Selenium – что такое?

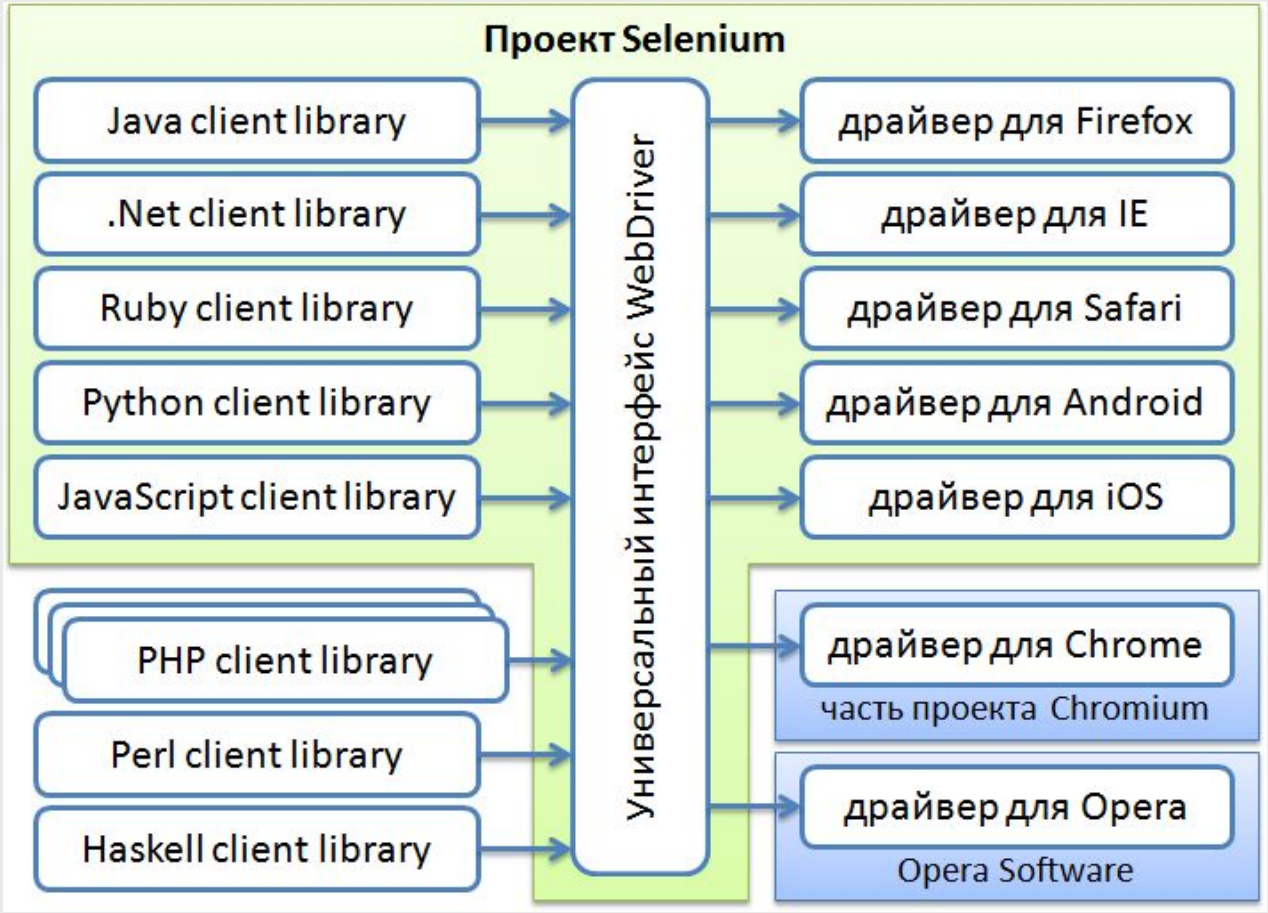
Selenium – это проект, в рамках которого разрабатывается серия программных продуктов с открытым исходным кодом (open source):
Selenium WebDriver,

- Selenium WebDriver,
- Selenium RC,
- Selenium Server,
- Selenium Grid,
- Selenium IDE.

Selenium WebDriver

Selenium WebDriver – это программная библиотека для управления браузерами. Часто употребляется также более короткое название WebDriver.

Это основной продукт, разрабатываемый в рамках проекта Selenium.

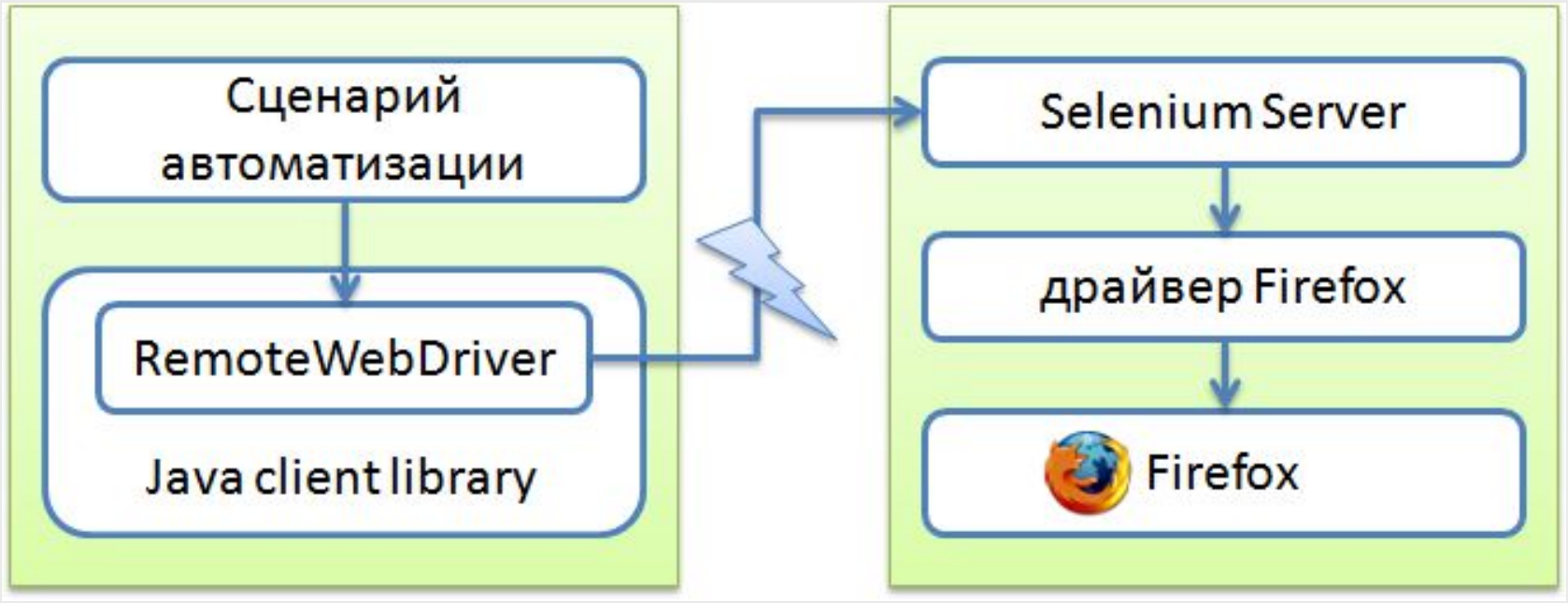


Selenium RC

Selenium RC – это предыдущая версия библиотеки для управления браузерами. Аббревиатура RC в названии этого продукта расшифровывается как Remote Control, то есть это средство для «удалённого» управления браузером.

Selenium Server

Selenium Server – это сервер, который позволяет управлять браузером с удалённой машины, по сети.



Selenium Grid

Selenium Grid – это кластер, состоящий из нескольких Selenium-серверов. Он предназначен для организации распределённой сети, позволяющей параллельно запускать много браузеров на большом количестве машин.

Selenium IDE

Selenium IDE – плагин к браузеру Firefox, который может записывать действия пользователя, воспроизводить их, а также генерировать код для WebDriver или Selenium RC, в котором выполняются те же самые действия. В общем, это «Selenium-рекордер».

Что такое Selenium WebDriver?

По назначению Selenium WebDriver представляет собой драйвер браузера, то есть программную библиотеку, которая позволяет разрабатывать программы, управляющие поведением браузера.

По своей сущности Selenium WebDriver представляет собой:

- **спецификацию программного интерфейса для управления браузером**,
- **референсные реализации этого интерфейса для нескольких браузеров**,
- **набор клиентских библиотек для этого интерфейса на нескольких языках программирования**.

Что такое Selenium WebDriver?

Selenium WebDriver, или просто 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.

Protractor

Protractor – **e2e** тест-фреймворк сделанный на основе WebDriverJS, который приходит на смену первоначальным e2e тестам **AngularJS**. Запускает тесты вреальном браузере. Может быть запущен как самостоятельный бинарник либо включен в тесты как библиотека.

Установка и запуск Protractor

Для установки достаточно установить npm-пакет:

```
npm install -g protractor
```

чтобы установить и запустить **Selenium**, который будет выполнять тесты, делаем:

```
webdriver-manager update
```

а потом:

```
webdriver-manager start
```

Для запуска **Protractor** делаем:

```
protractor conf.js
```

Файл конфигурации Protractor

```
exports.config = {
    // Адрес запускаемого selenium server
    seleniumAddress: 'http://localhost:4444/wd/hub',
    // Настройки экземпляра webdriver
    capabilities: {
        'browserName': 'chrome'
    },
    // Перечисляем пути к тестовым сценариям для выполнения
    specs: [`tests/my_spec.js'],
    // Опции Jasmine
    jasmineNodeOpts: {
        showColors: true,
        defaultTimeoutInterval: 30000
    }
};
```

Тестовый сценарий

```
describe('angularjs homepage', function() {
    it('should greet the named user', function() {
        browser.get('http://www.angularjs.org');
        element(by.model('yourName')).sendKeys('User');
        var greeting = element(by.binding('yourName'));
        expect(greeting.getText()).toEqual('Hello User!');
    });
});
```

Настройка конфигурации

ОПЦИЯ	ОПИСАНИЕ	ЗНАЧЕНИЕ ПО УМОЛЧАНИЮ
seleniumAddress	адрес запущенного селениум сервера (обычно http://localhost:4444/wd/hub)	null
allScriptsTimeout	таймаут для выполнения всех сценариев	11000
specs	пути к файлам сценариев тестов (относительно конфига)	['spec/*_spec.js']
exclude	исключения для предыдущего пункта	[]
capabilities	выбор браузера с параметрами. Более подробно тут	{'browserName': 'chrome'}
multiCapabilities	предыдущая опция для запуска тестов в нескольких браузерах	[]
baseUrl	стартовая страница приложения	http://localhost:8000
rootElement	элемент на котором инициализированно приложение (ng-app)	body
onPrepare	колбэк который будет выполнен, когда протрактор готов к работе, но тесты еще не начали выполняться	function() {}
params	параметры, которые будут внедрены в среду выполнения тестов (но не сами тесты)	{login: { user: 'Jane', password: '1234'}},
framework	фреймворк для тесто. возможные варианты: jasmine, cucumber, mocha	jasmine
onCleanUp	колбэк, когда тесты завершены	function(){}

Написание тестов

По умолчанию используется Jasmine фреймворк.

Глобальные переменные, которые добавляет **протрактор**:

- **protractor** – нэймспэйс-оболочка протрактора, которая содержит статические вспомогательные переменный и классы
- **browser** – оболочка вебдрайвера, используется для навигации и получение информации о странице
- **element** – вспомогательная функция для нахождения и взаимодействия с элементами
- **by** – коллекция стратегий поиска элементов (ccs selector, id, binding attribute)

Написание тестов

Основные методы-помощники:

- browser.**get**(targetUrl) – переход на указанный URL
- element(**by.css**(‘.error’)) – выбор элемента по css
- element(**by.model**(‘modelName’)) – выбор элемента по модели
- element(**by.binding**(‘variableName’)) – выбор элемента по баиндингу (*ng-bind*или *{{variableName}}*)
- element.**all**(**by.repeater**(‘item in items’)); – выбор списка элементов из ngRepeat
- element(by.model(‘modelName’)).**getText**() – получение текстового значения
- element(by.model(‘modelName’)).**getAttribute**(‘id’) – получение значение аттрибута
- element(by.model(‘modelName’)).**sendKeys**(‘Some text’) – задание значения
- element.all(by.repeater(‘item in items’)).**count**() – получение количества элементов в списке
- element.all(by.repeater(‘item in items’)).**get**(1) – получение одного элемента из списка
- element.all(by.repeater(‘item in items’)).**row**(1).**column**(‘title’) – получение значение title из 2й строки
- **browser**.**isElementPresent**(by.model(‘modelName’)) – проверка наличия элемента
- **$**(‘.info’) – короткий алиас к **element**(**by.css**(‘.info’))
- **$$**(‘option’) – короткий алиас к **element.all**(**by.css**(‘option’))

Создание снимков экрана

Вебдрайвер позволяет делать скриншоты с помощью метода**browser.takeScreenshot()**, который возвращает промис, который в свою очередь вернет **PNG** снимок экрана в формате **base64**:

```
browser.takeScreenshot().then(function (png) {
    //...
}
```

Для записи файла на диск можно написать свою вспомогательную функцию:

```
var fs = require('fs');
function writeScreenShot(data, filename) {
    var stream = fs.createWriteStream(filename);
    stream.write(new Buffer(data, 'base64'));
    stream.end();
}
```

и потом ее вызвать:

```
browser.takeScreenshot().then(function (png) {
    writeScreenShot(png, 'test_screen.png');
}
```

Jasmine

Jasmine – это BDD фреймворк для тестирования JavaScript кода. Он не зависит от других фреймворков и не требует наличия DOM. Имеет простой понятный синтаксис для написания тестовых сценариев.

Основными ключевыми словами при работе с Jasmine являются:

- **describe** — определение набора тестов, наборы могут быть вложенными
- **it** — определение теста внутри любого набора тестов
- **expect** — определяет ожидания, которые проверяются в тесте

Ключевые слова Jasmine

Ключевые слова **describe** и **it** являются обычными вызовами функций, которым передаются два параметра. Первый — название группы или теста, второй — функция содержащая код.

Для того чтобы отключить выполнение набора тестов или конкретного теста, необходимо воспользоваться ключевыми словами **xdescribe** и **xit** соответственно.

Jasmine имеет стандартный набор ожиданий для проверки результатов.

Для того чтобы избежать повторения при создании/удалении объектов и загрузки фикстур, необходимых для выполнения тестов, используются функции **beforeEach**/**afterEach**. Они запускаются перед/после каждого теста в наборе.