Web Applications Testing

1) Selenium WebDriver
2) Selenium IDE

Selenium Downloads

Selenium Client & WebDriver Language Bindings

In order to create scripts that interact with the Selenium Server (Selenium RC, Selenium Remote WebDriver) or create local Selenium WebDriver scripts, you need to make use of language-specific client drivers. These languages include both 1.x and 2.x style clients.

While language bindings for other languages exist, these are the core ones that are supported by the main project hosted on google code.

Language	Client Version	Release Date			
Java	3.8.1	2017-12-01	Download	Change log	Javadoc
C#	3.8.0	2017-12-01	Download	Change log	API docs
Ruby	3.8.0	2017-12-01	Download	Change log	API docs
Python	3.8.0	2017-11-30	Download	Change log	API docs
Javascript (Node)	3.6.0	2017-10-06	Download	Change log	API docs

C# NuGet

NuGet latest release is 3.8.0, Released on 2017-12-01

- WebDriver
- WebDriverBackedSelenium
- Support
- RC (Final version 3.1.0 Released 2017-02-16)

http://www.seleniumhq.org/download/

Selenium WebDriver

WebDriver – это инструмент для автоматизированного тестирования веб-приложений, в частности, для проверки того, что приложение работает в соответствии с ожиданиями.

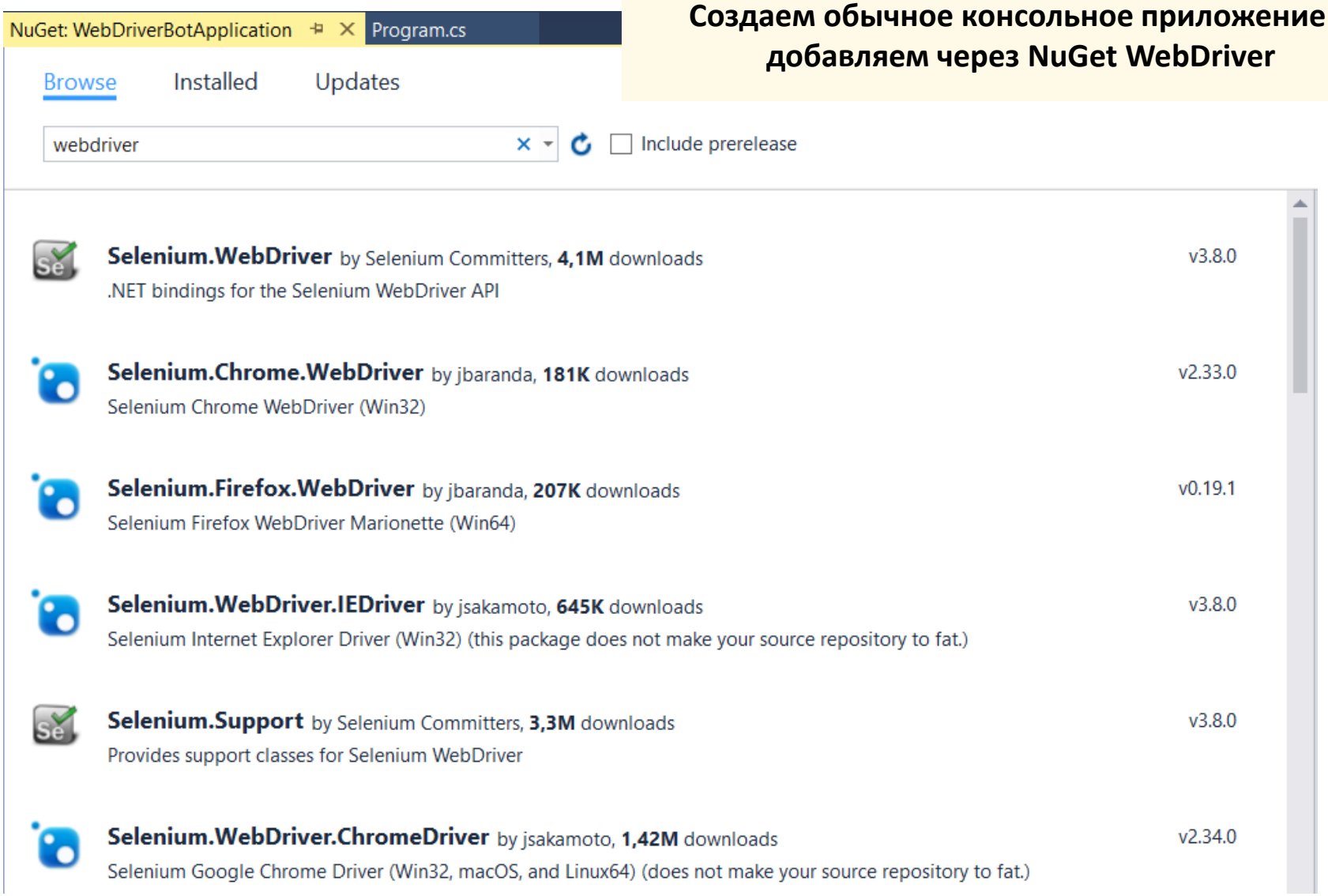
Этот инструмент задумывался таким образом, чтобы иметь удобный программный интерфейс (API), позволяющий повысить читаемость и упростить поддержку тестов, более легкий для изучения и понимания, чем Selenium RC (1.0) API.

WebDriver API не привязан ни к каким тестовым фреймворкам, что позволяет использовать любые фреймворки модульного тестирования, равно как и старый добрый метод Main.

http://selenium2.ru/docs/webdriver.html

Проект с WebDriver

Создаем обычное консольное приложение и добавляем через NuGet WebDriver



Пример теста (бот, без Assert)

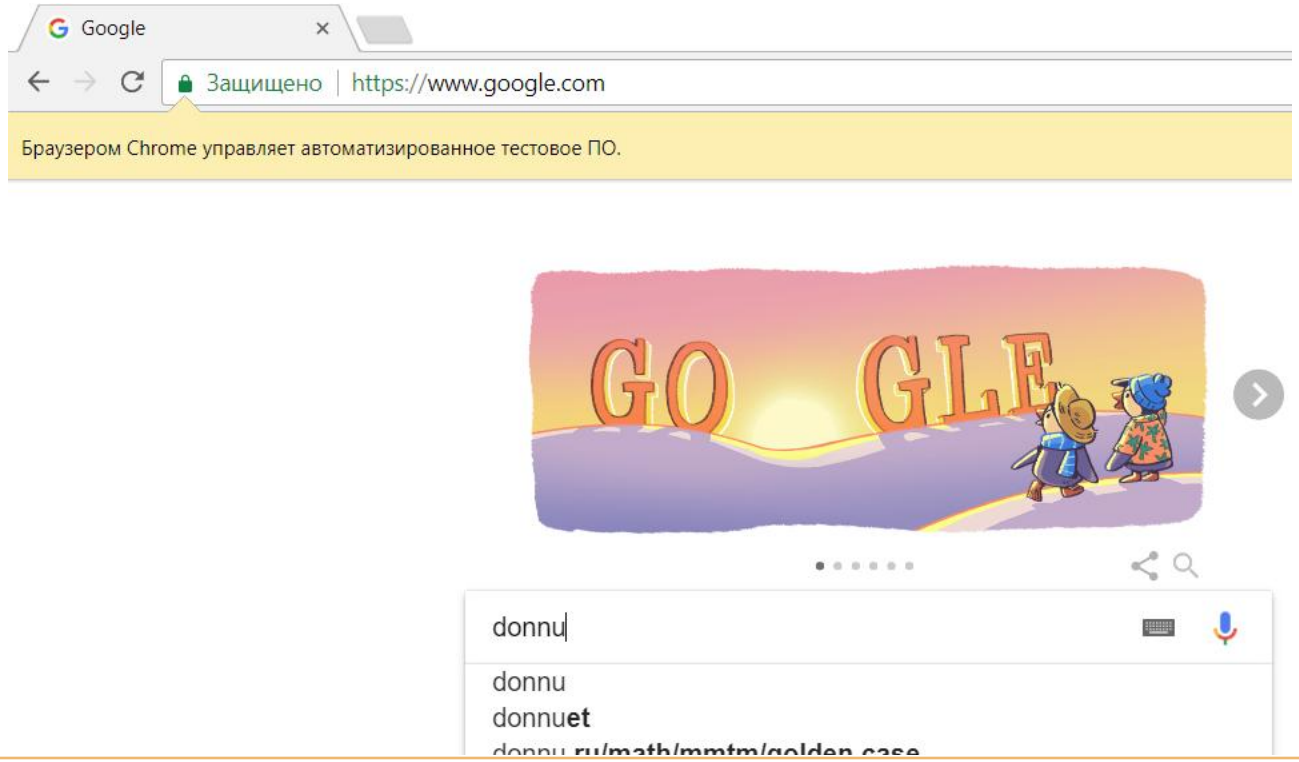
```
// Простой бот, который проверяет, что адрес donnu.ru
// присутствует в первых результатах по запросу "donnu"
// поисковиков: сначала Google, затем Yandex

class Program
{
    static readonly IWebDriver _driver = new ChromeDriver();

    static void Main()
    {
        const string addressToAppear = "donnu.ru";

        _driver.Navigate().GoToUrl(@"http://www.google.com");

        IWebElement search = _driver.FindElement(By.Id("lst-ib"));

        search.SendKeys("donnu");
        System.Threading.Thread.Sleep(250);         // for visualization
        search.SendKeys(Keys.Enter);

        // Not a good way of waiting
        // System.Threading.Thread.Sleep(1500);     // for waiting

        WebDriverWait wait = new WebDriverWait(_driver, TimeSpan.FromSeconds(10));

        wait.Until(d => d.FindElement(By.ClassName("_Rm")));
```

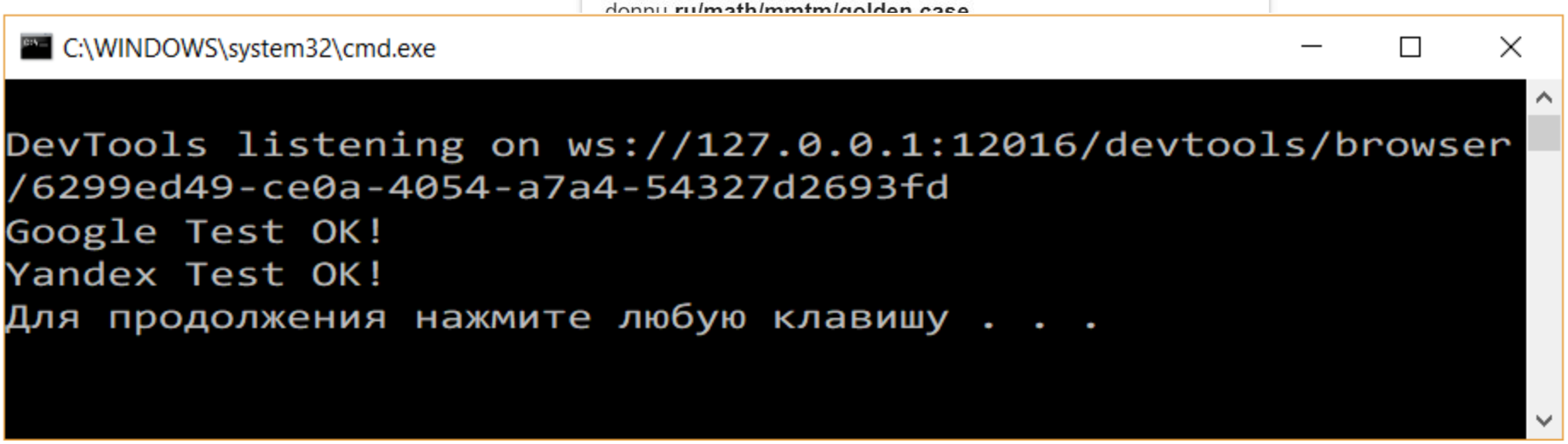
Пример теста (продолжение)

```
var entries = _driver.FindElements(By.ClassName("_Rm"));

if (entries.Any(entry => entry.Text.Contains(addressToAppear)))
{
    Console.WriteLine("Google Test OK!");
}
```

Пример теста (продолжение)

```
_driver.Navigate().GoToUrl(@"http://ya.ru");

wait.Until(d => d.FindElement(By.Id("text")));

search = _driver.FindElement(By.Id("text"));
search.SendKeys("donnu");
System.Threading.Thread.Sleep(250);     // for visualization
search.SendKeys(Keys.Enter);

wait.Until(d => d.FindElement(By.ClassName("serp-list")));

try
{
    entries = _driver.FindElements(By.XPath("//a[contains(@class, 'link')]/b"));

    if (entries.Any(entry => entry.Text.Contains(addressToAppear)))
    {
        Console.WriteLine("Yandex Test OK!");
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.GetType().Name + "\n" + ex.Message);
}
finally
{
    _driver.Quit();
}
```


Google
Защищено | https://www.google.com
Браузером Chrome управляет автоматизированное тестовое ПО.
GO GLE
donnu
donnu
donnuet


C:\WINDOWS\system32\cmd.exe
DevTools listening on ws://127.0.0.1:12016/devtools/browser/6299ed49-ce0a-4054-a7a4-54327d2693fd
Google Test OK!
Yandex Test OK!
Для продолжения нажмите любую клавишу . . .

Navigate Commands

За навигацию отвечает класс Navigate.

void GoToUrl(string url) – перейти по указанному адресу.
Пример: driver.Navigate().GoToUrl("some_url");

void Back() — вернуться на предыдущую страницу.
Пример: driver.Navigate().Back();

void Forward() — перейти на следующую страницу.
Пример: driver.Navigate().Forward();

void Refresh() — обновить страницу.
Пример: driver.Navigate().Refresh()

Browser Commands

void Close() — закрыть текущее окно. Закрывает браузер, если нету больше открытых окон;

void Dispose() — Member of System.IDisposable. С помощью данного метода можно удалять, освобождать или сбрасывать неуправляемые ресурсы;

Manage() — позволяет изменять настройки драйвера;

void Quit() выход из драйвера, закрытие всех окон связанных с ним;

OpenQA.Selenium.IWebElement FindElement(OpenQA.Selenium.By by) — поиск элемента на странице. Параметр By указывает на механизм поиска элемента. Возвращает первый найденный элемент, удовлетворяющий, условию поиска. Примеры представлены выше;

Browser Commands

System.Collections.ObjectModel.ReadOnlyCollection FindElements(OpenQA.Selenium.By by) — то же самое, что и FindElement, только возвращает все элементы, удовлетворяющие условию поиска;

OpenQA.Selenium.ITargetLocator SwitchTo() — переключение между popup- ами, алертами, окошками;

string PageSource — возвращает содержимое последней загруженной страницы;

string Title — возвращает Title текущей страницы;

string Url — устанавливает или возвращает Url текущего окна;

string CurrentWindowHandle — возвращает ссылку на текущее окно;

System.Collections.ObjectModel.ReadOnlyCollection WindowHandles — возвращает ссылки на все открытые окна браузера.

Примеры с Assert

Проверяем Title текущей страницы:

```
Assert.AreEqual("Bugs Catcher", driver.Title);
```

Проверяем, есть ли на текущей странице текст «Результаты поиска»:

```
Assert.IsTrue(driver.PageSource.Contains("Результаты поиска"));
```

Popups, Alerts

Для переключения между окнами, popup- ми, алертами нужно использовать метод SwitchTo(). Для работы с алертами предназначен класс IAlert.

Выбираем фрейм по имени:
driver.SwitchTo().Frame("frameName");

Обрабатываем алерт:
Alert alert = driver.SwitchTo().Alert();
alert.Accept();

Выбираем окно по имени:
driver.SwitchTo().Window("windowsName");

Работа с Cookies

WebDriver предоставляет необходимые методы работы с Cookies. Пример объявления переменной для работы с Cookies и удаление всех Cookies:

```
ICookieJar cookieJar = driver.Manage().Cookies;
cookieJar.DeleteAllCookies();
```

Действия над элементами страницы

void Click() — одиночное нажатие по элементу;
string GetAttribute(string attributeName) — возвращает значения атрибута;
string GetCssValue(string propertyName) — возвращает значение свойства CSS элемента;
void SendKeys(string text) — ввод текста в текстовые поля. При работе с текстовыми полями можно использовать функциональные клавиши, их работу обеспечивает класс Keys. Например: element.SendKeys("sds" + Keys.Up);
void Submit() — отправка формы на сервер;
bool Displayed — возвращает значение, которое указывает на то, является ли элемент невидимым;
bool Enabled — возвращает значение, которое указывает на то, является ли элемент видимым;
System.Drawing.Point Location — возвращает координаты элемента;
bool Selected — возвращает значение, является ли данный элемент (checkboxes, radio buttons) выбранным;
string TagName — возвращает имя тега элемента;
string Text — возвращает innerText элемента (без пробелов);
System.Drawing.Size Size — вовращает размеры элемента;
void Clear() — очистка содержимого текстового элемента.

Пример работы с select

```
using OpenQA.Selenium.Support.UI;
...
IWebElement element = driver.FindElement(By.Id("submit"));
SelectElement select = new SelectElement(driver.FindElement(By.XPath("//select")));
select.DeselectAll();
select.SelectByIndex(1);
select.SelectByText("TestText");
select.SelectByValue("Value2");
element.submit();
```

Пример сохранения скриншота

```
public void TakeScreenshot(IWebDriver driver, string saveLocation)
{
	ITakesScreenshot screenshotDriver = driver as ITakesScreenshot;
	Screenshot screenshot = screenshotDriver.GetScreenshot();
	screenshot.SaveAsFile(saveLocation, ImageFormat.Png);
}
...
IWebDriver driver = new SomeDriver();
driver.Navigate().GoToUrl("http://www.google.com");
TakeScreenshot(driver, @"C:\screenshot.png");
```

Пример запуска JavaScript

```
public object ExecuteJavaScript(string javaScript, params object[] args)
{
        Trace.WriteLine("Executes JavaScript", "Document");
        var javaScriptExecutor = (IJavaScriptExecutor) _driver;
        return javaScriptExecutor.ExecuteScript(javaScript, args);
}

...

ExecuteJavaScript("alert('hello world');");
```

Page Object

В WebDriver реализована возможность использования шаблонов PageObject и PageFactory. Суть данных шаблонов заключается в том, чтобы простейшие операции выделить в отдельные логические блоки, т.е. абстрагировать до уровня действий на определенной странице.

Например, у нас есть страница Login. Первым делом мы создадим отдельный класс Login в котором опишем все элементы страницы и методы их поиска. Далее для данного класса выделим все необходимые методы (действия на странице) используя уже ранее описанные элементы страницы. После чего реализованные методы можно будет использовать на более высоком уровне бизнес логики. Данный подход позволяет значительно снизить расходы на разработку и особенно на поддержку тестового фреймворка. А также сделает тесты более гибкими и реюзабельными. Для использования описанных выше шаблонов в реализации WebDriver необходимо подключить пространство имен OpenQA.Selenium.Support.PageObjects.

Page Object

```
IWebElement LoginButton = driver.FindElement(By.Id("LoginControl_LoginButton"));
IWebElement UserNameTextBox = driver.FindElement(By.Id("LoginControl_UserName"));
IWebElement PasswordTextBox = driver.FindElement(By.Id("LoginControl_Password"));

...

public void LoginAsAdministrator()
{
    UserNameTextBox.SendKeys("Administrator");
    PasswordTextBox.SendKeys("admin");
    LoginButton.Click();
}
```

Page Object

```
public class LoginPage
{
    [FindsBy(How = How.XPath, Using = USER_NAME_TEXT_FIELD)]
    public IWebElement userNameTextField;

    [FindsBy(How = How.XPath, Using = PASSWORD_TEXT_FIELD)]
    public IWebElement passwordTextField;

    [FindsBy(How = How.XPath, Using = LOGIN_BUTTON)]
    public IWebElement loginButton;

    public static LoginPage GetLoginPage(){...}
    public LoginPage TypeUserName(string userName){...}
    public LoginPage TypePassword(string password){...}
    public LandingPage ClickLoginButton(){...}

    public const string USER_NAME_TEXT_FIELD = "//input[@id='email']";
    public const string PASSWORD_TEXT_FIELD = "//input[@id='pass']";
    public const string LOGIN_BUTTON = "//label[@id='loginbutton']/input";
}
```

Page Factory

```
var driver = new ChromeDriver();

driver.Navigate().GoToUrl(StartingUrl);

var loginPage = PageFactory.InitElements<LoginPage>(driver);
loginPage = loginPage.TypeUserName("admin");
loginPage.TypePassword("12345");

var landingPage = loginPage.Submit();

Assert.That(landingPage.LoggedUser, Is.EqualTo("admin"));
```

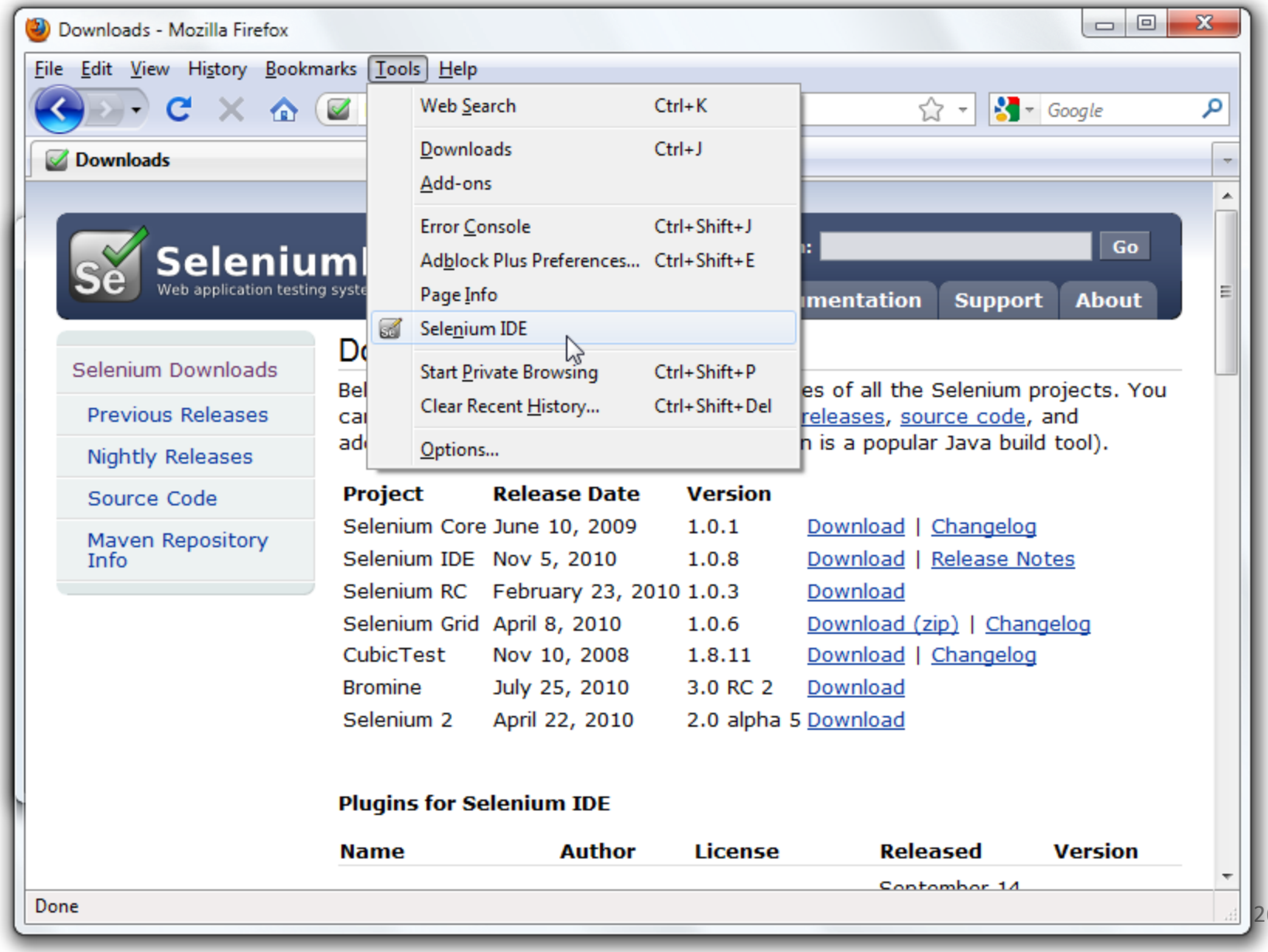
Еще пример Page Object

```
[TestFixture]
public class SeleniumTestClass
{
    private const string Keyword = "C#";

    private IWebDriver _driver;

    [OneTimeSetUp]
    public void Init()
    {
        _driver = new ChromeDriver();
        _driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(30);
    }

    [OneTimeTearDown]
    public void Quit()
    {
        _driver.Quit();
    }

    [Test]
    public void TestSearchResults()
    {
        var searchPage = new SeleniumSearchPage(_driver);
        searchPage.Navigate();

        var resultsPage = searchPage.Search(Keyword);

        Assert.That(resultsPage.Links, Does.Contain("Downloads"));
    }
}
```

```
public class SeleniumSearchPage
{
    private const string Url = @"http://www.seleniumhq.org/";

    private const string SearchQueryFieldId = "q";
    private const string SearchButtonId = "submit";

    private readonly IWebDriver _driver;

    [FindsBy(How = How.Id, Using = SearchQueryFieldId)]
    public IWebElement SearchQueryField;

    [FindsBy(How = How.Id, Using = SearchButtonId)]
    public IWebElement SearchButton;

    public SeleniumSearchPage(IWebDriver driver)
    {
        _driver = driver;
        PageFactory.InitElements(_driver, this);
    }

    public void Navigate()
    {
        _driver.Navigate().GoToUrl(Url);
    }

    public SearchResultsPage Search(string keyword)
    {
        SearchQueryField.SendKeys(keyword);
        SearchButton.Click();

        return new SearchResultsPage(_driver);
    }
}
```

```
public class SearchResultsPage
{
    [FindsBy(How = How.XPath, Using = "//a[@class='gs-title']")]
    public IList<IWebElement> LinkElements;

    public SearchResultsPage(IWebDriver driver)
    {
        var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(20));
        wait.Until(p => p.FindElement(By.Id("cse-body")));

        PageFactory.InitElements(driver, this);
    }

    public List<string> Links => LinkElements.Select(l => l.Text).ToList();
}
```

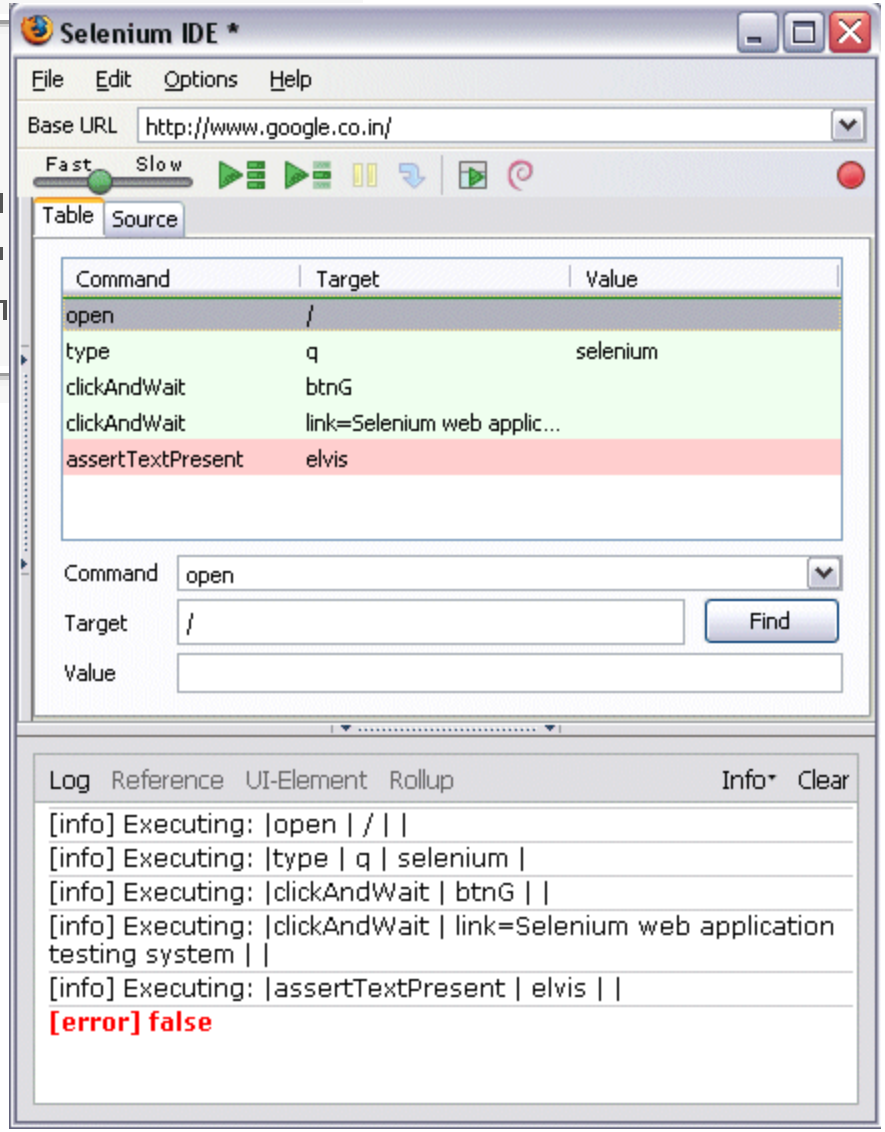
Selenium IDE



Окно Selenium IDE

Selenium IDE на службе функциональника

Тестировщик-ман
руки Selenium ID
неимоверный рул



Панель тестового сценария

Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

Selenium IDE

В Selenium существует три типа команд:

Действия (actions)
функциональное действие над тестируемым веб-приложением в браузере. Например, заполнение полей, нажатие на кнопку и другие;

Проверки (checks)
выполнение проверок на тестируемой странице. Например, проверка того, что определенное поле формы имеет указанное значение, или проверка заголовка окна;

Ожидания (wait)
организация как, сколько и какое событие Selenium будет дожидаться (ожидания загрузки страницы, ajax и т.д.).

Actions

open
открыть страницу в браузере по определенному адресу. Синтаксис команды — Open(string url). Пример — selenium.Open("http://blogs.logicsoftware.net/qa/");

click
произвести нажатие по элементу страницы. Синтаксис команды — Click(string locator). Пример использования — selenium.Click("LoginButton");

type
ввести значение в текстовое поле страницы. Синтаксис команды — Type(string locator, string value). Пример использования — selenium.Type("id_TextField_1", "test");

select
выбрать значение из выпадающего списка. Синтаксис команды — Select(string selectLocator, string optionLocator). Пример использования — selenium.select (TimeEntryTaskList, "Activity1") . В качестве опции для выбора элемента можно использовать следующие локаторы: label, value, id, index. При использовании локатора label для optionLocator, можно искать элемент по частичному совпадению label=regexp:Locator;

Locators in Selenium

id – в качестве локатора используется атрибут id элемента страницы;

name – в качестве локатора используется атрибут name элемента страницы;

identifier – используется атрибут id элемента, если по id-у элемент не найден, то поиск будет вестись по атрибуту name;

dom – поиска элемента происходит по DOM выражению;

xpath – используется для поиска элемента по XPath выражению;

link – поиск ссылок с указанным текстом;

css – данный тип локаторов основан на описаниях таблиц стилей (CSS).

Checks

verifyLocation / assertLocation
проверить адрес текущей страницы. Синтаксис команды — verifyLocation(URL);

verifyTitle / assertTitle
проверить значение Title страницы. Синтаксис команды — verifyTitle (Title);

verifyValue / assertValue
проверить значение элемента страницы. Синтаксис команды — verifyValue (locator, value);

verifyTextPresent / assertTextPresent
проверить, что страница содержит указанный в команде текст. Синтаксис команды — verifyTextPresent (value);

verifyElementPresent / assertElementPresent
проверить, есть ли на странице указанный элемент. Синтаксис команды — verifyElementPresent (locator).

Wait

WaitForCondition
ожидание выполнения определенного события на странице, указанного в параметре script (например, загрузка определенного элемента страницы, или страницы в целом). Синтаксис команды —WaitForCondition(string script, string timeout);

WaitForFrameToLoad
ждет загрузки фрэйма на странице указанное количество времени. Синтаксис команды —WaitForFrameToLoad(string frameAddress, string timeout);

WaitForPageToLoad
ждет загрузки страницы указанное количество времени. Синтаксис команды — WaitForPageToLoad(string timeout);

WaitForPopUp
ждет появления PopUp элемента страницы указанное количество времени. Синтаксис команды —WaitForPopUp(string windowID, string timeout);