement(new ByChained(By.TagName("input"), By.ClassName("class"));
```

Qualsiasi numero di `By` s può essere incatenato e usato come selezione del tipo AND (cioè tutti i messaggi `By` sono abbinati)

Selezione degli elementi prima che la pagina smetta di caricarsi

Quando si chiama `driver.Navigate().GoToUrl(url);` , l'esecuzione del codice si interrompe finché la pagina non è completamente caricata. Questo a volte non è necessario quando si desidera estrarre dati.

Nota: i seguenti esempi di codice potrebbero essere considerati hack. Non esiste un modo "ufficiale" per farlo.

Crea un nuovo thread

Crea e avvia una discussione per caricare una pagina web, quindi usa Aspetta .

C #

```
using (var driver = new ChromeDriver())
{
    new Thread(() =>
    {
        driver.Navigate().GoToUrl("http://stackoverflow.com");
    }).Start();

    new WebDriverWait(driver, TimeSpan.FromSeconds(10))
      .Until(ExpectedConditions.ElementIsVisible(By.XPath("//div[@class='summary']/h3/a")));
}
```

Usa i timeout

Usando un WebDriverTimeout, puoi caricare una pagina e, dopo un certo periodo di tempo, genererà un'eccezione, che impedirà il caricamento della pagina. Nel blocco catch, puoi usare Wait .

C #

```
using (var driver = new ChromeDriver())
{
    driver.Manage().Timeouts().SetPageLoadTimeout(TimeSpan.FromSeconds(5));

    try
    {
        driver.Navigate().GoToUrl("http://stackoverflow.com");
    }
    catch (WebDriverTimeoutException)
    {
        new WebDriverWait(driver, TimeSpan.FromSeconds(10))
          .Until(ExpectedConditions.ElementIsVisible
            (By.XPath("//div[@class='summary']/h3/a")));
    }
}
```

Il problema : quando si imposta il timeout troppo breve, la pagina smetterà di caricarsi indipendentemente dal fatto che l'elemento desiderato sia presente. Quando si imposta il timeout per troppo tempo, si andrà a negare il beneficio delle prestazioni.

Leggi Individuazione degli elementi Web online: https://riptutorial.com/it/selenium-webdriver/topic/3991/individuazione-degli-elementi-web

Capitolo 16: Interagire con le finestre del browser

Examples

Gestire la finestra attiva

C

Massimizzare la finestra

```
driver.Manage().Window.Maximize();
```

Questo è abbastanza semplice, assicura che la nostra finestra attualmente attiva sia massimizzata.

Posizione della finestra

```
driver.Manage().Window.Position = new System.Drawing.Point(1, 1);
```

Qui essenzialmente spostiamo la finestra attualmente attiva in una nuova posizione. Nel `Point` oggetto forniamo `x` ed `y` coordinate; questi vengono quindi utilizzati come offset dall'angolo in alto a sinistra dello schermo per determinare dove posizionare la finestra. Nota che puoi anche memorizzare la posizione della finestra in una variabile:

```
System.Drawing.Point windowPosition = driver.Manage().Window.Position;
```

Dimensione della finestra

L'impostazione e il recupero delle dimensioni della finestra utilizzano la stessa sintassi della posizione:

```
driver.Manage().Window.Size = new System.Drawing.Size(100, 200);
System.Drawing.Size windowSize = driver.Manage().Window.Size;
```

URL della finestra

Possiamo ottenere l'URL corrente della finestra attiva:

```
string url = driver.Url;
```

Possiamo anche impostare l'URL per la finestra attiva, che consentirà al conducente di navigare verso il nuovo valore:

```
driver.Url = "http://stackoverflow.com/";
```

Maniglie per finestre

Possiamo ottenere l'handle per la finestra corrente:

```
string handle = driver.CurrentWindowHandle;
```

E possiamo ottenere le maniglie per tutte le finestre aperte:

```
IList<String> handles = driver.WindowHandles;
```

Pitone

Massimizzare la finestra

```
driver.maximize_window()
```

Ottieni la posizione della finestra

```
driver.get_window_position() # returns {'y', 'x'} coordinates
```

Imposta la posizione della finestra

```
driver.set_window_position(x, y) # pass 'x' and 'y' coordinates as arguments
```

Ottieni dimensioni della finestra

```
driver.get_window_size() # returns {'width', 'height'} values
```

Imposta la dimensione della finestra

```
driver.set_window_size(width, height) # pass 'width' and 'height' values as arguments
```

Titolo della pagina corrente

```
driver.title
```

URL corrente

```
driver.current_url
```

Maniglie per finestre

```
driver.current_window_handle
```

Elenco delle finestre attualmente aperte

```
driver.window_handles
```

Chiusura della finestra del browser corrente

Passa alla nuova scheda aperta. Chiudi le finestre correnti (in questo caso la nuova scheda). Torna alla prima finestra.

GONIOMETRO:

```
browser.getAllWindowHandles().then(function (handles) {
    browser.driver.switchTo().window(handles[1]);
    browser.driver.close();
    browser.driver.switchTo().window(handles[0]);
});
```

Selenio JAVA:

```
        Set<String> handlesSet = driver.getWindowHandles();
        List<String> handlesList = new ArrayList<String>(handlesSet);
        driver.switchTo().window(handlesList.get(1));
        driver.close();
        driver.switchTo().window(handlesList.get(0));
```

Gestire più finestre

Pitone

Scenario più comunemente usato:

1. *apri la pagina in una nuova finestra*
2. *passare ad esso*
3. *fare qualcosa*
4. *chiudilo*
5. *torna alla finestra principale*

```
# Open "Google" page in parent window
driver.get("https://google.com")

driver.title # 'Google'

# Get parent window
parent_window = driver.current_window_handle

# Open "Bing" page in child window
driver.execute_script("window.open('https://bing.com')")

# Get list of all windows currently opened (parent + child)
all_windows = driver.window_handles
```

```
# Get child window
child_window = [window for window in all_windows if window != parent_window][0]

# Switch to child window
driver.switch_to.window(child_window)

driver.title # 'Bing'

# Close child window
driver.close()

# Switch back to parent window
driver.switch_to.window(parent_window)

driver.title # 'Google'
```

Leggi Interagire con le finestre del browser online: https://riptutorial.com/it/selenium-webdriver/topic/5181/interagire-con-le-finestre-del-browser

Capitolo 17: Interazione con l'elemento Web

Examples

C

Cancellare il contenuto dell'elemento (generalmente casella di testo)

```
interactionWebElement.Clear();
```

Immissione di dati sull'elemento (generalmente casella di testo)

```
interactionWebElement.SendKeys("Text");
```

Memorizzare il valore dell'elemento.

```
string valueinTextBox = interactionWebElement.GetAttribute("value");
```

Memorizzazione del testo dell'elemento.

```
string textOfElement = interactionWebElement.Text;
```

Cliccando su un elemento

```
interactionWebElement.Click();
```

Invio di un modulo

```
interactionWebElement.Submit();
```

Identificazione della visibilità di un elemento sulla pagina

```
bool isDisplayed=interactionWebElement.Displayed;
```

Identificazione dello stato di un elemento nella pagina

```
bool isEnabled = interactionWebElement.Enabled;

bool isSelected=interactionWebElement.Selected;
```

Individuazione elemento figlio di interactionWebElement

```
IWebElement childElement = interactionWebElement.FindElement(By.Id("childElementId"));
```

Individuazione degli elementi figlio di interactionWebElement

```
Ilist<IWebElement> childElements =
interactionWebElement.FindElements(By.TagName("childElementsTagName"));
```

Giava

Cancellazione del contenuto di un elemento Web: (nota: quando si simulano le azioni dell'utente nei test, è meglio inviare backspace, vedere l'azione successiva)

```
interactionWebElement.clear();
```

Immissione di dati - simulazione di invio di sequenze di tasti:

```
interactionWebElement.sendKeys("Text");
interactionWebElement.sendKeys(Keys.CONTROL + "c"); // copy to clipboard.
```

Ottenere il valore dell'attributo di un elemento:

```
interactionWebElement.getAttribute("value");
interactionWebElement.getAttribute("style");
```

Ottenere il testo dell'elemento:

```
String elementsText = interactionWebElement.getText();
```

Selezione dal menu a discesa:

```
Select dropDown = new Select(webElement);
dropDown.selectByValue(value);
```

Autoesplicativo:

```
interactionWebElement.click();
interactionWebElement.submit(); //for forms
interactionWebElement.isDisplayed();
interactionWebElement.isEnabled(); // for exampale - is clickable.
interactionWebElement.isSelected(); // for radio buttons.
```

Azioni che utilizzano `org.openqa.selenium.interactions.Actions` :

Drag & Drop:

```
Action dragAndDrop = builder.clickAndHold(someElement)
   .moveToElement(otherElement)
   .release(otherElement)
   .build();

dragAndDrop.perform();
```

Seleziona più:

```
Action selectMultiple = builder.keyDown(Keys.CONTROL)
   .click(someElement)
   .click(someOtherElement)
   .keyUp(Keys.CONTROL);

dragAndDrop.perform();
```

Auto esplicativo (usando il costruttore):

```
builder.doubleClick(webElement).perform();
builder.moveToElement(webElement).perform(); //hovering
```

Vedi qui per ulteriori esempi di azioni avanzate e un elenco completo.

Utilizzando Javascript:

```
// Scroll to view element:
((JavascriptExecutor) driver).executeJavaScript("arguments[0].scrollIntoView(true);",
webElement);
```

Leggi Interazione con l'elemento Web online: https://riptutorial.com/it/selenium-webdriver/topic/4280/interazione-con-l-elemento-web

Capitolo 18: Modello di oggetto della pagina

introduzione

Un ruolo significativo nell'automazione di siti Web e applicazioni Web implica l'identificazione di elementi sullo schermo e l'interazione con essi. Gli oggetti si trovano nel Selenio attraverso l'uso di localizzatori e la classe `By` . Questi locatori e interazioni sono posti all'interno di Page Objects come una buona pratica per evitare il codice duplicato e facilitare la manutenzione. Incapsula `WebElements` e suppone che contenga comportamenti e restituisca informazioni sulla pagina (o parte di una pagina in un'app Web).

Osservazioni

Il modello a oggetti di pagina è un modello in cui scriviamo classi orientate agli oggetti che fungono da interfaccia per una particolare vista della pagina web. Usiamo i metodi di quella classe di pagina per eseguire l'azione richiesta. Pochi anni fa, stavamo manipolando il codice HTML della pagina web direttamente nelle classi di test, il che era molto difficile da mantenere insieme ai cambiamenti apportati all'interfaccia utente.

Tuttavia, il fatto che il tuo codice sia organizzato in un modello di oggetto della pagina fornisce un'API specifica dell'applicazione, che consente di manipolare gli elementi della pagina senza scavare attorno all'HTML. La semplice Rue of thumb dice che l'oggetto della tua pagina dovrebbe avere tutto ciò che un umano può fare su quella pagina web. Ad esempio, per accedere al campo di testo su una pagina Web, è necessario un metodo per ottenere il testo e restituire la stringa dopo aver apportato tutte le modifiche.

Pochi punti importanti da tenere a mente durante la progettazione degli oggetti della pagina:

1. L'oggetto della pagina di solito non dovrebbe costruire solo per le pagine, ma dovresti preferirlo per costruire elementi significativi della pagina. Ad esempio, una pagina con più schede per mostrare diversi grafici dei tuoi accademici dovrebbe avere lo stesso numero di pagine del conteggio delle schede.

2. La navigazione da una vista ad un'altra dovrebbe restituire l'istanza delle classi di pagine.

3. I metodi di utilità che devono essere presenti solo per una vista o pagina Web specifica dovrebbero appartenere solo a quella classe di pagine.

4. I metodi di asserzione non dovrebbero essere presi in considerazione dalle classi di pagine, puoi avere metodi per restituire booleano ma non verificarli lì. Ad esempio, per verificare il nome completo dell'utente è possibile avere un metodo per ottenere il valore booleano:

```
public boolean hasDisplayedUserFullName (String userFullName) {
    return driver.findElement(By.xpath("xpathExpressionUsingFullName")).isDisplayed();
}
```

5. Se la tua pagina web è basata su iframe, preferisci avere anche le classi di pagine per iframe.

Vantaggi del modello di oggetto della pagina:

1. Separazione pulita tra codice di prova e codice di pagina
2. In caso di modifiche nell'interfaccia utente della pagina Web, non è necessario modificare il codice in più posizioni. Cambia solo nelle classi di pagine.
3. Nessun localizzatore di elementi sparsi.
4. Rende il codice più facile da capire
5. Facile manutenzione

Examples

Introduzione (Utilizzo di Java)

Un esempio per eseguire il test di accesso basato sul modello di oggetto Pagina:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;

/**
 * Class which models the view of Sign-In page
 */
public class SignInPage {

    @FindBy(id="username")
    private usernameInput;

    @FindBy(id="password")
    private passwordInput;

    @FindBy(id="signin")
    private signInButton;

    private WebDriver driver;

    public SignInPage(WebDriver driver) {
        this.driver = driver;
    }

    /**
     * Method to perform login
     */
    public HomePage performLogin(String username, String password) {
        usernameInput.sendKeys(username);
        passwordInput.sendKeys(password);
        signInButton.click();
        return PageFactory.initElements(driver, HomePage.class);
    }
}
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
/**
 * Class which models the view of home page
 */
public class HomePage {
    @FindBy(id="logout")
    private logoutLink;

    private WebDriver driver;

    public HomePage(WebDriver driver) {
        this.driver = driver;
    }

    /**
     * Method to log out
     */
    public SignInPage logout() {
        logoutLink.click();
        wait.ForPageToLoad();
        return PageFactory.initElements(driver, SignInPage.class);
    }
}

/**
 * Login test class
 */
public class LoginTest {
    public void testLogin() {
        SignInPage signInPage = new SignInPage(driver);
        HomePage homePage = signInPage.login(username, password);
        signInPage = homePage.logout();
    }
}
```

C

Gli oggetti Page devono contenere un comportamento, informazioni di ritorno per asserzioni e possibilmente un metodo per il metodo dello stato di pagina pronto all'inizializzazione. Il selenio supporta gli oggetti pagina usando le annotazioni. In C # è come segue:

```
using OpenQA.Selenium;
using OpenQA.Selenium.Support.PageObjects;
using OpenQA.Selenium.Support.UI;
using System;
using System.Collections.Generic;

public class WikipediaHomePage
{
    private IWebDriver driver;
    private int timeout = 10;
    private By pageLoadedElement = By.ClassName("central-featured-logo");

    [FindsBy(How = How.Id, Using = "searchInput")]
    [CacheLookup]
```

```
    private IWebElement searchInput;

    [FindsBy(How = How.CssSelector, Using = ".pure-button.pure-button-primary-progressive")]
    [CacheLookup]
    private IWebElement searchButton;

    public ResultsPage Search(string query)
    {
        searchInput.SendKeys(query);
        searchButton.Click();
    }

    public WikipediaHomePage VerifyPageLoaded()
    {
        new WebDriverWait(driver, TimeSpan.FromSeconds(timeout)).Until<bool>((drv) =>  return
drv.ExpectedConditions.ElementExists(pageLoadedElement));

        return this;
    }
}
```

gli appunti:

- `CacheLookup` salva l'elemento nella cache e salva la restituzione di un nuovo elemento per ogni chiamata. Questo migliora le prestazioni ma non è buono per gli elementi che cambiano dinamicamente.
- `searchButton` ha 2 nomi di classe e nessun ID, ecco perché non posso usare il nome della classe o l'id.
- Ho verificato che i miei locator mi restituiranno l'elemento che desidero utilizzando gli Strumenti per sviluppatori (per Chrome), mentre in altri browser è possibile utilizzare FireBug o simili.
- `Search()` metodo `Search()` restituisce un altro oggetto pagina (`ResultsPage`) mentre il clic di ricerca reindirizza a un'altra pagina.

Modello oggetto pagina Best Practices

- Crea file separati per intestazione e piè di pagina (poiché sono comuni a tutte le pagine e non ha senso farne parte di una singola pagina)
- Mantieni elementi comuni (come Cerca / Indietro / Avanti ecc.) In un file separato (l'idea è di rimuovere ogni tipo di duplicazione e mantenere la segregazione logica)
- Per Driver, è una buona idea creare una classe Driver separata e mantenere il driver come statico in modo che sia possibile accedervi su tutte le pagine! (Ho tutte le mie pagine web estendono DriverClass)
- Le funzioni utilizzate in PageObjects sono suddivise nel più piccolo chunk possibile tenendo presente la frequenza e il modo in cui verranno chiamate (il modo in cui hai fatto il login- anche se il login può essere scomposto in enterUsername e inserirePassword funzioni ma mantenerlo poiché la funzione di accesso è più logica perché nella maggior parte dei casi viene chiamata la funzione di login piuttosto che le chiamate separate per inserire usernameUn utente e immetterePassword funzioni)
- L'utilizzo di PageObjects segrega lo script di test dagli elementLocators
- Hanno funzioni di utilità nella cartella separata dei programmi di utilità (come DateUtil,

excelUtils ecc.)

- Avere configurazioni in una cartella conf separata (come l'impostazione dell'ambiente in cui i test devono essere eseguiti, la configurazione delle cartelle di output e di input)
- Incorporare screenCapture in caso di errore
- Avere una variabile di attesa statica nel DriverClass con un tempo di attesa implicito come si è fatto. Cerca sempre di avere attese condizionali piuttosto che attese statiche come: wait.until (ExpectedConditions). Ciò garantisce che l'attesa non rallenti l'esecuzione inutilmente.

Leggi Modello di oggetto della pagina online: https://riptutorial.com/it/selenium-webdriver/topic/4853/modello-di-oggetto-della-pagina

Capitolo 19: Navigare tra più fotogrammi

introduzione

Nelle pagine web contiene il numero di frame, il selenio considera Frame is seprate window, quindi accedi al contenuto presente nel frame per passare al frame. Molte volte abbiamo bisogno di una struttura Web in cui abbiamo frame con frame per navigare all'interno di frame windows. Selenium fornisce il metodo swithTo ().

Examples

Esempio di frame

```
<iframe "id="iframe_Login1">

    <iframe "id="iframe_Login2">

        <iframe "id="iframe_Login3">

        </iframe>

    </iframe>

</iframe>
```

Per passare al frame nel selenio usa il metodo swithTo () e frame ().

. Driver.switchTo () telaio (iframe_Login1); . Driver.switchTo () telaio (iframe_Login2); . Driver.switchTo () telaio (iframe_Login3);

Per tornare indietro possiamo usare parentFrame () e defaultContest ();

parentFrame (): modifica lo stato attivo sul contesto principale. Se il contesto corrente è il contesto di navigazione di livello superiore, il contesto rimane invariato.

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

defaultContent (): seleziona il primo fotogramma sulla pagina o il documento principale quando una pagina contiene iframe.

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

Leggi Navigare tra più fotogrammi online: https://riptutorial.com/it/selenium-webdriver/topic/9803/navigare-tra-piu-fotogrammi

Capitolo 20: Navigazione

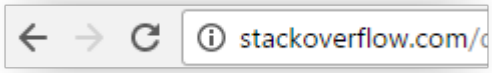
Sintassi

- **C #**
- void Indietro ()
- void Forward ()
- void GotToUrl (string URL)
- void Aggiorna ()
- **Pitone**
- driver.back ()
- driver.forward ()
- driver.get ("URL")
- driver.refresh ()
- **Giava**
- driver.navigate () back ().;
- . Driver.navigate () in avanti ();
- driver.navigate () per ("URL").;
- . Driver.navigate () refresh ();

Examples

Naviga () [C #]

È possibile navigare direttamente nel browser, ad esempio utilizzando i comandi della barra degli strumenti standard disponibili su tutti i browser:



È possibile creare un oggetto di navigazione chiamando `Navigate()` sul driver:

```
IWebDriver driver
INavigation navigation = driver.Navigate();
```

Un oggetto di navigazione consente di eseguire numerose azioni che navigano nel browser del Web:

```
//like pressing the back button
navigation.Back();
//like pressing the forward button on a browser
navigation.Forward();
//navigate to a new url in the current window
navigation.GoToUrl("www.stackoverflow.com");
//Like pressing the reload button
navigation.Refresh();
```

Navigate () [Java]

Per navigare verso qualsiasi URL:

```
driver.navigate().to("http://www.example.com");
```

Per tornare indietro:

```
driver.navigate().back();
```

Per spostare avanti:

```
driver.navigate().forward();
```

Per aggiornare la pagina:

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

Metodi del browser in WebDriver

WebDriver, l'interfaccia principale da utilizzare per i test, che rappresenta un browser web idealizzato. I metodi in questa classe si dividono in tre categorie:

- Controllo del browser stesso
- Selezione di WebElements
- Sussidi di debug

I metodi chiave sono get (String), che viene utilizzato per caricare una nuova pagina Web e i vari metodi simili a findElement (By), che viene utilizzato per trovare WebElements. In questo post impareremo i metodi di controllo del browser. ottenere

```
void get(java.lang.String url)
```

Carica una nuova pagina Web nella finestra corrente del browser. Questa operazione viene eseguita utilizzando un'operazione HTTP GET e il metodo verrà bloccato fino al completamento del caricamento. è meglio attendere fino al termine del timeout, poiché se la pagina sottostante dovesse cambiare mentre il test è in esecuzione, i risultati delle chiamate future contro questa interfaccia saranno contro la pagina appena caricata. **uso**

```
//Initialising driver
 WebDriver driver = new FirefoxDriver();

 //setting timeout for page load
 driver.manage().timeouts().pageLoadTimeout(20, TimeUnit.SECONDS);

 //Call Url in get method
 driver.get("https://www.google.com");
 //or
 driver.get("https://seleniumhq.org");
```

getCurrentUrl

```
java.lang.String getCurrentUrl()
```

Ottieni una stringa che rappresenti l'URL corrente a cui il browser sta guardando. Restituisce l'URL della pagina attualmente caricata nel browser.

uso

```
//Getting current url loaded in browser & comparing with expected url
 String pageURL = driver.getCurrentUrl();
 Assert.assertEquals(pageURL, "https://www.google.com");
```

getTitle

```
java.lang.String getTitle()
```

Restituisce il titolo della pagina corrente, con spaziatura iniziale e finale spogliata, o null se non è già impostata.

uso

```
//Getting current page title loaded in browser & comparing with expected title
 String pageTitle = driver.getTitle();
 Assert.assertEquals(pageTitle, "Google");

getPageSource

java.lang.String getPageSource()
```

Ottieni la fonte dell'ultima pagina caricata. Se la pagina è stata modificata dopo il caricamento (ad esempio, tramite Javascript) non è possibile garantire che il testo restituito sia quello della pagina modificata.

uso

```
//get the current page source
 String pageSource = driver.getPageSource();
```

vicino

```
void close()
```

Chiudi la finestra corrente, esci dal browser se è l'ultima finestra attualmente aperta. Se ci sono più finestre aperte con quell'istanza del driver, questo metodo chiuderà la finestra che sta avendo l'attuale focus su di esso.

uso

```
//Close the current window
     driver.close();
```

smettere

```
void quit()
```

Esce da questo driver, chiudendo ogni finestra associata. Dopo aver chiamato questo metodo, non possiamo usare nessun altro metodo usando la stessa istanza del driver.

uso

```
//Quit the current driver session / close all windows associated with driver
     driver.quit();
```

Questi sono tutti metodi molto utili disponibili in Selenium 2.0 per controllare il browser come richiesto.

Leggi Navigazione online: https://riptutorial.com/it/selenium-webdriver/topic/7272/navigazione

Capitolo 21: Prendendo Screenshots

introduzione

Acquisizione di screenshot e salvataggio in un determinato percorso

Sintassi

- File src = ((TakesScreenshot) driver) .getScreenshotAs (OutputType.FILE);
- FileUtils.copyFile (src, new File ("D: \ screenshot.png"));

Examples

GIAVA

Codice per prendere e salvare screenshot:

```
public class Sample
{
    public static void main (String[] args)
    {
        *//Initialize Browser*
        System.setProperty("webdriver.gecko.driver", "**E:\\path\\to\\geckodriver.exe**");
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("https://www.google.com/");

        //Take Screesnshot
        File src = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
        try {
            //Save Screenshot in destination file
            FileUtils.copyFile(src, new File("D:\\screenshot.png"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Prende Screenshot:

```
File src = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);
```

Memorizza Screenshot dalla sorgente alla destinazione:

```
FileUtils.copyFile(src, new File("D:\\screenshot.png"));
```

Leggi Prendendo Screenshots online: https://riptutorial.com/it/selenium-webdriver/topic/10094/prendendo-screenshots

Capitolo 22: Programma di base del selenio per web

introduzione

Questo argomento si propone di mostrare il programma di base del driver web in lingue supportate dal selenio come C #, Groovy, Java, Perl, PHP, Python e Ruby.

Il viaggio include l'apertura del driver del browser -> Pagina Google -> chiusura del browser

Examples

C

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace BasicWebdriver
{
    class WebDriverTest
    {
        static void Main()
        {
            using (var driver = new ChromeDriver())
            {
                driver.Navigate().GoToUrl("http://www.google.com");
            }
        }
    }
}
```

Il suddetto "programma" passerà alla home page di Google, quindi chiuderà il browser dopo aver caricato completamente la pagina.

```
using (var driver = new ChromeDriver())
```

Questo istanzia un nuovo oggetto WebDriver utilizzando l'interfaccia `IWebdriver` e crea una nuova istanza della finestra del browser. In questo esempio stiamo usando `ChromeDriver` (anche se questo potrebbe essere sostituito dal driver appropriato per qualsiasi browser volessimo usare). Stiamo `IWebDriver` wrapping con un'istruzione `using` , perché `IWebDriver` implementa `IDisposable` , quindi non è necessario digitare esplicitamente `driver.Quit();` .

Nel caso in cui non hai scaricato il tuo WebDriver usando NuGet , dovrai passare un argomento sotto forma di un percorso alla directory in cui si trova il driver stesso "chromedriver.exe".

---

navigazione

```
driver.Navigate().GoToUrl("http://www.google.com");
```

e

```
driver.Url = "http://www.google.com";
```

Entrambe queste linee fanno la stessa cosa. Istruiscono il guidatore a navigare verso un URL specifico e ad attendere che la pagina venga caricata prima di passare all'istruzione successiva.

Esistono altri metodi legati alla navigazione come `Back()` , `Forward()` o `Refresh()` .

Dopodiché, il blocco `using` chiude in sicurezza e smaltisce l'oggetto.

Pitone

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

def set_up_driver():
    path_to_chrome_driver = 'chromedriver'
    return webdriver.Chrome(executable_path=path_to_chrome_driver)

def get_google():
    driver = set_up_driver()
    driver.get('http://www.google.com')
    tear_down(driver)

def tear_down(driver):
    driver.quit()

if '__main__' == __name__:
    get_google()
```

Il suddetto "programma" passerà alla home page di Google e quindi chiuderà il browser prima di completarlo.

```
if '__main__' == __name__:
    get_google()
```

Per prima cosa abbiamo la nostra funzione principale, il nostro punto di ingresso nel programma, che chiama `get_google()` .

```
def get_google():
    driver = set_up_driver()
```

`get_google()` inizia quindi creando la nostra istanza `driver` tramite `set_up_driver()` :

```
def set_up_driver():
    path_to_chrome_driver = 'chromedriver'
    return webdriver.Chrome(executable_path=path_to_chrome_driver)
```

Per cui viene `chromedriver.exe` dove si trova `chromedriver.exe` e istanzia il nostro oggetto driver con questo percorso. Il resto di `get_google()` passa a Google:

```
driver.get('http://www.google.com')
```

E poi chiama `tear_down()` passando l'oggetto driver:

```
tear_down(driver)
```

`tear_down()` contiene semplicemente una riga per chiudere il nostro oggetto driver:

```
driver.quit()
```

Questo dice al guidatore di chiudere tutte le finestre del browser aperte e smaltire l'oggetto del browser, dato che non abbiamo altro codice dopo questa chiamata, questo termina efficacemente il programma.

Giava

Il codice sotto è di circa 3 passi.

1. Aprire un browser Chrome
2. Apertura di google page
3. Spegni il browser

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

public class WebDriverTest {
    public static void main(String args[]) {
        System.setProperty("webdriver.chrome.driver", "C:\\path\\to\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();

        driver.get("http://www.google.com");
        driver.quit();
    }
}
```

Il suddetto "programma" passerà alla home page di Google e quindi chiuderà il browser prima di completarlo.

```
System.setProperty("webdriver.chrome.driver", "C:\\path\\to\\chromedriver.exe");
WebDriver driver = new ChromeDriver();
```

La prima riga indica al sistema dove trovare l' `ChromeDriver` (chromedriver.exe). Creiamo quindi il nostro oggetto driver chiamando il `ChromeDriver()` , di nuovo potremmo chiamare il nostro costruttore qui per qualsiasi browser / piattaforma.

```
driver.get("http://www.google.com");
```

Ciò indica al nostro autista di navigare nell'URL specificato: http://www.google.com . L'API Java WebDriver fornisce il metodo `get()` direttamente sull'interfaccia WebDriver, sebbene sia possibile trovare ulteriori metodi di navigazione tramite il metodo `navigate()` , ad esempio `driver.navigate.back()` .

Una volta che la pagina ha terminato il caricamento, chiamiamo immediatamente:

```
driver.quit();
```

Questo dice al driver di chiudere tutte le finestre del browser aperte e di eliminare l'oggetto del driver, poiché non abbiamo altro codice dopo questa chiamata, questo termina efficacemente il programma.

```
driver.close();
```

È un'istruzione (non mostrata qui) al driver per chiudere solo la finestra attiva, in questo caso poiché abbiamo una sola finestra le istruzioni causerebbero risultati identici al chiamare `quit()` .

Java: best practice con classi di pagine

Usecase: accedi all'account FB

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.BeforeClass;
import org.testng.annotations.Test;

public class FaceBookLoginTest {
    private static WebDriver driver;
    HomePage homePage;
    LoginPage loginPage;
    @BeforeClass
    public void openFBPage(){
        driver = new FirefoxDriver();
        driver.get("https://www.facebook.com/");
        loginPage = new LoginPage(driver);
    }
    @Test
    public void loginToFB(){
       loginPage.enterUserName("username");
       loginPage.enterPassword("password");
       homePage = loginPage.clickLogin();
        System.out.println(homePage.getUserName());
    }

}
```

Classi di pagine: pagina di accesso e pagina iniziale pagina di accesso:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;
```

```
public class LoginPage {

    WebDriver driver;
    public LoginPage(WebDriver driver){
        this.driver = driver;
    }
    @FindBy(id="email")
    private WebElement loginTextBox;

    @FindBy(id="pass")
    private WebElement passwordTextBox;

    @FindBy(xpath = ".//input[@data-testid='royal_login_button']")
    private WebElement loginBtn;

    public void enterUserName(String userName){
        if(loginTextBox.isDisplayed()) {
            loginTextBox.clear();
            loginTextBox.sendKeys(userName);
        }
        else{
            System.out.println("Element is not loaded");
        }
    }
    public void enterPassword(String password){
        if(passwordTextBox.isDisplayed()) {
            passwordTextBox.clear();
            passwordTextBox.sendKeys(password);
        }
        else{
            System.out.println("Element is not loaded");
        }
    }
    public HomePage clickLogin(){
        if(loginBtn.isDisplayed()) {
            loginBtn.click();
        }
        return new HomePage(driver);
    }
}
```

Classe di home page:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.support.FindBy;

public class HomePage {

    WebDriver driver;
    public HomePage(WebDriver driver){
        this.driver = driver;
    }

    @FindBy(xpath=".//a[@data-testid='blue_bar_profile_link']/span")
    private WebElement userName;

    public String getUserName(){
        if(userName.isDisplayed()) {
            return userName.getText();
```

```
        }
        else {
            return "Username is not present";
        }
    }

}
```

Leggi Programma di base del selenio per web online: https://riptutorial.com/it/selenium-webdriver/topic/3990/programma-di-base-del-selenio-per-web

Capitolo 23: Rapporti HTML

introduzione

Questo argomento riguarda la creazione di report HTML per i test del selenio. Esistono vari tipi di plug-in disponibili per la creazione di report e quelli ampiamente utilizzati sono Allure, ExtentReports e ReportNG.

Examples

ExtentReports

Questo esempio copre l'implementazione di ExtentReports in Selenium utilizzando TestNG, Java e Maven.

ExtentReports sono disponibili in due versioni, community e commerciale. Per motivi di facilità e di dimostrazione, utilizzeremo la versione della community.

1. Dipendenza

Aggiungi la dipendenza nel tuo file Maven pom.xml per i rapporti di estensione.

```
<dependency>
     <groupId>com.aventstack</groupId>
     <artifactId>extentreports</artifactId>
     <version>3.0.6</version>
 </dependency>
```

2. Configura i plugin

Configura il plugin maven surefire come sotto in pom.xml

```
 <build>
 <defaultGoal>clean test</defaultGoal>
 <plugins>
     <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>3.6.1</version>
         <configuration>
             <source>${jdk.level}</source>
             <target>${jdk.level}</target>
         </configuration>
     </plugin>
     <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <version>2.19.1</version>
         <configuration>
             <suiteXmlFiles>
```

```
                    <suiteXmlFile>testng.xml</suiteXmlFile>
                </suiteXmlFiles>
            </configuration>
        </plugin>
    </plugins>
</build>
```

3. Test di esempio con ExtentReports

Ora crea un test con nome test.java

```
public class TestBase {
    WebDriver driver;

    ExtentReports extent;
    ExtentTest logger;
    ExtentHtmlReporter htmlReporter;
    String htmlReportPath = "C:\\Screenshots/MyOwnReport.html"; //Path for the HTML report to
be saved

    @BeforeTest
    public void setup(){
        htmlReporter = new ExtentHtmlReporter(htmlReportPath);
        extent = new ExtentReports();
        extent.attachReporter(htmlReporter);

        System.setProperty("webdriver.chrome.driver", "pathto/chromedriver.exe");
        driver = new ChromeDriver();

    }

    @Test
    public void test1(){
        driver.get("http://www.google.com/");
        logger.log(Status.INFO, "Opened site google.com");
        assertEquals(driver.getTitle()), "Google");
        logger.log(Status.PASS, "Google site loaded");
    }

    @AfterMethod
    public void getResult(ITestResult result) throws Exception {
        if (result.getStatus() == ITestResult.FAILURE)
        {
            logger.log(Status.FAIL, MarkupHelper.createLabel(result.getName() + " Test case
FAILED due to below issues:", ExtentColor.RED));
            logger.fail(result.getThrowable());
        }
        else if (result.getStatus() == ITestResult.SUCCESS)
        {
            logger.log(Status.PASS, MarkupHelper.createLabel(result.getName() + " Test Case
PASSED", ExtentColor.GREEN));
        }
        else if (result.getStatus() == ITestResult.SKIP)
        {
            logger.log(Status.SKIP, MarkupHelper.createLabel(result.getName() + " Test Case
SKIPPED", ExtentColor.BLUE));
        }
     }

    @AfterTest
```

```
    public void testend() throws Exception {
        extent.flush();
    }

    @AfterClass
    public void tearDown() throws Exception {
        driver.close();
    }
```

Allure Reports

Questo esempio copre l'implementazione di Allure Reports in Selenium utilizzando TestNG, Java e Maven.

Configurazione Maven

deposito

Aggiungi il seguente codice per configurare il repository jcenter

```
<repository>
            <id>jcenter</id>
            <name>bintray</name>
            <url>http://jcenter.bintray.com</url>
</repository>
```

Dipendenza

Aggiungi le seguenti dipendenze al tuo pom.xml

```
<dependency>
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>${aspectj.version}</version>
    </dependency>
    <dependency>
        <groupId>ru.yandex.qatools.allure</groupId>
        <artifactId>allure-testng-adaptor</artifactId>
        <version>1.5.4</version>
    </dependency>
```

Configurazione plug-in Surefire

```
<plugin>
        <groupId> org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.20</version>
        <configuration>
            <argLine>-
```

```
javaagent:${settings.localRepository}/org/aspectj/aspectjweaver/${aspectj.version}/aspectjweaver-
${aspectj.version}.jar
            </argLine>
            <properties>
                <property>
                    <name>listener</name>
                    <value>ru.yandex.qatools.allure.testng.AllureTestListener</value>
                </property>
            </properties>
            <suiteXmlFiles>testng.xml</suiteXmlFiles>
            <testFailureIgnore>false</testFailureIgnore>
        </configuration>
    </plugin>
```

Prova di esempio per il rapporto Allure

Creare un test di esempio con nome test.java

```
public class test{
    WebDriver driver;
    WebDriverWait wait;

    @BeforeMethod
    public void setup() {
        System.setProperty("webdriver.chrome.driver", "path to/chromedriver.exe");
        driver = new ChromeDriver();
        driver.get("https://www.google.com/");
        wait = new WebDriverWait(driver,50);
    }

    @Title("Title check")
    @Description("Checking the title of the loaded page.")
    @Test
    public void searchTest(){
        String title = driver.getTitle();
        LogUtil.log("Title Fetched: "+title);
        assertEquals(title,"Google");
        LogUtil.log("Test Passed. Expected: Google | Actual: "+title);
        System.out.println("Page Loaded");
    }

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

Nella classe precedente abbiamo utilizzato la classe LogUtiil. Questo è fatto semplicemente per registrare i **passaggi** nel nostro test. Di seguito è riportato il codice per lo stesso

LogUtil.java

```
public final class LogUtil {

    private LogUtil() {
    }
```

```
    @Step("{0}")
    public static void log(final String message){
        //intentionally empty
    }
}
```

Qui

@Title ("") aggiungerà il titolo al test in Allure Report

@Description ("") aggiungerà la descrizione al test

@Step ("") aggiungerà un passaggio nel report allure per il test

Durante l'esecuzione verrà generato un file xml nella cartella "target / allure-results /"

Relazione finale con Jenkins

Se stai usando Jenkins con il plugin Allure Report installato, allora Jenkins renderà automaticamente il report nel tuo lavoro.

Rapporto finale senza Jenkins

Per coloro che non hanno un Jenkins, usa la seguente linea di comando per creare il rapporto html. Allure CLI è un'applicazione Java quindi è disponibile per tutte le piattaforme. È necessario installare manualmente Java 1.7+ prima di utilizzare Allure CLI.

Debian

Per gli archivi basati su Debian forniamo un PPA quindi l'installazione è semplice: Installa Allure CLI per debian

```
$ sudo apt-add-repository ppa:yandex-qatools/allure-framework
$ sudo apt-get update
$ sudo apt-get install allure-commandline
```

Le distribuzioni supportate sono: Trusty e Precise. Dopo l'installazione avrai il comando allure disponibile.

Mac OS

È possibile installare Allure CLI tramite Homebrew.

```
$ brew tap qatools/formulas
$ brew install allure-commandline
```

Dopo l'installazione avrai il comando allure disponibile.

Windows e altri Unix

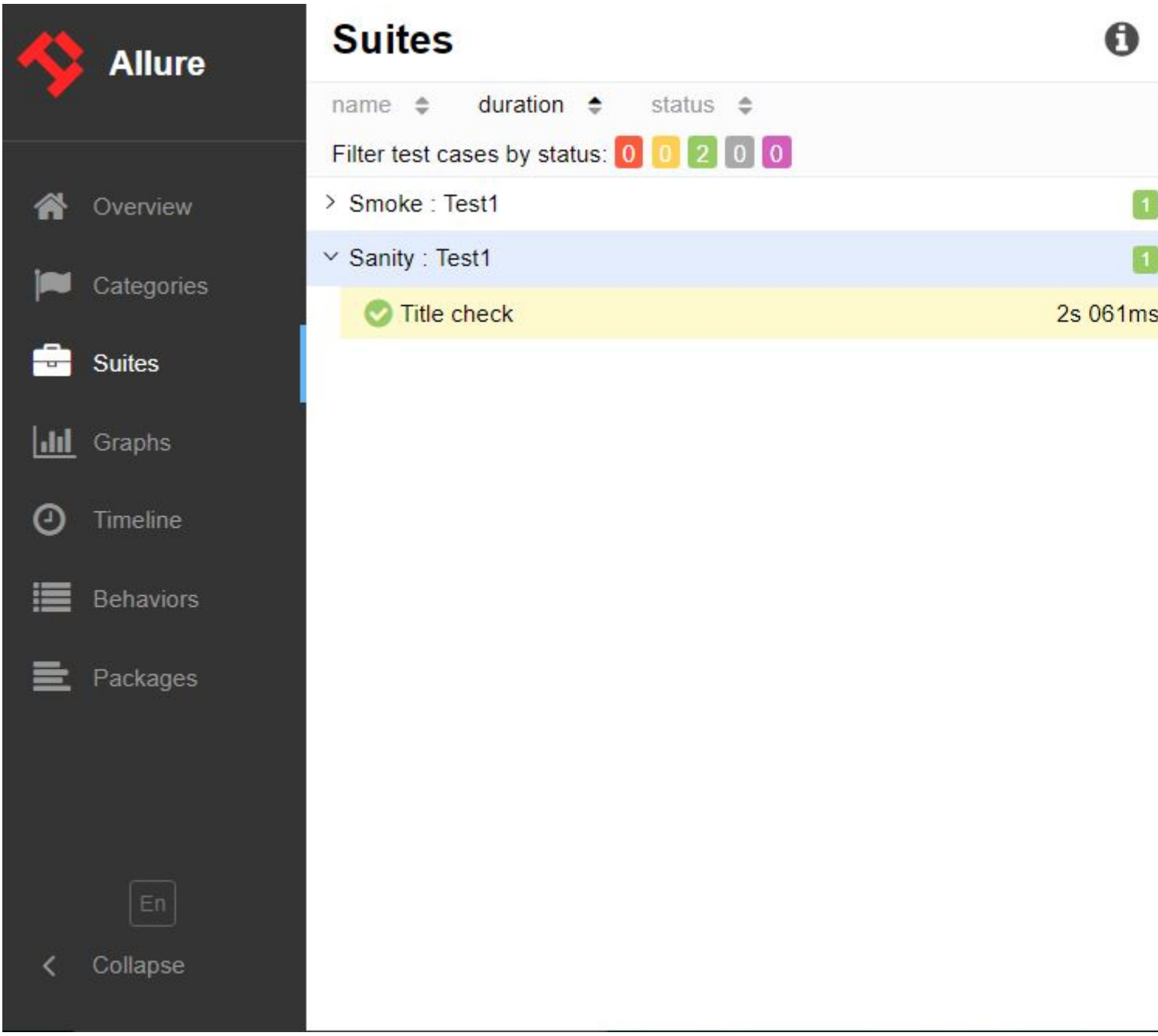
1. Scarica l'ultima versione come archivio zip da https://github.com/allure-framework/allure-

core/releases/latest .

2. Spacchettare l'archivio nella directory allure-commandline. Passare alla directory bin.
3. Usa allure.bat per Windows e allure per altre piattaforme Unix.

Nella Commandline / Terminale ora inserisci semplicemente la seguente sintassi e il rapporto verrà generato nella cartella allure-report

```
$ allure generate directory-with-results/
```



Leggi Rapporti HTML online: https://riptutorial.com/it/selenium-webdriver/topic/10721/rapporti-html

Capitolo 24: Robot nel selenio

Sintassi

- ritardo (int ms)
- keyPress (int keycode)
- keyRelease (int keycode)
- mouseMove (int x, int y)
- mousePress (pulsanti int)
- mouseRelease (pulsanti int)
- mouseWheel (int wheelAmt)

Parametri

Parametro	Dettagli
Signorina	È ora di dormire in millisecondi
chiave	Costante per premere il tasto specificato ad esempio per premere `A` codice è `VK_A` . Si prega di fare riferimento per maggiori dettagli: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html
x, y	Schermo coordintates
pulsanti	La maschera pulsante; una combinazione di una o più maschere di pulsanti del mouse
wheelAmt	Numero di tacche per spostare la rotellina del mouse, valore negativo per spostarsi in alto / in allontanamento dal valore positivo dell'utente per spostarsi in basso / verso l'utente

Osservazioni

Questa sezione contiene dettagli sull'implementazione dell'API Robot con Selenium Webdriver. La classe Robot viene utilizzata per generare input di sistema nativi quando il selenio non è in grado di farlo, ad esempio premendo il tasto destro del mouse, premendo il tasto F1, ecc.

Examples

Evento Keypress utilizzando Robot API (JAVA)

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
```

```
public class KeyBoardExample {
    public static void main(String[] args) {
        try {
            Robot robot = new Robot();
            robot.delay(3000);
            robot.keyPress(KeyEvent.VK_Q); //VK_Q for Q
        } catch (AWTException e) {
            e.printStackTrace();
        }
    }
}
```

Con selenio

A volte abbiamo bisogno di premere qualsiasi tasto per testare l'evento stampa chiave sull'applicazione web. Per un'istanza per testare il tasto INVIO nel modulo di accesso, possiamo scrivere qualcosa di simile in basso con Selenium WebDriver

```
import java.awt.AWTException;
import java.awt.Robot;
import java.awt.event.KeyEvent;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.testng.annotations.Test;

public class LoginTest {

    @Test
    public void testEnterKey() throws InterruptedException
    {
        WebDriver driver=new FirefoxDriver();
        Robot robot=null;
        driver.get("test-url");
        driver.manage().window().maximize();
        driver.findElement(By.xpath("xpath-expression")).click();
        driver.findElement(By.xpath("xpath-expression")).sendKeys("username");
        driver.findElement(By.xpath("xpath-expression")).sendKeys("password");
        try {
            robot=new Robot();
        } catch (AWTException e) {
            e.printStackTrace();
        }
        //Keyboard Activity Using Robot Class
        robot.keyPress(KeyEvent.VK_ENTER);
    }
}
```

Evento del mouse usando Robot API (JAVA)

Movimento del mouse:

```
import java.awt.Robot;

public class MouseClass {
 public static void main(String[] args) throws Exception {
```

```
     Robot robot = new Robot();

     // SET THE MOUSE X Y POSITION
     robot.mouseMove(300, 550);
     }
}
```

Premi il tasto sinistro / destro del mouse:

```
import java.awt.Robot;
import java.awt.event.InputEvent;

public class MouseEvent {
 public static void main(String[] args) throws Exception {
     Robot robot = new Robot();

     // LEFT CLICK
     robot.mousePress(InputEvent.BUTTON1_MASK);
     robot.mouseRelease(InputEvent.BUTTON1_MASK);

     // RIGHT CLICK
     robot.mousePress(InputEvent.BUTTON3_MASK);
     robot.mouseRelease(InputEvent.BUTTON3_MASK);
     }
}
```

Clicca e scorri la ruota:

```
import java.awt.Robot;
import java.awt.event.InputEvent;

public class MouseClass {
 public static void main(String[] args) throws Exception {
     Robot robot = new Robot();

     // MIDDLE WHEEL CLICK
     robot.mousePress(InputEvent.BUTTON3_DOWN_MASK);
     robot.mouseRelease(InputEvent.BUTTON3_DOWN_MASK);

     // SCROLL THE MOUSE WHEEL
     robot.mouseWheel(-100);
     }
}
```

Leggi Robot nel selenio online: https://riptutorial.com/it/selenium-webdriver/topic/4877/robot-nel-selenio

Capitolo 25: scorrimento

introduzione

Questo argomento fornirà diversi approcci su come eseguire lo scrolling con il `selenium`

Examples

Scorrimento con Python

1. Scorrimento verso l'elemento di destinazione (pulsante "SFOGLIA MODELLI" nella parte inferiore della pagina) con `Actions`

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
target = driver.find_element_by_link_text('BROWSE TEMPLATES')
actions = ActionChains(driver)
actions.move_to_element(target)
actions.perform()
```

2. Scorrimento verso l'elemento di destinazione (pulsante "SFOGLIA MODELLI" nella parte inferiore della pagina) con `JavaScript`

```
from selenium import webdriver

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
target = driver.find_element_by_link_text('BROWSE TEMPLATES')
driver.execute_script('arguments[0].scrollIntoView(true);', target)
```

3. Scorrimento verso l'elemento di destinazione (pulsante "SFOGLIA MODELLI" nella parte inferiore della pagina) con il metodo integrato

```
from selenium import webdriver

    driver = webdriver.Chrome()
    driver.get('http://www.w3schools.com/')
    target = driver.find_element_by_link_text('BROWSE TEMPLATES')
    target.location_once_scrolled_into_view
```

Nota che `location_once_scrolled_into_view` restituisce anche le coordinate `x` , `y` dell'elemento dopo lo scorrimento

4. Scorrere fino alla fine della pagina con i `Keys`

```
from selenium import webdriver
```

```
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.get('http://www.w3schools.com/')
driver.find_element_by_tag_name('body').send_keys(Keys.END) # Use send_keys(Keys.HOME) to
scroll up to the top of page
```

Si noti che anche `send_keys(Keys.DOWN)` / `send_keys(Keys.UP)` e `send_keys(Keys.PAGE_DOWN)` / `send_keys(Keys.PAGE_UP)` potrebbero essere utilizzati per lo scorrimento

Scorrimento diverso con java in modi diversi

> Qui di seguito la soluzione può essere utilizzata anche in altri linguaggi di programmazione supportati con alcune modifiche alla sintassi

---

1. Per **scorrere verso il basso** pagina / sezione / divisione nella pagina Web mentre c'è la barra di scorrimento personalizzata (non scorrere del browser). Clicca qui per la demo e controlla che la barra di scorrimento abbia il suo elemento indipendente.

   Nel codice di seguito indicato passare l'elemento della barra di scorrimento e richiedere i punti di scorrimento.

```
    public static boolean scroll_Page(WebElement webelement, int scrollPoints)
    {
    try
    {
        System.out.println("---------------- Started - scroll_Page ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        // drag downwards
        int numberOfPixelsToDragTheScrollbarDown = 10;
        for (int i = 10; i < scrollPoints; i = i + numberOfPixelsToDragTheScrollbarDown)
        {
            dragger.moveToElement(webelement).clickAndHold().moveByOffset(0,
numberOfPixelsToDragTheScrollbarDown).release(webelement).build().perform();
        }
        Thread.sleep(500);
        System.out.println("---------------- Ending - scroll_Page ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessfully done in scroll_Page -----
-----------");
        e.printStackTrace();
        return false;
    }
  }
```

---

2. Per fare **scorrere verso l'alto** pagina / sezione / divisione nella pagina web mentre c'è barra di scorrimento personalizzata (non scorrere del browser). Clicca qui per la demo e controlla che la barra di scorrimento abbia il suo elemento indipendente.

Nel codice di seguito indicato passare l'elemento della barra di scorrimento e richiedere i punti di scorrimento.

```
public static boolean scroll_Page_Up(WebElement webelement, int scrollPoints)
{
    try
    {
        System.out.println("---------------- Started - scroll_Page_Up ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);
        // drag upwards
        int numberOfPixelsToDragTheScrollbarUp = -10;
        for (int i = scrollPoints; i > 10; i = i + numberOfPixelsToDragTheScrollbarUp)
        {
            dragger.moveToElement(webelement).clickAndHold().moveByOffset(0,
numberOfPixelsToDragTheScrollbarUp).release(webelement).build().perform();
        }
        System.out.println("---------------- Ending - scroll_Page_Up ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessfully done in scroll_Page_Up---
-------------");
        e.printStackTrace();
        return false;
    }
}
```

---

3. Per scorrere verso il basso quando **più browser scorre** (browser incorporato) e si desidera scorrere verso il basso con il **tasto Pagina giù** . Clicca qui per la demo

   Nel codice sotto riportato, passa l'elemento della tua area di scorrimento come `<div>` e richiede il tasto page down.

```
 public static boolean pageDown_New(WebElement webeScrollArea, int iLoopCount)
 {
    try
    {
        System.out.println("---------------- Started - pageDown_New ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.PAGE_DOWN).build().perform();
        }
        System.out.println"---------------- Ending - pageDown_New ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do page down ----------------");
        return false;
    }
}
```

---

4. Per scorrere verso l'alto quando **più browser scorre** (browser incorporato) e si desidera scorrere verso l'alto con il **tasto Pagina SU** . Clicca qui per la demo

   Nel codice indicato sotto, passa l'elemento della tua area di scorrimento come `<div>` e richiede la chiave della pagina su.

```
public static boolean pageUp_New(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - pageUp_New ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.PAGE_UP).build().perform();
        }
        System.out.println("---------------- Ending - pageUp_New ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do page up ----------------");
        return false;
    }
}
```

---

5. Per scorrere verso il basso quando **più browser scorre** (browser incorporato) e si desidera scorrere verso il basso con il **solo tasto freccia giù** . Clicca qui per la demo

   Nel codice sotto riportato, passa l'elemento dell'area di scorrimento come `<div>` e richiede il tasto di riduzione.

```
public static boolean scrollDown_Keys(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - scrollDown_Keys ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {

dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.DOWN).build().perform();
        }
        System.out.println("---------------- Ending - scrollDown_Keys ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do scroll down with keys-------------
---");
        return false;
    }
```

```
}
```

---

6. Per scorrere verso l'alto quando **più browser scorre** (browser incorporato) e si desidera scorrere verso l'alto con il **solo tasto freccia su** . Clicca qui per la demo

   Nel codice sotto riportato, passa l'elemento dell'area di scorrimento come `<div>` e richiede la chiave su.

```
public static boolean scrollUp_Keys(WebElement webeScrollArea, int iLoopCount)
{
    try
    {
        System.out.println("---------------- Started - scrollUp_Keys ----------------");
        driver = ExecutionSetup.getDriver();
        dragger = new Actions(driver);

        for (int i = 0; i <= iLoopCount; i++)
        {
            dragger.moveToElement(webeScrollArea).click().sendKeys(Keys.UP).build().perform();
        }
        System.out.println("---------------- Ending - scrollUp_Keys ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- Not able to do scroll up with keys---------------
-");
        return false;
    }
}
```

---

7. Per scorrere verso l'alto / il basso quando il **browser scorre** (browser incorporato) e si desidera scorrere verso l'alto / verso il basso con il **solo punto fisso** . Clicca qui per la demo

   Nel codice indicato di seguito passa il tuo punto di scorrimento. Positivo significa in basso e negativo significa scorrere verso l'alto.

```
public static boolean scroll_without_WebE(int scrollPoint)
{
    JavascriptExecutor jse;
    try
    {
        System.out.println("---------------- Started - scroll_without_WebE ----------------");

        driver = ExecutionSetup.getDriver();
        jse = (JavascriptExecutor) driver;
        jse.executeScript("window.scrollBy(0," + scrollPoint + ")", "");

        System.out.println("---------------- Ending - scroll_without_WebE ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessful in scroll_without_WebE ----
```

```
------------");
        e.printStackTrace();
        return false;
    }
}
```

---

8. Per scorrere verso l'alto / verso il basso quando il **browser scorre** (browser incorporato) e si desidera scorrere verso l'alto / il basso su **Per rendere l'elemento nell'area visibile o lo scorrimento dinamico** . Clicca qui per la demo

   Nel codice indicato di seguito passa il tuo elemento.

```
public static boolean scroll_to_WebE(WebElement webe)
{
    try
    {
        System.out.println("---------------- Started - scroll_to_WebE ----------------");

        driver = ExecutionSetup.getDriver();
        ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", webe);

        System.out.println("---------------- Ending - scroll_to_WebE ----------------");
        return true;
    }
    catch (Exception e)
    {
        System.out.println("---------------- scroll is unsucessful in scroll_to_WebE ---------
-------");
        e.printStackTrace();
        return false;
    }
}
```

Nota: verificare il caso e utilizzare i metodi. Se manca qualche caso, fammi sapere.

Leggi scorrimento online: https://riptutorial.com/it/selenium-webdriver/topic/9063/scorrimento

Capitolo 26: Selenium-webdriver con Python, Ruby e Javascript insieme allo strumento CI

introduzione

Questo è un modo per eseguire test di selenio con CircleCI

Examples

Integrazione CircleCI con Selenium Python e Unittest2

Circle.yml

```
machine:
  python:
    # Python version to use - Selenium requires python 3.0 and above
    version: pypy-3.6.0
dependencies:
    pre:
        # Install pip packages
        - pip install selenium
        - pip install unittest
test:
  override:
    # Bash command to run main.py
    - python main.py
```

main.py

```
import unittest2

# Load and run all tests in testsuite matching regex provided
loader = unittest2.TestLoader()
# Finds all the tests in the same directory that have a filename that ends in test.py
testcases = loader.discover('.', pattern="*test.py")
test_runner = unittest2.runner.TextTestRunner()
# Checks that all tests ran
success = test_runner.run(testcases).wasSuccessful()
```

example_test.py

```
class example_test(unittest.TestCase):
    def test_something(self):
        # Make a new webdriver instance
        self.driver = webdriver.Chrome()
        # Goes to www.goooogle.com
        self.driver.get("https://www.google.com")
```

Leggi Selenium-webdriver con Python, Ruby e Javascript insieme allo strumento CI online:

https://riptutorial.com/it/selenium-webdriver/topic/10091/selenium-webdriver-con-python--ruby-e-javascript-insieme-allo-strumento-ci

Capitolo 27: Seleziona la classe

Sintassi

- **Giava**
- deselezionare tutto()
- deselectByIndex (int index)
- deselectByValue (valore java.lang.String)
- deselectByVisibleText (testo java.lang.String)
- getAllSelectedOptions ()
- getFirstSelectedOption ()
- GetOptions ()
- isMultiple ()
- selectByIndex (indice int)
- selectByValue (valore java.lang.String)
- selectByVisibleText (testo java.lang.String)

Parametri

parametri	Dettagli
indice	L'opzione a questo indice sarà selezionata
valore	Il valore da abbinare
testo	Il testo visibile da abbinare

Osservazioni

`Select` classe di Selenium WebDriver fornisce metodi utili per interagire con le opzioni `select` . L'utente può eseguire operazioni su un menu a discesa di selezione e anche deselezionare l'operazione utilizzando i metodi seguenti.

In **C #** la classe Select è in realtà `SelectElement`

Examples

Diversi modi per selezionare dall'elenco DropDown

Di seguito è riportata la pagina HTML

```
<html>
<head>
<title>Select Example by Index value</title>
```

```
</head>
<body>
<select name="Travel"><option value="0" selected> Please select</option>
<option value="1">Car</option>
<option value="2">Bike</option>
<option value="3">Cycle</option>
<option value="4">Walk</option>
</select>
</body>
</html>
```

GIAVA

Seleziona per indice

Per selezionare l'opzione per Indice usando Java

```
public class selectByIndexExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByIndex(1); //This will select the first 'Option' from 'Select' List i.e.
Car
    }
}
```

Seleziona per valore

```
public class selectByValueExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByValue("Bike"); //This will select the 'Option' from 'Select' List which
has value as "Bike".
        //NOTE: This will be case sensitive
    }
}
```

Seleziona per testo di visibilità

```
public class selectByVisibilityTextExample {
    WebDriver driver;
    @Test
    public void selectSamples()
    {
        driver = new FirefoxDriver();
        driver.get("URL GOES HERE");
        WebElement element=driver.findElement(By.name("Travel")); //This is the 'Select'
element locator
        Select sel=new Select(element);
        sel.selectByVisibleText("Cycle"); //This will select the 'Option' from 'Select' List
who's visibility text is "Cycle".
        //NOTE: This will be case sensitive
    }
}
```

C

Tutti gli esempi seguenti sono basati sull'interfaccia `IWebDriver` generica

Seleziona per indice

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByIndex(0);
```

Seleziona per valore

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByIndex("1");
//NOTE: This will be case sensitive
```

Seleziona per testo

```
IWebElement element=driver.FindElement(By.name("Travel"));
SelectElement selectElement = new SelectElement(title);
selectElement.SelectByText("Walk");
```

Leggi Seleziona la classe online: https://riptutorial.com/it/selenium-webdriver/topic/6426/seleziona-la-classe

Capitolo 28: Usando le annotazioni @FindBy in Java

Sintassi

- CLASS_NAME: @FindBy (className = "classname")
- CSS: @FindBy (css = "css")
- ID: @FindBy (id = "id")
- ID_OR_NAME: @FindBy (how = How.ID_OR_NAME, using = "idOrName")
- LINK_TEXT: @FindBy (linkText = "text")
- NOME: @FindBy (name = "nome")
- PARTIAL_LINK_TEXT: @FindBy (partialLinkText = "testo")
- TAG_NAME: @FindBy (tagName = "tagname")
- XPATH: @FindBy (xpath = "xpath")

Osservazioni

Nota che ci sono due modi per usare l'annotazione. Esempi:

```
@FindBy(id = "id")
```

e

```
@FindBy(how = How.ID, using  ="id")
```

sono uguali e entrambi cercano l'elemento in base al suo ID. In caso di `ID_OR_NAME` puoi utilizzare solo

```
@FindBy(how = How.ID_OR_NAME, using ="idOrName")
```

`PageFactory.initElements()` necessario utilizzare `PageFactory.initElements()` dopo l'istanza di oggetti di pagina per trovare gli elementi contrassegnati con `@FindBy` annotazione `@FindBy` .

Examples

Esempio di base

Supponiamo che stiamo usando il modello di oggetto Page . Classe di oggetto della pagina:

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.FindBy;
import org.openqa.selenium.support.PageFactory;
```

```
public class LoginPage {

    @FindBy(id="loginInput") //method used to find WebElement, in that case Id
    WebElement loginTextbox;

    @FindBy(id="passwordInput")
    WebElement passwordTextBox;

    //xpath example:
    @FindBy(xpath="//form[@id='loginForm']/button(contains(., 'Login')")
    WebElement loginButton;

    public void login(String username, String password){
        // login method prepared according to Page Object Pattern
        loginTextbox.sendKeys(username);
        passwordTextBox.sendKeys(password);
        loginButton.click();
        // because WebElements were declared with @FindBy, we can use them without
        // driver.find() method
    }
}
```

E classe di test:

```
class Tests{
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        LoginPage loginPage = new LoginPage();

        //PageFactory is used to find elements with @FindBy specified
        PageFactory.initElements(driver, loginPage);
        loginPage.login("user", "pass");
    }
}
```

Ci sono alcuni metodi per trovare WebElements usando @FindBy - controlla la sezione Sintassi.

Leggi Usando le annotazioni @FindBy in Java online: https://riptutorial.com/it/selenium-webdriver/topic/6777/usando-le-annotazioni--findby-in-java

Capitolo 29: Utilizzo di Selenium Webdriver con Java

introduzione

Il webdriver Selenium è un framework di automazione web che consente di testare la tua applicazione web su browser Web diversi. A differenza del Selenium IDE, il webdriver ti consente di sviluppare i tuoi casi di test in un linguaggio di programmazione di tua scelta. Supporta Java, .Net, PHP, Python, Perl, Ruby.

Examples

Apertura della finestra del browser con URL specifico utilizzando Selenium Webdriver in Java

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;

class test_webdriver{
    public static void main(String[] args) {
       WebDriver driver = new FirefoxDriver();
       driver.get("http://stackoverflow.com/");
       driver.close();
    }
}
```

Aprire una finestra del browser con il metodo to ()

Aprire un browser con il metodo to ().

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
class navigateWithTo{
    public static void main(String[] args) {
       WebDriver driver = new FirefoxDriver();
       driver.navigate().to("http://www.example.com");
       driver.close();
    }
}
```

Leggi Utilizzo di Selenium Webdriver con Java online: https://riptutorial.com/it/selenium-webdriver/topic/9158/utilizzo-di-selenium-webdriver-con-java

Titoli di coda

S. No	Capitoli	Contributors
1	Iniziare con selenio-webdriver	Abhilash Gupta, Alice, Community, Eugene S, iamdanchiv, Jakub Lokša, Josh, Kishor, Michal, Mohit Tater, Pan, Priyanshu Shekhar, rg702, Santoshsarma, Tomislav Nakic-Alfirevic, vikingben
2	Aspettare	Jakub Lokša, Josh, Kenil Fadia, Liam, Moshisho, noor, Sajal Singh, Saurav
3	Azioni (Emulazione di gesti complessi dell'utente)	Josh, Kenil Fadia, Liam, Priyanshu Shekhar, Tom Mc
4	Browser senza testa	Abhilash Gupta, Jakub Lokša, Liam, r_D, Tomislav Nakic-Alfirevic
5	Cambio di frame	Andersson, dreamwork801, Jakub Lokša, Java_deep, Jim Ashworth, Karthik Taduvai, Kyle Fairns, Liam, lloyd, Priyanshu Shekhar, SlightlyKosumi
6	Configurazione della griglia di selenio	mnoronha, Prasanna Selvaraj, selva, Thomas
7	Eccezioni in Selenium-WebDriver	Brydenr
8	Esecuzione di Javascript nella pagina	Brydenr, Liam
9	Gestione degli errori nell'automazione con selenio	Andersson
10	Gestisci un avviso	Andersson, Aurasphere, Priya, SlightlyKosumi
11	Gli ascoltatori	Erki M.
12	Griglia di selenio	Eugene S, Y-B Cause
13	Impostazione / acquisizione della dimensione della	Abhilash Gupta

	finestra del browser	
14	Impostazione del selenio e2e	Ashish Deshmukh
15	Individuazione degli elementi Web	alecxe, daOnlyBG, Jakub Lokša, Josh, Liam, Łukasz Piaszczyk, Moshisho, NarendraR, noor, Priya, Sakshi Singla, Siva
16	Interagire con le finestre del browser	Andersson, Josh, Sakshi Singla
17	Interazione con l'elemento Web	Jakub Lokša, Liam, Moshisho, Siva, Sudha Velan
18	Modello di oggetto della pagina	JeffC, Josh, Moshisho, Priyanshu Shekhar, Sakshi Singla
19	Navigare tra più fotogrammi	Pavan T, Raghvendra
20	Navigazione	Andersson, Liam, Santoshsarma, viralpatel
21	Prendendo Screenshots	Abhilash Gupta, Sanchit
22	Programma di base del selenio per web	Jakub Lokša, Josh, Liam, Priya, Sudha Velan, Thomas, vikingben
23	Rapporti HTML	Ashish Deshmukh
24	Robot nel selenio	Priyanshu Shekhar
25	scorrimento	Andersson, Sagar007
26	Selenium-webdriver con Python, Ruby e Javascript insieme allo strumento CI	Brydenr
27	Seleziona la classe	Gaurav Lad, Liam
28	Usando le annotazioni @FindBy in Java	Alex Wittig, Łukasz Piaszczyk
29	Utilizzo di Selenium Webdriver con Java	r_D, the_coder