Шпаргалка для проекта автотестов с использованием Selenium (пример с Chrome)

Это шпаргалка содержит в себе достаточное количество методов из библиотеки selenium Webdriver чтобы автоматизировать простой сценарий. Однако за полной документацией обратитесь на ресурс https://www.selenium.dev/documentation/webdriver/

Зависимости

Для разработки простого проекта UI автотестов потребуется минимальный набор инструментов: драйвер браузера, сам браузер и тестовый фреймворк. Вы можете использовать любой тестовый фреймворк, ниже представлены предложенные мной инструменты.

- Библиотека Selenium Webdriver
- Веб-браузер Google Chrome
- Бинарный файл вебдрайвера для требуемой версии и типа браузера, подходящий вашей системе. Хромдрайвер можно скачать с сайта хромиума https://chromedriver.chromium.org/
- Фреймворк для тестирования: Nunit для .NET, JUnit для Java, Pytest для Python

Инициализация

.NET

- Подключение библиотеки для взаимодействия с вебдрайвером

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

- Запуск (инициализация) вебдрайвера

```
this.driver = new ChromeDriver();
```

Python

- Подключение библиотеки для взаимодействия с вебдрайвером

```
from selenium import webdriver
```

- Запуск (инициализация) вебдрайвера

```
driver = webdriver.Chrome()
```

Поиск элементов на странице

.NET

Найти один элемент

- По классу `driver.FindElement(By.ClassName("main"));`

- По CSS селектору `driver.FindElement(By.CssSelector("h1"));`
- По Id элемента`driver.FindElement(By.Id("138562"));`
- По тексту гиперссылки `driver.FindElement(By.LinkText("СКАЧАТЬ БЕСПЛАТНО БЕЗ СМС!"));`
- По имени (аттрибут name) элемента `driver.FindElement(By.Name("name"));`
- По части текста гиперссылки `driver.FindElement(By.PartialLinkText("СКАЧАТЬ"));`
- По типу элемента (пример: найти поле ввода input)

```
driver.FindElement(By.TagName("input"));
```

- Поиск по XPath

```
driver.FindElement(By.XPath("//*[@id='editor']"));
```

- "Относительный" локатор

```
var emailLocator =
RelativeBy.WithLocator(By.TagName("input")).Above(By.Id("password"));
```

Найти множество элементов (можно использовать любой локатор).

```
IReadOnlyCollection<IWebElement> anchors = driver.FindElements(By.TagName("a"));
```

Возвращается коллекция элементов. Если элементов не нашлось, возвращает пустую коллекцию;

Найти элемент внутри другого элемента (можно использовать любой локатор)

```
var div = .driver.FindElement(By.TagName("div")).FindElement(By.TagName("a"));
```

Таким образом можно выстраивать цепочки из действий с элементами. Но будьте осторожны - такой подход приведет к нестабильным тестам, поскольку одна ошибка приведет к провалу всей цепи. Лучше использовать "обертки" и искать элементы более гибко

Python

По аналогии с предыдущим примером:

Найти один элемент

```
driver.find_element(By.CLASS_NAME, "main")

driver.find_element(By.CSS_SELECTOR, "h1")

driver.find_element(By.ID, "138562")

driver.find_element(By.LINK_TEXT, "СКАЧАТЬ БЕСПЛАТНО БЕЗ СМС!")

driver.find_element(By.NAME, "name")
```

```
driver.find_element(By.PARTIAL_LINK_TEXT, "СКАЧАТЬ")

driver.find_element(By.TAG_NAME, "tag name")

driver.find_element(By.XPATH, "//*[@id='editor']")
```

"Относительный" локатор `email_locator = locate_with(By.TAG_NAME, "input").above({By.ID: "password"})`

Найти множество элементов

`driver.find_elements(By.XPATH, '//button')``

Найти элемент внутри элемента

```
element = driver.find_element_by_cssselector("css=div[title='div2']")

element.find_element_by_cssselector("p[@class='test']").text
```

Взаимодействие с элементами на странице

.NET

- Кликнуть `element.Click();`
- Напечатать `element.SendKeys("любой текст, но также можно отправлять и сочетания клавиш!");`
- Нажать Enter `element.SendKeys(Keys.ENTER)`
- Очистить поле ввода `element.Clear();`
- Отображен ли элемент? `bool isDisplayed = element.Displayed;`
- Выбран ли элемент? `bool isSelected = element.Selected;`
- Прокрутить страницу до выбранного элемента

```
IWebElement link = driver.FindElement(By.PartialLinkText("СКАЧАТЬ"));

string js = string.Format("window.scroll(0, {0});", link.Location.Y);

((IJavaScriptExecutor)driver).ExecuteScript(js);
```

- Подождать какое-то время появления элемента (неявное ожидание). Не рекомендуется. Слишком медленно, и неясно будет ли работать каждый раз.

```
System.Threading.Thread.Sleep(5000);
```

- Подождать появления элемента на странице

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(30));
```

```
wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.XPath("//*[@id='popup']/div[1]/input")));
```

Хорошей практикой является написание оберток для методов ожиданий. Таким образом, уменьшается количество повторений кода и обернутые ожидания можно настроить под конкретное веб приложение, что удобно.

Python

Аналогично,

- Кликнуть `button_element.click()`
- Напечатать `element.send_keys("любой текст, но также можно отправлять и сочетания клавиш!")`
- Нажать кнопку `element.sendKeys(Keys.RETURN);`
- Очистить поле ввода `element.clear()`
- Проверить видимость `element.is_displayed()`
- Прокрутить страницу до элемента `driver.execute_script("arguments[0].scrollIntoView();", element)`
- Подождать элемент неявно (требует импорта `import time`)

```
time.sleep(2.5)
```

- Подождать элемент явно (сперва требует импорт `from selenium.webdriver.support import expected_conditions as EC`)

```
WebDriverWait(browser, 20).until(EC.element_to_be_clickable((By.CSS_SELECTOR, ".reply-button"))).click()
```

Прочие действия в браузере

.NET

- Перейти по URL адресу `driver.Navigate().GoToUrl(@"http://google.com");`
- Перейти назад `driver.Navigate().Back();`
- Обновить страницу `driver.Navigate().Refresh();`
- Перейти вперед `driver.Navigate().Forward();`
- Развернуть окно на весь экран `driver.Manage().Window.Maximize();`
- Переключиться на другой фрейм

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

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

Python

- Перейти по url `driver.get("https://www.google.com/")`
- Обновить страницу `driver.refresh()`
- Вперед `driver.forward()`
- Назад `driver.back()`
- Во весь экран`driver.maximize_window()`
- Переключиться на другой фрейм `driver.switch_to.frame(driver.find_element_by_name(name))`

XPath выражения

`//a` - двойная дробь говорит о том что элемент находится в любом месте DOM дерева

`//div/a` - здесь `a` прямой потомок элемента `div`

`//div//a` - здесь `a` *может быть* прямым или непрямым потомком элемента `div`

`//*` - `*` это обозначение "wildcard" элемента, т.е. любого элемента.

`//input[@type=submit]` - `@` обозначает аттрибут элемента. У элемента может быть множество элементов

Таблица ниже показывает как можно найти элементы по признаку вложенности в другие элементы.

Дочерние элементы	**Их селекторы**
h1	//h1
div p	//div//p
ul > li	//ul/li
ul > li > a	//ul/li/a
div > *	//div/*
:root	/
:root > body	/body

Таблица ниже показывает как найти элемент по аттрибуту, например его id или классу

Аттрибуты	**Их Селекторы**
#id	//*[@id="12345"]
.class	//*[@class="main"]
input[type="submit"]	//input[@type="submit"]
a#abc[for="xyz"]	//a[@id="abc"][@for="xyz"]

Таблица ниже показывает как найти какой-либо элемент по порядку, если их несколько. Например, первый, последний, или n-ный

Порядок элементов	Селекторы
ul > li:first-of-type	//ul/li[1]
ul > li:nth-of-type(2)	//ul/li[2]
ul > li:last-of-type	//ul/li[last()]
li#id:first-of-type	//li[1][@id="id"]
a:first-child	//*[1][name()="a"]
a:last-child	//*[last()][name()="a"]

Таблица ниже показывает как можно использовать встроенные функции для поиска содержимого элементов, например классов или текста

Операции с текстом	Селекторы
contains()	//div[contains(@class,"head")]
text()	//div[contains(text(),'любой текст')]
starts-with()	//div[starts-with(@class,"head")]
ends-with()	//div[ends-with(@class,"head")]

Для более полного ознакомления с выражениями XPath можно обратиться к различным cheatshees вроде https://devhints.io/xpath. Полная спецификация находится на https://www.w3.org/TR/2017/REC-xpath-31-20170321/