Бесплатная электронная книга

УЧУСЬ selenium-webdriver

Free unaffiliated eBook created from
Stack Overflow contributors.

#selenium-webdriver

Около

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

глава 1: Начало работы с selenium-webdriver

замечания

В этом разделе представлен обзор того, что такое selenium-webdriver, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в селен-webdriver и ссылаться на связанные темы. Поскольку документация для selenium-webdriver является новой, вам может потребоваться создать начальные версии этих связанных тем.

Версии

Версия	Дата выхода
0.0.1	2016-08-03

Examples

Установка или настройка

Чтобы начать использовать WebDriver, вам нужно будет получить соответствующий драйвер с сайта Selenium : Selenium HQ Downloads . Отсюда вам нужно загрузить драйвер, относящийся к браузерам и / или платформам, на которых вы пытаетесь запустить WebDriver, например, если вы тестировали в Chrome, сайт Selenium направит вас на:

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

Чтобы загрузить `chromedriver.exe` .

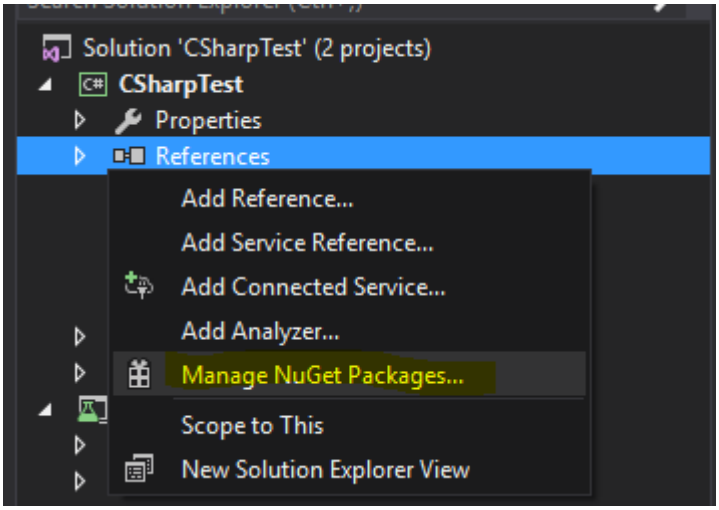
Наконец, прежде чем использовать WebDriver, вам нужно будет загрузить соответствующие языковые привязки, например, если вы используете C #, вы можете получить доступ к загрузке со страницы загрузки Selenium HQ для получения необходимых файлов DLL или, в качестве альтернативы, загрузить их в виде пакетов в Visual Studio через менеджер пакетов NuGet.

Необходимые файлы теперь должны быть загружены, для получения информации о том, как начать использовать WebDriver, обратитесь к другой документации `selenium-webdriver` .

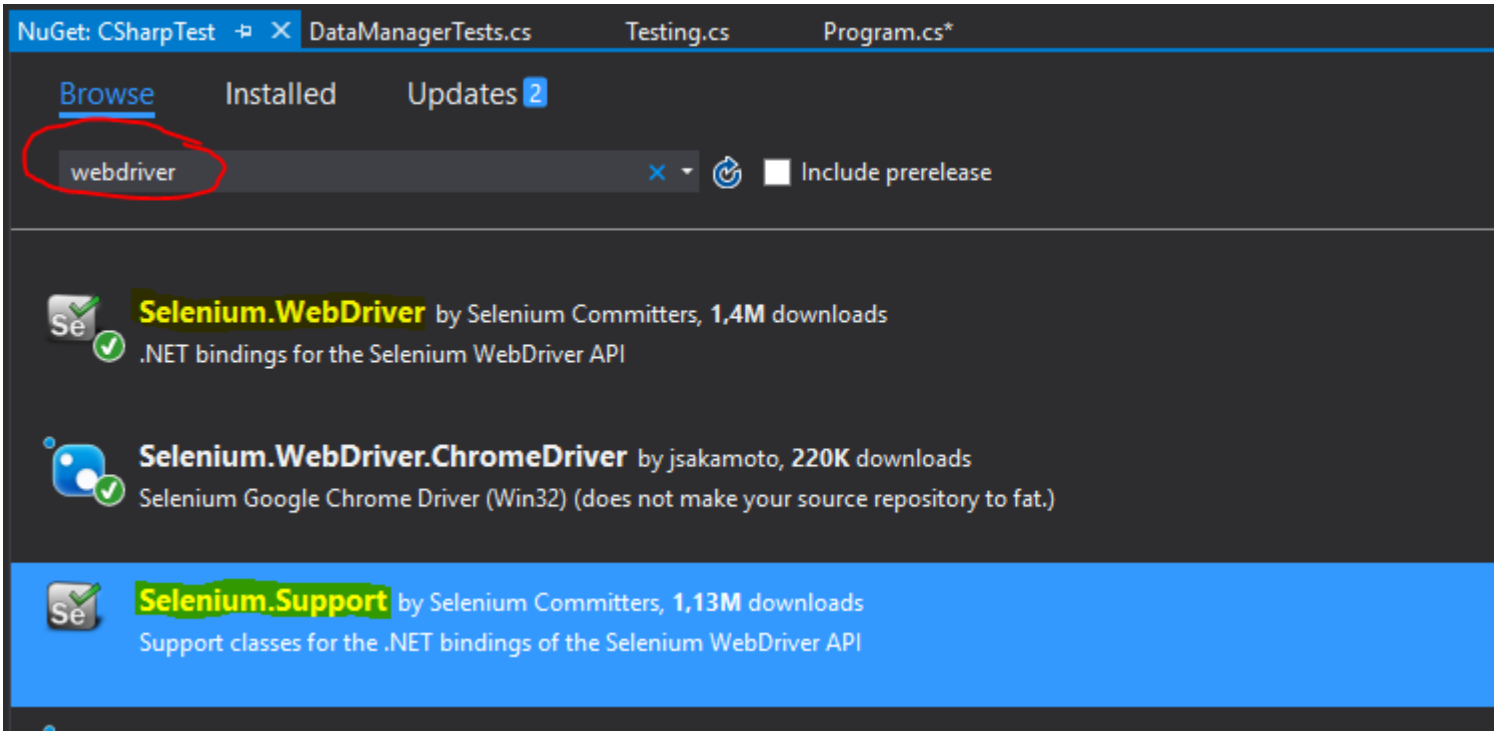
Для Visual Studio [NuGet]

Самый простой способ установки Selenium WebDriver - использовать диспетчер пакетов NuGet.

В своем проекте щелкните правой кнопкой мыши «Ссылки» и нажмите «Управление пакетами NuGet», как показано:



Затем введите в поле поиска « *webdriver* ». Затем вы должны увидеть что-то вроде этого:



Установите « **Selenium.WebDriver** » и « **Selenium.Support** » (пакет поддержки включает дополнительные ресурсы, такие как Wait), нажав кнопку «Установить» с правой стороны.

Затем вы можете установить свои WebDrivers, которые хотите использовать, например, один из них:

- Selenium.WebDriver.ChromeDriver (Google Chrome)
- PhantomJS (безголовый)
-

Что такое Selenium WebDriver?

Selenium - это набор инструментов, предназначенных для автоматизации браузеров. Он обычно используется для тестирования веб-приложений на нескольких платформах. Есть несколько инструментов, доступных под зонтиком Селена, таких как Selenium WebDriver (ex-Selenium RC), Selenium IDE и Selenium Grid.

WebDriver - это *интерфейс* удаленного управления, который позволяет вам манипулировать элементами DOM на веб-страницах, а также управлять поведением пользовательских агентов. Этот интерфейс обеспечивает проводной протокол, нечувствительный к языку , который был реализован для различных платформ, таких как:

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

а также другие реализации:

- EventFiringWebDriver
- HtmlUnitDriver
- PhantomJSDriver
- RemoteWebDriver

Selenium WebDriver является одним из инструментов Selenium, который предоставляет объектно-ориентированные API на различных языках, чтобы обеспечить больший контроль и применение стандартных методов разработки программного обеспечения. Чтобы точно смоделировать способ взаимодействия пользователя с веб-приложением, он использует « Native OS Level Events» в качестве противопоставления «Синтезированным событиям JavaScript».

Ссылки для ссылки:

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

Установка или настройка для Java

Чтобы писать тесты с использованием Selenium Webdriver и Java в качестве языка программирования, вам необходимо загрузить JAR-файлы Selenium Webdriver с веб-сайта Selenium.

Существует несколько способов настройки Java-проекта для веб-сервера Selenium, одним из самых простых из которых является использование Maven. Maven загружает требуемые привязки Java для Selenium webdriver, включая все зависимости. Другой способ - загрузить JAR-файлы и импортировать их в свой проект.

Шаги по настройке проекта Selenium Webdriver с использованием Maven:

1. Установите maven на окне Windows после этого документа: https://maven.apache.org/install.html
2. Создайте папку с именем `selenium-learing`
3. Создайте файл в папку выше, используя любой текстовый редактор с именем `pom.xml`
4. Скопировать ниже содержимого в `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>
```

Примечание . Убедитесь, что версия, указанная выше, является последней. Вы можете проверить последнюю версию здесь: http://docs.seleniumhq.org/download/maven.jsp

5. Используя командную строку, запустите команду ниже в каталог проекта.

   ```
   mvn clean install
   ```

Выше команда загрузит все необходимые зависимости и добавит в проект.

6. Напишите команду ниже для создания проекта eclipse, который можно импортировать в среду Eclipse.

   ```
   mvn eclipse:eclipse
   ```

7. Чтобы импортировать проект в eclipse ide, вы можете выполнить следующие шаги

   Открыть Elipse -> Файл -> Импорт -> Общие -> Существующий проект в рабочую

область -> Далее -> Обзор -> Найти папку с pom.xml -> Ok -> Готово

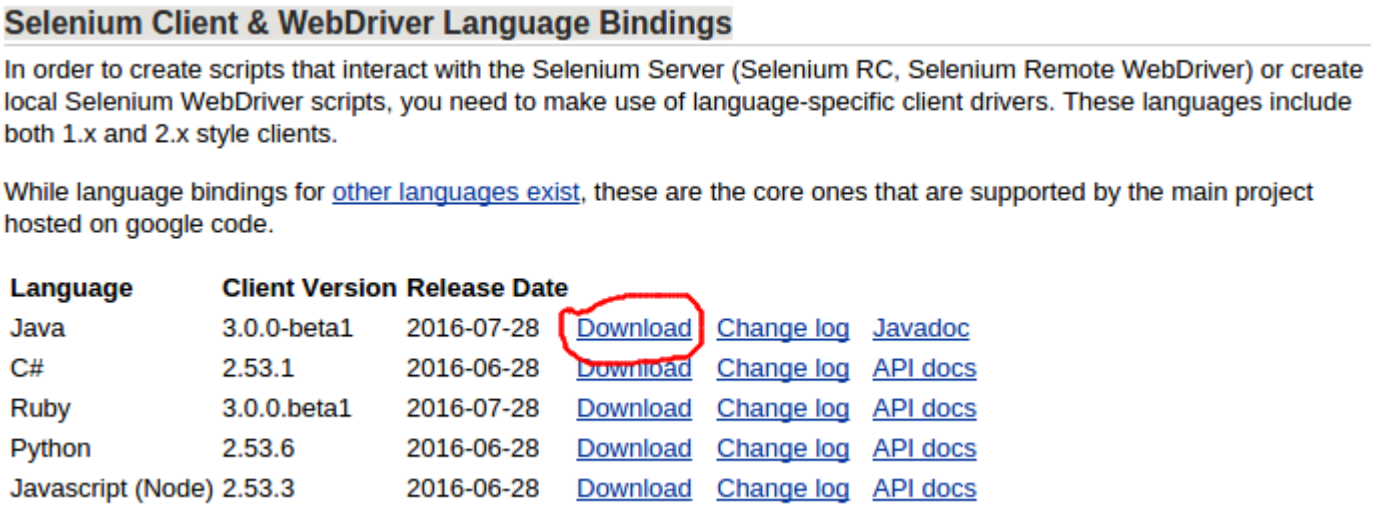
Установите плагин m2eclipse, щелкнув правой кнопкой мыши на своем проекте и выберите Maven -> Enable Dependency Management.

Шаги по настройке проекта Selenium Webdriver с использованием файлов Jar

1. Создайте новый проект в Eclipse, выполнив следующие шаги.

   Открыть Elipse -> Файл -> Создать -> Проект Java -> Предоставить имя (обучение по селену) -> Готово

2. Загрузите файлы jar с http://www.seleniumhq.org/download/ . Вам необходимо загрузить как **Selenium Standalone Server, так** и **Selenium Client & WebDriver Language Bindings** . Поскольку этот документ говорит о Java, вам нужно загрузить только jar из раздела Java. Взгляните на прилагаемый скриншот.



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

Примечание. Автономный сервер Selenium необходим только в том случае, если вы хотите использовать удаленный сервер для запуска тестов. Поскольку этот документ находится выше, вы можете создать проект, чтобы все было на месте.

3. Банки загрузятся в zip-файл, распакуйте их. Вы должны иметь возможность видеть `.jar` напрямую.

4. В eclipse щелкните правой кнопкой мыши проект, который вы создали на шаге 1, и выполните следующие шаги.

   Свойства -> Путь сборки Java -> Выбрать вкладку «Библиотеки» -> Нажмите «Добавить внешние банки» -> Найти папку распакованного jar, которую вы загрузили выше -> Выбрать все банки из папки `lib` -> Нажмите «OK» -> «Снова нажмите« Добавить внешние банки »- > Найдите одну и ту же распакованную папку -> Выберите банку, которая находится за пределами папки lib (`client-combined-3.0.0-beta1-nodeps.jar`) -> Ok

Аналогичным образом добавьте `Selenium Standalone Server` после вышеуказанного шага.

5. Теперь вы можете начать писать код селена в свой проект.

PS : Над документацией основана бета-версия selenium-3.0.0, поэтому имена указанных файлов jar могут меняться с версией.

Прочитайте Начало работы с selenium-webdriver онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/878/начало-работы-с-selenium-webdriver

глава 2: Selenium-webdriver с Python, Ruby и Javascript вместе с инструментом CI

Вступление

Это один из способов проведения тестов селена с помощью CircleCI

Examples

Интеграция CircleCI с Selenium Python и Unittest2

Circle.yml

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

main.py

```
import unittest2

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

example_test.py

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

Прочитайте Selenium-webdriver с Python, Ruby и Javascript вместе с инструментом CI

онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/10091/selenium-webdriver-c-python--ruby-и-javascript-вместе-с-инструментом-ci

глава 3: Без заголовков

Examples

PhantomJS [C #]

`PhantomJS` является полнофункциональным браузером без браузера **с** поддержкой JavaScript.

Перед запуском вам нужно будет загрузить драйвер PhantomJS и обязательно поместите его в начало своего кода:

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

Отлично, теперь на инициализацию:

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

Это просто создаст новый экземпляр класса PhantomJSDriver. Затем вы можете использовать его так же, как каждый WebDriver, например:

```
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);
    }
}
```

Это прекрасно работает. Тем не менее, проблема, с которой вы, вероятно, сталкивались, заключается в том, что при работе с пользовательским интерфейсом `PhantomJS` открывает новое консольное окно, которое в большинстве случаев действительно не `PhantomJS` . К счастью, мы можем скрыть окно и даже немного улучшить производительность с помощью `PhantomJSOptions` и `PhantomJSDriverService` . Полный рабочий пример ниже:

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

Pro tip: нажмите на класс (например, `PhantomJSDriverService`) и нажмите F12, чтобы увидеть, что именно они содержат, и краткое описание того, что они делают.

SimpleBrowser [C #]

`SimpleBrowser` - это легкий WebDriver **без** поддержки JavaScript.

Это значительно быстрее, чем вышеупомянутый `PhantomJS` , однако, когда дело доходит до функциональности, оно ограничено простыми задачами без каких-либо причудливых функций.

Во-первых, вам нужно будет скачать пакет SimpleBrowser.WebDriver , а затем поместить этот код в начало:

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

Теперь, вот краткий пример того, как его использовать:

```
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);
    }
}
```

Безглавой браузер в Java

HTMLUnitDriver

HTMLUnitDriver - самая легкая версия браузера без браузера (без GUI) для Webdriver на основе HtmlUnit. Он моделирует документы HTML и предоставляет API, который позволяет вам вызывать страницы, заполнять формы, ссылки на клики и т. Д., Как и в обычном браузере. Он поддерживает JavaScript и работает с библиотеками AJAX. Он используется для тестирования и извлечения данных с веб-сайта.

Пример. Использование HTMLUnitDriver для получения списка вопросов из `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();
                }
    }
```

Он аналогичен любому другому браузеру (Mozilla Firefox, Google Chrome, IE), но у него нет графического интерфейса, выполнение на экране не отображается.

Прочитайте Без заголовков онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/3931/без-заголовков

глава 4: Взаимодействие с веб-элементом

Examples

C

Очистка содержимого элемента (обычно текстовое поле)

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

Ввод данных в элемент (обычно текстовое поле)

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

Хранение значения элемента.

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

Хранение текста элемента.

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

Нажатие на элемент

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

Отправка формы

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

Идентификация видимости элемента на странице

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

Идентификация состояния элемента на странице

```
bool isEnabled = interactionWebElement.Enabled;

bool isSelected=interactionWebElement.Selected;
```

Поиск дочернего элемента взаимодействияWebElement

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

Поиск дочерних элементов взаимодействияWebElement

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

Джава

Очистка содержимого веб-элемента: (обратите внимание - при имитации действий пользователя в тестах лучше отправить обратное пространство, см. Следующее действие)

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

Ввод данных - имитация отправки нажатий клавиш:

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

Получение значения атрибута элемента:

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

Получение текста элемента:

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

Выбор из раскрывающегося списка:

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

Самостоятельное объяснение:

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

Действия с использованием `org.openqa.selenium.interactions.Actions` :

Перетаскивания:

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

```
dragAndDrop.perform();
```

Выберите несколько:

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

dragAndDrop.perform();
```

Самостоятельное объяснение (с использованием строителя):

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

См. Здесь дополнительные примеры расширенных действий и полный список.

Использование Javascript:

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

Прочитайте Взаимодействие с веб-элементом онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/4280/взаимодействие-с-веб-элементом

глава 5: Взаимодействие с окнами браузера

Examples

Управление активным окном

C

Максимизация окна

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

Это довольно просто, гарантирует, что наше текущее активное окно будет максимизировано.

Положение окна

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

Здесь мы по существу перемещаем текущее активное окно в новую позицию. В объекте `Point` мы предоставляем координаты `x` и `y` ; они затем используются в качестве смещений в верхнем левом углу экрана для определения места размещения окна. Обратите внимание, что вы также можете сохранить позицию окна в переменной:

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

Размер окна

Установка и получение размера окна использует тот же синтаксис, что и позиция:

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

URL окна

Мы можем получить текущий URL активного окна:

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

Мы также можем установить URL для активного окна, которое заставит драйвер перейти к новому значению:

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

Оконные ручки

Мы можем получить ручку для текущего окна:

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

И мы можем получить ручки для всех открытых окон:

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

ПИТОН

Максимизация окна

```
driver.maximize_window()
```

Получить положение окна

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

Установить положение окна

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

Получить размер окна

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

Установить размер окна

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

Название текущей страницы

```
driver.title
```

Текущий URL

```
driver.current_url
```

Оконные ручки

```
driver.current_window_handle
```

Список открытых окон

```
driver.window_handles
```

Закрытие текущего окна браузера

Переключитесь на новую открытую вкладку. Закройте текущие окна (в этом случае новая вкладка). Вернитесь в первое окно.

ТРАНСПОРТИР:

```
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));
```

Обработка нескольких окон

ПИТОН

Наиболее часто используемый сценарий:

1. *открыть страницу в новом окне*
2. *переключиться на него*
3. *сделай что-нибудь*
4. *закрой его*
5. *вернуться к родительскому окну*

```
# 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'
```

Прочитайте Взаимодействие с окнами браузера онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/5181/взаимодействие-с-окнами-браузера

глава 6: Выбрать класс

Синтаксис

- **Джава**
- убрать выделение со всего()
- deselectByIndex (int index)
- deselectByValue (значение java.lang.String)
- deselectByVisibleText (текст java.lang.String)
- getAllSelectedOptions ()
- getFirstSelectedOption ()
- getOptions ()
- isMultiple ()
- selectByIndex (int index)
- selectByValue (значение java.lang.String)
- selectByVisibleText (текст java.lang.String)

параметры

параметры	подробности
индекс	Будет выбран вариант этого индекса.
значение	Значение, соответствующее
текст	Видимый текст соответствует

замечания

`Select` класс Selenium WebDriver предоставляет полезные методы для взаимодействия с опциями `select` . Пользователь может выполнять операции над выпадающим меню, а также отменять операцию, используя приведенные ниже методы.

В **C #** класс Select на самом деле `SelectElement`

Examples

Различные способы выбора из списка DropDown

Ниже и HTML-страница

```
<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>
```

ДЖАВА

Выбрать по индексу

Чтобы выбрать параметр по индексу с помощью 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
    }
}
```

Выбрать по значению

```
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
    }
}
```

Выберите текст видимости

```
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

Все приведенные ниже примеры основаны на общем интерфейсе `IWebDriver`

Выбрать по индексу

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

Выбрать по значению

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

Выбрать по тексту

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

Прочитайте Выбрать класс онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/6426/выбрать-класс

глава 7: Выполнение Javascript на странице

Синтаксис

- объект ExecuteAsyncScript (строковый скрипт, params object [] args);
- объект ExecuteScript (строковый скрипт, params object [] args);

Examples

C

Чтобы выполнить JavaScript в экземпляре `IWebDriver` вам нужно `IWebDriver` в новый интерфейс, `IJavaScriptExecutor`

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

Теперь вы можете получить доступ ко всем методам, доступным в экземпляре `IJavaScriptExecutor` которые позволяют выполнять Javascript, например:

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

питон

Чтобы выполнить Javascript в python, используйте `execute_script("javascript script here")` . execute_script вызывается в экземпляре webdriver и может быть любым допустимым javascript.

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

Джава

Чтобы выполнить Javascript на Java, создайте новый webdriver, который поддерживает Javascript. Чтобы использовать `executeScript()` , либо драйвер должен быть `executeScript()` в `JavascriptExecutor` , либо новая переменная может быть установлена на значение драйвера, используемого в `executeScript()` : `((JavascriptExecutor)driver)` . `driver.executeScript()` принимает строку, которая является действительной Javascript.

```
WebDriver driver = new ChromeDriver();
```

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

Рубин

```
require "selenium-webdriver"

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

Прочитайте Выполнение Javascript на странице онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/6986/выполнение-javascript-на-странице

глава 8: Действия (Эмуляция сложных жестов пользователя)

Вступление

Класс `Actions` дает нам способ точно понять, как пользователь будет взаимодействовать с веб-страницей / элементами. Используя экземпляр этого класса, вы можете описать ряд действий, таких как щелчок, двойное нажатие, перетаскивание, нажатие клавиш и т. Д. После того, как эти действия описаны, чтобы выполнить действия, вы должны вызывать, чтобы строить действия (`.Build()`), а затем инструктируйте их выполнить (`.Perform()`). Поэтому мы должны описывать, строить, выполнять. Ниже будут рассмотрены примеры ниже.

Синтаксис

- dragAndDrop (источник WebElement, цель WebElement)
- dragAndDropBy (источник WebElement, int xOffset, int yOffset)
- выполнить ()

параметры

параметры	подробности
источник	Элемент для эмуляции кнопки вниз.
цель	Элемент для перемещения и выключения мыши.
xOffset	x координата для перемещения.
YOffset	y координируйте движение.

замечания

В этом разделе содержится информация о классе действий Selenium WebDriver. Класс Actions предоставляет вам удобные методы для выполнения сложных жестов пользователя, таких как перетаскивание, удержание и клик и т. Д.

Examples

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();
        }
    }
}
```

Вышеупомянутый `IWebElement` , `source` и перетащить его, и перетащите его во вторую `target` `IWebElement` .

Джава

Перетаскивание с использованием источника и целевого веб-элемента.

Метод удобства, который выполняет щелчок и удерживание в местоположении исходного элемента, перемещается в местоположение целевого элемента, а затем освобождает мышь.

```
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();
    }
}
```

Перетащите элемент и опустите его с заданным смещением.

Метод удобства, который выполняет щелчок и удерживание в местоположении исходного элемента, перемещается по заданному смещению (x и y, оба целых числа), а затем освобождает мышь.

```
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
```

Перейти к элементу

C

Предположим, вы хотите проверить, что, когда вы наведете над элементом, отобразится список переходов. Вы можете проверить содержимое этого списка или выбрать опцию из списка.

Сначала создайте действие, чтобы навести курсор на элемент *(например, у моего элемента есть текст ссылки «Admin»)* :

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

В приведенном выше примере:

- Вы создали действие `mouseHover`
- Вы сказали `driver` перейти к определенному элементу
- Отсюда вы можете выполнять другие `Actions` с объектом `mouseHover` или продолжать тестирование с помощью своего объекта- `driver`

Этот подход особенно полезен, если щелчок на элементе выполняет другую функцию, чем зависание над ним.

Полный пример:

```
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);
```

Прочитайте Действия (Эмуляция сложных жестов пользователя) онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/4849/действия--эмуляция-сложных-жестов-пользователя-

глава 9: Исключения в Selenium-WebDriver

Вступление

Существует ряд исключений, которые можно использовать при использовании webdriver. Приведенные ниже примеры призваны дать представление о том, что они означают.

Examples

Исключения на Python

Документация по исключению селена

ElementNotInteractableException: выбрасывается, когда элемент присутствует в DOM, но взаимодействие с этим элементом ударит по другому элементу из-за порядка рисования

- **ElementNotSelectableException: выбрано** при попытке выбрать неэлектируемый элемент. Примеры невыбираемых элементов:
  - скрипт
- **ElementNotVisibleException: Брошено,** когда элемент присутствует в DOM, но он не отображается, и с ним невозможно взаимодействовать. Чаще всего встречаются при попытке щелкнуть или прочитать текст элемента, который скрыт от представления.
- **ErrorInResponseException: выбрасывается** при возникновении ошибки на стороне сервера. Это может произойти при общении с расширением firefox или удаленным сервером драйверов.
- **ImeActivationFailedException: Брошено** при активации движка IME.
- **ImeNotAvailableException:** брошен, когда поддержка IME недоступна. Это исключение выбрасывается для каждого вызова метода, связанного с IME, если поддержка IME недоступна на компьютере.
- **InvalidArgumentException:** аргументы, переданные команде, являются недействительными или неверными.
- **InvalidCookieDomainException: выбрано** при попытке добавить файл cookie в другом домене, чем текущий URL.
- **InvalidElementStateException: выбрано,** когда действие приведет к недопустимому состоянию элемента. Подклассы:
  - ElementNotInteractableException
  - ElementNotSelectableException
  - ElementNotVisibleException
- **InvalidSelectorException:** выбрано, когда селектор, который используется для поиска элемента, не возвращает WebElement. В настоящее время это происходит только тогда, когда селектор является выражением xpath, и он либо синтаксически недействителен (т. Е. Это не выражение xpath), либо выражение не выбирает

WebElements (например, «count (// input)»).

- **InvalidSwitchToTargetException:** Брошено, когда цель переключения кадра или окна не существует.
- **MoveTargetOutOfBoundsException: Брошено,** когда объект, предоставленный методу moveChains (), недействителен, то есть вне документа.
- **NoAlertPresentException: Брошено** при переключении без уведомления. Это может быть вызвано вызовом операции в классе Alert (), когда предупреждение еще не отображается на экране.
- **NoSuchAttributeException: Брошено,** когда атрибут элемента не найден. Вы можете проверить, существует ли атрибут в конкретном браузере, с которым вы тестируете. Некоторые браузеры могут иметь разные имена свойств для одного и того же свойства. (IE8 .innerText против Firefox .textContent)
- **NoSuchElementException: Брошено,** когда элемент не найден. Если вы столкнулись с этим исключением, вы можете проверить следующее:
  - Проверьте свой селектор, используемый в вашем find_by ...
  - Элемент может пока не отображаться на экране во время операции поиска (веб-страница по-прежнему загружается) см. Selenium.webdriver.support.wait.WebDriverWait () для того, как писать ожидающую оболочку, чтобы ждать появления элемента.
- **NoSuchFrameException: Брошено,** когда цель кадра для переключения не существует.
- **NoSuchWindowException: Брошено,** когда цель, которую нужно переключить, не существует. Чтобы найти текущий набор активных оконных дескрипторов, вы можете получить список активных оконных дескрипторов следующим образом:
  `print driver.window_handles`
- **RemoteDriverServerException:**
- **StaleElementReferenceException: Брошено,** когда ссылка на элемент теперь «устарела». Stale означает, что элемент больше не отображается в DOM страницы. Возможные причины исключения StaleElementReferenceException включают, но не ограничиваются:
  - Вы больше не на одной странице, или страница, возможно, обновилась с момента ее нахождения.
  - Элемент, возможно, был удален и повторно добавлен на экран, так как он был расположен. Такие, как перемещаемый элемент. Обычно это может происходить с использованием структуры javascript, когда значения обновляются и узел перестраивается.
  - Элемент, возможно, находился внутри iframe или другого контекста, который был обновлен.
- **TimeoutException: Брошено,** когда команда не завершается за достаточно **короткое** время.
- **UnableToSetCookieException: выбрасывается,** когда драйверу не удается установить файл cookie.
- **UnexpectedAlertPresentException:** вызывается при появлении неожиданного

предупреждения. Обычно возникает, когда ожидаемый модал блокирует форму webdriver, выполняя любые другие команды.

- **UnexpectedTagNameException:** выбрано, когда класс поддержки не получил ожидаемый веб-элемент.
- Исключение **WebDriverException:** исключение базового webdriver. Все исключения webdriver либо используют WebDriverException, либо InvalidStateException как родительский класс.

Прочитайте Исключения в Selenium-WebDriver онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/10546/исключения-в-selenium-webdriver

глава 10: Использование Selenium Webdriver с Java

Вступление

Selenium webdriver - это система веб-автоматизации, которая позволяет тестировать ваше веб-приложение на разных веб-браузерах. В отличие от Selenium IDE, webdriver позволяет вам разрабатывать собственные тестовые примеры на языке программирования по вашему выбору. Он поддерживает Java, .Net, PHP, Python, Perl, Ruby.

Examples

Открытие окна браузера с определенным URL-адресом с использованием Selenium Webdriver в 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();
    }
}
```

Открытие окна браузера с помощью метода ()

Открытие браузера с помощью метода ().

```
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();
    }
}
```

Прочитайте Использование Selenium Webdriver с Java онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/9158/использование-selenium-webdriver-с-java

глава 11: Использование аннотаций @ FindBy в Java

Синтаксис

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

замечания

Обратите внимание, что есть два способа использования аннотации. Примеры:

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

а также

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

равны, и оба ищут элемент по его идентификатору. В случае `ID_OR_NAME` вы можете использовать

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

`PageFactory.initElements()` должен использоваться после создания объекта объекта страницы, чтобы найти элементы, помеченные аннотацией `@FindBy` .

Examples

Основной пример

Предположим, что мы используем объектную модель страницы . Страница Класс объекта:

```
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
    }
}
```

И класс тестов:

```
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");
    }
}
```

Существует несколько методов поиска WebElements с помощью @FindBy - проверьте раздел Синтаксис.

Прочитайте Использование аннотаций @FindBy в Java онлайн:
https://riptutorial.com/ru/selenium-webdriver/topic/6777/использование-аннотаций--findby-в-java

глава 12: Конфигурация сетки селена

Вступление

Selenium Grid - это среда для запуска теста, распределенного по целому ряду тестовых устройств. Он используется для тестирования веб-приложений. Его можно написать тесты на разных популярных языках программирования, включая C #, Groovy, Java, Perl, PHP, Python и Ruby. Тестирование можно проводить с помощью ряда веб-браузеров на таких платформах, как Windows, Linux и OS X.

Это программное обеспечение с открытым исходным кодом, выпущенное под лицензией Apache 2.0: веб-разработчики могут загружать и использовать его бесплатно.

Синтаксис

- для запуска jar-файла следующий синтаксис для каждого файла jar
- `java -jar <jar-file-full-name>.jar -<your parameters if any>`

параметры

параметры	подробности
роль	Это то, что говорит селену, который он был `hub` или `node`
порт	Это нужно указать, какой порт следует прослушивать `hub` или `node` .
хаб	Этот параметр используется в `node` для указания URL-адреса концентратора
browserName	Его использовали в `node` чтобы указать имя браузера, например firefox, chrome, internet explorer
MaxInstances	Его где указывается экземпляр браузера, например. 5 означает, что будет 5 экземпляров браузера, который будет указан указанным пользователем.
nodeConfig	Файл конфигурации Json для узла. Здесь вы можете указать роль, порт и т. Д.
hubConfig	Файл конфигурации Json для узла. Здесь вы можете указать роль, порт, максимальные экземпляры и т. Д.

Examples

Код Java для Selenium Grid

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

Создание узла и узла Selenium Grid

Создание концентратора

Быстрая конфигурация для настройки концентратора и узла в сетке селена.
Дополнительную информацию см. В документе : Grid 2 docs

Требования

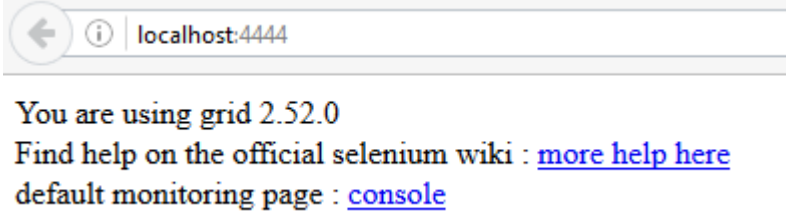
Чтобы настроить концентратор сетки, вам необходимо протекать:

- Селен-серверные автономные-.jar

Создание концентратора

Чтобы создать концентратор, вам нужно запустить сервер selenium.

1. Скачать Selenium-server-standalone-.jar
2. Откройте терминал и перейдите к папке, где Selenium-server-standalone-.jar
3. Выполните следующую команду:
    1. Для конфигурации по умолчанию `java -jar selenium-server-standalone-<Version>.jar -role hub`
    2. Для конфигурации Json `java -jar selenium-server-standalone-<Version>.jar -role hub -hubConfig hubConfig.json`
4. Откройте http: // localhost: 4444 / вы увидите сообщение а



На `console` клика -> `View config` для просмотра конфигурации для сведений о узле.

Создание узла

Требования

Чтобы настроить концентратор сетки, вам необходимо протекать:

- Селен-серверные автономные-.jar
- Webdrivers
  - Драйвер Chrome
  - Драйвер FireFox
  - Драйвер Microsoft Edge
- Браузеры
  - Хром
  - Fire Fox
  - Microsoft Edge (Windows 10)

Создание узла

Теперь создать узлы для концентратора

1. Скачать Selenium-server-standalone-.jar
2. Загрузите браузеры, которые хотите протестировать в
3. Загрузите драйверы для браузеров, которые вы хотите протестировать в
4. Откройте новый терминал и перейдите к местоположению файла jar selenium server
5. Выполните следующую команду:
   1. для конфигурации по умолчанию `java -jar selenium-server-standalone-<VERSION NUMBER>.jar -role node`
   2. Для конфигурации Json `java -jar selenium-server-standalone-<Version>.jar -role node -nodeConfig nodeConfig.json`
6. Теперь перейдите по адресу http: // localhost: 4444 / grid / console, чтобы просмотреть сведения об узле

Конфигурация через Json

Пример конфигурации концентратора:

```
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"
}
```

Пример конфигурации узла

```
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": {}
}
```

Прочитайте Конфигурация сетки селена онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/2504/конфигурация-сетки-селена

глава 13: навигация

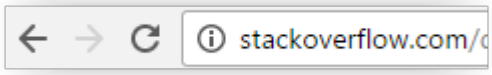
Синтаксис

- **C #**
- void Назад ()
- void Forward ()
- void GotToUrl (строковый url)
- void Обновить ()
- **питон**
- driver.back ()
- driver.forward ()
- driver.get ("URL")
- driver.refresh ()
- **Джава**
- driver.navigate () назад ().
- . Driver.navigate () вперед ();
- driver.navigate () к ("URL").
- . Driver.navigate () обновления ();

Examples

Навигация () [C #]

Можно напрямую перемещаться по браузеру, например, используя стандартные команды панели инструментов, доступные во всех браузерах:



Вы можете создать навигационный объект, вызвав `Navigate()` в драйвере:

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

Объект навигации позволяет выполнять многочисленные действия, которые перемещаются по обозревателю через Интернет:

```
//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();
```

Навигация () [Java]

Для перехода на любой URL:

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

Для перемещения назад:

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

Для перемещения вперед:

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

Для обновления страницы:

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

Методы браузера в WebDriver

WebDriver, основной интерфейс для тестирования, представляющий собой идеализированный веб-браузер. Методы этого класса делятся на три категории:

- Контроль самого браузера
- Выбор WebElements
- Отладочные средства

Ключевыми методами являются get (String), который используется для загрузки новой веб-страницы, и различные методы, подобные findElement (By), которые используются для поиска WebElements. В этой статье мы рассмотрим методы управления браузером. получить

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

Загрузите новую веб-страницу в текущее окно браузера. Это делается с помощью операции HTTP GET, и метод будет блокироваться до завершения загрузки. лучше подождать, пока этот тайм-аут не закончится, поскольку, если базовая страница изменится, пока ваш тест выполнит результаты будущих вызовов против этого интерфейса, будет против только что загруженной страницы. **использование**

```
//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()
```

Получить строку, представляющую текущий URL-адрес, на который смотрит браузер. Он возвращает URL-адрес страницы, загруженной в настоящее время в браузере.

использование

```
//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()
```

Он возвращает заголовок текущей страницы, когда лидирующие и конечные пробелы лишены, или null, если один из них еще не установлен.

использование

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

getPageSource

java.lang.String getPageSource()
```

Получите источник последней загруженной страницы. Если страница была изменена после загрузки (например, по Javascript), нет гарантии, что возвращенный текст будет изменен на странице с измененной.

использование

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

близко

```
void close()
```

Закройте текущее окно, выйдя из браузера, если это последнее открытое окно. Если открыто более одного окна с этим экземпляром драйвера, этот метод закроет окно, в котором он будет сосредоточен на нем.

использование

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

уволиться

```
void quit()
```

Выключает этот драйвер, закрывая каждое связанное окно. После вызова этого метода мы не сможем использовать какой-либо другой метод, используя тот же самый экземпляр драйвера.

использование

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

Все это очень полезные методы, доступные в Selenium 2.0 для управления браузером по мере необходимости.

Прочитайте навигация онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/7272/навигация

глава 14: Настройка / Получение размера окна браузера

Вступление

Установка или получение размера окна любого браузера во время автоматизации

Синтаксис

- . Driver.manage () окно () максимального ().
- driver.manage (). window (). setSize (*DimensionObject*);
- driver.manage (). окно (). GETSIZE ()

Examples

ДЖАВА

Установите максимальный размер окна браузера:

```
//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();
```

Установите определенный размер окна:

```
//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);
```

Размер окна браузера:

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

Прочитайте Настройка / Получение размера окна браузера онлайн:
https://riptutorial.com/ru/selenium-webdriver/topic/10093/настройка---получение-размера-окна-браузера

глава 15: Настройка Selenium e2e

Вступление

В этом разделе рассказывается о сквозной установке Selenium, то есть Selenium Webdriver + TestNG + Maven + Jenkins.

Дополнительную информацию можно найти в разделе « Отчеты HTML»

Examples

Настройка TestNG

TestNG - это обновленная тестовая среда для junit. Мы собираемся использовать **testng.xml** для вызова наборов тестов. Это полезно, когда мы будем использовать CI впереди.

testng.xml

В корневой папке вашего проекта создайте xml-файл с именем testng.xml. Обратите внимание, что имя может быть другим, но для удобства его везде используют как «testng».

Ниже приведен простой код для файла 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>
```

Настройка Maven

TBD. Как настроить pom.xml для вызова testng.xml

Настройка Jenkins

TBD. Покроет установку Jenkins для вытягивания кода из git / bitbucket и т. Д.

Прочитайте Настройка Selenium e2e онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/10724/настройка-selenium-e2e

глава 16: Обработка ошибок при автоматизации с использованием селена

Examples

питон

`WebDriverException` - это базовое исключение `Selenium-WebDriver` которое может использоваться для обнаружения всех других исключений `Selenium-WebDriver`

Чтобы уловить исключение, он должен быть импортирован первым:

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

а потом:

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

Точно так же вы можете импортировать другие более конкретные исключения:

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

Если вы хотите только извлечь сообщение об исключении:

```
from selenium.common.exceptions import UnexpectedAlertPresentException

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

Прочитайте Обработка ошибок при автоматизации с использованием селена онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/9548/обработка-ошибок-при-автоматизации-с-использованием-селена

глава 17: Обращение с предупреждением

Examples

Селен с Java

Вот как обрабатывать всплывающее оповещение в Java с помощью Selenium:

Существует 3 типа всплывающих окон.

1. **Simple alert** : alert («Это простое предупреждение»);
2. **Предупреждение о подтверждении** : var popuResult = подтвердить («Подтвердить всплывающее окно с помощью кнопки« OK »и« Отмена »);
3. **Быстрое предупреждение** : var person = prompt («Вам нравится stackoverflow?», «Да / Нет»);

Его дозвон, какой тип всплывающего окна необходимо обрабатывать в тестовом примере.

Либо вы можете

1. accept () Чтобы принять оповещение
2. Увольнять () Отклонить предупреждение
3. getText () Чтобы получить текст предупреждения
4. sendKeys () Чтобы написать текст для предупреждения

Для простого предупреждения:

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

Для подтверждения подтверждения:

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

Для быстрого оповещения:

```
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();
```

в соответствии с вашими потребностями.

Другой способ, которым вы можете это сделать, - обернуть свой код внутри try-catch:

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

C

Вот как закрыть всплывающее оповещение в C # с Selenium:

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

в соответствии с вашими потребностями.

Другой способ, которым вы можете это сделать, - обернуть свой код внутри try-catch:

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

ПИТОН

Существует несколько способов переключения всплывающих окон в `Python` :

1. *Устаревший* :

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

2. *Использование* `switch_to` :

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

3. *Использование* `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. *Объявив экземпляр класса* `Alert` :

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

alert = Alert(driver)
```

Чтобы заполнить поле ввода во всплывающем окне, вызванное `prompt()` `JavaScript` `prompt()` :

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

Чтобы подтвердить всплывающее диалоговое окно *:

```
alert.accept()
```

Уволить:

```
alert.dismiss()
```

Чтобы получить текст из всплывающего окна:

```
alert.text
```

** PS* `alert.dismiss()` *может использоваться для подтверждения всплывающих окон, вызванных* `JavaScript` `alert()` *а также* `alert.confirm()`

Прочитайте Обращение с предупреждением онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/6048/обращение-с-предупреждением

глава 18: Объектная модель страницы

Вступление

Значительная роль в автоматизации веб-сайтов и веб-приложений заключается в идентификации элементов на экране и взаимодействии с ними. Элементы находятся в Selenium с помощью локаторов и класса `By` . Эти локаторы и взаимодействия помещаются внутри объектов страницы как наилучшая практика, чтобы избежать дублирования кода и упростить обслуживание. Он инкапсулирует `WebElements` и предполагает содержать информацию о поведении и возврате на странице (или часть страницы в веб-приложении).

замечания

Объектная модель страницы - это шаблон, в котором мы пишем объектно-ориентированные классы, которые служат интерфейсом для определенного вида веб-страницы. Мы используем методы этого класса страницы для выполнения требуемого действия. Несколько лет назад мы манипулировали HTML-кодом веб-страницы в тестовых классах, что было очень сложно поддерживать вместе с хрупкими изменениями в пользовательском интерфейсе.

Однако, если ваш код организован таким образом, что шаблон объекта страницы предоставляет API-интерфейс, специфичный для приложения, позволяющий вам манипулировать элементами страницы без копания вокруг HTML. Основная ручка большого пальца говорит, что ваш объект страницы должен иметь все, что человек может сделать на этой веб-странице. Например, чтобы получить доступ к текстовому полю на веб-странице, вы должны использовать метод для получения текста и строки возврата после выполнения всех изменений.

Несколько важных моментов, которые вы должны учитывать при разработке объектов страницы:

1. Объект страницы обычно не должен строиться только для страниц, но вы должны предпочесть строить его для значительных элементов страницы. Например, страница с несколькими вкладками для отображения разных диаграмм ваших академиков должна иметь одинаковое количество страниц, чем количество вкладок.

2. Переход с одного вида на другой должен возвращать экземпляр классов страниц.

3. Методы утилиты, которые должны быть доступны только для определенного вида или веб-страницы, должны принадлежать только этому классу.

4. Методы утверждения не должны заботиться о классах страниц, вы можете иметь методы для возврата к логическим, но не проверяйте их там. Например, чтобы

проверить полное имя пользователя, вы можете получить метод для получения логического значения:

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

5. Если ваша веб-страница основана на iframe, предпочитайте также классы страниц для iframes.

Преимущества шаблона объекта страницы:

1. Чистое разделение между тестовым кодом и кодом страницы
2. В случае каких-либо изменений в пользовательском интерфейсе веб-страницы не нужно менять код в нескольких местах. Измените только классы страниц.
3. Нет разбросанных локаторов элементов.
4. Делает код более понятным
5. Простое обслуживание

Examples

Введение (использование Java)

Пример выполнения теста входа на основе шаблона объекта страницы:

```
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

Объекты страницы должны содержать поведение, возвращаемую информацию для утверждений и, возможно, метод для состояния готовности страницы при инициализации. Selenium поддерживает объекты страницы с помощью аннотаций. В C # это выглядит следующим образом:

```
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;
    }
}
```

заметки:

- `CacheLookup` сохраняет элемент в кеше и сохраняет возврат нового элемента к каждому вызову. Это повышает производительность, но не подходит для динамически изменяющихся элементов.
- `searchButton` имеет 2 имени класса и идентификатор, поэтому я не могу использовать имя класса или идентификатор.
- Я подтвердил, что мои локаторы возвратят мне элемент, который я хочу использовать Инструменты разработчика (для Chrome), в других браузерах вы можете использовать FireBug или аналогичный.
- Метод `Search()` возвращает другой объект страницы (`ResultsPage`), поскольку поисковый щелчок перенаправляет вас на другую страницу.

Объектная модель страницы оптимальной практики

- Создавайте отдельные файлы для верхнего и нижнего колонтитула (поскольку они распространены для всех страниц, и нет смысла делать их частью одной страницы)
- Сохраняйте общие элементы (например, Поиск / Назад / Далее и т. Д.) В отдельном файле (Идея состоит в том, чтобы удалить любое дублирование и сохранить логику сегрегации)
- Для драйвера рекомендуется создать отдельный класс драйвера и сохранить

драйвер как статический, чтобы он мог быть доступен на всех страницах! (У меня есть все мои веб-страницы, расширяющие DriverClass)

- Функции, используемые в PageObjects, разбиты на наименьший возможный фрагмент, учитывая частоту и способ их вызова (способ, которым вы это сделали для входа в систему, хотя логин может быть разбит на enterUsername и enterPassword, но все еще сохраняет его поскольку функция входа более логична, поскольку в большинстве случаев функция Login будет вызываться вместо отдельных вызовов для enterUsername и enterPassword)
- Использование самого PageObjects отделяет тестовый скрипт от elementLocators
- Утилиты функции в отдельной папке utils (например, DateUtil, excelUtils и т. Д.)
- У вас есть конфигурации в отдельной папке conf (например, настройка среды, на которой должны выполняться тесты, настройка выходных и входных папок)
- Включить screenCapture при сбое
- Имейте переменную статического ожидания в DriverClass с некоторым неявным временем ожидания, как вы это делали. Всегда старайтесь иметь условные ожидания, а не статические ожидания: wait.until (ExpectedConditions). Это гарантирует, что ожидание не замедляет выполнение без необходимости.

Прочитайте Объектная модель страницы онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/4853/объектная-модель-страницы

глава 19: Отчеты HTML

Вступление

В этом разделе рассматривается создание отчетов HTML для тестов селена. Для отчетности доступны различные типы плагинов, и широко используются Allure, ExtentReports и ReportNG.

Examples

ExtentReports

Этот пример охватывает реализацию ExtentReports в Selenium с использованием TestNG, Java и Maven.

ExtentReports доступны в двух версиях, сообщества и рекламы. Для удобства и демонстрации мы будем использовать версию сообщества.

1. Зависимость

Добавьте зависимость в файл Maven pom.xml для отчетов о степени.

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

2. Настройка плагинов

Настройте плагин maven surefire, как показано ниже в 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. Пример теста с ExtentReports

Теперь создайте тест с именем 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 Отчеты

Этот пример охватывает реализацию отчетов Allure в Selenium с использованием TestNG, Java и Maven.

Конфигурация Maven

вместилище

Добавьте следующий код для настройки репозитория jcenter

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

зависимость

Добавьте следующие зависимости к вашему 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>
```

Конфигурация плагина 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>
```

Пример теста для отчета Allure

Создайте образец теста с именем 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();
    }
}
```

В вышеприведенном классе мы использовали класс LogUtiil. Это просто делается для регистрации **шагов** в нашем тесте. Ниже приведен код для того же

LogUtil.java

```
public final class LogUtil {

    private LogUtil() {
    }

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

Вот

@Title ("") добавит название к вашему тесту в Allure Report

@Description ("") добавит описание к вашему тесту

@Step ("") добавит шаг в отчете о соблазнении для теста

При выполнении xml-файл будет сгенерирован в папке «target / allure-results /»

Заключительный отчет с Дженкинсом

Если вы работаете в Jenkins с плагином Allure Report, то Jenkins автоматически отобразит отчет в вашей работе.

Заключительный отчет без Дженкинса

Для тех, кто не имеет Jenkins, используйте следующую команду для создания html-отчета. Allure CLI - это Java-приложение, поэтому оно доступно для всех платформ. Перед использованием Allure CLI вам необходимо вручную установить Java 1.7+.

Debian

Для репозиториев на базе Debian мы предоставляем PPA, поэтому установка проста: установите Allure CLI для debian

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

Поддерживаемые дистрибутивы: надежные и точные. После установки вы получите команду allure.

Mac OS

Вы можете установить Allure CLI через Homebrew.

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

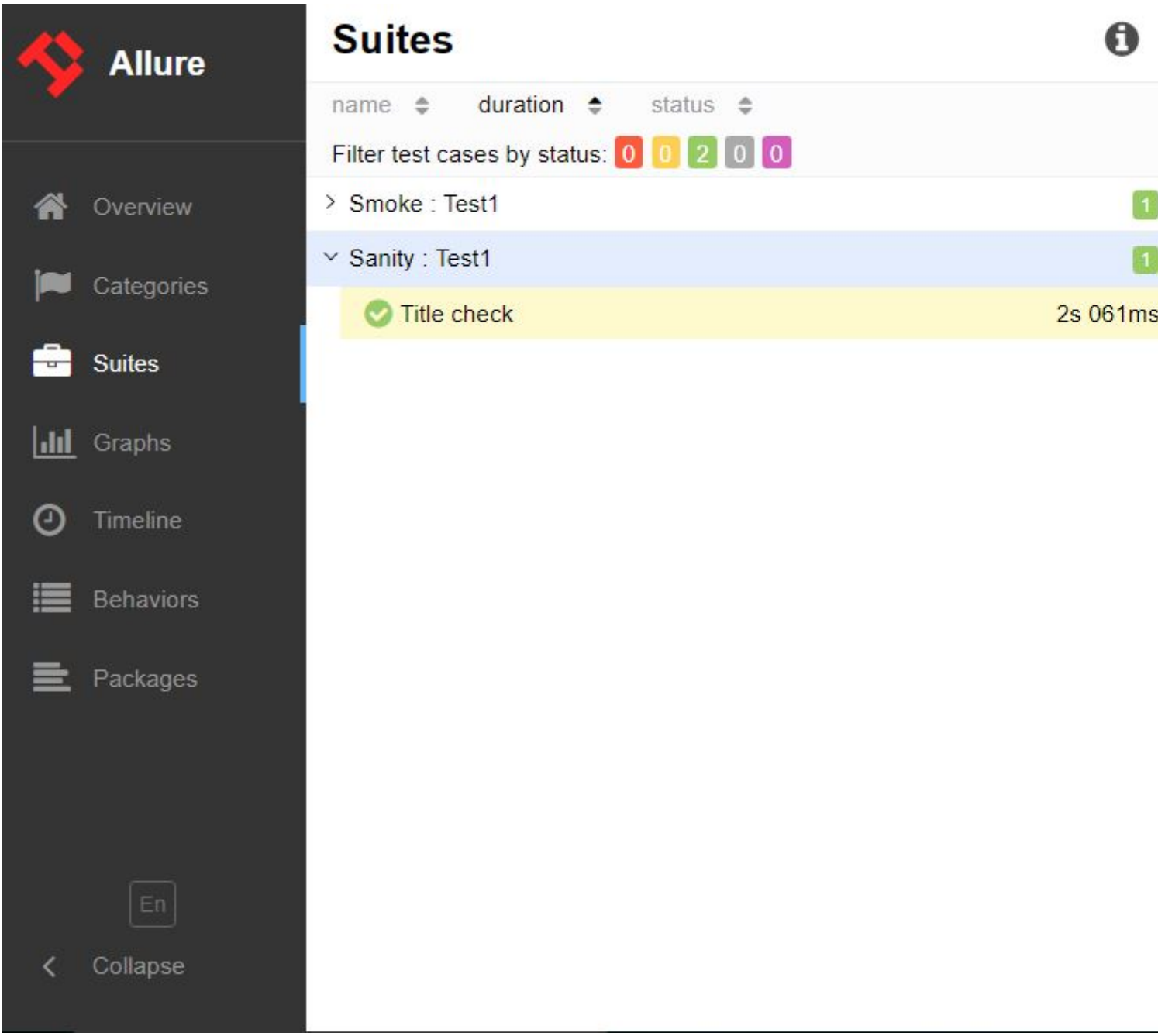
После установки вы получите команду allure.

Windows и другие Unix

1. Загрузите последнюю версию в виде zip-архива с https://github.com/allure-framework/allure-core/releases/latest .
2. Распакуйте архив в каталог allure-commandline. Перейдите в каталог bin.
3. Используйте allure.bat для Windows и allure для других платформ Unix.

В Commandline / Terminal теперь просто введите следующий синтаксис, и отчет будет сгенерирован в папку allure-report

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



Прочитайте Отчеты HTML онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/10721/отчеты-html

глава 20: Переключающие рамки

Синтаксис

- **Джава**
- driver.switchTo (). frame (String name);
- driver.switchTo (). frame (String id);
- driver.switchTo (). frame (int index);
- frame.switchTo (). frame (элемент WebElement frameElement);
- . Driver.switchTo () defaultContent ();
- **C #**
- driver.SwitchTo (). Frame (int frameIndex);
- driver.SwitchTo (). Frame (элемент интерфейса IWebElement);
- driver.SwitchTo (). Frame (string frameName);
- . Driver.SwitchTo () DefaultContent ();
- **питон**
- driver.switch_to_frame (nameOrId)
- driver.switch_to.frame (nameOrId)
- driver.switch_to_frame (индекс)
- driver.switch_to.frame (индекс)
- driver.switch_to_frame (frameElement)
- driver.switch_to.frame (frameElement)
- driver.switch_to_default_content ()
- driver.switch_to.default_content ()
- **JavaScript**
- driver.switchTo (). рама (nameOrId)
- driver.switchTo (). кадр (индекс)
- driver.switchTo (). defaultContent ()

параметры

параметр	подробности
nameOrId	Выберите фрейм по имени.
индекс	Выберите кадр по индексу на основе нуля.
frameElement	Выберите фрейм, используя ранее расположенный WebElement

Examples

Чтобы переключиться на фрейм с помощью Java

Для экземпляра, если исходный HTML-код html-представления или элемента обернут iframe следующим образом:

```
<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 />
```

... затем, чтобы выполнить любое действие на веб-элементах iframe, вам нужно сначала переключить фокус на iframe, используя любой из следующих методов:

Использование идентификатора кадра (должно использоваться только в том случае, если вы знаете идентификатор iframe).

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

Использование имени фрейма (должно использоваться только в том случае, если вы знаете имя iframe).

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

Использование индекса кадра (следует использовать, только если у вас нет идентификатора или имени iframe), где индекс определяет позицию iframe среди всех кадров.

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

Примечание. Если на странице есть три кадра, то первый кадр будет иметь индекс 0, второй - по индексу 1, а третий - по индексу 2.

Использование ранее расположенного webelement (должно использоваться только в том случае, если вы уже разместили фрейм и вернули его как `WebElement`).

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

Итак, нажмите на якорь `Red Ball` :

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

Чтобы выйти из фрейма с помощью Java

Чтобы переключить фокус на основной или первый кадр страницы. Вы должны использовать синтаксис ниже.

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

Переключитесь на фрейм с помощью C

1. Переключитесь на кадр по индексу.

Здесь мы переходим к индексу 1. Индекс относится к порядку кадров на странице. Это должно использоваться как последнее средство, поскольку идентификатор кадра или имена намного надежнее.

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

2. Переключитесь на кадр по имени

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

3. Переключитесь на кадр по заголовку, идентификатору или другим, передав IWebElement

Если вы хотите переключиться на фрейм по идентификатору или названию, вам необходимо передать в качестве элемента веб-элемент:

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

Также обратите внимание, что если ваш фрейм занимает несколько секунд, вы можете использовать ожидание :

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

Выйдите из рамки:

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

Чтобы выйти из рамки с помощью C

Чтобы переключить фокус на основной или первый кадр страницы. Вы должны использовать синтаксис ниже.

. WebDriver.SwitchTo () DefaultContent ();

Переключение между дочерними рамками родительской рамки.

У вас есть родительская рамка (Frame-Parent). и 2 дочерних кадра (Frame_Son, Frame_Daughter). Давайте посмотрим на различные условия и способы их обработки.

1. От родителя к сыну или дочери:

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

2. От сыня к родительскому: если родительский фрейм по умолчанию, переключитесь на кадр по умолчанию, иначе от переключателя фрейма по умолчанию до родительского фрейма. Но вы не можете напрямую переключиться с сына на родителя.

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

3. От сына дочери: если твоя сестра совершит какую-то ошибку, не кричите на нее, просто протяните руку своему родителю. Аналогично, вы даете управление родительскому кадру, а затем дочернему кадру.

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

Подождите, пока ваши кадры загрузятся

В некоторых случаях ваш фрейм может не отображаться сразу, и вам, вероятно, придется подождать, пока он будет загружен для переключения. Или у вас будет NoSuchFrameException.

Поэтому всегда нужно выбирать, прежде чем переключаться. Следующим является идеальный способ подождать, пока загрузится фрейм.

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

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

// выдает исключение только в том случае, если ваш фрейм не отображается с вашим временем ожидания 300 секунд}

Прочитайте Переключающие рамки онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/4589/переключающие-рамки

глава 21: Перемещение между несколькими кадрами

Вступление

На веб-страницах содержится количество кадров, селен считать Frame - это отдельное окно, поэтому доступ к содержимому, присутствующему в кадре, необходимо переключить в кадр. Много раз нам нужна веб-структура, в которой у нас есть фрейм с рамкой для навигации внутри окон рамки. Selenium предоставляет метод swithTo ().

Examples

Пример рамы

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

    <iframe "id="iframe_Login2">

        <iframe "id="iframe_Login3">

        </iframe>

    </iframe>

</iframe>
```

Чтобы переключиться на кадр в селене, используйте метод swithTo () и frame ().

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

Чтобы вернуться назад, мы можем использовать parentFrame () и defaultContest ();

parentFrame (): изменить фокус на родительский контекст. Если текущий контекст является контекстом просмотра верхнего уровня, контекст остается неизменным.

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

defaultContent (): выбирает либо первый кадр на странице, либо основной документ, когда страница содержит фреймы.

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

Прочитайте Перемещение между несколькими кадрами онлайн:

https://riptutorial.com/ru/selenium-webdriver/topic/9803/перемещение-между-несколькими-кадрами

глава 22: Подождите

Examples

Типы ожидания в селене WebDriver

При запуске любого веб-приложения необходимо учитывать время загрузки. Если ваш код пытается получить доступ к любому элементу, который еще не загружен, WebDriver выдаст исключение, и ваш скрипт остановится.

Существует три типа ожиданий -

- **Неявные ожидания**
- **Явные ожидания**
- **Свободные ожидания**

Неявные ожидания используются для установки времени ожидания во всей программе, в то время как явные ожидания используются только на определенных участках.

Неявное ожидание

Неявное ожидание - сказать WebDriver, чтобы опросить DOM в течение определенного времени, пытаясь найти элемент или элементы, если они не доступны сразу. Неявные ожидания - это в основном ваш способ сообщить WebDriver о задержке, которую вы хотите увидеть, если указанный веб-элемент отсутствует, что ищет WebDriver. Значение по умолчанию равно 0. После установки неявный ожидание задается для жизни экземпляра объекта WebDriver. Неявное ожидание объявляется в части экземпляра кода с использованием следующего фрагмента.

Пример в **Java** :

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

Пример в **C #** :

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

Таким образом, в этом случае вы сообщаете WebDriver, что он должен ждать 15 секунд, если указанный элемент не доступен в пользовательском интерфейсе (DOM).

Явное ожидание

Вы можете столкнуться с экземплярами, когда какой-то элемент занимает больше времени для загрузки. Установка неявного ожидания таких случаев не имеет смысла, поскольку браузер будет ждать ненужных за одно и то же время для каждого элемента, увеличивая время автоматизации. Явное ожидание помогает здесь, минуя неявное ожидание для некоторых конкретных элементов.

Явные ожидания - это интеллектуальные ожидания, которые ограничены определенным веб-элементом. Используя явные ожидания, вы в основном говорите WebDriver на максимуме, чтобы ждать X единиц времени, прежде чем он сдастся.

Явные ожидания выполняются с использованием классов WebDriverWait и ExpectedConditions. В приведенном ниже примере мы подождем до 10 секунд для элемента, идентификатор которого является именем пользователя, чтобы стать видимым, прежде чем переходить к следующей команде. Вот шаги.

Пример в **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”);
```

Класс ExpectedConditions имеет некоторые предопределенные общие условия для ожидания элемента. Нажмите здесь, чтобы просмотреть список этих условий в привязке Java.

Пример в **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);
}
```

В этом примере система будет ждать 10 секунд, пока элемент не будет виден. Если элемент не будет отображаться после таймаута, WebDriver выкинет `WebDriverTimeoutException` .

Обратите внимание: если элемент отображается до истечения 10-секундного тайм-аута, система немедленно продолжит дальнейший процесс.

Свободное ожидание

В отличие от неявного и явного ожидания, свободное ожидание использует два параметра. Значение тайм-аута и частота опроса. Скажем, у нас есть значение тайм-аута как 30 секунд, а частота опроса - 2 секунды. WebDriver будет проверять элемент через каждые 2 секунды до значения тайм-аута (30 секунд). После превышения значения тайм-аута без какого-либо результата генерируется исключение. Ниже приведен пример кода, который демонстрирует безупречное ожидание.

Пример в **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"));
    }
});
```

Еще одно преимущество использования свободного ожидания - мы можем игнорировать определенные типы исключений (например, NoSuchElementExceptions) во время ожидания. Из-за всех этих положений, свободное время полезно в приложениях AJAX, а также в сценариях, когда время загрузки элемента часто колеблется. Стратегическое использование быстрого ожидания значительно улучшает усилия по автоматизации.

Различные типы явных условий ожидания

В явном ожидании вы ожидаете, что это произойдет. Например, вы хотите подождать, пока элемент будет доступен для клика.

Вот демонстрация нескольких общих проблем.

Обратите внимание: во всех этих примерах вы можете использовать любой параметр « `By` *качестве локатора», например, имя* `classname` , `xpath` , `link text` , `tag name` *или* `cssSelector`

Подождите, пока элемент не будет виден

Например, если вашему веб-сайту требуется некоторое время для загрузки, вы можете подождать, пока страница завершит загрузку, и ваш элемент будет виден WebDriver.

C #

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

Джава

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

Подождите, пока элемент больше не будет виден

То же, что и раньше, но наоборот.

C #

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

Джава

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

Подождите, пока текст не будет указан в указанном элементе

C #

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

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

Джава

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

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

Если вы перейдете к данной ссылке выше, вы увидите все условия ожидания.

Разница между использованием этих условий ожидания заключается в их входном параметре.

Это означает, что вам необходимо передать WebElement, если его входным параметром является WebElement, вам нужно передать локатор элемента, если он принимает локатор By в качестве его входного параметра.

Выберите разумно, какое условие ожидания вы хотите использовать.

Ожидание заявок Ajax на завершение

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);
            }
        }
    }
}
```

Этот пример полезен для страниц, на которых выполняются запросы ajax, здесь мы

используем `IJavaScriptExecutor` для запуска нашего собственного кода JavaScript. Как это в `while` петли он будет продолжать работать , пока `ajaxIsComplete == true` и поэтому оператор возврата выполняется.

Мы проверяем, что все запросы ajax завершены, подтвердив, что `jQuery.active` равен `0` . Это работает, потому что каждый раз, когда выполняется новый запрос ajax, `jQuery.active` увеличивается и каждый раз, когда запрос дополняет его, уменьшается, из этого можно сделать вывод, что когда `jQuery.active == 0` все запросы ajax должны быть завершены.

Свободное время ожидания

Свободное ожидание является суперклассом **явного** ожидания (`WebDriverWait`), который более настраивается, поскольку он может принимать аргумент функции ожидания. Я передам **имплицитное** ожидание, так как лучше избегать этого.

Использование (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;
  }
});
```

Когда вы используете Explicit wait с его значениями по умолчанию, это просто `FluentWait<WebDriver>` со значениями по умолчанию: `DEFAULT_SLEEP_TIMEOUT = 500;` и игнорирование `NotFoundException` .

Свободное ожидание

Каждый экземпляр FluentWait определяет максимальное время ожидания условия, а также частоту проверки состояния. Кроме того, пользователь может настроить ожидание, чтобы игнорировать определенные типы исключений во время ожидания, например, NoSuchElementExceptions при поиске элемента на странице. Он связан с драйвером.

```
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"));
```

Прочитайте Подождите онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/4435/подождите

глава 23: Поиск веб-элементов

Синтаксис

- ByChained (params By [] bys)

замечания

Элементы находятся в Selenium с помощью *локаторов* и класса `By` . Чтобы создать надежный проект автоматизации с Selenium, нужно использовать локаторы для Web Elements. Локаторы должны быть **описательными, уникальными и вряд ли измениться,** поэтому вы не получите ложных срабатываний в тестах, например. Приоритет заключается в использовании:

1. **ID** - поскольку он уникален, и вы получите именно тот элемент, который вы хотите.
2. **Имя класса** - оно описательно и может быть уникальным в данном контексте.
3. **CSS** (более высокая производительность, чем xpath) - для более сложных селекторов.
4. **XPATH** - где CSS нельзя использовать (XPATH Axis), например `div::parent` .

Остальные локаторы подвержены изменениям или рендерингу, и их предпочтительно избегать.

Правило большого пальца: если ваш код не может найти определенный элемент, одна из причин может заключаться в том, что ваш код не дождался загрузки всех элементов DOM. Подумайте о том, чтобы ваша программа «подождала» в течение короткого периода времени (попробуйте 3-5 секунд, а затем медленно увеличивайте по мере необходимости) перед поиском указанного элемента. Вот пример в Python, взятый из этого вопроса :

```
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

Поиск элементов страницы с помощью WebDriver

Чтобы взаимодействовать с WebElements на веб-странице, сначала нам нужно определить местоположение элемента.

K является ключевым словом доступны селеном.

Вы можете найти элементы By ..

1. **По** *идентификатору*
2. **По** *названию класса*
3. **По** *TagName*
4. **По** *имени*
5. **По** *тексту ссылки*
6. **По** *частичной текстовой ссылке*
7. **С помощью** *CSS Selector*
8. **По** *XPath*
9. **Использование** *JavaScript*

Рассмотрим пример ниже сценария

```
<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" />
```

В приведенном выше коде имя пользователя и пароль задаются с использованием идентификаторов. Теперь вы собираетесь идентифицировать элементы с id.

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

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

Поскольку селен поддерживает 7 разных языков, этот документ дает вам представление о местонахождении элементов на всех языках.

По идентификатору

Пример того, как найти элемент с помощью 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"));
```

По названию класса

Пример того, как найти элемент с именем класса:

```
<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"));
```

По названию тега

Пример того, как найти элемент с именем тега:

```
<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"));
```

По имени

Пример того, как найти элемент с именем:

```
<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"));
```

По тексту ссылки

Пример того, как найти элемент, используя текст ссылки:

```
<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"));
```

По частичной текстовой ссылке

Пример того, как найти элемент с использованием текста частичной ссылки:

```
<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"));
```

С помощью селекторов CSS

Пример того, как найти элемент с помощью селекторов 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"));
```

Вот статья о создании селекторов CSS: http://www.w3schools.com/cssref/css_selectors.asp

---

По XPath

Пример того, как найти элемент с помощью 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"));
```

Вот статья о XPath: http://www.w3schools.com/xsl/xpath_intro.asp

---

Использование JavaScript

Вы можете выполнить произвольный javascript для поиска элемента и до тех пор, пока вы вернете элемент DOM, он будет автоматически преобразован в объект WebElement.

Простой пример на странице с загруженным 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 -
```

Обратите внимание: этот метод не будет работать, если ваш WebDriver не поддерживает JavaScript, например SimpleBrowser .

Выбор по нескольким критериям [C #]

Также можно использовать селекторы вместе. Это делается с помощью объекта `OpenQA.Selenium.Support.PageObjects.ByChained` :

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

Любое число `By` s может быть закодировано и используется как выбор типа И (т. Е. Все `By` s совпадают)

Выбор элементов перед загрузкой страницы

При вызове `driver.Navigate().GoToUrl(url);` , выполнение кода останавливается, пока страница не будет полностью загружена. Иногда это необязательно, когда вы просто хотите извлечь данные.

Примечание. Образцы кода ниже можно считать хаками. Нет никакого «официального» способа сделать это.

Создать новый поток

Создайте и запустите поток для загрузки веб-страницы, затем используйте 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")));
}
```

Использовать таймауты

Используя WebDriverTimeout, вы можете загрузить страницу, и через определенный промежуток времени она выдает исключение, которое заставит страницу останавливать загрузку. В блоке catch вы можете использовать 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")));
    }
}
```

Проблема . Если вы установите слишком короткий тайм-аут, страница перестанет загружаться независимо от того, присутствует ли ваш желаемый элемент. Когда вы установите слишком высокий тайм-аут, вы откажетесь от производительности.

Прочитайте Поиск веб-элементов онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/3991/поиск-веб-элементов

глава 24: Программа Basic Selenium Webdriver

Вступление

В этом разделе показана основная программа для веб-драйверов на языках, поддерживающих селен, таких как C #, Groovy, Java, Perl, PHP, Python и Ruby.

Путешествие включает в себя открытие драйвера браузера -> Страница Google -> выключение браузера

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");
            }
        }
    }
}
```

Вышеупомянутая «программа» переместится на главную страницу Google, а затем закроет браузер после полной загрузки страницы.

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

Это создает новый объект WebDriver с использованием интерфейса `IWebdriver` и создает новый экземпляр окна браузера. В этом примере мы используем `ChromeDriver` (хотя это может быть заменено соответствующим драйвером для любого браузера, который мы хотели использовать). Мы обертываем это с `using` инструкции `using` , потому что `IWebDriver` реализует `IDisposable` , поэтому не нужно явно вводить `driver.Quit();` ,

Если вы не загрузили свой WebDriver с помощью NuGet , вам необходимо передать аргумент в виде пути к каталогу, в котором находится сам драйвер «chromedriver.exe».

навигационный

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

а также

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

Обе эти линии делают то же самое. Они инструктируют драйвер для перехода к определенному URL-адресу и дождаться загрузки страницы до того, как он перейдет к следующему оператору.

Существуют другие методы, связанные с навигацией, такие как `Back()` , `Forward()` или `Refresh()` .

После этого блок `using` безопасно завершает работу и удаляет объект.

ПИТОН

```
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()
```

Вышеупомянутая «программа» перейдет на главную страницу Google, а затем закроет браузер до завершения.

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

Сначала у нас есть наша основная функция, наша точка входа в программу, которая вызывает `get_google()` .

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

`get_google()` затем начинается с создания нашего экземпляра `driver` через `set_up_driver()` :

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

Здесь мы указываем, где находится `chromedriver.exe` , и `chromedriver.exe` экземпляр нашего объекта драйвера с помощью этого пути. Остальная часть `get_google()` переместится в Google:

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

А затем вызывает `tear_down()` передающую объект драйвера:

```
tear_down(driver)
```

`tear_down()` просто содержит одну строку для закрытия нашего объекта драйвера:

```
driver.quit()
```

Это говорит о том, что драйвер закрывает все открытые окна браузера и удаляет объект браузера, так как у нас нет другого кода после этого вызова, это фактически завершает работу программы.

Джава

Код ниже - всего 3 шага.

1. Открытие хромового браузера
2. Открытие страницы google
3. Выключение браузера

```
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();
    }
}
```

Вышеупомянутая «программа» перейдет на главную страницу Google, а затем закроет браузер до завершения.

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

Первая строка сообщает системе, где можно найти исполняемый файл `ChromeDriver` (chromedriver.exe). Затем мы создаем наш объект драйвера, вызывая конструктор `ChromeDriver()` , и снова мы можем вызвать наш конструктор здесь для любого браузера / платформы.

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

Это говорит нашему драйверу перейти к указанному URL- адресу : http://www.google.com . API Java WebDriver предоставляет метод `get()` непосредственно в интерфейсе WebDriver, хотя дальнейшие методы навигации можно найти с помощью метода `navigate()` , например `driver.navigate.back()` .

Как только страница закончит загрузку, мы сразу вызываем:

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

Это говорит о том, что драйвер закрывает все открытые окна браузера и удаляет объект драйвера, так как у нас нет другого кода после этого вызова, это фактически завершает работу программы.

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

Является инструкцией (не показана здесь) драйверу, чтобы закрыть только активное окно, в этом случае, поскольку у нас есть только одно окно, инструкции приведут к идентичным результатам для вызова `quit()` .

Java - Лучшая практика с классами страниц

Usecase: Войдите в учетную запись 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());
    }

}
```

Классы страниц: Вход в систему Страница и главная страница Класс входной страницы:

```
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);
    }
}
```

Класс домашней страницы:

```
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";
        }
    }

}
```

Прочитайте Программа Basic Selenium Webdriver онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/3990/программа-basic-selenium-webdriver

глава 25: Робот в селене

Синтаксис

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

параметры

параметр	подробности
Миз	Время спать в миллисекундах
ключевой код	Константа для нажатия указанной клавиши, например, для нажатия кода `A` - это `VK_A` . Пожалуйста, обратитесь за дополнительной информацией: https://docs.oracle.com/javase/7/docs/api/java/awt/event/KeyEvent.html
x, y	Координаты экрана
кнопки	Маска кнопки; комбинация одной или нескольких масок кнопки мыши
wheelAmt	Количество вырезов для перемещения колеса мыши, отрицательное значение для перемещения вверх / в сторону от положительного значения пользователя для перемещения вниз / по направлению к пользователю

замечания

В этом разделе содержится информация о реализации Robot API с Selenium Webdriver. Класс Robot используется для генерации собственного входа системы, когда селен не способен это сделать, например, нажав правую клавишу мыши, нажав клавишу F1 и т. Д.

Examples

Событие Keypress с использованием Robot API (JAVA)

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

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

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

С селеном

Иногда нам нужно нажать любую клавишу, чтобы проверить событие нажатия клавиши в веб-приложении. Для экземпляра, чтобы проверить ключ ENTER на форме входа, мы можем написать что-то вроде ниже с 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);
    }
}
```

Событие мыши с использованием Robot API (JAVA)

Движение мыши:

```
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);
     }
}
```

Нажмите левую / правую кнопку мыши:

```
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);
     }
}
```

Нажмите и прокрутите колесо:

```
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);
     }
}
```

Прочитайте Робот в селене онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/4877/робот-в-селене

глава 26: Селеновая сетка

Examples

Конфигурация узла

Конфигурация Selenium Grid Node находится на самом узле и содержит информацию о настройке сети и возможностях узла. Конфигурация может применяться различными способами:

- Настройка по умолчанию
- Конфигурация JSON
- Конфигурация командной строки

Конфигурация JSON

Конфигурация узла в файле JSON разделена на 2 раздела:

- возможности
- конфигурация

Возможности определяют такие области, как типы и версии браузера, местоположения бинарных файлов браузера, количество максимальных экземпляров каждого типа браузера.

Конфигурация имеет дело с такими настройками, как адреса узлов и узлов и портов.

Ниже приведен пример файла конфигурации 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
  }
}
```

Как создать узел

Чтобы создать узел, сначала нужно иметь концентратор. Если у вас нет концентратора, вы можете создать его так:

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

Затем вы можете создать узел:

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

Подробнее здесь: https://github.com/SeleniumHQ/selenium/wiki/Grid2

Прочитайте Селеновая сетка онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/1359/селеновая-сетка

глава 27: Скриншоты

Вступление

Выполнение скриншотов и сохранение в определенном пути

Синтаксис

- Файл src = ((TakesScreenshot))) .getScreenshotAs (OutputType.FILE);
- FileUtils.copyFile (src, новый файл ("D: \ screenshot.png"));

Examples

ДЖАВА

Код для снятия и сохранения снимка экрана:

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

Делает скриншот:

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

Магазины Скриншот из источника в пункт назначения:

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

Прочитайте Скриншоты онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/10094/

скриншоты

глава 28: Скроллинг

Вступление

В этой теме будут предложены несколько способов выполнения прокрутки с `selenium`

Examples

Прокрутка с использованием Python

1. Прокрутка к целевому элементу (кнопка «BROWSE TEMPLATES» внизу страницы) с помощью `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. Прокрутка к целевому элементу (кнопка «BROWSE TEMPLATES» внизу страницы) с помощью `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. Прокрутка к целевому элементу (кнопка «BROWSE TEMPLATES» внизу страницы) со встроенным методом

```
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
```

Обратите внимание, что `location_once_scrolled_into_view` также возвращает `x` , `y` координаты элемента после прокрутки

4. Прокрутка в нижней части страницы с помощью `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
```

Обратите внимание, что `send_keys(Keys.DOWN)` / `send_keys(Keys.UP)` и `send_keys(Keys.PAGE_DOWN)` / `send_keys(Keys.PAGE_UP)` также могут использоваться для прокрутки

Различные прокрутки с использованием java разными способами

> Ниже дать решение можно также использовать в других поддерживаемых языках программирования с некоторыми изменениями синтаксиса

---

1. **Выполнение Прокрутка вниз по** страницам / разделу / разделению на веб-странице, в то время как есть настраиваемая полоса прокрутки (не прокрутка браузера). Нажмите здесь. Для демонстрации и проверки полосы прокрутки имеет свой независимый элемент.

   В приведенном ниже коде передайте элемент полосы прокрутки и укажите точки прокрутки.

```
    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. **Выполнение прокрутки вверх** страницы / раздела / деления на веб-странице, пока есть пользовательская полоса прокрутки (не прокрутка браузера). Нажмите здесь. Для демонстрации и проверки полосы прокрутки имеет свой независимый элемент.

   В приведенном ниже коде передайте элемент полосы прокрутки и укажите точки прокрутки.

```
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. Чтобы выполнить прокрутку вниз, когда **несколько прокрутки браузера** (Встроенный браузер), и вы хотите прокрутить вниз с помощью **клавиши «Вниз»** . Нажмите здесь для демонстрации

   В приведенном ниже коде передайте свой элемент области прокрутки, например `<div>` и нажмите кнопку «вниз».

```
 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. Чтобы выполнить прокрутку вверх, когда **несколько прокрутки браузера** (Встроенный браузер), и вы хотите прокрутить вверх с помощью **клавиши UP UP** . Нажмите здесь для демонстрации

   В нижеприведенном коде передайте свой элемент области прокрутки, например `<div>` и нажмите клавишу вверх.

```
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. Чтобы выполнить прокрутку вниз, когда **несколько прокрутки браузера** (Встроенный браузер), и вы хотите прокрутить вниз с помощью **клавиши «Только стрелка вниз»** . Нажмите здесь для демонстрации

   В приведенном ниже коде передайте свой элемент области прокрутки, например `<div>` и нажмите клавишу «вниз».

```
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. Чтобы выполнить прокрутку вверх, когда **несколько прокрутки браузера** (Встроенный браузер), и вы хотите прокручивать вверх с помощью **клавиши «Только стрелка вверх»** . Нажмите здесь для демонстрации

   В приведенном ниже коде передайте свой элемент области прокрутки, например `<div>` и введите требуемый ключ.

```
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. Чтобы выполнить прокрутку вверх / вниз при **прокрутке браузера** (встроенный браузер), и вы хотите прокручивать вверх / вниз **только с фиксированной точкой** . Нажмите здесь для демонстрации

   В приведенном ниже коде передайте свою точку прокрутки. Положительное означает, что вниз и отрицательные средства прокручиваются вверх.

```
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. Чтобы выполнить прокрутку вверх / вниз при **прокрутке браузера** (встроенный браузер), и вы хотите прокручивать вверх / вниз, чтобы **сделать элемент в видимой области или динамическом прокрутке** . Нажмите здесь для демонстрации

   В приведенном ниже коде передайте свой элемент.

```
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;
    }
}
```

Примечание. Пожалуйста, проверьте свой случай и используйте методы. Если какой-либо случай отсутствует, дайте мне знать.

Прочитайте Скроллинг онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/9063/скроллинг

глава 29: Слушатели

Examples

JUnit

Если вы используете JUnit для выполнения, вы можете расширить класс `TestWatcher` :

```
public class TestRules extends TestWatcher {

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

Поэтому в своем тестовом классе вы можете просто называть его:

```
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

Использование EventFiringWebDriver . Вы можете присоединить к нему WebDriverEventListener и переопределить методы, то есть метод onException:

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

Прочитайте Слушатели онлайн: https://riptutorial.com/ru/selenium-webdriver/topic/8226/слушатели

кредиты

S. No	Главы	Contributors
1	Начало работы с 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	Selenium-webdriver с Python, Ruby и Javascript вместе с инструментом CI	Brydenr
3	Без заголовков	Abhilash Gupta, Jakub Lokša, Liam, r_D, Tomislav Nakic-Alfirevic
4	Взаимодействие с веб-элементом	Jakub Lokša, Liam, Moshisho, Siva, Sudha Velan
5	Взаимодействие с окнами браузера	Andersson, Josh, Sakshi Singla
6	Выбрать класс	Gaurav Lad, Liam
7	Выполнение Javascript на странице	Brydenr, Liam
8	Действия (Эмуляция сложных жестов пользователя)	Josh, Kenil Fadia, Liam, Priyanshu Shekhar, Tom Mc
9	Исключения в Selenium-WebDriver	Brydenr
10	Использование Selenium Webdriver с Java	r_D, the_coder
11	Использование аннотаций @FindBy в Java	Alex Wittig, Łukasz Piaszczyk

12	Конфигурация сетки селена	mnoronha, Prasanna Selvaraj, selva, Thomas
13	навигация	Andersson, Liam, Santoshsarma, viralpatel
14	Настройка / Получение размера окна браузера	Abhilash Gupta
15	Настройка Selenium e2e	Ashish Deshmukh
16	Обработка ошибок при автоматизации с использованием селена	Andersson
17	Обращение с предупреждением	Andersson, Aurasphere, Priya, SlightlyKosumi
18	Объектная модель страницы	JeffC, Josh, Moshisho, Priyanshu Shekhar, Sakshi Singla
19	Отчеты HTML	Ashish Deshmukh
20	Переключающие рамки	Andersson, dreamwork801, Jakub Lokša, Java_deep, Jim Ashworth, Karthik Taduvai, Kyle Fairns, Liam, lloyd, Priyanshu Shekhar, SlightlyKosumi
21	Перемещение между несколькими кадрами	Pavan T, Raghvendra
22	Подождите	Jakub Lokša, Josh, Kenil Fadia, Liam, Moshisho, noor, Sajal Singh, Saurav
23	Поиск веб-элементов	alecxe, daOnlyBG, Jakub Lokša, Josh, Liam, Łukasz Piaszczyk, Moshisho, NarendraR, noor, Priya, Sakshi Singla, Siva
24	Программа Basic Selenium Webdriver	Jakub Lokša, Josh, Liam, Priya, Sudha Velan, Thomas, vikingben
25	Робот в селене	Priyanshu Shekhar
26	Селеновая сетка	Eugene S, Y-B Cause
27	Скриншоты	Abhilash Gupta, Sanchit

28	Скроллинг	Andersson, Sagar007
29	Слушатели	Erki M.