Способы поиска элементов при работе с Selenium Webdriver

Когда вы открываете веб-страницу в браузере, он получает исходный текст HTML и разбирает (парсит) его . Браузер строит модель структуры документа и использует её, чтобы нарисовать страницу на экране.
Это представление документа и есть одна из игрушек, доступных в песочнице JavaScript. Вы можете читать её и изменять. Она изменяется в реальном времени – как только вы её подправляете, страница на экране обновляется, отражая изменения.
Структура данных, использующаяся браузером для представления документа, отражает его форму. Для каждой коробки есть объект, с которым мы можем взаимодействовать и узнавать про него разные данные – какой тег он представляет, какие коробки и текст содержит. Это представление называется Document Object Model (объектная модель документа), или сокращённо DOM.

DOM (от англ. Document Object Model — «объектная модель документа») — это независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому HTML-, XHTML- и XML-документов, а также изменять содержимое, структуру и оформление таких документов.

Поиск элементов по локаторам

Поскольку Webdriver - это инструмент для автоматизации веб приложений, то большая часть работы с ним это работа с веб элементами (WebElements). **WebElements** - ни что иное, как DOM объекты, находящиеся на веб странице. А для того, чтобы осуществлять какие-то действия над DOM объектами / веб елементами необходимо их точным образом определить(найти)

1.Поиск по локатору By.Id:

Пример:

```
<div id="">
     <p>some content</p>
</div>
```

Форма записи для поиска элемента:

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

2.Поиск по локатору By.name:

Пример:

```
<div name="">
     <p>some content</p>
</div>
```

Форма записи для поиска элемента:

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

3. Поиск элемента по локатору By.className:

Пример:

```
<img class="">
```

Форма записи для поиска элемента:

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

4. By.TagName:

Пример:

```
<div>
     <a class="logo" ref="...">...</a>
     <a class="support" ref="...">...</a>
</div>
```

   :

```
List<WebElement> elements = driver.findElement(By.tagName("a")); //
List<WebElement> elements = driver.findElements(By.tagName("a")); //
```

5. By.LinkText:

Пример:

```
<div>
     <a ref="...">text</a>
     <a ref="...">Another text</a>
</div>
```

Форма записи для поиска элемента:

```
WebElement element = driver.findElement(By.linkText("text"));
```

CSS локаторы и XPath локаторы

Ранее мы уже изучали локаторы, с помощью которых можно идентифицировать элементы на web странице. В этой части мы пристальнее рассмотрим наиболее сложные из них, а именно CSS, Xpath.

CSS локаторы

Многие браузеры реализуют CSS движок, чтобы разработчики смогли применять CSS таблицы в своих проектах. Что позволяет разделить между собой контент страницы с её оформлением. В CSS есть паттерны, согласно которым стили, создаваемые разработчиком, применяются к элементам страницы (DOM). Эти паттерны называются локаторы (selectors). Selenium WebDriver использует тот же принцип для нахождения элементов. И он намного быстрее, чем поиск элементов на основе XPath, который мы рассмотрим чуть позже.

Существует несколько способов построения CSS локатора, но самый простой это установить специальные расширения в ваш браузер(например Ranorex Selocity):

```
WebElement userName = driver.findElement(By.cssSelector(".home-link.home-logo__link > div[role='img']"));
```

Так же можно построить CSS локатор на основе одного из параметров элемента, по которому нет поиска среди обычных локаторов :

```
WebElement userName = driver.findElement(By.cssSelector("[role='img']"));
```

XPath локаторы

XPath (XML path) - это язык для нодов (nodes) в XML документе. Так как многие браузеры поддерживают XHTML, то мы можем использовать XPath для нахождения элементов на web страницах.

Xpath можно так же получить с помощью различных расширений, или построить самому. Так же важно знать, что по XPath можно искать в 2 режимах: по всей странице(указав абсолютный путь) или среди потомков указанного элемента(указав //).

```
//    a   "link"  :
driver.findElement(By.id("some-id")).findElement(By.xpath("a[@class='link']"));

//    a   "link"     id="some-id":
driver.findElement(By.id("some-id")).findElement(By.xpath(".//a[@class='link']"));
```

Что лучше использовать?

Здесь каждый решает сам, ведь есть плюсы и минусы у каждого. Например у CSS нет возможности искать среди потомков, но зато по исследованиям тесты с CSS локаторами быстрее выполняются. Так же стоит заметить, что бывают случаи когда элемент не удается найти по CSS локатору и тогда приходится использовать XPath.