Selenium

Петухов Илья (МГТУ)
iproha94@gmail.com

Что такое Selenium?

- набор инструментов, предназначенных для автоматизации веб браузеров на различных платформах:

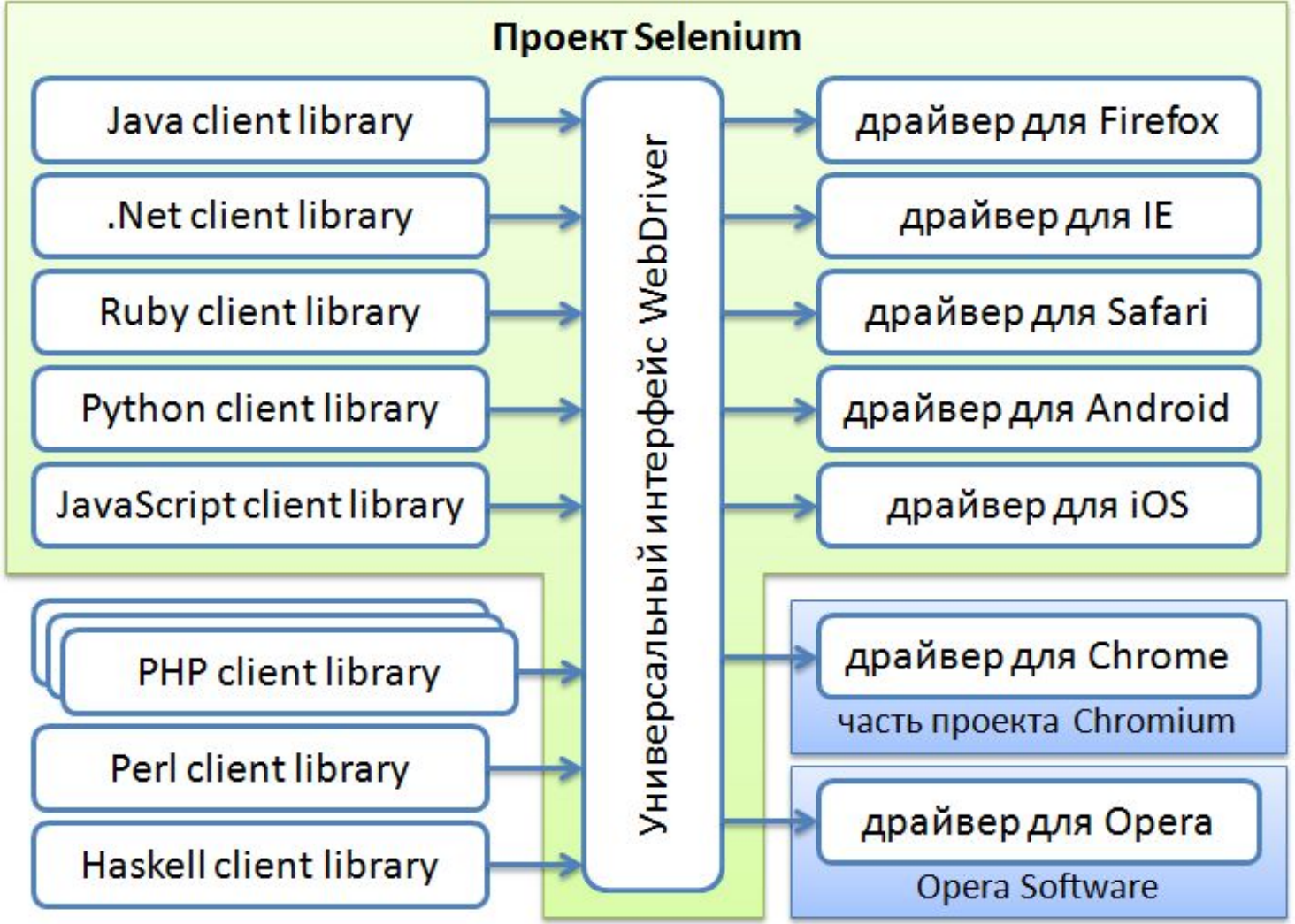
- Selenium RC
- Selenium WebDriver
- Selenium Server
- Selenium Grid
- Selenium IDE

Selenium RC

Эта версия с функциональной точки зрения значительно уступает WebDriver. Сейчас она находится в законсервированном состоянии, не развивается и даже известные баги не исправляются. А всем, кто сталкивается с ограничениями Selenium RC, предлагается переходить на использование WebDriver.

Selenium WebDriver

- может автоматизировать множество разнообразных браузеров на разных платформах, используя различные языки программирования


Проект Selenium
Java client library
.Net client library
Ruby client library
Python client library
JavaScript client library
Универсальный интерфейс WebDriver
драйвер для Firefox
драйвер для IE
драйвер для Safari
драйвер для Android
драйвер для iOS
PHP client library
Perl client library
Haskell client library
драйвер для Chrome
часть проекта Chromium
драйвер для Opera
Opera Software

Selenium WebDriver

WebDriver – это драйвер браузера, то есть не имеющая пользовательского интерфейса программная библиотека, которая позволяет различным другим программам взаимодействовать с браузером, управлять его поведением, получать от браузера какие-то данные и заставлять браузер выполнять какие-то команды.

=>WebDriver не имеет прямого отношения к тестированию

“Мне вообще безразлично, кто и зачем хочет управлять браузером”

© Selenium W.D.

Selenium WebDriver

- автоматизировать какие-то рутинные задачи
- сделать ботов, которые будут флудить в форумах
- сделать скрипт, который автоматически снимает скриншоты для документации
- автоматизация тестирования GUI

Для работы с Webdriver необходимо 3 основных программных компонента:

1. Браузер, работу которого пользователь хочет автоматизировать.
2. Для управления браузером совершенно необходим driver браузера. Driver запускает браузер и отправляет ему команды, а также закрывает его.
3. Скрипт/тест, который содержит набор команд на определённом языке программирования для драйвера браузера.

Важная сущность №1 webdriver (создание)

- самая важная сущность, ответственная за управление браузером. Основной ход скрипта/теста строится именно вокруг экземпляра этой сущности.

```
let webdriver = require('selenium-webdriver');
let driver = new webdriver
    .Builder()
    .forBrowser('chrome')
    .build();
```

Важная сущность №1 webdriver (возможности)

- driver.get(“www.yandex.ru”);
- driver.findElement({важная сущность №3})
- driver.findElements({важная сущность №3})
- driver.quit();

Важная сущность №2: Webelement

- вторая важная сущность, представляющая собой абстракцию над веб-элементом (кнопки, ссылки, поля ввода и др.). Webelement инкапсулирует методы для взаимодействия пользователя с элементами и получения их текущего статуса.

- driver.findElement({важная сущность №3})
  .sendKeys(“iu7-71”);
- driver.findElement({важная сущность №3})
  .click()

Важная сущность №3: By

- абстракция над локатором веб-элемента. Этот класс инкапсулирует информацию о селекторе(например, CSS), а также сам локатор элемента, то есть всю информацию, необходимую для нахождения нужного элемента на странице.

- By.className("button_indent")
- By.id('team-name')

Ошибка:

- Uncaught NoSuchElementError:

  Unable to locate element:

  *[id="my-btn-id"]

Решение:

- Явные ожидания
- Неявные ожидания

Явные ожидания

- используется "здесь и сейчас" на один конкретный поиск элемента.
- худший пример такого кода — использование команды time.sleep()

```
var myBtn = driver.wait(
    webdriver.until.elementLocated(By.id('id-btn')),
    10000
);

myBtn.click();
```

Неявные ожидания

- устанавливается один раз в коде вне операции поиска и действительно до изменения.
- Значение по умолчанию - 0, т.е. никакого ожидания.
- применяется ко всем последующим операциям поиска неявно (т.е. скрытно, без указания напрямую в методе поиска, как мы видели в примере выше).

```
driver.manage().timeouts().implicitlyWait(5000);
```

“Абсолютно не претендую на то, что мой совет лучший. Но зато он - простой. Используйте всегда 4 секунды ImplicitlyWait, не дожидаясь проблем. А остальное - только, если понимаете зачем.”

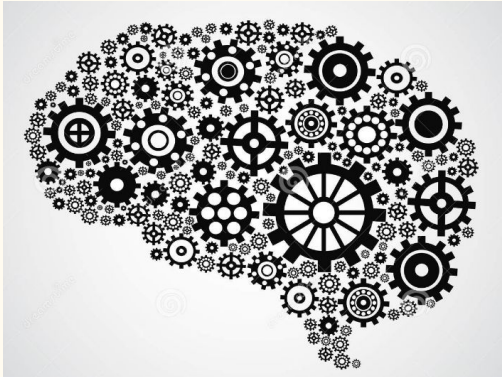
- Какой-то человек из Интернета.

Page Object Pattern

- один из наиболее полезных и используемых архитектурных решений в автоматизации. Данный шаблон проектирования помогает инкапсулировать работу с отдельными элементами страницы, что позволяет уменьшить количество кода и упростить его поддержку.

Основные преимущества:

- разделение кода тестов и описания страниц
- объединение всех действий по работе с веб-страницей в одном месте



Page Object Pattern (пример)

```
let mainPage = require('./pages/mySearcher).mainPage;
mainPage = new mainPage(driver);

mainPage.open();
mainPage.fillQuery("котики");
mainPage.search();
let arr = mainPage.getMisspellMessages();

assert.isTrue(arr.length > 0);
```

Демонстрация работы

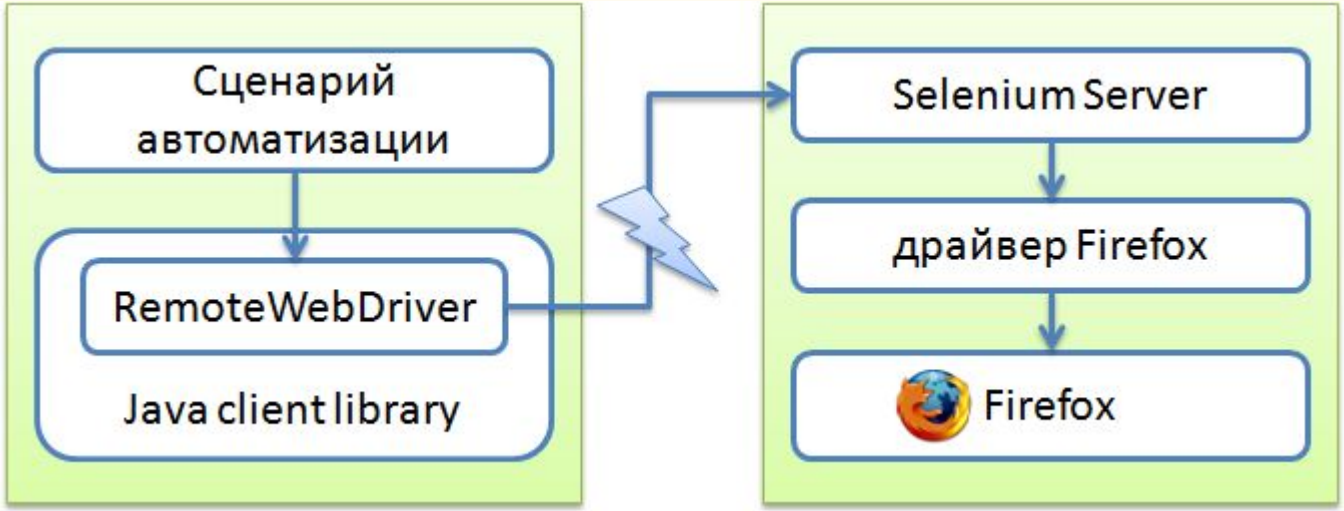
Selenium Server

- позволяет управлять браузером с удалённой машины, по сети

---

1. на той машине, где должен работать браузер, устанавливается и запускается сервер
2. на другой машине запускается программа, которая, используя специальный драйвер RemoteWebDriver, соединяется с сервером и отправляет ему команды
3. сервер в свою очередь запускает браузер и выполняет в нём эти команды, используя драйвер, соответствующий этому браузеру

Selenium Server



Selenium Grid

- это кластер, состоящий из нескольких Selenium-серверов

- для организации распределённой сети, позволяющей параллельно запускать много браузеров на большом количестве машин

Selenium IDE

- плагин к браузеру Firefox

- записывать действия пользователя
- воспроизводить их
- генерировать код для WebDriver
- выполняются те же самые действия

Ссылки

http://www.seleniumhq.org/

https://seleniumhq.github.io/selenium/docs/api/javascript/

https://comaqa.gitbooks.io/selenium-webdriver-lectures/

Спасибо за внимание!