Selenium - Команды

В Selenium есть несколько основных команд:

- Actions
- Accessors
- Assertions

Локаторы

Элемент Locators помогают Selenium идентифицировать элемент HTML, на который ссылается команда. Все эти локаторы можно идентифицировать с помощью плагина FirePath и FireBug Mozilla.

- **identifier = id** Выберите элемент с указанным атрибутом «id», а если нет совпадения, выберите первый элемент, атрибут @name которого - id.
- **id = id** Выберите элемент с указанным атрибутом id.
- **name = name** Выберите первый элемент с указанным атрибутом name
- **dom = javascriptExpression** Selenium находит элемент, оценивая указанную строку, которая позволяет нам перемещаться по объектной модели документа HTML с использованием JavaScript. Пользователи не могут вернуть значение, но могут оцениваться как выражение в блоке.
- **xpath = xpathExpression** Найдите элемент, используя выражение XPath.
- **link = textPattern** Выберите элемент ссылки (в тегах привязки), который содержит текст, соответствующий указанному шаблону.
- **css = cssSelectorSyntax** Выберите элемент с помощью селектора css.

Actions

Действия - это команды, которые управляют состоянием приложения. После выполнения, если действие не выполняется, выполнение текущего теста прекращается.

Команда	Описание
click (locator)	Щелчок по ссылке, кнопке, флажку или переключателю
clickAt (locator,coordString)	Щелчок по элементу с помощью локатора и координат
close()	Имитирует пользователя, нажимая кнопку «закрыть» в строке заголовка всплывающего окна или вкладки.

contextMenuAt (locator,coordString)	Имитирует открытие контекстного меню указанного элемента из указанного местоположения
doubleClick (locator)	Двойной щелчок на веб-элементе на основе указанного элемента.
dragAndDrop (locator,movementsString)	Перетаскивает элемент, а затем отбрасывает его на заданное расстояние.
dragAndDropToObject (Dragobject,dropobject)	Перетаскивает элемент и отбрасывает его на другой элемент.
echo (message)	Распечатывает указанное сообщение на консоли, которое используется для отладки.
fireEvent (locator,eventName)	Явно имитируйте событие, чтобы вызвать соответствующий обработчик «onevent»
focus (locator)	Перемещение фокуса на указанный элемент
highlight (locator)	Изменяет цвет фона указанного элемента на желтый. Коротко, что полезно для целей отладки.
mouseDown (locator)	Имитирует пользователя, нажимая левую кнопку мыши на указанном элементе.
mouseDownAt (locator,coordString)	Имитирует пользователя, нажимая левую кнопку мыши в указанном месте на указанном элементе.
mouseUp (locator)	Имитирует событие, которое происходит, когда пользователь отпускает кнопку мыши
mouseUpAt (locator,coordString)	Имитирует событие, которое происходит, когда пользователь отпускает кнопку мыши в указанном месте.
open (url)	Открывает URL-адрес в указанном браузере и принимает как относительные, так и абсолютные URL-адреса.
openWindow (url,windowID)	Открывает всплывающее окно. После открытия окна пользователю необходимо активировать его с помощью команды selectWindow.
pause (waitTime)	Ожидает заданное время (в миллисекундах)
refresh()	Имитирует пользователя, нажимая кнопку «Обновить» в своем браузере.
select (selectLocator,optionLocator)	Выберите опцию из раскрывающегося списка с помощью локатора параметров.
selectWindow (windowID)	Выбирает всплывающее окно с помощью локатора окон; как только всплывающее окно выбрано, все фокусы сдвигаются в это окно.

store (expression,variableName)	Имя переменной, в которой должен быть сохранен результат, и выражение - это значение для хранения
type (locator,value)	Устанавливает значение поля ввода, аналогичное действию ввода пользователя.
typeKeys (locator,value)	Имитирует события нажатия клавиш на указанном элементе, как если бы вы набрали значение key-by-key.
waitForCondition (script,timeout)	Выполняет указанный фрагмент JavaScript несколько раз, пока не будет оценен «true».
waitForPageToLoad (timeout)	Ожидает загрузки новой страницы.
waitForPopUp (windowID,timeout)	Ожидает появления и загрузки всплывающего окна.
windowFocus()	Придает фокус выбранному окну
windowMaximize()	Изменить размер выбранного окна, чтобы отобразить весь экран

Accessors

Аксессоры оценивают состояние приложения и сохраняют результаты в переменной, которая используется в утверждениях.

Команда	**Описание**
assertErrorOnNext (message)	Pings Selenium ожидает ошибку при следующем выполнении команды с ожидаемым сообщением.
storeAllButtons (variableName)	Возвращает идентификаторы всех кнопок на странице.
storeAllFields (variableName)	Возвращает идентификаторы всех полей ввода на странице.
storeAllLinks (variableName)	Возвращает идентификаторы всех ссылок на странице.
storeAllWindowIds (variableName)	Возвращает идентификаторы всех окон, о которых браузер знает в массиве.
storeAllWindowTitles (variableName)	Возвращает имена всех окон, о которых обозреватель знает в массиве.
storeAllWindowNames (variableName)	Возвращает названия всех окон, о которых обозреватель знает в массиве.

storeAttribute (attributeLocator, variableName)	Возвращает значение атрибута элемента. Значение атрибута может отличаться в разных браузерах.
storeBodyText (variableName)	Получает весь текст страницы.
storeConfirmation (variableName)	Извлекает сообщение диалогового окна подтверждения JavaScript, сгенерированного во время предыдущего действия.
storeElementIndex (locator, variableName)	Получить относительный индекс элемента к его родительскому (начиная с 0).
storeLocation (variableName)	Получает абсолютный URL текущей страницы.
storeSelectedIds (selectLocator,variableName)	Получает все идентификаторы элементов для выбранных параметров в указанном элементе select или multi-select.
storeSelectedIndex (selectLocator, variableName)	Возвращает индекс (номер опции, начиная с 0) для выбранного параметра в указанном элементе выбора.
storeSelectedLabel (selectLocator, variableName)	Получает ярлык (видимый текст) для выбранного параметра в указанном элементе выбора.
storeSelectedValue (selectLocator,variableName)	Возвращает значение (атрибут value) для выбранного параметра в указанном элементе select.
storeSelectOptions (selectLocator,variableName)	Получает все метки в указанном выпадающем списке.
storeTable (tableCellAddress, variableName)	Получает текст из ячейки таблицы. Синтаксис cellAddress: tableLocator.row.column, где строка и столбец начинаются с 0.
storeText (locator, variableName)	Возвращает текст элемента. Это работает для любого элемента, содержащего текст.
storeTitle (variableName)	Возвращает заголовок текущей страницы.
storeValue (locator,variableName)	Получает (обрезанное пробелами) значение поля ввода.
storeChecked (locator, variableName)	Возвращает, включена ли кнопка переключения (флажок / радио).
storeElementPresent (locator, variableName)	Проверяет, что указанный элемент находится где-то на странице.

storeTextPresent (pattern, variableName)	Проверяет, что указанный текстовый шаблон отображается где-то на отображаемой странице, отображаемой пользователю.
storeVisible (locator, variableName)	Определяет, является ли указанный элемент видимым.

Assertions

Утверждения позволяют нам проверять состояние приложения и сравнивать с ожидаемым. Он используется в трех режимах, а именно: - утверждать (assert), проверять (verify) и ждать (waitfor).

Команда	**Описание**
waitForErrorOnNext (message)	Ожидает ошибки; используется с accessor assertErrorOnNext.
verifySelected(selectLocator, optionLocator)	Проверяет, что выбранная опция раскрывающегося списка удовлетворяет параметруSpecifier.
waitForSelected (selectLocator, optionLocator)	Ожидает получения выбранной опции; используется с аксессуаром assertSelected.
waitForNotSelected (selectLocator, optionLocator)	Ожидает, что выбор не будет выбран; используется с аксессуаром assertSelected.
verifyAlert (pattern)	Проверяет текст предупреждения; используется с аксессуаром storeAlert.
waitForAlert (pattern)	Ожидает оповещения; используется с хранилищем аксессуаров.
verifyAllButtons (pattern)	Проверяет кнопку; используется с аксессуарами storeAllButtons.
waitForAllButtons (pattern)	Ожидает нажатия кнопки; используется с аксессуарами storeAllButtons.
verifyAllLinks (pattern)	Проверяет все ссылки; используется с хранилищем AccessAllLinks.
waitForAllLinks (pattern)	Ожидает всех ссылок; используется с хранилищем AccessAllLinks.
verifyAllWindowIds (pattern)	Проверяет идентификатор окна; используется с хранилищем AccessAllWindowIds.

waitForAllWindowIds (pattern)	Ожидает идентификатор окна; используется с хранилищем AccessAllWindowIds.
verifyAttribute(attributeLocator, pattern)	Проверяет атрибут элемента; используется с атрибутом storeAttribute.
waitForAttribute(attributeLocator, pattern)	Ожидает атрибут элемента; используется с атрибутом storeAttribute.
verifyBodyText(pattern)	Проверяет основной текст; используется с хранилищем-хранилищемBodyText.
waitForBodyText(pattern)	Ожидает текст тела; используется с хранилищем-хранилищемBodyText.
waitForConfirmation(pattern)	Ожидает подтверждения; используется с хранилищем аксессуаровConfirmationPresent.

Команды WebDriver

В следующей таблице перечислены некоторые из наиболее часто используемых команд в WebDriver вместе с их синтаксисом.

Команда	**Описание**
driver.get("URL")	Чтобы перейти к приложению.
element.sendKeys("inputtext")	Введите текст в поле ввода.
element.clear()	Очистите содержимое из окна ввода.
select.deselectAll()	Отмените выбор всех ОПЦИЙ из первого SELECT на
select.selectByVisibleText("some text")	Выберите OPTION с вводом, указанным пользовате
driver.switchTo().window("windowName")	Переместите фокус из одного окна в другое.
driver.switchTo().frame("frameName")	Качели от кадра к кадру.
driver.switchTo().alert()	Помогает в обработке предупреждений.
driver.navigate().to("URL")	Перейдите к URL-адресу.

driver.navigate().forward()	Перемещение вперед.
driver.navigate().back()	Чтобы вернуться назад.
driver.close()	Закрывает текущий браузер, связанный с драйверо
driver.quit()	Выключает драйвер и закрывает все связанные окн драйвера.
driver.refresh()	Обновляет текущую страницу.

Selenium - Locators

Расположение элементов в Selenium WebDriver выполняется с помощью методов `findElement()` и `findElements()`, предоставляемых классами `WebDriver` и `WebElement`.

- `findElement()` возвращает объект WebElement на основе заданного критерия поиска или заканчивает выдачу исключения, если он не находит элемент, соответствующий критериям поиска.
- `findElements()` возвращает список WebElements, соответствующий критериям поиска. Если элементы не найдены, он возвращает пустой список.

В следующей таблице представлен весь синтаксис для размещения элементов в Selenium WebDriver.

Метод	Синтаксис	Описание
По идентификатору	`driver.findElement(By.id (<element ID>))`	Определяет элемент, использующий атрибут ID
По имени	`driver.findElement(By.name (<element name>))`	Располагает элемент, используя атрибут Name
По имени класса	`driver.findElement(By.className (<element class>))`	Располагает элемент, используя атрибут Class
По имени тега	`driver.findElement(By.tagName (<htmltagname>))`	Располагает элемент, используя тег HTML
По тексту ссылки	`driver.findElement(By.linkText (<linktext>))`	Определяет ссылку, используя текст ссылки

Посредством частичного текста ссылки	`driver.findElement(By.partialLinkText (<linktext>))`	Определяет местонахождение ссылки с использованием частичного текста ссылки
По CSS	`driver.findElement(By.cssSelector (<css selector>))`	Располагает элемент с помощью селектора CSS
По XPath	`driver.findElement(By.xpath (<xpath>))`	Располагает элемент с помощью запроса XPath

Selenium - User Interactions

Selenium WebDriver является наиболее часто используемым инструментом среди всех инструментов, доступных в наборе инструментов Selenium. Поэтому важно понять, как использовать Selenium для взаимодействия с веб-приложениями. В этом модуле давайте разобраться, как взаимодействовать с объектами графического интерфейса, используя Selenium webDriver.

Нам нужно взаимодействовать с приложением, используя некоторые базовые действия или даже некоторые продвинутые действия пользователя, разрабатывая пользовательские функции, для которых нет предопределенных команд.

Text Box Interaction

Мы можем поместить значения в текстовое поле, используя метод `sendkeys`. Аналогичным образом, мы также можем извлекать текст из текстового поля, используя команду `getattribute (value)`.

Radio Button Selection

Мы можем выбрать опцию переключателя с помощью метода `click` и отменить выбор, используя тот же метод `click`.

Check Box Selection

Мы можем выбрать флажок с помощью метода `click` и снять отметку с помощью одного и того же метода `click`.

Drop Down Item Selection

Мы можем выбрать вариант с помощью
методов `selectByVisibleText` или `selectByIndex` или `selectByValue`.

Synchronization

Чтобы синхронизировать выполнение сценария и приложения, нам нужно подождать после выполнения соответствующих действий. Thread.Sleep - это статическое ожидание, и это не очень хороший способ использования в скриптах, поскольку это сон без каких-либо условий.

```
Thread.Sleep(7000);
```

Явное ожидание - `explicit wait` ожидает определенного условия, прежде чем продолжить. Он используется главным образом, когда мы хотим щелкнуть или действовать на объект, как только он станет видимым.

Неявное ожидание - `implicit wait` используется в тех случаях, когда WebDriver не может найти объект немедленно из-за его недоступности. WebDriver будет ожидать указанное неявное время ожидания, и он не попытается найти элемент снова в течение указанного периода времени.

Как только указанный срок будет скрещен, webDriver попытается снова выполнить поиск элемента в последний раз. После успеха он выполняет исполнение; при сбое он выдает исключение.

Это своего рода глобальное ожидание, которое означает, что ожидание применимо для всего драйвера. Следовательно, жесткое кодирование этого ожидания для более длительных периодов времени будет препятствовать времени выполнения.

Свободное время ожидания - экземпляр `FluentWait` определяет максимальный промежуток времени для ожидания выполнения условия, а также частоту проверки существования условия объекта.

Keyboard Actions

Ниже приведены методы выполнения действий клавиатуры:

- **sendKeys** - Отправляет ключи в представление клавиатуры в браузере. Специальные клавиши, которые не являются текстом, представленными как Ключи, распознаются как часть последовательностей символов, так и индивидуально.
- **pressKey** - нажмите клавишу на клавиатуре, которая не является текстом. Клавиши, такие как функциональные клавиши «F1», «F2», «Tab», «Control» и т. Д. Если keyToPress представляет собой последовательность символов, разные реализации драйверов могут выбрасывать исключение или читать только первый символ в последовательность.

- **releaseKey** - отпустите клавишу на клавиатуре после выполнения события нажатия клавиши. Как правило, это удобно для текстовых символов.

Mouse Actions

Ниже перечислены некоторые из ключевых действий мыши, которые можно встретить в большинстве приложений:

- **Click** - Выполняет щелчок. Мы также можем выполнить щелчок на основе координат.
- **contextClick** - выполняет контекст щелчком / щелчком правой кнопки мыши по элементу или на основе координат
- **doubleClick** - выполняет двойной щелчок на веб-элементе или на основе координат. Если он оставлен пустым, он выполняет двойной щелчок по текущему местоположению.
- **mouseDown** - Выполняет действие мыши по элементу или на основе координат.
- **mouseMove** - Выполняет действие перемещения мыши на элементе или на основе координат.
- **mouseUp** - освобождает мышь, а затем мышь-вниз и действует на основе координат.