eBook Gratuit

APPRENEZ selenium-webdriver

eBook gratuit non affilié créé à partir des **contributeurs de Stack Overflow.**

#selenium-webdriver

Table des matières

À propos

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

Chapitre 1: Démarrer avec selenium-webdriver

Remarques

Cette section fournit une vue d'ensemble de ce qu'est le sélénium-webdriver et pourquoi un développeur peut vouloir l'utiliser.

Il devrait également mentionner tous les grands sujets au sein de sélénium-webdriver, et établir un lien avec les sujets connexes. Comme la documentation de selenium-webdriver est nouvelle, vous devrez peut-être créer des versions initiales de ces rubriques connexes.

Versions

Version	Date de sortie
0.0.1	2016-08-03

Examples

Installation ou configuration

Pour commencer à utiliser WebDriver, vous devez obtenir le pilote approprié sur le site Selenium : Téléchargements Selenium HQ . À partir de là, vous devez télécharger le pilote correspondant au (x) navigateur (s) et / ou à la (aux) plate-forme (s) sur lesquels vous essayez d'exécuter WebDriver. Par exemple, si vous testiez Chrome, le site Selenium vous indiquera:

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

Pour télécharger `chromedriver.exe` .

Enfin, avant de pouvoir utiliser WebDriver, vous devez télécharger les liaisons de langue appropriées, par exemple si vous utilisez C #, vous pouvez accéder au téléchargement depuis la page Selenium HQ Downloads pour obtenir les fichiers .dll requis ou les télécharger en tant que packages dans Visual Studio. via le gestionnaire de paquets NuGet.

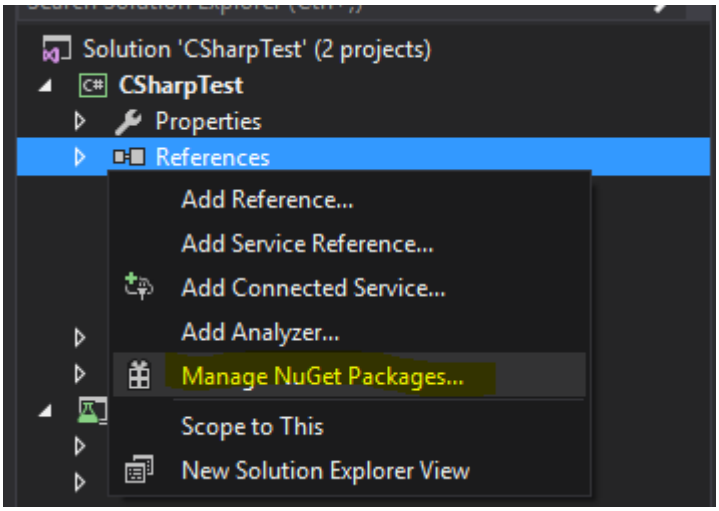
Les fichiers requis doivent maintenant être téléchargés. Pour plus d'informations sur l'utilisation de WebDriver, reportez-vous à la documentation de `selenium-webdriver` .

Pour Visual Studio [NuGet]

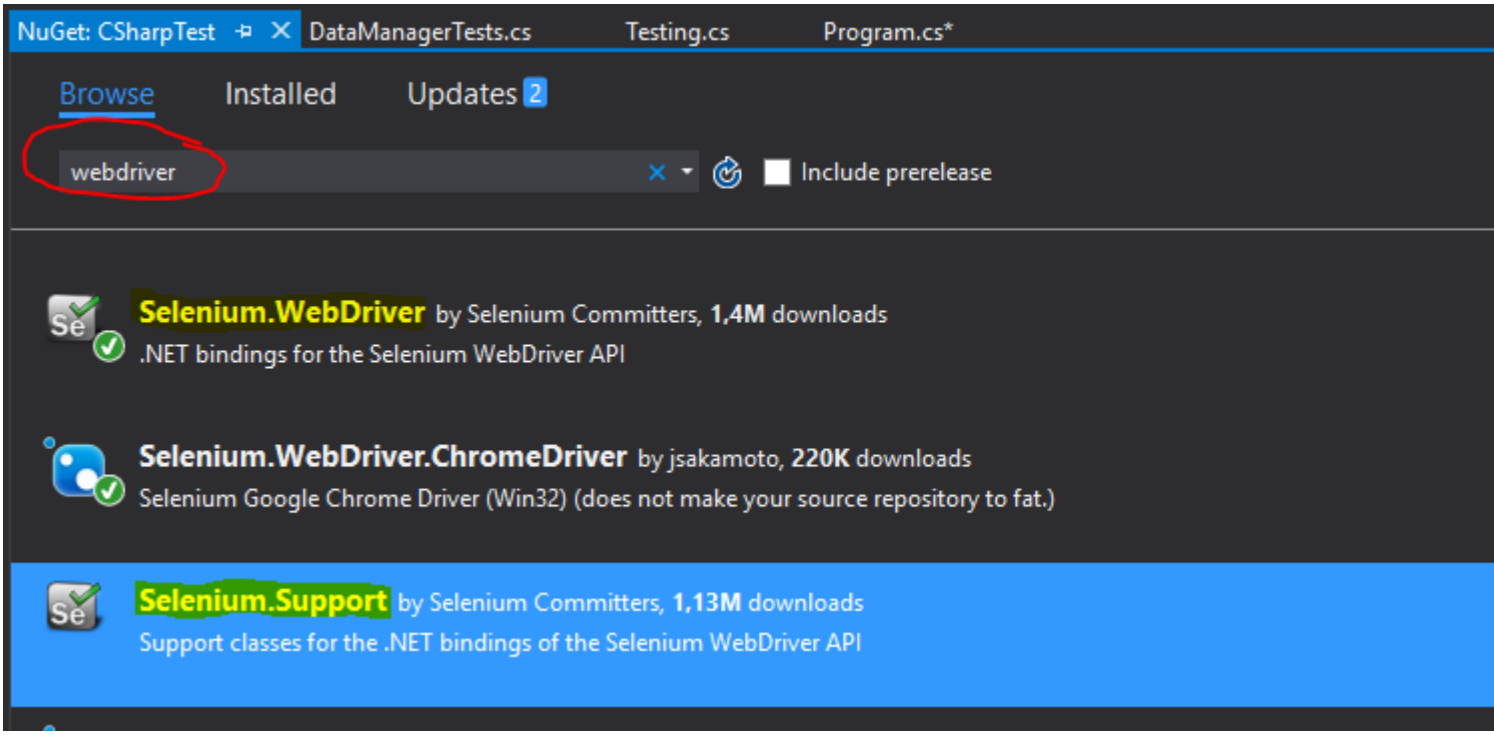
La méthode la plus simple pour installer Selenium WebDriver consiste à utiliser un gestionnaire de

paquets NuGet.

Dans votre projet, cliquez avec le bouton droit sur "Références" et cliquez sur "Gérer les packages NuGet" comme indiqué:



Ensuite, tapez dans la zone de recherche " *webdriver* ". Vous devriez alors voir quelque chose comme ceci:



Installez " **Selenium.WebDriver** ", et " **Selenium.Support** " (le package de support inclut des ressources supplémentaires, telles que Wait) en cliquant sur le bouton Installer sur le côté droit.

Ensuite, vous pouvez installer vos WebDrivers que vous souhaitez utiliser, comme l'un de ceux-ci:

- Selenium.WebDriver.ChromeDriver (Google Chrome)
- PhantomJS (sans tête)
-

Qu'est-ce que Selenium WebDriver?

Selenium est un ensemble d'outils conçus pour automatiser les navigateurs. Il est couramment utilisé pour les tests d'applications Web sur plusieurs plates-formes. Quelques outils sont disponibles sous le parapluie Selenium, tels que Selenium WebDriver (ex-Selenium RC), Selenium IDE et Selenium Grid.

WebDriver est une *interface de* contrôle à distance qui vous permet de manipuler des éléments DOM dans des pages Web et de contrôler le comportement des agents utilisateurs. Cette interface fournit un protocole de connexion indépendant du langage qui a été implémenté pour diverses plates-formes telles que:

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

ainsi que d'autres implémentations:

- EventFiringWebDriver
- HtmlUnitDriver
- PhantomJSDriver
- RemoteWebDriver

Selenium WebDriver est l'un des outils de Selenium qui fournit des API orientées objet dans divers langages pour permettre un meilleur contrôle et l'application de pratiques de développement de logiciels standard. Pour simuler avec précision la manière dont un utilisateur interagira avec une application Web, il utilise les «événements de niveau du système d'exploitation natif», contrairement aux «événements JavaScript synthétisés».

Liens à consulter:

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

Installation ou configuration pour Java

Pour écrire des tests en utilisant Selenium Webdriver et Java comme langage de programmation, vous devez télécharger les fichiers JAR de Selenium Webdriver sur le site Web de Selenium.

Il existe plusieurs façons de configurer un projet Java pour le webdriver Selenium, l'un des plus faciles à utiliser est d'utiliser Maven. Maven télécharge les liaisons Java requises pour Selenium webdriver, y compris toutes les dépendances. L'autre méthode consiste à télécharger les fichiers JAR et à les importer dans votre projet.

Étapes pour configurer le projet Selenium Webdriver à l'aide de Maven:

1. Installez maven sur la fenêtre windows suivant ce document: https://maven.apache.org/install.html
2. Créer un dossier avec le nom `selenium-learing`
3. Créez un fichier dans le dossier ci-dessus en utilisant n'importe quel éditeur de texte nommé `pom.xml`
4. Copiez le contenu ci-dessous dans `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>
```

Remarque : assurez-vous que la version que vous avez spécifiée ci-dessus est la plus récente. Vous pouvez vérifier la dernière version à partir d'ici: http://docs.seleniumhq.org/download/maven.jsp

5. À l'aide de la ligne de commande, exécutez la commande ci-dessous dans le répertoire du projet.

   ```
   mvn clean install
   ```

La commande ci-dessus téléchargera toutes les dépendances requises et les ajoutera ensuite au projet.

6. Écrivez ci-dessous la commande pour générer un projet eclipse que vous pouvez importer dans l'IDE Eclipse.

   ```
   mvn eclipse:eclipse
   ```

7. Pour importer le projet dans eclipse ide, vous pouvez suivre les étapes ci-dessous

   Ouvrez Elipse -> Fichier -> Importer -> Général -> Projet existant dans l'espace de travail -> Suivant -> Parcourir -> Localisez le dossier contenant pom.xml -> Ok -> Terminer

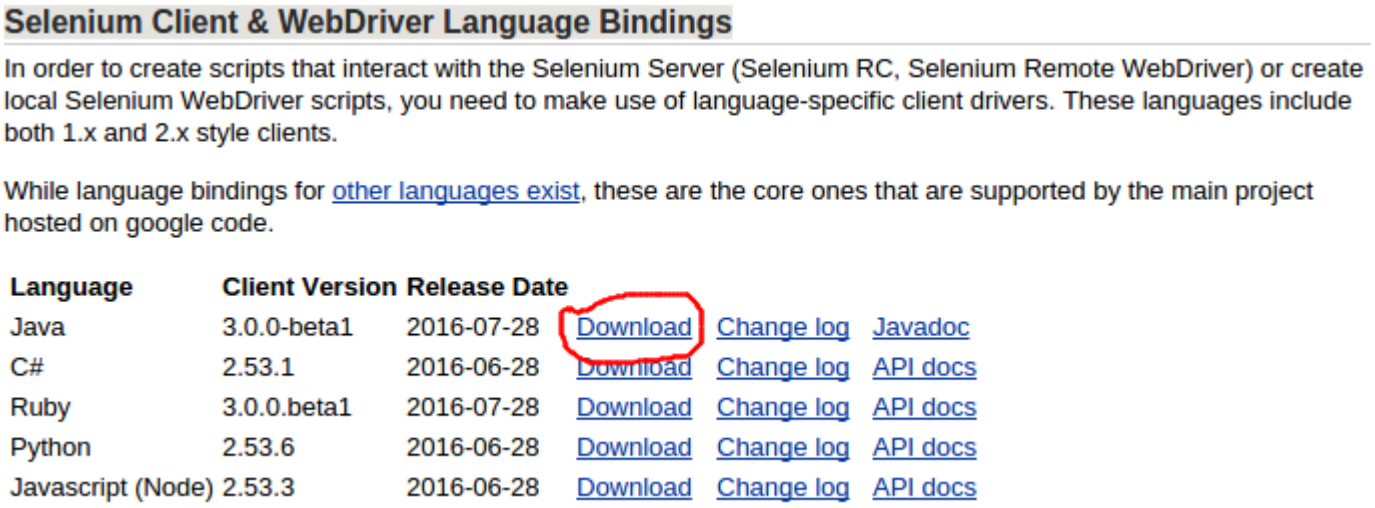
Installez le plug-in m2eclipse en cliquant avec le bouton droit sur votre projet et sélectionnez Maven -> Activer la gestion des dépendances.

Étapes pour configurer le projet Selenium Webdriver à l'aide de fichiers Jar

1. Créez un nouveau projet dans Eclipse en suivant les étapes ci-dessous.

   Ouvrir Elipse -> Fichier -> Nouveau -> Projet Java -> Fournir un nom (apprentissage du sélénium) -> Terminer

2. Téléchargez les fichiers jar depuis http://www.seleniumhq.org/download/ . Vous devez télécharger à la fois **Selenium Standalone Server** et **Selenium Client & WebDriver Language Bindings** . Puisque ce document parle de Java, vous devez télécharger uniquement jar depuis la section Java. Jetez un coup d'oeil dans la capture d'écran ci-jointe.



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

Remarque: Selenium Standalone Server est requis uniquement si vous souhaitez utiliser le serveur distant pour exécuter les tests. Comme ce document est tout au-dessus de la mise en place du projet, il est donc préférable d'avoir tout en place.

3. Les pots seront téléchargés dans un fichier zip, décompressez-les. Vous devriez pouvoir voir directement `.jar`

4. Dans éclipse, cliquez avec le bouton droit sur le projet que vous avez créé à l'étape 1 et suivez les étapes ci-dessous.

   Propriétés -> Java Build Path -> Sélectionnez l'onglet Bibliothèques -> Cliquez sur Ajouter des fichiers Jars externes -> Localisez le dossier JAR décompressé que vous avez téléchargé ci-dessus -> Sélectionnez tous les fichiers `lib` dossier `lib` -> Cliquez sur OK > Localisez le même dossier décompressé -> Sélectionnez le fichier jar qui se trouve en dehors du dossier lib (`client-combined-3.0.0-beta1-nodeps.jar`) -> Ok

De même, ajoutez le `Selenium Standalone Server` après l'étape ci-dessus.

5. Vous pouvez maintenant commencer à écrire du code de sélénium dans votre projet.

PS : la documentation ci-dessus est basée sur la version bêta de sélénium-3.0.0, donc les noms des fichiers JAR spécifiés peuvent changer avec la version.

Lire Démarrer avec selenium-webdriver en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/878/demarrer-avec-selenium-webdriver

Chapitre 2: Actions (émulation de gestes complexes)

Introduction

La classe `Actions` nous permet d'imiter avec précision la manière dont un utilisateur interagirait avec une page Web ou des éléments. En utilisant une instance de cette classe, vous pouvez décrire une série d'actions, telles que cliquer, double-cliquer, faire glisser, appuyer sur des touches, etc. Une fois ces actions décrites, vous devez appeler les actions pour exécuter les actions. (`.Build()`), puis leur demander d'être exécuté (`.Perform()`). Donc, nous devons décrire, construire, exécuter. Les exemples ci-dessous développeront cela.

Syntaxe

- dragAndDrop (source WebElement, cible WebElement)
- dragAndDropBy (source WebElement, int xOffset, int yOffset)
- effectuer()

Paramètres

Paramètres	Détails
la source	Élément à émuler bouton vers.
cible	Élément à déplacer et à relâcher la souris.
xOffset	x coordonne pour aller à.
yOffset	y coordonne pour passer à.

Remarques

Cette section contient des informations sur la classe Actions de Selenium WebDriver. La classe Actions vous fournit des méthodes pratiques pour effectuer des gestes complexes tels que le glisser-déposer, le maintien et le clic, etc.

Examples

Glisser déposer

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();
        }
    }
}
```

Ce qui précède trouvera une `source IWebElement` et la déposera dans la seconde `target IWebElement` .

Java

Glissez et déposez en utilisant le webelement source et cible.

Une méthode pratique qui effectue un clic sur l'emplacement de l'élément source, se déplace à l'emplacement de l'élément cible, puis libère la souris.

```
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();
    }
}
```

Faites glisser un élément et déposez-le à un décalage donné.

Une méthode pratique qui effectue un click-and-hold à l'emplacement de l'élément source, se

déplace d'un décalage donné (x et y, les deux entiers), puis libère la souris.

```
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
```

Déplacer vers l'élément

C

Supposons que vous vouliez vérifier que lorsque vous survolez un élément, une liste déroulante est affichée. Vous voudrez peut-être vérifier le contenu de cette liste ou peut-être sélectionner une option dans la liste.

Commencez par créer une action, pour survoler l'élément *(par exemple, mon élément a le texte du lien "Admin")* :

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

Dans l'exemple ci-dessus:

- Vous avez créé la `mouseHover`
- Vous avez dit au `driver` de passer à un élément spécifique
- De là, vous pouvez effectuer d'autres `Actions` avec l'objet `mouseHover` ou continuer à tester avec votre objet `driver`

Cette approche est particulièrement utile lorsque le clic sur un élément exécute une fonction différente de celle du survol.

Un exemple complet:

```
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);
```

Lire Actions (émulation de gestes complexes) en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/4849/actions--emulation-de-gestes-complexes-

Chapitre 3: Attendez

Examples

Types d'attentes dans Selenium WebDriver

Lors de l'exécution d'une application Web, il est nécessaire de prendre en compte le temps de chargement. Si votre code tente d'accéder à un élément qui n'est pas encore chargé, WebDriver lancera une exception et votre script s'arrêtera.

Il existe trois types de Waits -

- **Attentes implicites**
- **Attentes explicites**
- **Attentes Courantes**

Les attentes implicites sont utilisées pour définir le temps d'attente tout au long du programme, tandis que les attentes explicites ne sont utilisées que sur des parties spécifiques.

Attente implicite

Une attente implicite consiste à demander à WebDriver d'interroger le DOM pendant un certain temps lorsqu'il tente de trouver un élément ou des éléments s'ils ne sont pas immédiatement disponibles. Les attentes implicites sont essentiellement votre façon de dire à WebDriver la latence que vous souhaitez voir si l'élément Web spécifié n'est pas présent et que WebDriver recherche. Le paramètre par défaut est 0. Une fois définie, l'attente implicite est définie pour la durée de vie de l'instance de l'objet WebDriver. L'attente implicite est déclarée dans la partie instanciation du code à l'aide de l'extrait de code suivant.

Exemple en **Java** :

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

Exemple en **C #** :

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

Donc, dans ce cas, vous dites à WebDriver qu'il doit attendre 15 secondes dans le cas où l'élément spécifié n'est pas disponible sur l'interface utilisateur (DOM).

Attente explicite

Vous pouvez rencontrer des cas où un élément prend plus de temps à charger. Définir l'attente implicite pour de tels cas n'a pas de sens car le navigateur attendra inutilement le même temps pour chaque élément, augmentant ainsi le temps d'automatisation. L'attente explicite aide ici en contournant l'attente implicite pour certains éléments spécifiques.

Les attentes explicites sont des attentes intelligentes limitées à un élément Web particulier. En utilisant des attentes explicites, vous indiquez à WebDriver au maximum qu'il faut attendre X unités de temps avant d'abandonner.

Les attentes explicites sont effectuées à l'aide des classes WebDriverWait et ExpectedConditions. Dans l'exemple ci-dessous, nous allons attendre jusqu'à 10 secondes pour qu'un élément dont l'identifiant est un nom d'utilisateur soit visible avant de passer à la commande suivante. Voici les étapes.

Exemple en **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 a des conditions communes prédéfinies pour attendre un élément. Cliquez ici pour voir la liste de ces conditions dans la liaison Java.

Exemple en **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);
}
```

Dans cet exemple, le système attendra 10 secondes jusqu'à ce que l'élément soit visible. Si

l'élément n'est pas visible après l'expiration du délai, le WebDriver lancera une `WebDriverTimeoutException` .

Remarque: Si l'élément est visible avant l'expiration du délai de 10 secondes, le système procédera immédiatement à un traitement ultérieur.

Attendez

Contrairement à l'attente implicite et explicite, l'attente fluide utilise deux paramètres. Valeur de temporisation et fréquence d'interrogation. Disons que nous avons une valeur de délai d'attente de 30 secondes et une fréquence d'interrogation de 2 secondes. WebDriver vérifie l'élément après toutes les 2 secondes jusqu'à la valeur du délai d'attente (30 secondes). Une fois que la valeur du délai d'expiration est dépassée sans aucun résultat, une exception est levée. Vous trouverez ci-dessous un exemple de code indiquant la mise en œuvre d'une attente fluide.

Exemple en **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 autre avantage d'utiliser l'attente fluide est que nous pouvons ignorer des types spécifiques d'exceptions (par ex. NoSuchElementExceptions) en attendant. En raison de toutes ces dispositions, l'attente fluide est utile dans les applications AJAX ainsi que dans les scénarios où le temps de chargement des éléments varie souvent. L'utilisation stratégique d'attentes fluides améliore considérablement les efforts d'automatisation.

Différents types de conditions d'attente explicites

En attente explicite, vous vous attendez à une condition. Par exemple, vous voulez attendre qu'un élément soit cliquable.

Voici une démonstration de quelques problèmes communs.

S'il vous plaît noter: Dans tous ces exemples , vous pouvez utiliser tout `By` *un localisateur, comme* `classname` , `xpath` , `link text` , `tag name` la `cssSelector` `tag name` *ou* `cssSelector`

Attendez que l'élément soit visible

Par exemple, si le chargement de votre site Web prend un certain temps, vous pouvez attendre

que la page se charge complètement et que votre élément soit visible par le WebDriver.

C #

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

Java

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

Attendez que l'élément ne soit plus visible

Comme avant, mais inversé.

C #

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

Java

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

Attendez que du texte soit présent dans l'élément spécifié

C #

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

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

Java

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

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

Si vous allez au lien ci-dessus, vous verrez toutes les conditions d'attente.

La différence entre l'utilisation de ces conditions d'attente réside dans leur paramètre d'entrée.

Cela signifie que vous devez passer WebElement si son paramètre d'entrée est WebElement,

vous devez passer le localisateur d'élément s'il utilise le localisateur By comme paramètre d'entrée.

Choisissez judicieusement le type de condition d'attente que vous souhaitez utiliser.

En attente de demandes Ajax à remplir

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);
            }
        }
    }
}
```

Cet exemple est utile pour les pages où des requêtes ajax sont faites, ici nous utilisons le `IJavaScriptExecutor` pour exécuter notre propre code JavaScript. Comme il est dans un `while` en boucle continue à fonctionner jusqu'à ce que `ajaxIsComplete == true` et ainsi l'instruction de retour est exécutée.

Nous vérifions que toutes les requêtes ajax sont complètes en confirmant que `jQuery.active` est égal à `0` . Cela fonctionne car chaque fois qu'une nouvelle requête ajax est faite, `jQuery.active` est incrémenté et chaque fois qu'une requête est complétée, il est décrémenté, de là on peut déduire que lorsque `jQuery.active == 0` toutes les requêtes ajax doivent être complètes.

Fluent Wait

Fluent wait est une super-classe d'attente **explicite** (`WebDriverWait`) qui est plus configurable car elle peut accepter un argument pour la fonction wait. Je vais laisser **une** attente **implicite** , car il est préférable de l'éviter.

Utilisation (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;
  }
});
```

Lorsque vous utilisez l' attente explicite avec ses valeurs par défaut, il s'agit simplement d'un `FluentWait<WebDriver>` avec les valeurs par défaut suivantes: `DEFAULT_SLEEP_TIMEOUT = 500;` et en ignorant `NotFoundException` .

Attendez

Chaque instance de FluentWait définit la durée maximale d'attente d'une condition, ainsi que la fréquence à laquelle vérifier la condition. En outre, l'utilisateur peut configurer l'attente pour ignorer certains types d'exceptions en attente, telles que NoSuchElementExceptions lors de la recherche d'un élément sur la page. Il est associé au pilote.

```
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"));
```

Lire Attendez en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/4435/attendez

Chapitre 4: Changement de cadre

Syntaxe

- **Java**
- driver.switchTo (). frame (nom de la chaîne);
- driver.switchTo (). frame (identifiant de chaîne);
- driver.switchTo (). frame (int index);
- driver.switchTo (). frame (WebElement frameElement);
- driver.switchTo (). defaultContent ();
- **C #**
- driver.SwitchTo (). Frame (int frameIndex);
- driver.SwitchTo (). Frame (frameElement IWebElement);
- driver.SwitchTo (). Frame (string FrameName);
- driver.SwitchTo (). DefaultContent ();
- **Python**
- driver.switch_to_frame (nameOrId)
- driver.switch_to.frame (nameOrId)
- driver.switch_to_frame (index)
- driver.switch_to.frame (index)
- driver.switch_to_frame (frameElement)
- driver.switch_to.frame (frameElement)
- driver.switch_to_default_content ()
- driver.switch_to.default_content ()
- **JavaScript**
- driver.switchTo (). frame (nameOrId)
- driver.switchTo (). frame (index)
- driver.switchTo (). defaultContent ()

Paramètres

paramètre	détails
nameOrId	Sélectionnez un cadre par son nom.
indice	Sélectionnez un cadre par son index de base.
frameElement	Sélectionnez un cadre en utilisant son WebElement précédemment localisé

Examples

Pour basculer vers un cadre en utilisant Java

Pour une instance, si le code source html d'une vue ou d'un élément html est enveloppé par un iframe comme celui-ci:

```
<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 />
```

... alors pour effectuer une action quelconque sur les éléments web de l'iframe, vous devez d'abord basculer le focus sur l'iframe, en utilisant l'une des méthodes ci-dessous:

Utiliser un identifiant de trame (doit être utilisé uniquement si vous connaissez l'id de l'iframe).

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

Utiliser le nom du cadre (doit être utilisé uniquement si vous connaissez le nom de l'iframe).

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

Utilisation de l'index d'images (doit être utilisé uniquement si vous ne possédez pas l'id ou le nom de l'iframe), où l'index définit la position de l'iframe parmi toutes les images.

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

Remarque: Si vous avez trois images dans la page, la première sera à l'index 0, la deuxième à l'index 1 et la troisième à l'index 2.

Utilisation de webelement précédemment localisé (doit être utilisé uniquement si vous avez déjà localisé le cadre et l'avez renvoyé en tant que `WebElement`).

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

Alors, pour cliquer sur l'ancre `Red Ball` :

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

Pour sortir d'un cadre en Java

Pour basculer le focus sur le document principal ou la première image de la page. Vous devez utiliser la syntaxe ci-dessous.

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

Passer à un cadre en utilisant C

1. Passez à un cadre par index.

Ici, nous passons à l'index 1. L'index fait référence à l'ordre des images sur la page. Cela devrait être utilisé en dernier recours, car l'identifiant ou les noms des images sont beaucoup plus fiables.

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

2. Passer à un cadre par nom

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

3. Passez à une image par titre, identifiant ou autres en transmettant IWebElement

Si vous souhaitez basculer vers un cadre par identifiant ou par titre, vous devez transmettre un élément Web en tant que paramètre:

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

Notez également que si votre cadre prend quelques secondes pour apparaître, vous devrez peut-être attendre :

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

Sortez d'un cadre:

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

Pour sortir d'un cadre en utilisant C

Pour basculer le focus sur le document principal ou la première image de la page. Vous devez utiliser la syntaxe ci-dessous.

webDriver.SwitchTo (). DefaultContent ();

Basculer entre les images enfants d'une image parent.

Considérez que vous avez un cadre parent (Frame-Parent). et 2 cadres enfants (Frame_Son, Frame_Daughter). Permet de voir diverses conditions et comment gérer.

1. Du parent au fils ou à la fille:

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

2.De Fils à Parent: Si le parent est le cadre par défaut, passez au cadre par défaut, sinon, du basculement du cadre par défaut au cadre parent. Mais vous ne pouvez pas passer directement de fils à Parent.

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

3.De fils à fille: Si votre sœur fait une erreur, ne lui criez pas, contactez votre parent. De même, vous donnez le contrôle à l'image parent et ensuite à l'image fille.

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

Attendez que vos cadres se chargent

Dans de nombreux cas, votre cadre risque de ne pas apparaître immédiatement et vous devrez probablement attendre qu'il soit chargé pour basculer. Ou bien vous aurez une exception NoSuchFrameException.

C'est donc toujours un bon choix d'attendre avant de passer. Voici un moyen idéal d'attendre qu'un bloc soit chargé.

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

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

// déclenche une exception uniquement si votre cadre n'est pas visible avec 300 secondes dans votre temps d'attente}

Lire Changement de cadre en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/4589/changement-de-cadre

Chapitre 5: Configuration de la grille de sélénium

Introduction

Selenium Grid est un framework pour exécuter des tests distribués sur toute une gamme de périphériques de test. Il est utilisé pour tester des applications Web. Il est possible d'écrire des tests dans différents langages de programmation populaires, notamment C #, Groovy, Java, Perl, PHP, Python et Ruby. Les tests peuvent être exécutés sur une gamme de navigateurs Web sur des plates-formes telles que Windows, Linux et OS X.

C'est un logiciel open-source, publié sous la licence Apache 2.0: les développeurs Web peuvent le télécharger et l'utiliser sans frais.

Syntaxe

- pour exécuter le fichier jar, voici la syntaxe de chaque fichier jar
- `java -jar <jar-file-full-name>.jar -<your parameters if any>`

Paramètres

Paramètres	Détails
rôle	Est-ce ce qui dit le sélénium qu'il était `hub` ou `node`
Port	Ceci permet de spécifier le port que le `hub` ou le `node` doit écouter.
centre	Ce paramètre est utilisé dans le `node` pour spécifier l'URL du concentrateur
browserName	Son utilisé dans le `node` pour spécifier le nom du navigateur comme Firefox, Chrome, Internet Explorer
maxInstances	C'est là que l'instance du navigateur est spécifiée, par exemple. 5 signifie qu'il y aura 5 instances du navigateur que l'utilisateur spécifie sera présent.
nodeConfig	Un fichier de configuration Json pour le noeud. Vous pouvez spécifier le rôle, le port, etc. ici
hubConfig	Un fichier de configuration Json pour le noeud. Vous pouvez spécifier le rôle, le port, les instances max, etc. ici

Examples

Code Java pour Selenium Grid

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

Création d'un hub et d'un noeud Selenium Grid

Créer un hub

Une configuration rapide pour la configuration d'un concentrateur et d'un nœud dans une grille de sélénium. Pour plus d'informations, voir: Grid 2 docs

Exigences

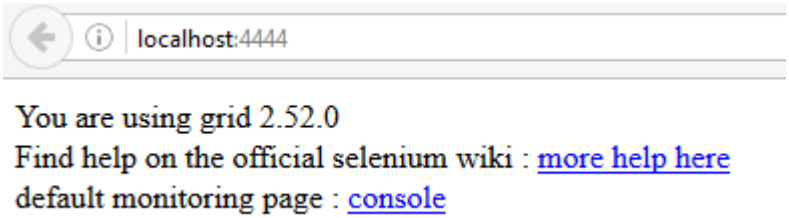
Pour configurer un hub de grille, vous avez besoin de ce qui suit:

- Selenium-server-standalone-.jar

Créer le hub

Pour créer un concentrateur, vous devez exécuter le serveur sélénium.

1. Télécharger Selenium-server-standalone-.jar
2. Ouvrez votre terminal et naviguez jusqu'au dossier où Selenium-server-standalone-.jar est
3. Exécutez la commande suivante:
    1. Pour la configuration par défaut `java -jar selenium-server-standalone-<Version>.jar -role hub`
    2. Pour la configuration Json `java -jar selenium-server-standalone-<Version>.jar -role hub -hubConfig hubConfig.json`
4. Ouvrez http: // localhost: 4444 / vous verrez un message suivant



En cliquant sur `console` -> `View config` pour afficher la configuration des détails du concentrateur.

Créer un noeud

Exigences

Pour configurer un hub de grille, vous avez besoin de ce qui suit:

- Selenium-server-standalone-.jar
- Webdrivers
    - Pilote chrome
    - Pilote firefox
    - Pilote Microsoft Edge
- Les navigateurs
    - Chrome
    - FireFox
    - Microsoft Edge (Windows 10)

Création du noeud

Maintenant, pour créer des nœuds pour le concentrateur

1. Télécharger Selenium-server-standalone-.jar
2. Téléchargez les navigateurs que vous souhaitez tester
3. Téléchargez les pilotes pour les navigateurs que vous souhaitez tester
4. Ouvrir un nouveau terminal et accéder à l'emplacement du fichier jar du serveur sélénium
5. Exécutez la commande suivante:
    1. pour la configuration par défaut `java -jar selenium-server-standalone-<VERSION NUMBER>.jar -role node`
    2. Pour la configuration Json `java -jar selenium-server-standalone-<Version>.jar -role node -nodeConfig nodeConfig.json`
6. Maintenant, allez à http: // localhost: 4444 / grid / console pour voir les détails du noeud

Configuration via Json

Un exemple de configuration pour un concentrateur:

```
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"
}
```

Un exemple de configuration pour un noeud

```
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": {}
}
```

Lire Configuration de la grille de sélénium en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/2504/configuration-de-la-grille-de-selenium

Chapitre 6: Configuration du sélénium e2e

Introduction

Cette rubrique couvre la configuration complète de Selenium, à savoir Selenium Webdriver + TestNG + Maven + Jenkins.

Pour l'ajout d'un rapport, veuillez vous reporter à la rubrique Rapports HTML

Examples

Configuration du test

TestNG est votre framework de test mis à jour pour junit. Nous allons utiliser **testng.xml** pour invoquer des suites de tests. Ceci est utile lorsque nous allons utiliser CI en avant.

testng.xml

Dans le dossier racine de votre projet, créez un fichier xml portant le nom testng.xml. Notez que ce nom peut également être différent, mais par commodité, il est utilisé comme "test" partout.

Voici le code simple pour le fichier 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>
```

Configuration Maven

À déterminer Comment configurer pom.xml pour appeler testng.xml

Configuration de Jenkins

À déterminer Va couvrir la configuration de Jenkins pour tirer du code de git / bitbucket etc.

Lire Configuration du sélénium e2e en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/10724/configuration-du-selenium-e2e

Chapitre 7: Défilement

Introduction

Cette rubrique proposera plusieurs approches pour effectuer le défilement avec le `selenium`

Examples

Défilement avec Python

1. Faire défiler jusqu'à l'élément cible (bouton "BROWSE TEMPLATES" en bas de la page) avec `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. Faire défiler jusqu'à l'élément cible (bouton "BROWSE TEMPLATES" au bas de la page) avec `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. Faire défiler jusqu'à l'élément cible (bouton "BROWSE TEMPLATES" en bas de la page) avec la méthode intégrée

```
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
```

Notez que `location_once_scrolled_into_view` renvoie également les coordonnées `x` , `y` de l'élément après le défilement

4. Faire défiler vers le bas de la page avec les `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
```

Notez que `send_keys(Keys.DOWN)` / `send_keys(Keys.UP)` et `send_keys(Keys.PAGE_DOWN)` / `send_keys(Keys.PAGE_UP)` peuvent également être utilisés pour le défilement.

Défilement différent utilisant java de différentes manières

> Ci-dessous donne la solution peut également être utilisé dans un autre langage de programmation pris en charge avec quelques modifications de syntaxe

---

1. Pour faire **défiler la** page / section / division dans la page Web alors qu'il existe une barre de défilement personnalisée (pas de défilement du navigateur). Cliquez ici Pour la démonstration et la vérification, la barre de défilement a son élément indépendant.

   Dans le code ci-dessous, passez votre élément de barre de défilement et demandez des points de défilement.

```
    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. Pour faire **défiler la** page / section / division dans la page Web alors qu'il existe une barre de défilement personnalisée (pas de défilement du navigateur). Cliquez ici Pour la démonstration et la vérification, la barre de défilement a son élément indépendant.

Dans le code ci-dessous, passez votre élément de barre de défilement et demandez des points de défilement.

```
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. Pour faire défiler vers le bas lorsque **plusieurs navigateurs font défiler** (navigateur intégré) et que vous souhaitez faire défiler la **page** avec la **touche Bas** . Cliquez ici pour la démonstration

   Dans le code donné ci-dessous, passez votre élément de zone de défilement tel que `<div>` et exigez une touche vers le bas.

```
 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. Pour faire défiler vers le haut lorsque **plusieurs navigateurs font défiler** (navigateur intégré) et que vous souhaitez faire défiler vers le haut avec la **touche Page UP** . Cliquez ici pour la démonstration

   Dans le code donné ci-dessous, passez votre élément de zone de défilement tel que `<div>` et exigez une clé de page précédente.

```
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. Pour faire défiler vers le bas lorsque **plusieurs navigateurs défilent** (navigateur intégré) et que vous souhaitez faire défiler la liste avec la **touche flèche vers le** bas **uniquement** . Cliquez ici pour la démonstration

   Dans le code ci-dessous, passez votre élément de zone de défilement comme `<div>` et nécessitez une touche bas.

```
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. Pour faire défiler vers le haut lorsque **plusieurs navigateurs défilent** (navigateur intégré) et que vous souhaitez faire défiler vers le haut avec une **seule flèche vers le haut** . Cliquez ici pour la démonstration

   Dans le code ci-dessous, passez votre élément de zone de défilement comme `<div>` et nécessitez la touche Haut.

```
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. Pour faire défiler vers le haut / bas lorsque le **navigateur fait défiler** (navigateur intégré) et que vous souhaitez faire défiler vers le haut / bas avec **un point fixe uniquement** . Cliquez ici pour la démonstration

   En dessous du code donné, passez votre point de défilement. Positif signifie vers le bas et négatif signifie défiler vers le haut.

```
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. Pour faire défiler vers le haut / bas lorsque le **navigateur fait défiler** (navigateur intégré) et que vous souhaitez faire défiler l' **élément** Haut / Bas vers **Pour créer une zone visible ou un défilement dynamique** . Cliquez ici pour la démonstration

   En dessous du code donné passez votre élément.

```
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;
    }
}
```

Remarque: Veuillez vérifier votre cas et utiliser vos méthodes. Si quelque cas manque, faites le moi savoir.

Lire Défilement en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/9063/defilement

Chapitre 8: Exceptions dans Selenium-WebDriver

Introduction

Il existe un certain nombre d'exceptions pouvant être générées lors de l'utilisation d'un Webdriver. Les exemples ci-dessous sont destinés à donner une idée de leur signification.

Examples

Exceptions Python

Documentation sur les exceptions de sélénium

ElementNotInteractableException: déclenché lorsqu'un élément est présent dans le DOM, mais que les interactions avec cet élément atteignent un autre élément en raison de l'ordre de peinture

- **ElementNotSelectableException:** déclenché lors de la tentative de sélection d'un élément non sélectionnable. Exemples d'éléments non sélectionnables:
  - scénario
- **ElementNotVisibleException:** lancé lorsqu'un élément est présent dans le DOM, mais qu'il n'est pas visible et qu'il n'est donc pas possible d'interagir avec lui. Le plus souvent rencontré lorsque vous essayez de cliquer ou de lire le texte d'un élément masqué.
- **ErrorInResponseException: déclenché** lorsqu'une erreur s'est produite côté serveur. Cela peut se produire lors de la communication avec l'extension firefox ou le serveur de pilote distant.
- **ImeActivationFailedException:** déclenché lorsque l'activation d'un moteur IME a échoué.
- **ImeNotAvailableException: levée** lorsque la prise en charge d'IME n'est pas disponible. Cette exception est levée pour chaque appel de méthode lié à IME si la prise en charge d'IME n'est pas disponible sur l'ordinateur.
- **InvalidArgumentException:** Les arguments transmis à une commande sont non valides ou mal formés.
- **InvalidCookieDomainException:** Lancé lors de la tentative d'ajout d'un cookie sous un domaine différent de l'URL actuelle.
- **InvalidElementStateException: déclenché** lorsqu'une action entraînerait un état non valide pour un élément. Sous-classes:
  - ElementNotInteractableException
  - ElementNotSelectableException
  - ElementNotVisibleException
- **InvalidSelectorException: déclenché** lorsque le sélecteur utilisé pour rechercher un élément ne renvoie pas un WebElement. Actuellement, cela ne se produit que lorsque le sélecteur est une expression xpath et qu'il est invalide sur le plan syntaxique (c.-à-d. Qu'il ne s'agit pas d'une expression xpath) ou que l'expression ne sélectionne pas WebElements

(par exemple, «count (// input)»).

- **InvalidSwitchToTargetException: L'existence d'** une cible de fenêtre ou de cadre à basculer n'existe pas.
- **MoveTargetOutOfBoundsException: déclenché** lorsque la cible fournie à la méthode ActionsChains move () n'est pas valide, c'est-à-dire hors document.
- **NoAlertPresentException:** Lancé lors du passage à aucune alerte présentée. Cela peut être dû à l'appel d'une opération sur la classe Alert () lorsqu'une alerte n'est pas encore à l'écran.
- **NoSuchAttributeException: levée** lorsque l'attribut de l'élément est introuvable. Vous voudrez peut-être vérifier si l'attribut existe dans le navigateur que vous testez. Certains navigateurs peuvent avoir des noms de propriété différents pour la même propriété. (.InnerText vs Firefox .textContent) de IE8
- **NoSuchElementException: levée** lorsque l'élément est introuvable. Si vous rencontrez cette exception, vous souhaiterez peut-être vérifier les éléments suivants:
    - Vérifiez votre sélecteur utilisé dans votre find_by ...
    - L'élément n'est peut-être pas encore à l'écran au moment de l'opération de recherche, (la page Web est toujours en cours de chargement) voir selenium.webdriver.support.wait.WebDriverWait () pour savoir comment écrire un wrapper pour attendre qu'un élément apparaisse.
- **NoSuchFrameException: levée** lorsque la cible de l'image à basculer n'existe pas.
- **NoSuchWindowException: levée** lorsque la cible de la fenêtre à basculer n'existe pas. Pour trouver l'ensemble actuel des poignées de fenêtre actives, vous pouvez obtenir la liste des poignées de fenêtre actives de la manière suivante:
  `print driver.window_handles`
- **RemoteDriverServerException:**
- **StaleElementReferenceException: déclenché** lorsqu'une référence à un élément est désormais «obsolète». Stale signifie que l'élément n'apparaît plus dans le DOM de la page. Les causes possibles de StaleElementReferenceException sont notamment les suivantes:
    - Vous n'êtes plus sur la même page ou la page peut avoir été actualisée depuis la localisation de l'élément.
    - L'élément peut avoir été supprimé et ajouté à nouveau à l'écran depuis sa localisation. Comme un élément en cours de déplacement. Cela peut se produire généralement avec un framework javascript lorsque les valeurs sont mises à jour et que le noeud est reconstruit.
    - L'élément peut avoir été à l'intérieur d'un iframe ou d'un autre contexte qui a été rafraîchi.
- **TimeoutException: déclenché** lorsqu'une commande ne se termine pas suffisamment longtemps.
- **UnableToSetCookieException: déclenché** lorsqu'un pilote ne parvient pas à définir un cookie.
- **UnexpectedAlertPresentException:** déclenché lorsqu'une alerte inattendue est apparue. Généralement déclenché quand un modal attendu bloque la forme de webdriver en exécutant d'autres commandes.
- **UnexpectedTagNameException:** déclenché lorsqu'une classe de support n'a pas reçu d'élément Web attendu.
- **WebDriverException:** exception **Webdriver de** base. Toutes les exceptions webdriver utilisent WebDriverException ou InvalidStateException comme classe parente.

Lire Exceptions dans Selenium-WebDriver en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/10546/exceptions-dans-selenium-webdriver

Chapitre 9: Exécution de Javascript dans la page

Syntaxe

- objet ExecuteAsyncScript (script de chaîne, params object [] args);
- object ExecuteScript (script de chaîne, params object [] args);

Examples

C

Pour exécuter JavaScript dans une instance `IWebDriver` , vous devez `IWebDriver` le `IWebDriver` en une nouvelle interface, `IJavaScriptExecutor`

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

Vous pouvez maintenant accéder à toutes les méthodes disponibles sur l'instance `IJavaScriptExecutor` qui vous permettent d'exécuter Javascript, par exemple:

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

Python

Pour exécuter Javascript en python, utilisez `execute_script("javascript script here")` . execute_script est appelé sur une instance webdriver et peut être n'importe quel javascript valide.

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

Java

Pour exécuter Javascript en Java, créez un nouveau pilote Web qui prend en charge Javascript. Pour utiliser la fonction `executeScript()` , le pilote doit être `executeScript()` en un `JavascriptExecutor` ou une nouvelle variable peut être définie sur la valeur du pilote converti: `((JavascriptExecutor)driver)` pilote `((JavascriptExecutor)driver) . driver.executeScript()` une chaîne de caractères Javascript valide.

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

Rubis

```
require "selenium-webdriver"

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

Lire Exécution de Javascript dans la page en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/6986/execution-de-javascript-dans-la-page

Chapitre 10: Gérer une alerte

Examples

Sélénium avec Java

Voici comment gérer une alerte contextuelle en Java avec Selenium:

Il existe 3 types de popups.

1. **Alerte simple** : alerte ("Ceci est une alerte simple");
2. **Alerte de confirmation** : var popuResult = confirmer ("Confirmer avec les boutons OK et Annuler");
3. **Alerte d'invite** : var person = prompt ("Aimez-vous stackoverflow?", "Oui / Non");

Son utilisateur doit savoir quel type de popup doit être manipulé dans son test élémentaire.

Soit vous pouvez

1. accepter () accepter l'alerte
2. rejeter () rejeter l'alerte
3. getText () Pour obtenir le texte de l'alerte
4. sendKeys () Pour écrire du texte dans l'alerte

Pour une alerte simple:

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

Pour une alerte de confirmation:

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

Pour une alerte rapide:

```
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();
```

selon vos besoins.

Une autre façon de procéder est d'emballer votre code dans un try-catch:

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

C

Voici comment fermer une alerte contextuelle en C # avec Selenium:

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

selon vos besoins.

Une autre façon de procéder est d'emballer votre code dans un try-catch:

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

Python

Il existe plusieurs façons de basculer en alerte pop-up dans `Python` :

1. *Déconseillé* :

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

2. *En utilisant* `switch_to` :

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

3. *Utiliser* `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. *En déclarant l'instance de la classe* `Alert` :

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

alert = Alert(driver)
```

Pour remplir le champ de saisie dans la fenêtre contextuelle déclenchée par l' `prompt()` `JavaScript` `prompt()` :

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

Pour confirmer la boîte de dialogue contextuelle *:

```
alert.accept()
```

De rejeter:

```
alert.dismiss()
```

Pour obtenir du texte à partir de la fenêtre contextuelle:

```
alert.text
```

* *PS* `alert.dismiss()` *pourrait être utilisé pour confirmer les pop-ups déclenchés par* `JavaScript` `alert()` `JavaScript` `alert()` *ainsi que* `alert.confirm()`

Lire Gérer une alerte en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/6048/gerer-une-alerte

Chapitre 11: Grille de sélénium

Examples

Configuration du noeud

La configuration du nœud Selenium Grid réside sur le nœud lui-même et contient les informations sur la configuration du réseau et les capacités des nœuds. La configuration peut être appliquée de différentes manières:

- Configuration par défaut
- Configuration JSON
- Configuration en ligne de commande

Configuration JSON

La configuration du noeud dans le fichier JSON est divisée en 2 sections:

- Les capacités
- Configuration

Les fonctionnalités définissent des domaines tels que les types de navigateur et les versions pris en charge, l'emplacement des fichiers binaires du navigateur, le nombre d'instances maximales de chaque type de navigateur.

La configuration concerne les paramètres tels que les adresses de concentrateur et de noeud et les ports.

Voici un exemple de fichier de configuration 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
  }
}
```

Comment créer un noeud

Pour créer un nœud, vous devez d'abord disposer d'un concentrateur. Si vous n'avez pas de hub, vous pouvez le créer comme ceci:

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

Ensuite, vous pouvez créer un nœud:

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

Plus d'infos ici: https://github.com/SeleniumHQ/selenium/wiki/Grid2

Lire Grille de sélénium en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/1359/grille-de-selenium

Chapitre 12: Interaction avec l'élément Web

Examples

C

Effacer le contenu de l'élément (généralement la zone de texte)

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

Saisie de données dans l'élément (zone de texte générale)

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

Stocker la valeur de l'élément.

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

Stocker le texte de l'élément.

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

Cliquer sur un élément

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

Soumission d'un formulaire

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

Identifier la visibilité d'un élément sur la page

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

Identifier l'état d'un élément sur la page

```
bool isEnabled = interactionWebElement.Enabled;

bool isSelected=interactionWebElement.Selected;
```

Localisation d'un élément enfant d'interactionWebElement

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

Localisation des éléments enfants de l'interaction WebElement

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

Java

Effacement du contenu d'un élément Web: (remarque - lors de la simulation des actions utilisateur dans les tests, il est préférable d'envoyer un retour arrière, voir l'action suivante)

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

Saisie de données - simulation des séquences d'envoi:

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

Obtenir la valeur de l'attribut d'un élément:

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

Obtenir le texte de l'élément:

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

Sélection à partir du menu déroulant:

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

Auto explicatif:

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

Actions utilisant `org.openqa.selenium.interactions.Actions` :

Drag & Drop:

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

dragAndDrop.perform();
```

Sélectionnez multiple:

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

dragAndDrop.perform();
```

Auto explicatif (à l'aide du constructeur):

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

Voir ici pour plus d'exemples d'actions avancées et une liste complète.

En utilisant Javascript:

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

Lire Interaction avec l'élément Web en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/4280/interaction-avec-l-element-web

Chapitre 13: Interaction avec la ou les fenêtres du navigateur

Examples

Gestion de la fenêtre active

C

Maximiser la fenêtre

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

Ceci est assez simple, garantit que notre fenêtre active est maximisée.

Position de la fenêtre

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

Ici, nous déplaçons essentiellement la fenêtre actuellement active vers une nouvelle position. Dans l'objet `Point` , nous fournissons les `x` et `y` ; ceux-ci sont ensuite utilisés comme décalages du coin supérieur gauche de l'écran pour déterminer où la fenêtre doit être placée. Notez que vous pouvez également stocker la position de la fenêtre dans une variable:

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

Taille de la fenêtre

Définir et obtenir la taille de la fenêtre utilise la même syntaxe que la position:

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

URL de la fenêtre

Nous pouvons obtenir l'URL actuelle de la fenêtre active:

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

Nous pouvons également définir l'URL de la fenêtre active, ce qui fera naviguer le pilote vers la nouvelle valeur:

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

Poignées de fenêtre

Nous pouvons obtenir le handle pour la fenêtre en cours:

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

Et nous pouvons obtenir les poignées pour toutes les fenêtres ouvertes:

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

Python

Maximiser la fenêtre

```
driver.maximize_window()
```

Obtenir la position de la fenêtre

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

Définir la position de la fenêtre

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

Obtenir la taille de la fenêtre

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

Définir la taille de la fenêtre

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

Titre de la page en cours

```
driver.title
```

URL actuelle

```
driver.current_url
```

Poignées de fenêtre

```
driver.current_window_handle
```

Liste des fenêtres actuellement ouvertes

```
driver.window_handles
```

Fermer la fenêtre du navigateur en cours

Passez au nouvel onglet ouvert. Fermez les fenêtres actuelles (dans ce cas, le nouvel onglet). Revenez à la première fenêtre.

RAPPORTEUR:

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

JAVA Selenium:

```
        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));
```

Manipuler plusieurs fenêtres

Python

Scénario le plus couramment utilisé:

1. *ouvrir la page dans une nouvelle fenêtre*
2. *passer à elle*
3. *faire quelque chose*
4. *Ferme le*
5. *revenir à la fenêtre parente*

```
# 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'
```

Lire Interaction avec la ou les fenêtres du navigateur en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/5181/interaction-avec-la-ou-les-fenetres-du-navigateur

Chapitre 14: La navigation

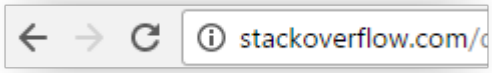
Syntaxe

- **C #**
- annuler retour ()
- annuler Forward ()
- annuler GotToUrl (URL de chaîne)
- void Refresh ()
- **Python**
- driver.back ()
- driver.forward ()
- driver.get ("URL")
- driver.refresh ()
- **Java**
- driver.navigate (). back ();
- driver.navigate (). forward ();
- driver.navigate (). to ("URL");
- driver.navigate (). refresh ();

Examples

Naviguer () [C #]

Il est possible de naviguer directement dans le navigateur, par exemple en utilisant les commandes standard de la barre d'outils disponibles sur tous les navigateurs:



Vous pouvez créer un objet de navigation en appelant `Navigate()` sur le pilote:

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

Un objet de navigation vous permet d'effectuer de nombreuses actions qui parcourent le navigateur sur le 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();
```

Naviguer () [Java]

Pour naviguer dans n'importe quelle URL:

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

Pour reculer:

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

Pour aller de l'avant:

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

Pour rafraîchir la page:

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

Méthodes de navigateur dans WebDriver

WebDriver, l'interface principale à utiliser pour les tests, qui représente un navigateur Web idéalisé. Les méthodes de cette classe se répartissent en trois catégories:

- Contrôle du navigateur lui-même
- Sélection des WebElements
- Aide au débogage

Les méthodes clés sont get (String), qui est utilisé pour charger une nouvelle page Web, et les différentes méthodes similaires à findElement (By), qui sont utilisées pour rechercher les WebElements. Dans cet article, nous allons apprendre les méthodes de contrôle du navigateur. obtenir

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

Chargez une nouvelle page Web dans la fenêtre du navigateur en cours. Ceci est fait en utilisant une opération HTTP GET, et la méthode bloquera jusqu'à ce que le chargement soit terminé. il est préférable d'attendre la fin de ce délai, car si la page sous-jacente change pendant que votre test s'exécute, les résultats des futurs appels sur cette interface seront appliqués à la page qui vient d'être chargée. **Usage**

```
//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()
```

Obtenez une chaîne représentant l'URL en cours que le navigateur recherche. Il renvoie l'URL de la page actuellement chargée dans le navigateur.

Usage

```
//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()
```

Il renvoie le titre de la page en cours, avec des espaces en début et en fin de page supprimés ou null si aucun n'est déjà défini.

Usage

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

getPageSource

java.lang.String getPageSource()
```

Récupère la source de la dernière page chargée. Si la page a été modifiée après le chargement (par exemple, par Javascript), rien ne garantit que le texte renvoyé est celui de la page modifiée.

Usage

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

Fermer

```
void close()
```

Fermez la fenêtre en cours, quittez le navigateur si c'est la dernière fenêtre actuellement ouverte. Si plusieurs fenêtres sont ouvertes avec cette instance de pilote, cette méthode ferme la fenêtre sur laquelle elle est active.

Usage

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

quitter

```
void quit()
```

Quitte ce pilote en fermant toutes les fenêtres associées. Après avoir appelé cette méthode, nous ne pouvons utiliser aucune autre méthode utilisant la même instance de pilote.

Usage

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

Ce sont toutes des méthodes très utiles disponibles dans Selenium 2.0 pour contrôler le navigateur si nécessaire.

Lire La navigation en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/7272/la-navigation

Chapitre 15: Les auditeurs

Examples

JUnit

Si vous utilisez JUnit pour exécuter, vous pouvez étendre la classe `TestWatcher` :

```
public class TestRules extends TestWatcher {

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

Donc, dans votre classe de test, vous pouvez simplement l'appeler:

```
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

Utiliser le EventFiringWebDriver . Vous pouvez y attacher WebDriverEventListener et remplacer les méthodes, à savoir la méthode onException:

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

Lire Les auditeurs en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/8226/les-auditeurs

Chapitre 16: Localisation d'éléments Web

Syntaxe

- ByChained (params By [] bys)

Remarques

Les objets se trouvent dans Selenium grâce à l'utilisation de *localisateurs* et de la classe `By` . Afin de réaliser un projet d'automatisation robuste avec Selenium, il convient d'utiliser intelligemment les localisateurs pour Web Elements. Les localisateurs doivent être **descriptifs, uniques et peu susceptibles de changer** . Par exemple, vous n'obtiendrez pas de faux positifs dans les tests. La priorité est d'utiliser:

1. **ID** - puisqu'il est unique et que vous obtenez exactement l'élément souhaité.
2. **Nom de classe** - Il est descriptif et peut être unique dans un contexte donné.
3. **CSS** (meilleures performances que xpath) - Pour les sélecteurs plus compliqués.
4. **XPATH** - Où CSS ne peut pas être utilisé (axe XPATH), par exemple `div::parent` .

Les autres localisateurs sont sujets à des changements ou à un rendu, et sont de préférence évités.

Règle de base: si votre code ne peut pas localiser un élément particulier, l'une des raisons pourrait être que votre code n'a pas attendu tous les éléments DOM à télécharger. Envisagez de dire à votre programme d'attendre pendant une courte période (essayez 3 à 5 secondes, puis augmentez lentement si nécessaire) avant de rechercher cet élément. Voici un exemple en Python, tiré de cette question :

```
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

Localisation des éléments de page à l'aide de WebDriver

Pour interagir avec WebElements dans une page Web, nous devons d'abord identifier

l'emplacement de l'élément.

Par est le mot-clé disponible en sélénium.

Vous pouvez localiser les éléments par ..

1. **Par** *identifiant*
2. **Par** *nom de classe*
3. **Par** *tagName*
4. **Par** *nom*
5. **Par** *lien texte*
6. **Par** *texte de lien partiel*
7. **Par** *sélecteur CSS*
8. **Par** *XPath*
9. **Utiliser** *JavaScript*

Considérons l'exemple de script ci-dessous

```
<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" />
```

Dans le code ci-dessus, le nom d'utilisateur et le mot de passe sont définis à l'aide des identifiants. Maintenant, vous allez identifier les éléments avec id.

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

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

Comme le sélénium supporte 7 langues différentes, ce document vous donne une idée pour localiser les éléments dans toutes les langues.

Par identifiant

Exemple de recherche d'un élément à l'aide de 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"));
```

Par nom de classe

Exemple de recherche d'un élément en utilisant le nom de 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"));
```

Par nom de tag

Exemple de recherche d'un élément en utilisant le nom de 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"));
```

De nom

Exemple de recherche d'un élément en utilisant name:

```
<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"));
```

Par lien texte

Exemple de recherche d'un élément à l'aide du texte du lien:

```
<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"));
```

Par texte de lien partiel

Exemple de recherche d'un élément à l'aide d'un texte de lien partiel:

```
<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"));
```

Par sélecteurs CSS

Exemple de recherche d'un élément à l'aide des sélecteurs 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"));
```

Voici un article sur la création de sélecteurs CSS:
http://www.w3schools.com/cssref/css_selectors.asp

Par XPath

Exemple de recherche d'un élément à l'aide de 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"));
```

Voici un article sur XPath: http://www.w3schools.com/xsl/xpath_intro.asp

---

Utiliser JavaScript

Vous pouvez exécuter un javascript arbitraire pour trouver un élément et tant que vous retournez un élément DOM, il sera automatiquement converti en objet WebElement.

Exemple simple sur une page chargée en 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 -
```

Remarque: cette méthode ne fonctionnera pas si votre WebDriver particulier ne prend pas en charge JavaScript, tel que SimpleBrowser .

Sélection par plusieurs critères [C #]

Il est également possible d'utiliser des sélecteurs ensemble. Cela se fait à l'aide de l'objet `OpenQA.Selenium.Support.PageObjects.ByChained` :

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

Un nombre quelconque de `By` s peuvent être chaînés et sont utilisés comme sélection de type ET (tous les `By` s sont mis en correspondance)

Sélection d'éléments avant que la page ne se charge

Lorsque vous appelez `driver.Navigate().GoToUrl(url);` , l'exécution du code s'arrête jusqu'à ce que la page soit complètement chargée. Cela est parfois inutile lorsque vous souhaitez simplement extraire des données.

Remarque: Les exemples de code ci-dessous peuvent être considérés comme des hacks. Il n'y a pas de moyen "officiel" de le faire.

Créer un nouveau fil de discussion

Créez et lancez un thread pour charger une page Web, puis utilisez Wait .

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")));
}
```

Utiliser les délais d'attente

En utilisant un WebDriverTimeout, vous pouvez charger une page et, après un certain temps, elle lancera une exception, ce qui arrêtera le chargement de la page. Dans le bloc catch, vous pouvez utiliser 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")));
    }
}
```

Le problème : Lorsque vous définissez un délai trop court, la page cessera de se charger, qu'il y ait ou non l'élément souhaité. Lorsque vous définissez le délai d'attente trop longtemps, vous allez annuler l'avantage de performance.

Lire Localisation d'éléments Web en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/3991/localisation-d-elements-web

Chapitre 17: Modèle d'objet de page

Introduction

Un rôle important dans l'automatisation des sites Web et des applications Web consiste à identifier les éléments à l'écran et à interagir avec eux. Les objets se trouvent dans Selenium grâce à l'utilisation de localisateurs et de la classe `By` . Ces localisateurs et interactions sont placés dans les objets de la page pour éviter le code en double et faciliter la maintenance. Il encapsule les `WebElements` et suppose de contenir le comportement et de renvoyer des informations sur la page (ou une partie d'une page dans une application Web).

Remarques

Le modèle d'objet de page est un modèle où l'on écrit des classes orientées objet qui servent d'interface à une vue particulière de la page Web. Nous utilisons les méthodes de cette classe de page pour effectuer l'action requise. Il y a quelques années, nous manipulions directement le code HTML de la page Web dans des classes de test, ce qui était très difficile à maintenir et risquait de provoquer des modifications de l'interface utilisateur.

Cependant, le fait d'avoir votre code organisé de la même manière que le modèle d'objets de page fournit une API spécifique à l'application, vous permettant de manipuler les éléments de la page sans contourner le HTML. Le principe de base de la page dit: votre objet page doit avoir tout ce qu'un humain peut faire sur cette page Web. Par exemple, pour accéder au champ de texte d'une page Web, vous devez utiliser une méthode pour obtenir le texte et retourner la chaîne après avoir effectué toutes les modifications.

Quelques points importants à retenir lors de la conception des objets de la page:

1. L'objet de la page ne devrait généralement pas être construit uniquement pour les pages, mais vous devriez plutôt le construire pour des éléments significatifs de la page. Par exemple, une page comportant plusieurs onglets pour afficher différents graphiques de vos universitaires doit avoir le même nombre de pages que le nombre d'onglets.

2. La navigation d'une vue à l'autre doit renvoyer l'instance des classes de page.

3. Les méthodes utilitaires requises uniquement pour une vue ou une page Web spécifique doivent appartenir uniquement à cette classe de page.

4. Les méthodes d'assertion ne doivent pas être prises en compte par les classes de page, vous pouvez avoir des méthodes pour renvoyer booléen sans les vérifier. Par exemple, pour vérifier le nom complet de l'utilisateur, vous pouvez avoir une méthode pour obtenir une valeur booléenne:

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

5. Si votre page Web est basée sur iframe, préférez également avoir des classes de page pour les iframes.

Avantages du modèle d'objet de page:

1. Séparation propre entre le code de test et le code de page
2. En cas de modification de l'interface utilisateur de la page Web, il n'est pas nécessaire de modifier votre code à plusieurs endroits. Changer uniquement dans les classes de page.
3. Pas de localisateur d'éléments dispersés.
4. Facilite la compréhension du code
5. Entretien facile

Examples

Introduction (Utilisation de Java)

Un exemple pour effectuer un test de connexion basé sur le modèle d'objet de la page:

```
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

Les objets de page doivent contenir le comportement, renvoyer des informations pour les assertions et éventuellement une méthode pour la méthode d'état de page prête lors de l'initialisation. Selenium prend en charge les objets de page à l'aide d'annotations. En C # c'est comme suit:

```
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;
    }
}
```

Remarques:

- `CacheLookup` enregistre l'élément dans le cache et enregistre le retour d'un nouvel élément à chaque appel. Cela améliore les performances mais ne permet pas de modifier dynamiquement les éléments.
- `searchButton` a 2 noms de classes et pas d'ID, c'est pourquoi je ne peux pas utiliser le nom de la classe ou l'id.
- J'ai vérifié que mes localisateurs me renverraient l'élément que je souhaite utiliser avec Developer Tools (pour Chrome). Dans d'autres navigateurs, vous pouvez utiliser FireBug ou similaire.
- `Search()` méthode `Search()` renvoie un autre objet de page (`ResultsPage`) car le clic de recherche vous redirige vers une autre page.

Modèle d'objet de page Meilleures pratiques

- Créez des fichiers séparés pour l'en-tête et le pied de page (car ils sont communs à toutes les pages et cela n'a aucun sens de les intégrer à une seule page)
- Conserve les éléments communs (comme Search / Back / Next, etc.) dans un fichier séparé (L'idée est de supprimer tout type de duplication et de garder la ségrégation logique)
- Pour Driver, c'est une bonne idée de créer une classe de pilote distincte et de garder le pilote statique afin de pouvoir l’accéder à toutes les pages! (J'ai toutes mes pages Web pour étendre DriverClass)
- Les fonctions utilisées dans PageObjects sont décomposées en morceaux les plus petits possibles en tenant compte de la fréquence et de la manière dont elles seront appelées (la manière dont vous vous êtes connecté - bien que la connexion puisse être divisée en fonctions enterUsername et enterPassword comme la fonction de connexion est plus logique car dans la majorité des cas, la fonction de connexion est appelée plutôt que des appels séparés aux fonctions enterUsername et enterPassword)
- L'utilisation de PageObjects lui-même sépare le script de test de elementLocators

- Avoir des fonctions utilitaires dans un dossier utils séparé (comme DateUtil, excelUtils, etc.)
- Avoir des configurations dans un dossier conf distinct (comme définir l'environnement sur lequel les tests doivent être exécutés, configurer les dossiers de sortie et d'entrée)
- Incorporer screenCapture en cas d'échec
- Avoir une variable d'attente statique dans DriverClass avec un temps d'attente implicite comme vous l'avez fait Essayez toujours d'avoir des attentes conditionnelles plutôt que des attentes statiques comme: wait.until (ExpectedConditions). Cela garantit que l'attente ne ralentit pas inutilement l'exécution.

Lire Modèle d'objet de page en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/4853/modele-d-objet-de-page

Chapitre 18: Navigateurs sans tête

Examples

PhantomJS [C #]

`PhantomJS` est un navigateur Web sans tête **avec** prise en charge de JavaScript.

Avant de commencer, vous devrez télécharger un pilote PhantomJS et assurez-vous de le placer au début de votre code:

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

Super, maintenant sur l'initialisation:

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

Cela créera simplement une nouvelle instance de la classe PhantomJSDriver. Vous pouvez ensuite l'utiliser de la même manière que chaque WebDriver, par exemple:

```
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);
    }
}
```

Cela fonctionne bien. Cependant, le problème que vous avez probablement rencontré est que, lorsque vous utilisez l'interface utilisateur, `PhantomJS` ouvre une nouvelle fenêtre de console, ce qui n'est pas vraiment souhaité dans la plupart des cas. Heureusement, nous pouvons masquer la fenêtre et même améliorer légèrement les performances en utilisant `PhantomJSOptions` et `PhantomJSDriverService` . Exemple de travail complet ci-dessous:

```
// 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);
    }
}
```

Astuce Pro: cliquez sur une classe (par exemple, le `PhantomJSDriverService`) et appuyez sur F12 pour voir exactement ce qu'elles contiennent ainsi qu'une brève description de ce qu'elles font.

SimpleBrowser [C #]

`SimpleBrowser` est un WebDriver léger **sans** support JavaScript.

Il est considérablement plus rapide qu'un `PhantomJS` mentionné ci-dessus, mais en ce qui concerne les fonctionnalités, il se limite à des tâches simples sans fonctionnalités sophistiquées.

Tout d'abord, vous devrez télécharger le package SimpleBrowser.WebDriver , puis mettre ce code au début:

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

Maintenant, voici un court exemple d'utilisation:

```
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);
    }
}
```

Navigateur sans tête en Java

HTMLUnitDriver

HTMLUnitDriver est l'implémentation la plus légère d'un navigateur sans interface graphique (sans interface graphique) pour Webdriver basé sur HtmlUnit. Il modélise les documents HTML et fournit

une API qui vous permet d'appeler des pages, de remplir des formulaires, de cliquer sur des liens, etc., comme vous le faites dans votre navigateur habituel. Il prend en charge JavaScript et fonctionne avec les bibliothèques AJAX. Il est utilisé pour tester et récupérer des données à partir du site Web.

Exemple: Utilisation de HTMLUnitDriver pour récupérer une liste de questions sur `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();
                }
    }
```

Il est identique à tout autre navigateur (Mozilla Firefox, Google Chrome, IE), mais il n’a pas d’interface graphique, l’exécution n’est pas visible à l’écran.

Lire Navigateurs sans tête en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/3931/navigateurs-sans-tete

Chapitre 19: Naviguer entre plusieurs images

Introduction

Dans les pages Web qui contiennent un nombre d'images, le sélénium considère que la fenêtre Image est une fenêtre distincte. Plusieurs fois, nous avons besoin d'une structure Web où nous avons un cadre avec cadre pour naviguer dans les fenêtres de cadre. Selenium fournit la méthode swithTo ().

Examples

Exemple de cadre

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

    <iframe "id="iframe_Login2">

        <iframe "id="iframe_Login3">

        </iframe>

    </iframe>

</iframe>
```

Pour basculer dans le cadre en sélénium, utilisez la méthode swithTo () et frame ().

driver.switchTo (). frame (iframe_Login1); driver.switchTo (). frame (iframe_Login2); driver.switchTo (). frame (iframe_Login3);

Pour revenir en arrière, nous pouvons utiliser parentFrame () et defaultContest ();

parentFrame (): change le focus sur le contexte parent. Si le contexte actuel est le contexte de navigation de niveau supérieur, le contexte reste inchangé.

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

defaultContent (): Sélectionne la première image de la page ou le document principal lorsqu'une page contient des iframes.

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

Lire Naviguer entre plusieurs images en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/9803/naviguer-entre-plusieurs-images

Chapitre 20: Prendre des captures d'écran

Introduction

Prendre des captures d'écran et enregistrer un chemin particulier

Syntaxe

- Fichier src = (pilote (TakesScreenshot)) .getScreenshotAs (OutputType.FILE);
- FileUtils.copyFile (src, nouveau fichier ("D: \ screenshot.png"));

Examples

JAVA

Code à prendre et enregistrer la capture d'écran:

```
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();
        }
    }
}
```

Capture d'écran:

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

Capture d'écran de la source à la destination:

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

Lire Prendre des captures d'écran en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/10094/prendre-des-captures-d-ecran

Chapitre 21: Programme de base Selenium Webdriver

Introduction

Ce sujet a pour but de montrer le programme de base du pilote Web dans les langages supportés par sélénium comme C #, Groovy, Java, Perl, PHP, Python et Ruby.

Journey inclut l'ouverture d'un pilote de navigateur -> Google Page -> arrêter le navigateur

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");
            }
        }
    }
}
```

Le programme ci-dessus permet d'accéder à la page d'accueil de Google, puis de fermer le navigateur après le chargement complet de la page.

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

Cela instancie un nouvel objet WebDriver à l'aide de l'interface `IWebdriver` et crée une nouvelle instance de fenêtre de navigateur. Dans cet exemple, nous utilisons `ChromeDriver` (bien que cela puisse être remplacé par le pilote approprié pour le navigateur que nous souhaitons utiliser). Nous enveloppons cela avec une instruction `using` , car `IWebDriver` implémente `IDisposable` , ne nécessitant donc pas de taper explicitement `driver.Quit();` .

Si vous n'avez pas téléchargé votre WebDriver à l'aide de NuGet , vous devrez transmettre un argument sous la forme d'un chemin vers le répertoire où se trouve le pilote "chromedriver.exe".

Naviguer

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

et

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

Ces deux lignes font la même chose. Ils indiquent au pilote de naviguer vers une URL spécifique et d'attendre que la page soit chargée avant de passer à l'instruction suivante.

Il existe d'autres méthodes liées à la navigation, telles que `Back()` , `Forward()` ou `Refresh()` .

Après cela, le bloc `using` ferme en toute sécurité et dispose de l'objet.

Python

```
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()
```

Le «programme» ci-dessus permet d'accéder à la page d'accueil de Google, puis de fermer le navigateur avant de terminer.

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

Nous avons d'abord notre fonction principale, notre point d'entrée dans le programme, qui appelle `get_google()` .

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

`get_google()` commence alors par créer notre instance de `driver` via `set_up_driver()` :

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

Nous `chromedriver.exe` où se trouve `chromedriver.exe` et instancions notre objet pilote avec ce chemin. Le reste de `get_google()` navigue vers Google:

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

Et appelle ensuite `tear_down()` passant l'objet du pilote:

```
tear_down(driver)
```

`tear_down()` contient simplement une ligne pour arrêter notre objet pilote:

```
driver.quit()
```

Cela indique au pilote de fermer toutes les fenêtres de navigateur ouvertes et de disposer de l'objet du navigateur, car nous n'avons pas d'autre code après cet appel, ce qui met effectivement fin au programme.

Java

Le code ci-dessous est composé d'environ 3 étapes.

1. Ouvrir un navigateur chrome
2. Ouverture de la page google
3. Arrêter le navigateur

```
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();
    }
}
```

Le «programme» ci-dessus permet d'accéder à la page d'accueil de Google, puis de fermer le navigateur avant de terminer.

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

La première ligne indique au système où trouver l' `ChromeDriver` (chromedriver.exe). Nous créons ensuite notre objet pilote en appelant le constructeur `ChromeDriver()` encore, nous pourrions appeler notre constructeur ici pour tout navigateur / plate-forme.

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

Cela indique à notre pilote de naviguer jusqu'à l'URL spécifiée: http://www.google.com . L'API Java WebDriver fournit la méthode `get()` directement sur l'interface WebDriver, bien que d'autres méthodes de navigation puissent être trouvées via la méthode `navigate()` , par exemple `driver.navigate.back()` .

Une fois le chargement de la page terminé, nous appelons immédiatement:

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

Cela dit au pilote de fermer toutes les fenêtres de navigateur ouvertes et de se débarrasser de l'objet pilote, car nous n'avons pas d'autre code après cet appel, ce qui met effectivement fin au programme.

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

Est une instruction (non montrée ici) pour le pilote de fermer uniquement la fenêtre active, dans ce cas, comme nous n'avons qu'une seule fenêtre, les instructions provoqueraient des résultats identiques à l'appel à `quit()` .

Java - Meilleures pratiques avec les classes de page

Usecase: Connexion au compte 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());
    }

}
```

Classes de pages: Login Page & Page de connexion à la page d'accueil:

```
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 de page d'accueil:

```
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";
        }
    }

}
```

Lire Programme de base Selenium Webdriver en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/3990/programme-de-base-selenium-webdriver

Chapitre 22: Rapports HTML

Introduction

Cette rubrique traite de la création de rapports HTML pour les tests de sélénium. Il existe différents types de plug-ins disponibles pour les rapports et les plus utilisés sont Allure, ExtentReports et ReportNG.

Examples

ExtentReports

Cet exemple couvre l'implémentation d'ExtentReports dans Selenium à l'aide de TestNG, Java et Maven.

ExtentReports est disponible en deux versions, communautaire et commerciale. Pour la facilité et la démonstration, nous utiliserons la version communautaire.

1. dépendance

Ajoutez la dépendance dans votre fichier Maven pom.xml pour les rapports d'étendue.

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

2. Configurez les plugins

Configurez le plugin maven surefire comme ci-dessous dans 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. Exemple de test avec ExtentReports

Maintenant, créez un test avec le nom 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

Cet exemple concerne l'implémentation d'Allure Reports dans Selenium à l'aide de TestNG, Java et Maven.

Configuration Maven

Dépôt

Ajouter le code suivant pour configurer le référentiel jcenter

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

Dépendance

Ajouter les dépendances suivantes à votre 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>
```

Configuration du 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>
```

Exemple de test pour le rapport Allure

Créer un exemple de test avec le nom 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();
    }
}
```

Dans la classe ci-dessus, nous avons utilisé la classe LogUtiil. Ceci est simplement fait pour enregistrer les **étapes** de notre test. Voici le code pour le même

LogUtil.java

```
public final class LogUtil {

    private LogUtil() {
    }
```

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

Ici

@Titre ("") ajoutera le titre à votre test dans Allure Report

@Description ("") ajoutera la description à votre test

@Step ("") ajoutera une étape dans le rapport d'allure pour le test

A l'exécution, un fichier xml sera généré dans le dossier "target / allure-results /"

Rapport final avec Jenkins

Si vous utilisez Jenkins avec le plug-in Allure Report, Jenkins affichera automatiquement le rapport dans votre travail.

Rapport final sans Jenkins

Pour ceux qui n'ont pas de Jenkins, utilisez la ligne de commande suivante pour créer le rapport HTML. Allure CLI est une application Java disponible pour toutes les plates-formes. Vous devez installer manuellement Java 1.7+ avant d'utiliser Allure CLI.

Debian

Pour les référentiels basés sur Debian, nous fournissons un PPA afin que l'installation soit simple: Installez Allure CLI pour debian

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

Les distributions prises en charge sont les suivantes: Trusty et Precise. Après l'installation, vous aurez la commande allure disponible.

Mac OS

Vous pouvez installer la CLI Allure via Homebrew.

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

Après l'installation, vous aurez la commande allure disponible.

Windows et autres Unix

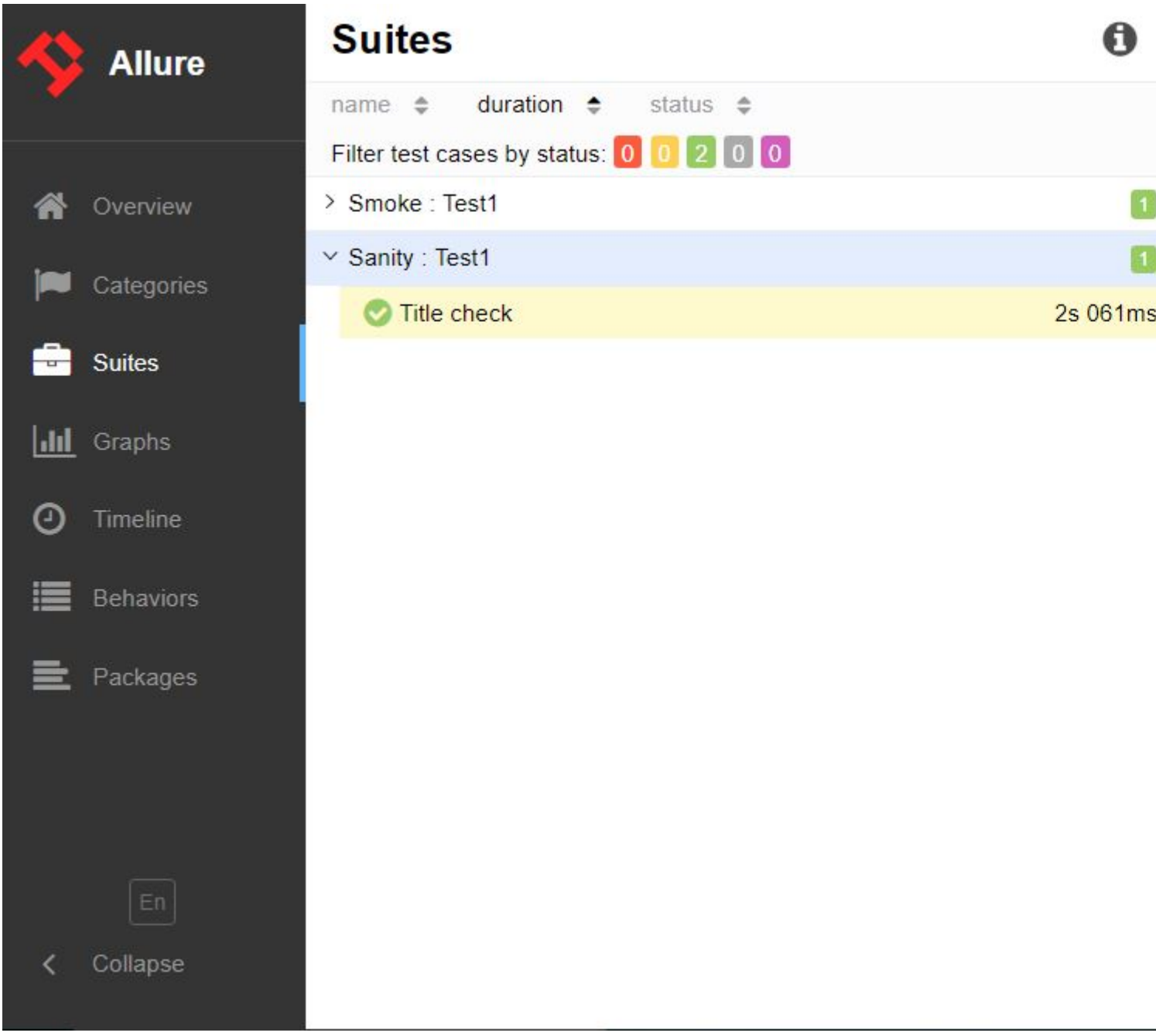
1. Téléchargez la dernière version sous forme d'archive zip à l' adresse

https://github.com/allure-framework/allure-core/releases/latest .

2. Décompressez l'archive dans le répertoire allure-commandline. Accédez au répertoire bin.
3. Utilisez allure.bat pour Windows et allure pour d'autres plates-formes Unix.

Dans la ligne de commande / terminal, entrez simplement la syntaxe suivante et le rapport sera généré dans le dossier allure-report

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



Lire Rapports HTML en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/10721/rapports-html

Chapitre 23: Réglage / Obtention de la taille de la fenêtre du navigateur

Introduction

Définir ou obtenir la taille de la fenêtre de n'importe quel navigateur lors de l'automatisation

Syntaxe

- driver.manage (). window (). maxim ();
- driver.manage (). window (). setSize (*DimensionObject*);
- driver.manage (). window (). getSize ()

Examples

JAVA

Définir à la taille maximale de la fenêtre du navigateur:

```
//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();
```

Définir la taille de la fenêtre spécifique:

```
//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);
```

Obtenir la taille de la fenêtre du navigateur:

```
//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());
```

Lire Réglage / Obtention de la taille de la fenêtre du navigateur en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/10093/reglage---obtention-de-la-taille-de-la-fenetre-du-navigateur

Chapitre 24: Robot en sélénium

Syntaxe

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

Paramètres

Paramètre	Détails
Mme	Il est temps de dormir en millisecondes
code clé	Constante pour appuyer sur la touche spécifiée par exemple pour appuyer sur `A` code est `VK_A` . Veuillez vous référer pour plus de détails: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html
x, y	Coordonnées de l'écran
boutons	Le masque de bouton; une combinaison d'un ou plusieurs masques de bouton de souris
WheelAmt	Nombre d'encoches pour déplacer la molette de la souris, valeur négative pour monter / descendre de la valeur positive de l'utilisateur pour descendre / vers l'utilisateur

Remarques

Cette section contient des détails sur l'implémentation de l'API Robot avec Selenium Webdriver. La classe Robot est utilisée pour générer une entrée système native lorsque le sélénium n'est pas capable de le faire, par exemple en appuyant sur la touche droite de la souris, en appuyant sur la touche F1, etc.

Examples

Evénement Keypress utilisant l'API Robot (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();
        }
    }
}
```

Avec sélénium

Parfois, nous devons appuyer sur une touche pour tester l'événement de presse clé sur une application Web. Pour qu'une instance teste la touche ENTRÉE sur le formulaire de connexion, nous pouvons écrire quelque chose comme ci-dessous avec 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);
    }
}
```

Evénement Souris utilisant l'API Robot (JAVA)

Mouvement de la souris:

```
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);
     }
}
```

Appuyez sur le bouton gauche / droit de la souris:

```
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);
     }
}
```

Cliquez et faites défiler la roue:

```
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);
     }
}
```

Lire Robot en sélénium en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/4877/robot-en-selenium

Chapitre 25: Sélectionner une classe

Syntaxe

- **Java**
- tout déselectionner()
- deselectByIndex (int index)
- deselectByValue (valeur java.lang.String)
- deselectByVisibleText (texte java.lang.String)
- getAllSelectedOptions ()
- getFirstSelectedOption ()
- getOptions ()
- isMultiple ()
- selectByIndex (int index)
- selectByValue (valeur java.lang.String)
- selectByVisibleText (texte java.lang.String)

Paramètres

Paramètres	Détails
indice	L'option à cet index sera sélectionnée
valeur	La valeur à comparer
texte	Le texte visible à assortir

Remarques

`Select` class of Selenium WebDriver fournit des méthodes utiles pour interagir avec `select` options. L'utilisateur peut effectuer des opérations sur une liste déroulante de sélection et également sélectionner une opération en utilisant les méthodes ci-dessous.

En **C #** la classe Select est en fait `SelectElement`

Examples

Différentes façons de sélectionner à partir de la liste DropDown

Ci-dessous la page 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>
```

JAVA

Sélectionner par index

Pour sélectionner l'option par index en utilisant 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
    }
}
```

Sélectionner par valeur

```
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
    }
}
```

Sélectionner par texte de visibilité

```
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

Tous les exemples ci-dessous sont basés sur l'interface générique `IWebDriver`

Sélectionner par index

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

Sélectionner par valeur

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

Sélectionner par texte

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

Lire Sélectionner une classe en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/6426/selectionner-une-classe

Chapitre 26: Selenium-webdriver avec Python, Ruby et Javascript avec l'outil CI

Introduction

Ceci est une façon d'exécuter des tests de sélénium avec CircleCI

Examples

Intégration CircleCI avec Selenium Python et 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()
```

exemple_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")
```

Lire Selenium-webdriver avec Python, Ruby et Javascript avec l'outil CI en ligne:

https://riptutorial.com/fr/selenium-webdriver/topic/10091/selenium-webdriver-avec-python--ruby-et-javascript-avec-l-outil-ci

Chapitre 27: Traitement des erreurs dans l'automatisation à l'aide de Selenium

Examples

Python

`WebDriverException` est une exception `Selenium-WebDriver` base pouvant être utilisée pour intercepter toutes les autres exceptions `Selenium-WebDriver`

Pour pouvoir intercepter des exceptions, il faut d'abord les importer:

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

et alors:

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

De la même manière, vous pouvez importer d'autres exceptions plus spécifiques:

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

Si vous souhaitez extraire le message d'exception uniquement:

```
from selenium.common.exceptions import UnexpectedAlertPresentException

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

Lire Traitement des erreurs dans l'automatisation à l'aide de Selenium en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/9548/traitement-des-erreurs-dans-l-automatisation-a-l-aide-de-selenium

Chapitre 28: Utilisation de Selenium Webdriver avec Java

Introduction

Selenium webdriver est un framework d'automatisation Web qui vous permet de tester votre application Web sur différents navigateurs Web. Contrairement à Selenium IDE, webdriver vous permet de développer vos propres scénarios de test dans le langage de programmation de votre choix. Il supporte Java, .Net, PHP, Python, Perl, Ruby.

Examples

Ouverture de la fenêtre du navigateur avec une URL spécifique à l'aide de Selenium Webdriver en 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();
    }
}
```

Ouvrir une fenêtre de navigateur avec la méthode to ()

Ouvrir un navigateur avec la méthode 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();
    }
}
```

Lire Utilisation de Selenium Webdriver avec Java en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/9158/utilisation-de-selenium-webdriver-avec-java

Chapitre 29: Utiliser les annotations @FindBy en Java

Syntaxe

- CLASS_NAME: @FindBy (className = "classname")
- CSS: @FindBy (css = "css")
- ID: @FindBy (id = "id")
- ID_OR_NAME: @FindBy (comment = How.ID_OR_NAME, en utilisant = "idOrName")
- LINK_TEXT: @FindBy (linkText = "text")
- NOM: @FindBy (name = "name")
- PARTIAL_LINK_TEXT: @FindBy (partialLinkText = "text")
- TAG_NAME: @FindBy (tagName = "tagname")
- XPATH: @FindBy (xpath = "xpath")

Remarques

Notez qu'il existe deux manières d'utiliser l'annotation. Exemples:

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

et

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

sont égaux et recherchent tous les deux leur identifiant. Dans le cas de `ID_OR_NAME` vous ne pouvez utiliser que

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

`PageFactory.initElements()` doit être utilisé après l'instanciation de l'objet page pour rechercher des éléments marqués avec `@FindBy` annotation `@FindBy` .

Examples

Exemple de base

Supposons que nous utilisons un modèle d'objet de page . Classe d'objet page:

```
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
    }
}
```

Et classe de tests:

```
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");
    }
}
```

Il existe peu de méthodes pour trouver les WebElements à l'aide de la section @FindBy - check Syntax.

Lire Utiliser les annotations @FindBy en Java en ligne: https://riptutorial.com/fr/selenium-webdriver/topic/6777/utiliser-les-annotations--findby-en-java

Crédits

S. No	Chapitres	Contributeurs
1	Démarrer avec selenium-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	Actions (émulation de gestes complexes)	Josh, Kenil Fadia, Liam, Priyanshu Shekhar, Tom Mc
3	Attendez	Jakub Lokša, Josh, Kenil Fadia, Liam, Moshisho, noor, Sajal Singh, Saurav
4	Changement de cadre	Andersson, dreamwork801, Jakub Lokša, Java_deep, Jim Ashworth, Karthik Taduvai, Kyle Fairns, Liam, lloyd, Priyanshu Shekhar, SlightlyKosumi
5	Configuration de la grille de sélénium	mnoronha, Prasanna Selvaraj, selva, Thomas
6	Configuration du sélénium e2e	Ashish Deshmukh
7	Défilement	Andersson, Sagar007
8	Exceptions dans Selenium-WebDriver	Brydenr
9	Exécution de Javascript dans la page	Brydenr, Liam
10	Gérer une alerte	Andersson, Aurasphere, Priya, SlightlyKosumi
11	Grille de sélénium	Eugene S, Y-B Cause
12	Interaction avec l'élément Web	Jakub Lokša, Liam, Moshisho, Siva, Sudha Velan
13	Interaction avec la ou les fenêtres du navigateur	Andersson, Josh, Sakshi Singla
14	La navigation	Andersson, Liam, Santoshsarma, viralpatel

15	Les auditeurs	Erki M.
16	Localisation d'éléments Web	alecxe, daOnlyBG, Jakub Lokša, Josh, Liam, Łukasz Piaszczyk, Moshisho, NarendraR, noor, Priya, Sakshi Singla, Siva
17	Modèle d'objet de page	JeffC, Josh, Moshisho, Priyanshu Shekhar, Sakshi Singla
18	Navigateurs sans tête	Abhilash Gupta, Jakub Lokša, Liam, r_D, Tomislav Nakic-Alfirevic
19	Naviguer entre plusieurs images	Pavan T, Raghvendra
20	Prendre des captures d'écran	Abhilash Gupta, Sanchit
21	Programme de base Selenium Webdriver	Jakub Lokša, Josh, Liam, Priya, Sudha Velan, Thomas, vikingben
22	Rapports HTML	Ashish Deshmukh
23	Réglage / Obtention de la taille de la fenêtre du navigateur	Abhilash Gupta
24	Robot en sélénium	Priyanshu Shekhar
25	Sélectionner une classe	Gaurav Lad, Liam
26	Selenium-webdriver avec Python, Ruby et Javascript avec l'outil CI	Brydenr
27	Traitement des erreurs dans l'automatisation à l'aide de Selenium	Andersson
28	Utilisation de Selenium Webdriver avec Java	r_D, the_coder
29	Utiliser les annotations @FindBy en Java	Alex Wittig, Łukasz Piaszczyk