

В.В. Герасимов, А.Я. Беличенко

**РАЗВИТИЕ ИНСТРУМЕНТА АВТОМАТИЗИРОВАННОГО
ТЕСТИРОВАНИЯ – SELENIUM**

Аннотация. Проведен анализ особенностей разных поколений популярного инструмента для автоматизации действий веб-браузера Selenium. Указаны преимущества, которые дает библиотека Selenide — обертка над Selenium WebDriver.

Ключевые слова: WebDriver, Selenium, Selenide, браузер, веб-приложение, тест.

Постановка проблемы. При изменении работающего кода всегда существует вероятность того, что изменения сломают отлаженную систему. Чтобы избежать этого, после правок программу тестируют для выявления ошибок. Тесты помогают разработчику своевременно выявить проблему и починить ее. Поэтому желательно покрывать свой код тестами [1]. Тем не менее, тесты не всегда могут обнаружить логические ошибки и из-за этого может пострадать работоспособность программы. К тому же процесс этот занимает время и не всегда тестер может заметить ту или иную ошибку. Отличным решением сложившейся ситуации является применение автоматизированного тестирования [2].

Одним из широко используемых инструментов, используемых для автоматизации тестирования, является Selenium — инструмент для автоматизации действий веб-браузера [3].

Целью данной работы является рассмотрение используемых в настоящее время поколений популярного инструмента Selenium, анализ их функциональных особенностей, возможностей и удобства работы с ним.

Основная часть. Selenium — инструмент для автоматизированного управления браузерами. Наиболее популярной областью применения Selenium является автоматизация тестирования веб-приложений. На данный момент существует три поколения данного фреймворка, которые довольно-таки отличаются друг от друга [4].

Selenium 1 (Selenium RC) состоит из двух частей (рис. 1):

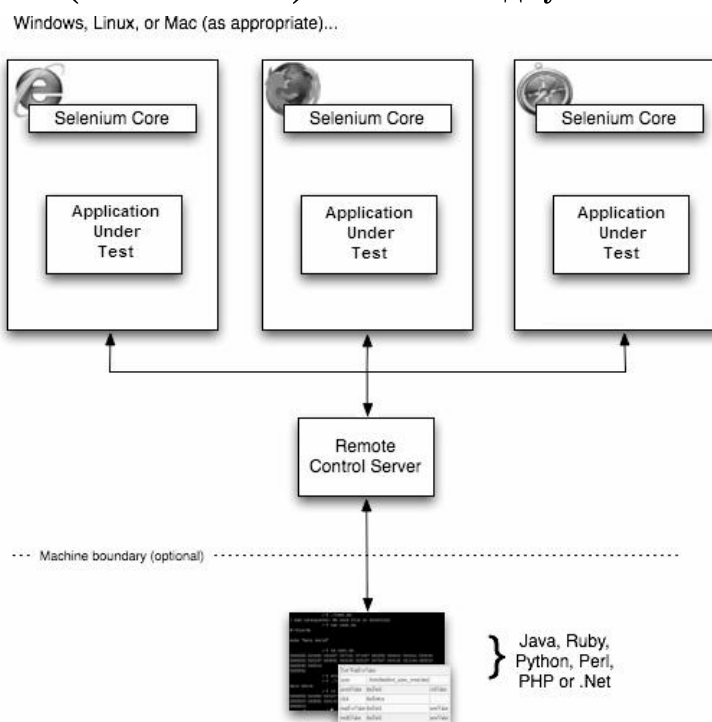


Рисунок 1 - Схема работы Selenium RC

1. Selenium Server принимает команды из вашего теста, интерпретирует их, и возвращает вашей программе результаты их выполнения. Команды передаются через HTTP-запросы, так что любой язык программирования, который их может посылать, может использоваться для Selenium (например, C#, Java, Ruby, PHP, Python). Кроме того, сервер в момент открытия тестовой программой браузера, автоматически собирает и разворачивает ядро Selenium в браузере.

2. Клиентские библиотеки обеспечивают интерфейс между каждым из языков программирования и сервером Selenium RC.

В Selenium 2.0 интегрируется WebDriver API, который поддерживает большее количество браузеров и лучше решает проблемы тестирования веб-приложений (табл. 1) [5]. WebDriver — это инструмент для автоматизированного тестирования веб-приложений, в частности, для проверки того, что приложение работает в соответствии с ожиданиями. Этот инструмент задумывался таким образом, чтобы иметь удобный программный интерфейс (API), позволяющий повысить читаемость и упростить поддержку тестов, более легкий для изучения и понимания, чем Selenium RC (1.0) API.

Сравнение Selenium RC и WebDriver

Параметр	Selenium RC	WebDriver
Способ взаимодействия с браузером	В браузер внедряется специальная универсальная библиотека Selenium Core, написанная на JavaScript	Взаимодействие с браузером происходит через нативный интерфейс, через который выполняются команды, частично реализованные на JavaScript, а частично нативные
Набор команд	Большой, хаотически сложившийся в результате эволюции	Минимальный, соответствующий тому, какие действия может выполнять пользователь, в основном всё сводится к click и sendKeys
События, генерируемые в браузере в результате тестовых воздействий	Многие события не генерируются, нужно это делать искусственно, иногда явно вызывая операцию fireEvent	События генерируются в точности такие же, как при ручном выполнении, потому что воздействие на браузер максимально точно эмулирует действия пользователя
Удаленное управление браузером	Да, с использованием протокола Selenium Remote Control	Да, с использованием протокола WebDriver JSON
Поиск элементов	Всегда от корня дерева DOM	Можно задавать контекст поиска – либо все дерево DOM, либо отдельный элемент и его поддереву (то есть вложенные элементы)
Ожидание появления элемента (перед совершением некоторого действия с ним)	Эксплицитное, с использованием класса Wait	Либо эксплицитное, с использованием класса WebDriverWait, либо имплицитное — все команды поиска элементов автоматически становятся ожидающими
Работа с невидимыми элементами	Возможна	Невозможна (потому что пользователь вручную не может этого делать)

WebDriver API не привязан ни к каким тестовым фреймворкам, что позволяет использовать любые фреймворки модульного тестирования, равно как и старый добрый метод “main”. Одним из главных преимуществ этой версии является то, что Selenium 2.0 обладает полной обратной совместимостью. Всё, что было в версии 1.0, по-прежнему присутствует и может использоваться в версии 2.0.

Пример теста, написанного на разных версиях:

Selenium 1:

```
public class SeleniumRCScriptSample {
    public static void main(String args[]) {
        Selenium selenium = new DefaultSelenium("localhost", 4444, "*firefox", "
https://www.facebook.com");
        selenium.start();
        selenium.open("/");
        selenium.windowMaximize();
        Thread.sleep(5000);
        selenium.type("type='email'", "contact.automateapps@gmail.com");
        selenium.type("type='password'", "password");
        selenium.click("value='Log In'");
        selenium.stop();
    }
}
```

Selenium 2:

```
public class MigratedScriptFromRCToWebDriver {
    public static void main(String args[]) {
        WebDriver driver = new FirefoxDriver();
        String baseUrl = "https://www.facebook.com";
        Selenium selenium = new WebDriverBackedSelenium(driver, baseUrl);
        selenium.type("type='email'", "contact.automateapps@gmail.com");
        selenium.type("type='password'", "password");
        selenium.click("value='Log In'");
        selenium.stop();
    }
}
```

Для WebDriver API, Selenium 3.0 не привнесло никаких изменений, так что код менять не надо, достаточно просто подключить новую версию и всё. В тоже время, пользователям Selenium Grid придётся обновить конфигурационные json-файлы, так как их формат поменялся. Изменился также набор опций командной строки. Основным изменением в Selenium 3.0 является полное отсутствие драйвера предыдущего поколения, Selenium Core. Продолжать использовать Selenium RC API можно, но при этом всё равно будет использоваться реализация на базе Selenium WebDriver. Кроме того, в новой версии драйверы для всех браузеров разрабатывали не создатели Selenium, а сами производители браузеров, так как именно они лучше всех знают особенности своих браузеров. Другое значительное достижение — создание на базе WebDriver спецификации интерфейса управления

браузерами. Эта спецификация разрабатывается под эгидой комитета W3C. Mozilla является первой компанией, которая пытается разработать новый драйвер в соответствии со спецификацией W3C. Вследствие того, что разработка драйвера продвигается быстрее разработки Selenium 3, для тестирования в данном браузере стоит обновлять версию *geckodriver*. До версии 45 драйвер, используемый для автоматизации Firefox, был расширением, включенным в каждый клиент. Но это расширение было удалено, вероятно, из-за изменения политики, которая теперь требует, чтобы все расширения были подписаны Mozilla.

Marionette — это новый драйвер, который поставляется вместе с Firefox. У этого драйвера есть собственный протокол, который несовместим с протоколом Selenium/WebDriver. Драйвер Gecko — это сервер приложений, реализующий протокол Selenium/WebDriver. Он перехватывает команды Selenium и перенаправляет их в драйвер Marionette.

Подключение Marionette- и Gecko- драйверов для 45+ версий Firefox:

```
WebDriver driver;  
DesiredCapabilities capabilities;  
System.setProperty("webdriver.gecko.driver", "PATH_TO_GECKODRIVER");  
System.setProperty("webdriver.firefox.marionette", "PATH_TO_GECKODRIVER");  
capabilities.setCapability("marionette", true);  
driver = WebDriverFactory.getDriver(capabilities);
```

Кроме того, перестали поддерживаться нативные события (*nativeEvents*), так как было обнаружено, что под UNIX-системами они некорректно себя вели.

Selenide — это обёртка вокруг Selenium WebDriver, позволяющая быстро и просто его использовать при написании тестов, сосредоточившись на логике, а не суете с браузером [6]. Прежде всего, это выражается в том, что больше не надо самому объявлять и конфигурировать драйвер — Selenide берет это на себя. Драйвер в Selenium имел свойство перегружаться после нескольких страниц, при этом все, что находилось в буфере обмена, терялось. Если в ОС под управлением Windows можно было сохранять данные из буфера обмена, то для пользователей UNIX такое решение не подойдет из-за того, что в Linux два буфера обмена. С Selenide данная проблема ушла. Выбор радиокнопки, выбор элемента из выпадающего списка, создание

снимка экрана, очистка кэша браузера и т.п. — то, чего не было в Selenium, появляется с данной библиотекой. Вторым главным свойством является то, что все, что было доступно при голом Selenium, продолжает быть доступным для использования.

Selenide предлагает лаконичный и мощный API, который поможет вам писать короткие и хорошо читаемые тесты. Поддержка Ajax позволит не заботиться о необходимости ожидания, так как определенные методы по умолчанию до 4-х секунд будут ожидать загрузки того или иного элемента.

Еще одной очень удобной особенностью являются автоматические скриншоты, которые можно делать как в случае падения теста, так и при положительном результате. Вместе со скриншотом сохраняется и html страница, на которой упал тест:

- при падении:

```
@Rule
```

```
public ScreenShooter makeScreenshotOnFailure = ScreenShooter.failedTests();
```

- при успешном завершении:

```
@Rule
```

```
public ScreenShooter makeScreenshotOnSuccess = ScreenShooter.succeededTests();
```

Кроме локального хранения, можно настроить систему непрерывного развертывания Jenkins на сохранение скриншотов.

С Selenide нет проблемы с отчетами благодаря встроенному механизму — в консоль будет выводиться, сколько времени затрачено на тот или иной метод, что также позволит понимать, что именно долго отработывает в веб-приложении:

```
@Rule
```

```
public TestRule report = new TextReport();
```

Еще одним приятным дополнением является тот факт, что больше нет никаких проблем с запуском тестов в разных браузерах (поддерживаются все популярные браузеры):

```
Configuration.browser = "chrome";
```

Почти что единственным недостатком Selenide является малая, по сравнению с Selenium, поддержка языков — пока лишь Java и Python.

Синтаксис Selenide схож с синтаксисом JQuery, а тест, ранее написанный на Selenium, имеет вид:

```
public class SelenideTest {
```

```
    @Test
```

```

public void Test() {
    open("https://www.facebook.com");

    $(By.cssSelector("[type='email']").setValue("contact.automateapps@gmail.com");
    $("[type='password']").setValue("password");
    $("[value='Log In']").click();
}
}

```

Невооруженным взглядом можно заметить, что тест занимает в два раза меньше места, чем при использовании Selenium.

Выводы. Автоматизированное тестирование, в первую очередь, следует использовать для тестирования корпоративных веб-приложений, поскольку у них огромное количество возможных вариантов событий, проверять каждое из них вручную займет очень много времени, а при интенсивной разработке подобная процедура может повторяться несколько раз в день.

Автоматизированное тестирование незаслуженно низко оценивается современным обществом, во многом из-за того, что оно выполняет “черновую работу”, которая не на виду. Тем не менее, при правильном построении автотестирования, именно эта отрасль позволит сэкономить большое количество ресурсов. Знание фреймворка Selenium позволит разработчику писать тесты на разных языках программирования. Владение технологией Selenide позволит использовать гораздо больше методов и возможностей, чем чистый Selenium, сэкономить время и затраты на работу с драйвером и полностью переключить внимание на логику.

ЛИТЕРАТУРА

1. Матвеева Н.А., Герасимов В.В., Игнатъева Д.О. Исследование технологий модульного тестирования на платформе Java // Системні технології. Регіональний міжвузівський збірник наукових праць. — Вип. 1 (102). — Дніпропетровськ, 2016. С. 49 – 55.
2. Герасимов В.В., Кронфельд М.Ф., Озерова Д.М. Исследование технологий автоматизированного тестирования // Системні технології. Регіональний міжвузівський збірник наукових праць. — Вип. 1 (96). — Дніпропетровськ, 2015. С. 130 – 136.
3. Selenium – Web Browser Automation — Режим доступа: <http://www.seleniumhq.org/>
4. Selenium – автоматизация веб-приложений — Режим доступа: <https://selenium2.ru/>
5. Selenium 2. Remote Control vs Webdriver — Режим доступа: <http://habrahabr.ru/post/151715/>
6. Selenide: лаконичные и стабильные UI тесты на Java - Режим доступа: <http://ru.seleniumide.org/>