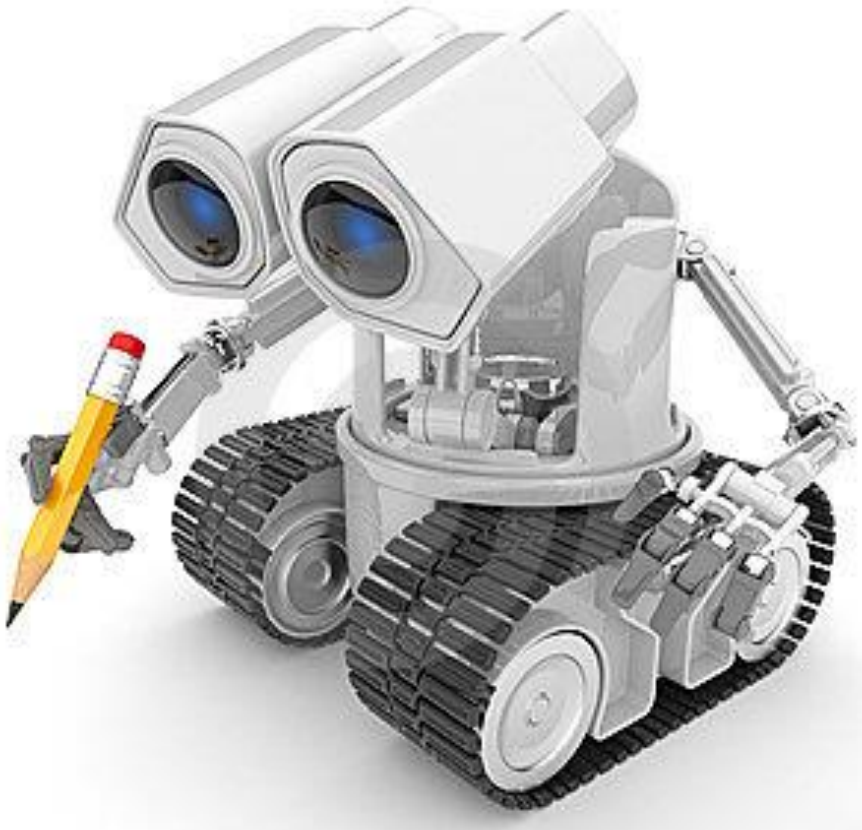
Автоматизация тестирования



Автоматизированное тестирование

Автоматизированное тестирование – это набор техник, подходов, методов и инструментальных средств, позволяющих исключить участие человека в реализации некоторых задач по тестированию ПО.

Из этого определения следует, что

Автоматизация тестирования – НЕ является стадией процесса тестирования, не следует за или перед каким-то этапом, а является процессом, пронизывающим большинство стадий тестирования

Инструментальное средство автоматизированного тестирования *(test automation tool)* – программа (или набор программ), позволяющая создавать, редактировать, отлаживать и выполнять автоматизированные тесты, а также собирать статистику их выполнения.

Области автоматизации

Проверка ссылок

- Проверка ссылок в любом веб-ориентированном приложении является неотъемлемым этапом тестирования.

- Эта задача в тестировании может быть полностью автоматизирована (в том числе с применением бесплатных инструментов).

Области автоматизации

Проверка работоспособности стандартного, типичного для множества проектов функционала

- Есть функции, повторяющиеся во множестве проектов. Они полностью идентичны или обладают незначительными отличиями. Автоматизация тестирования таких функций экономит много времени.
- Пример таких функций:
  - Аутентификация пользователя.
  - Регистрация пользователя.
  - Добавление товара в корзину.

Области автоматизации

Проверка стандартных элементов управления

- Изолированная проверка стандартных элементов управления хорошо автоматизируется как в настольных приложениях, так и в веб-ориентированных.

Базовая проверка безопасности

- К базовой проверке можно отнести: исследование известных уязвимостей тех или иных версий ПО, неверно выставленные права доступа (открыты какие-то файлы, которые не должны быть доступны извне), использование паролей по умолчанию, разрешённый доступ к СУБД не с localhost и т.п.

Что ещё следует автоматизировать

- **Smoke test** для крупных систем, где приходится выполнять большое количество примитивных трудоёмких задач
- **Регрессионное тестирование**. Достаточно большой процент тестов повторяется снова и снова на каждом цикле тестирования. Их автоматизация может сэкономить в конечном итоге много сил и времени.
- **Конфигурационное тестирование** в контексте проверки работоспособности приложения при тех или иных настройках
- Часто **повторяющиеся тесты**, простые для автоматизации
- Длинные **утомительные** для человека **тесты**. Заполнение форм, регистрация тысяч пользователей и т.п. задачи крайне утомительны для человека и являются неприятной рутиной, которой хотелось бы избежать.
- **Нагрузочное, стрессовое тестирование и тестирование производительности.** Этот вид тестирования относится к числу тех, в которых без автоматизации обойтись невозможно или почти невозможно. Так, например, вручную крайне тяжело проэмулировать работу нескольких десятков тысяч пользователей, одновременно собирая полную статистику по работе каждого из них и реакции приложения.

Условия автоматизации

- Как правило, автоматизация тестирования связана с работой с интерфейсом приложения, поэтому не следует приступать к автоматизации, пока интерфейс нестабилен
- К автоматизации следует приступать тогда, когда ручные тесты достигли некоторого заданного (высокого) уровня качества. В противном случае мы в дополнение к «ручному хаосу» получим «автоматизированный хаос». Автоматизация тестирования требует наличия инструментальных средств, соответствующих поставленной задаче
- Автоматизация тестирования требует подхода, позволяющего гибко адаптировать автоматизированные тесты к изменениям в приложении. Следует помнить о проблемах синхронизации средства автоматизированного тестирования и тестируемого приложения по времени (иными словами – средство автоматизации тестирования должно уметь «понимать», перешло ли приложение в новое состояние, а не выполнять операции вслепую)

Достоинства автоматизации

- **Скорость**. Да, компьютер, в общем случае, работает быстрее человека.
- **Надёжность**. Да, компьютер допускает меньше ошибок в силу отсутствия т.н. «человеческого фактора».
- **Мощность**. Например, человек никогда не сможет вручную добавить в БД 100 миллиардов записей, состоящих из 500 полей каждая.

Преимущества перед ручным тестированием

- Автоматизированные тесты выполняются быстрее, а это значит, что можно сэкономить время и деньги.
- Средства автоматизации тестирования прекрасно умеют собирать числовую (в т.ч. статистическую) информацию и представлять её в удобной для понимания человеком форме.
- Средства автоматизации тестирования в сложных ошибочных ситуациях способны обратиться к DLL, БД, удалённому серверу и т.п. по альтернативным путям (например, в обход стандартного клиента), что позволяет диагностировать сложные ошибки в клиент-серверных и распределённых приложениях.

Недостатки автоматизации тестирования

- Автоматизированное регрессионное тестирование реже обнаруживает новые ошибки, т.к. тесты (особенно, в случае жёстко заданных тестовых данных и путей выполнения) проходят строго по одному и тому же пути. В случае качественной работы команды разработчиков основное внимание на проекте следует уделять новой функциональности, в то время как автоматизированные тесты чаще всего направлены на проверку ранее реализованной.
- Недостаточная продуманность вопросов поддержки автоматизированных тестов приводит к тому, что в случае серьёзных изменений в приложении многие автоматизированные тесты становятся бесполезными и/или требуют серьёзной переработки.
- На автоматизацию часто возлагают неоправданные надежды, что ведёт к срывам сроков и прочим проблемам.
- Сложно и нецелесообразно автоматизировать проверку графического интерфейса (верстки, расположения элементов на экране и т.д.).

Почему не всё следует автоматизировать

- Итак, регрессионное тестирование и иные часто повторяющиеся тесты наилучшие кандидаты на автоматизацию. Следующим наилучшим кандидатом на автоматизацию являются длительные утомительные для человека тесты, не требующие сложной логики для выполнения и анализа результатов (заполнение больших форм, проверка корректности ссылок и т.д.).
- **НЕ следует автоматизировать тесты со сложной логикой оценки времени выполнения** (особенно в веб-приложениях). Даже современные средства автоматизации зачастую неспособны верно оценить происходящее с приложением и выдают неверные результаты.
- **НЕ следует автоматизировать тесты, требующие постоянного вмешательства оператора**.
- **НЕ надо автоматизировать всё подряд**!

Процесс автоматизации

- Процесс автоматизации – долгий и дорогой процесс. Как правило, на разработку автоматизированных тестов уходит в 5-10 раз больше времени, чем на создание и однократное выполнение аналогичных ручных тестов.
- Поэтому, приступая к автоматизации, следует учесть:
  - ✓ Затраты времени на ручное выполнение тестов
  - ✓ Количество повторений тестов
  - ✓ Затраты времени на отладку и обновление автоматизированных тестов
  - ✓ Затраты времени на выполнение автоматизированных тестов
  - ✓ Затраты времени на поддержку автоматизированных тестов

ИТАК, подводя итог:

- Автоматизация требует ручного тестирования
- Не следует приступать к автоматизации, если времени не хватает даже на ручное тестирование
- Автоматизация требует полного переключения на неё части команды тестировщиков, а потому следует хорошо продумывать использование человеческих ресурсов
- Автоматизация – это не только выполнение тестов, это ещё и управление, поддержка, написание отчётов об ошибках и о результатах тестирования, управление тестовым окружением и т.д.
- Каждый проект требует своего подхода к автоматизации
- Не следует пытаться автоматизировать всё подряд
- Следует управлять процессом автоматизации как отдельным проектом внутри проекта
- При разработке автоматизированных тестов следует особое внимание уделять вопросам поддержки тестов

Технология Record&Playback

- Одной из наиболее распространённых и простых для понимания технологий автоматизации тестирования является технология **Record&Playback** («Записать и воспроизвести»)
- Суть данной технологии заключается в том, что средство автоматизации тестирования позволяет выполнить с тестируемым приложением некоторый набор действий, которые будут записаны на специальном языке программирования (аналогично работает запись макросов в приложениях Microsoft Office, где макрос записывается на языке Visual Basic)

Различные популярные средства автоматизации

- Selenium
- Appium
- Sikuli
- TestComplete
- Katalon Studio

Selenium

Selenium - это набор средств автоматизации тестирования, который состоит из различных компонентов:

- ✓ **Selenium Core** это ядро (движок) для запусков тестов, на котором основывается вся линейка инструментов Selenium.
- ✓ **Selenium RC** *Remote Control* был основным проектом Selenium, который длился долгое время до появления Selenium WebDriver
- ✓ **Selenium IDE** расширение к браузеру, которое позволяет записывать, редактировать, отлаживать и выполнять тесты
- ✓ **Selenium Grid** это инструмент, который распределяет тесты на нескольких физических или виртуальных машинах, чтобы выполнять скрипты параллельно
- ✓ **Selenium Webdriver** программная библиотека, которая позволяет разрабатывать программы, управляющие поведением браузера

Selenium IDE

Загрузить Selenium IDE можно в браузерах Chrome, Firefox и Edge

Начало использования

- Создан на основе библиотеки, написанной на JacaScript в 2004 году. До 2017 года работал исключительно только в браузере Firefox
- Firefox, начиная с версии Firefox 55, перестал поддерживать Selenium IDE .(2017)
- В 2018 году использование Selenium IDE вновь стало возможным.

НО! Теперь работать с ним можно не только в браузере Firefox и в Chrome.

С 2021 работа возможна в браузере Microsoft Edge.

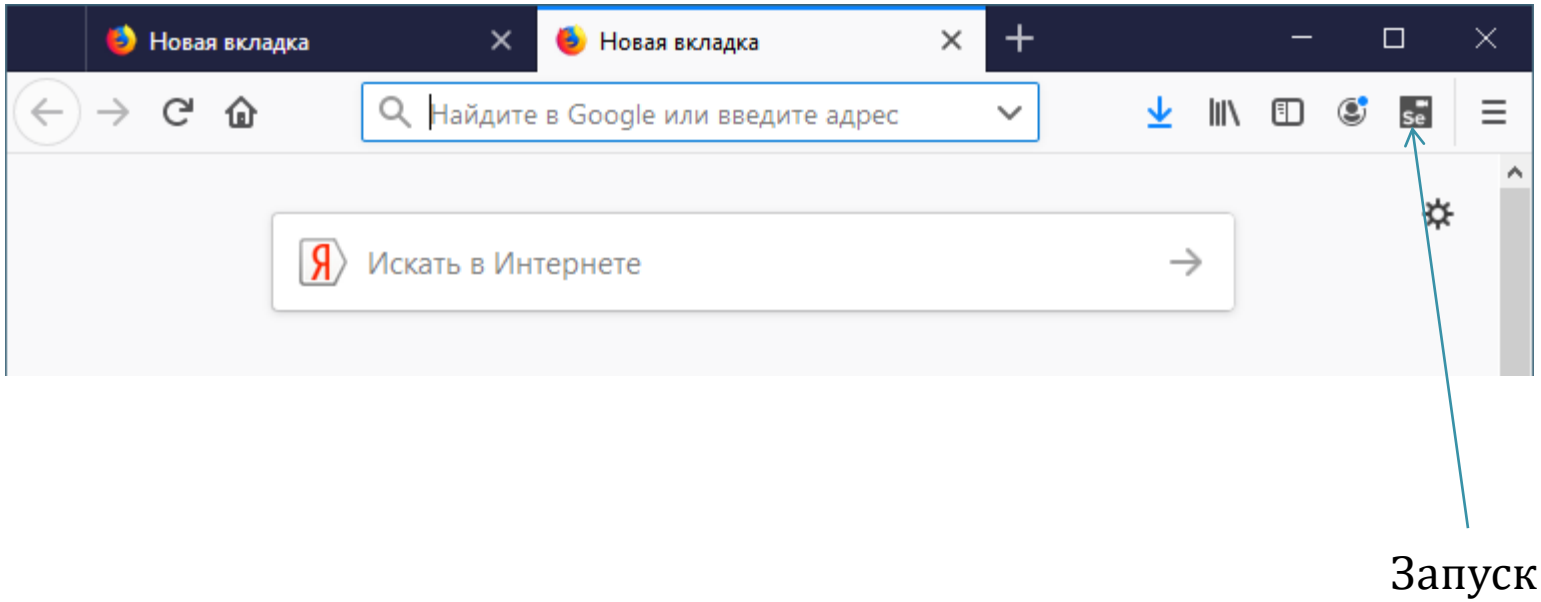
- https://www.seleniumhq.org - официальный сайт

Selenium IDE

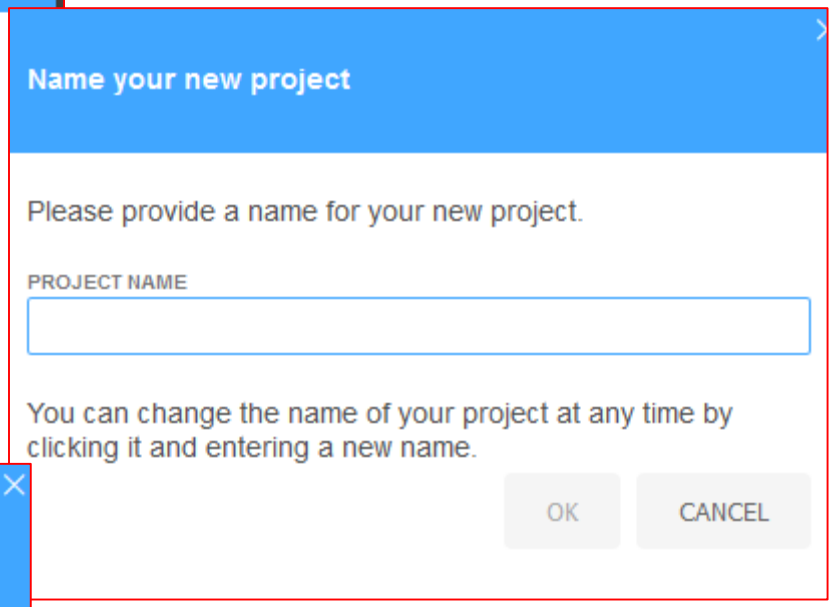
Это интегрированная среда для разработки и выполнения скриптов, представленная в виде расширения для браузеров FireFox, Chrome и Edge.

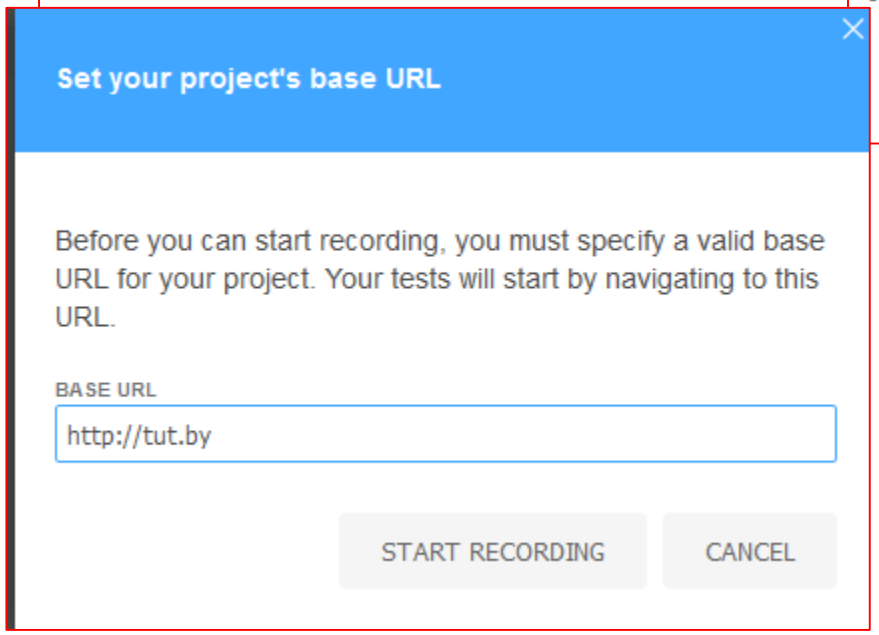
Запуск Selenium IDE

После установки расширение **Selenium IDE** доступен в браузере на панели инструментов:



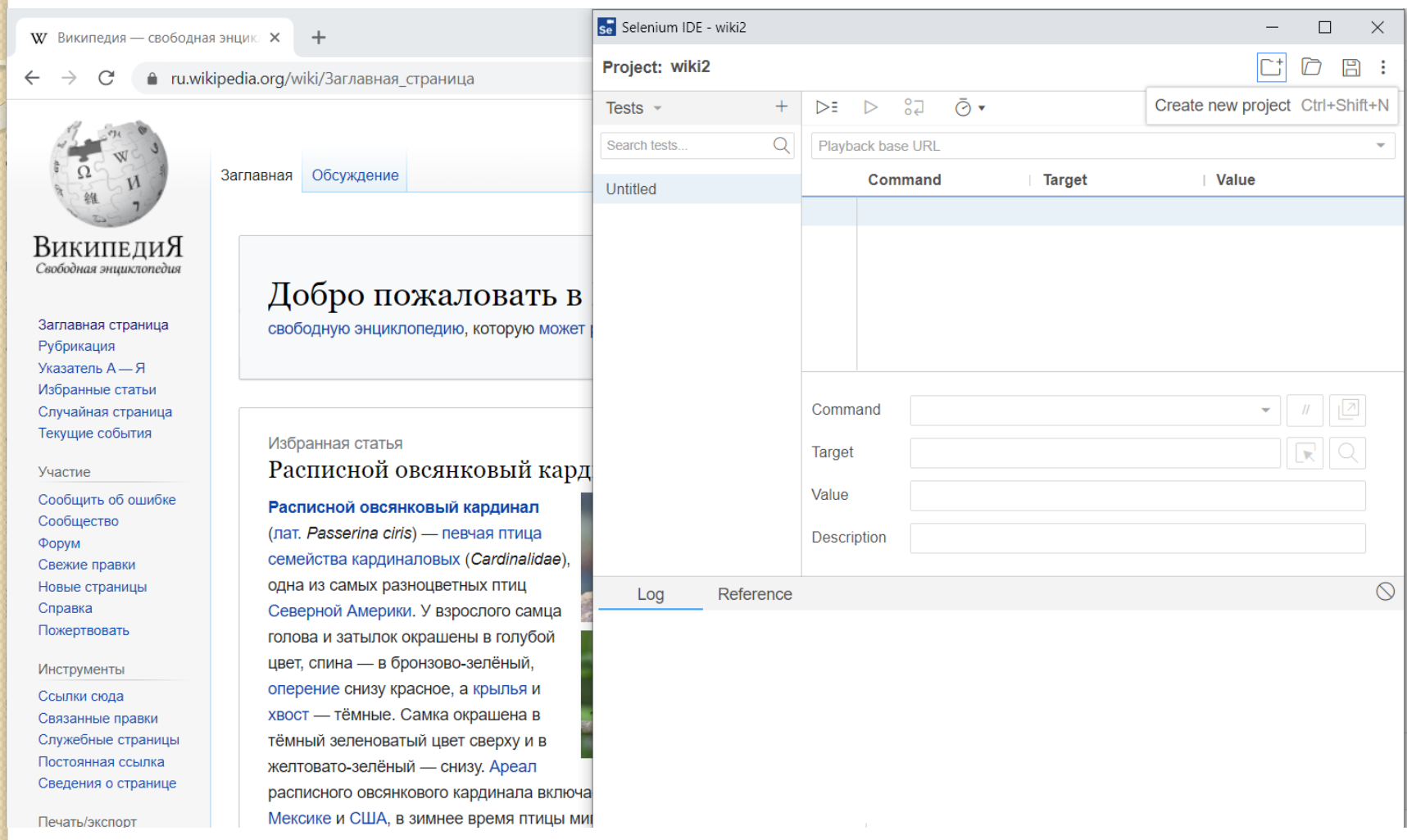
Начало работы

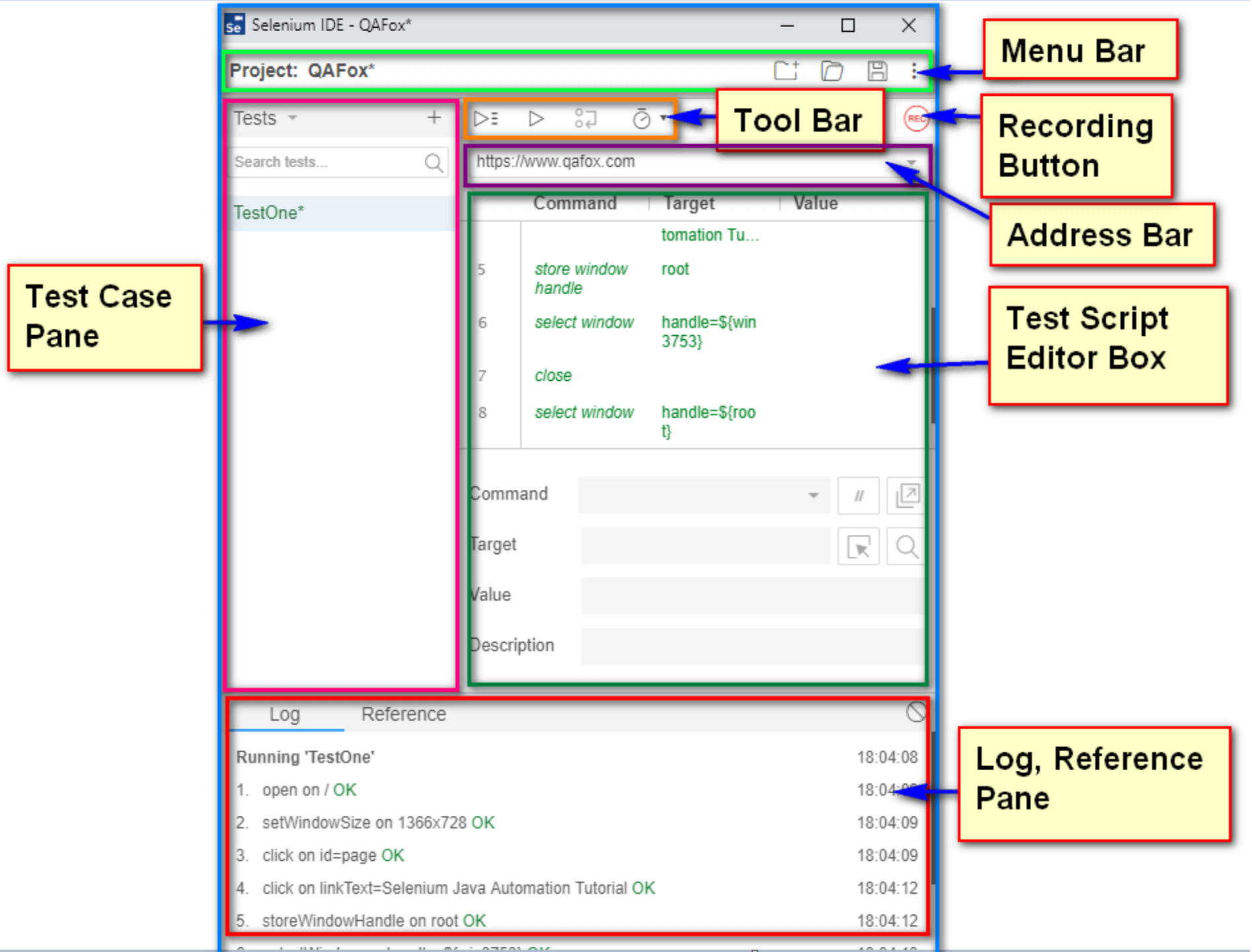




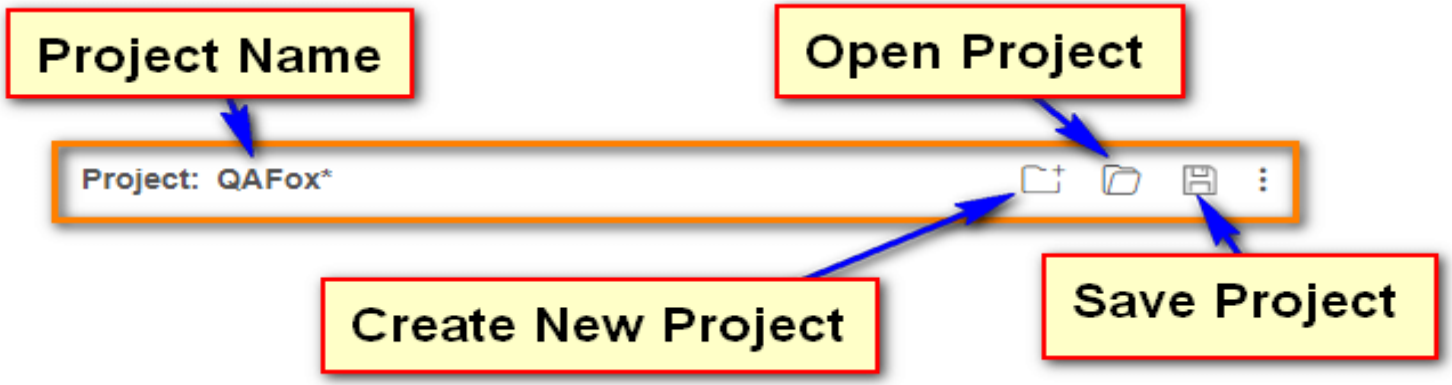
После этого откроется окно браузера с указанным Web-приложением и расширением **Selenium IDE**

Начало работы




Selenium IDE - QAFox*
Project: QAFox*
Menu Bar
Tool Bar
Recording Button
Address Bar
Test Case Pane
Test Script Editor Box
Log, Reference Pane
Tests
Search tests...
TestOne*
https://www.qafox.com
Command
Target
Value
tomation Tu...
5 store window handle root
6 select window handle=${win 3753}
7 close
8 select window handle=${roo t}
Command
Target
Value
Description
Log
Reference
Running 'TestOne' 18:04:08
1. open on / OK
2. setWindowSize on 1366x728 OK 18:04:09
3. click on id=page OK 18:04:09
4. click on linkText=Selenium Java Automation Tutorial OK 18:04:12
5. storeWindowHandle on root OK 18:04:12

Selenium IDE. Строка меню



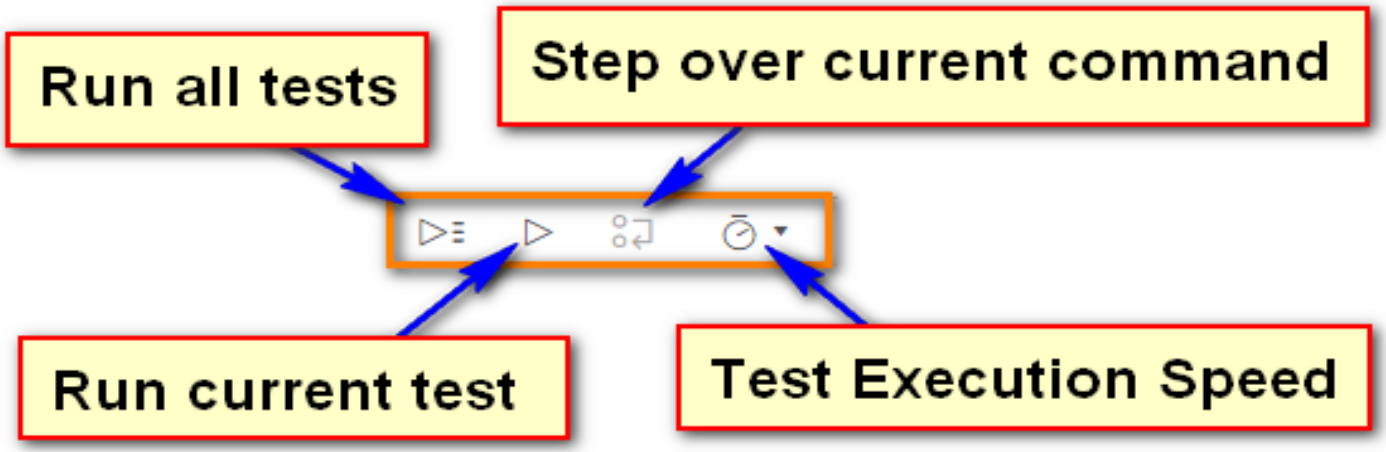
Имя проекта: здесь будет отображаться имя, которое вы дали при создании нового проекта в Selenium IDE. Мы также можем переименовать имя проекта, щелкнув существующее отображаемое имя.

Создать новый проект: если вы хотите создать новый проект, нам нужно использовать эту опцию.

Открыть проект: если на вашем компьютере сохранен какой-либо существующий проект Selenium IDE, вы можете использовать эту опцию, чтобы открыть проект.

Сохранить проект: после того, как вы внесли какие-либо изменения в проект, вы можете использовать эту опцию, чтобы сохранить проект на своем компьютере.

Selenium IDE. Панель инструментов



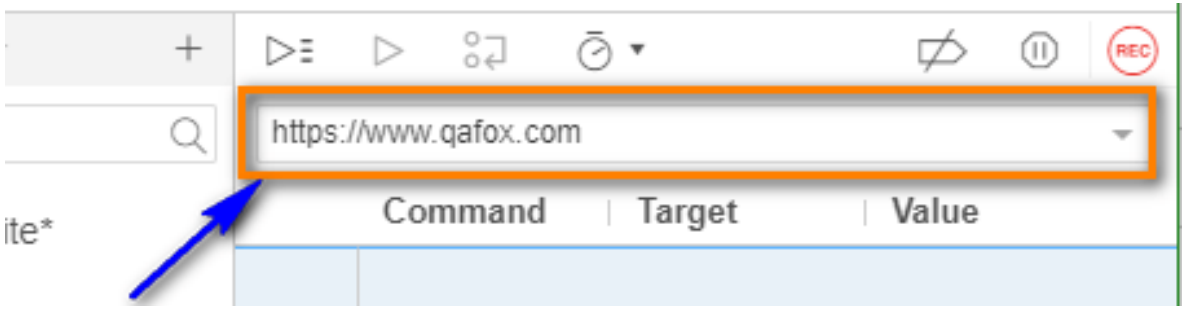
Запустить все тесты: с помощью этой опции можно выполнить все тесты в наборе тестов одновременно.

Запустить текущий тест: с помощью этой опции можно запустить текущий загруженный тест.

Шаг за шагом: используется для пошагового использования варианта использования. Он в основном используется для отладки вариантов использования.

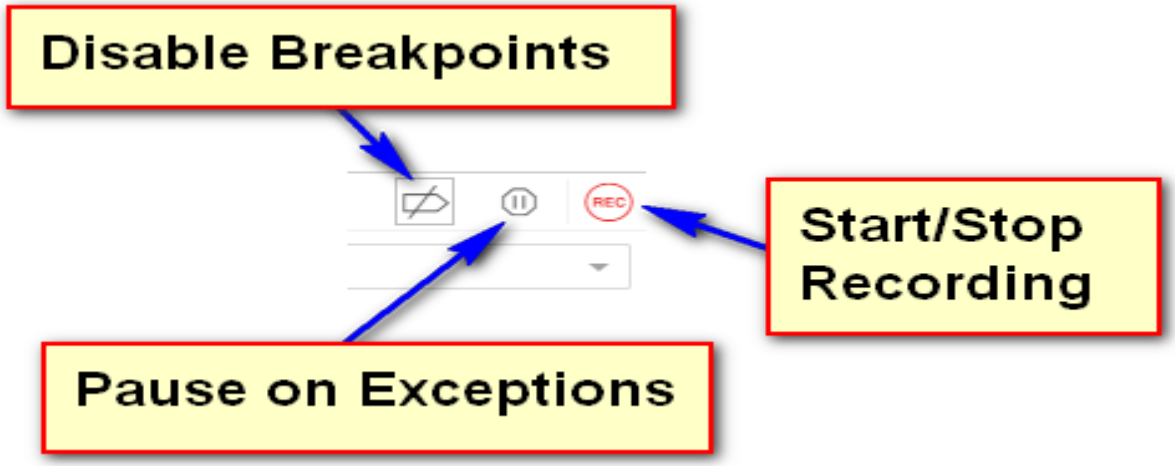
Контроль скорости: используется для контроля скорости выполнения сценария использования.

Selenium IDE. Адресная строка



В адресной строке отображается текущий URL-адрес, а при нажатии связанного с ним раскрывающегося списка отображаются все ранее посещенные веб-сайты в рамках автоматизации.

Selenium IDE. Доп. инструменты

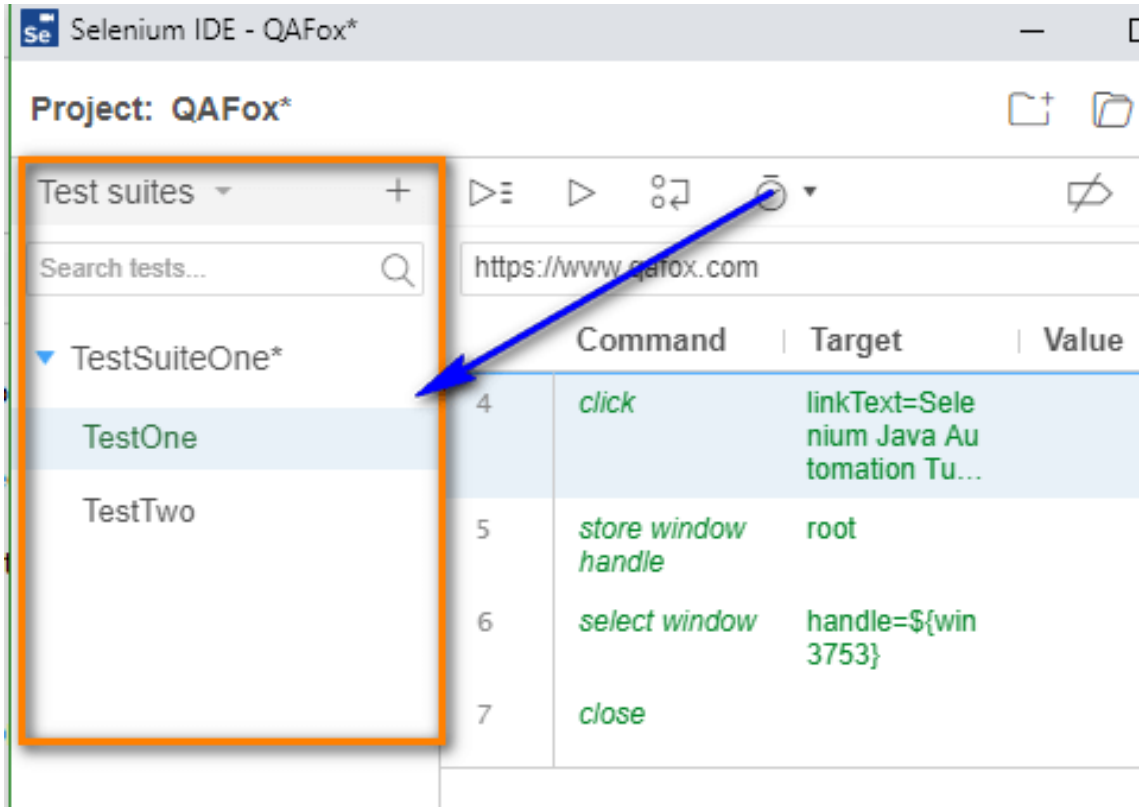


Отключить точки останова: все шаги, для которых установлены точки останова для отладки, не будут учитываться во время выполнения при использовании этой опции.

Пауза при исключениях: если во время выполнения возникают какие-либо исключения, вы можете использовать эту опцию, чтобы остановить тестовые случаи при получении каких-либо исключений.

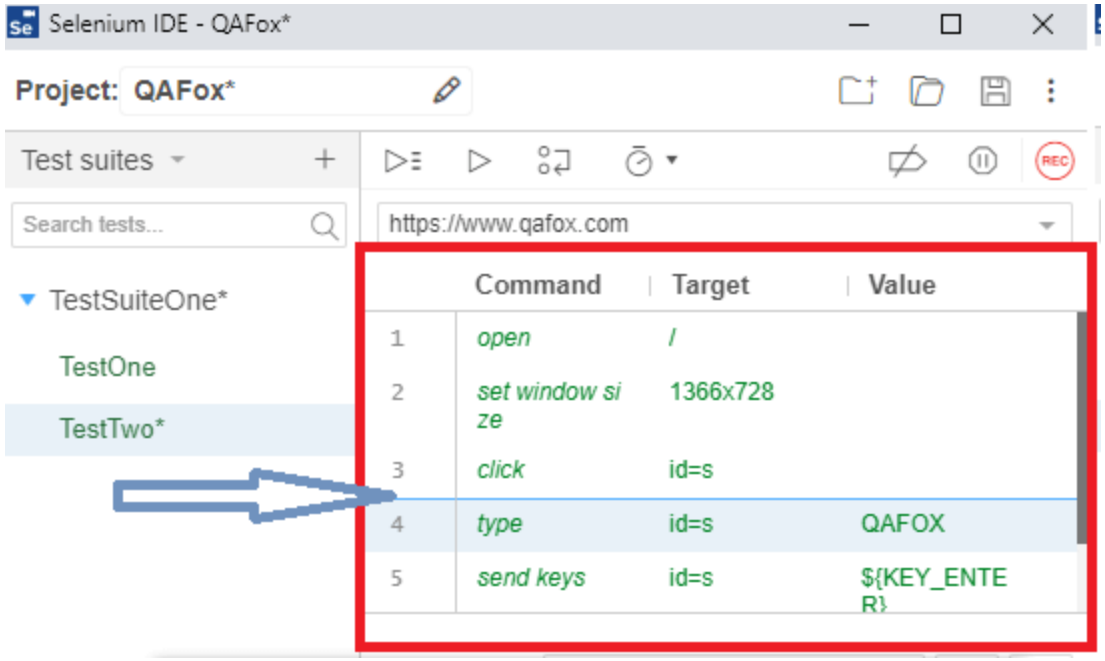
Начать/остановить запись: действия, выполняемые в приложении, будут записываться, когда эта запись включена, и не будут записываться при остановке этой опции записи.

Selenium IDE. Панель тестовых сценариев

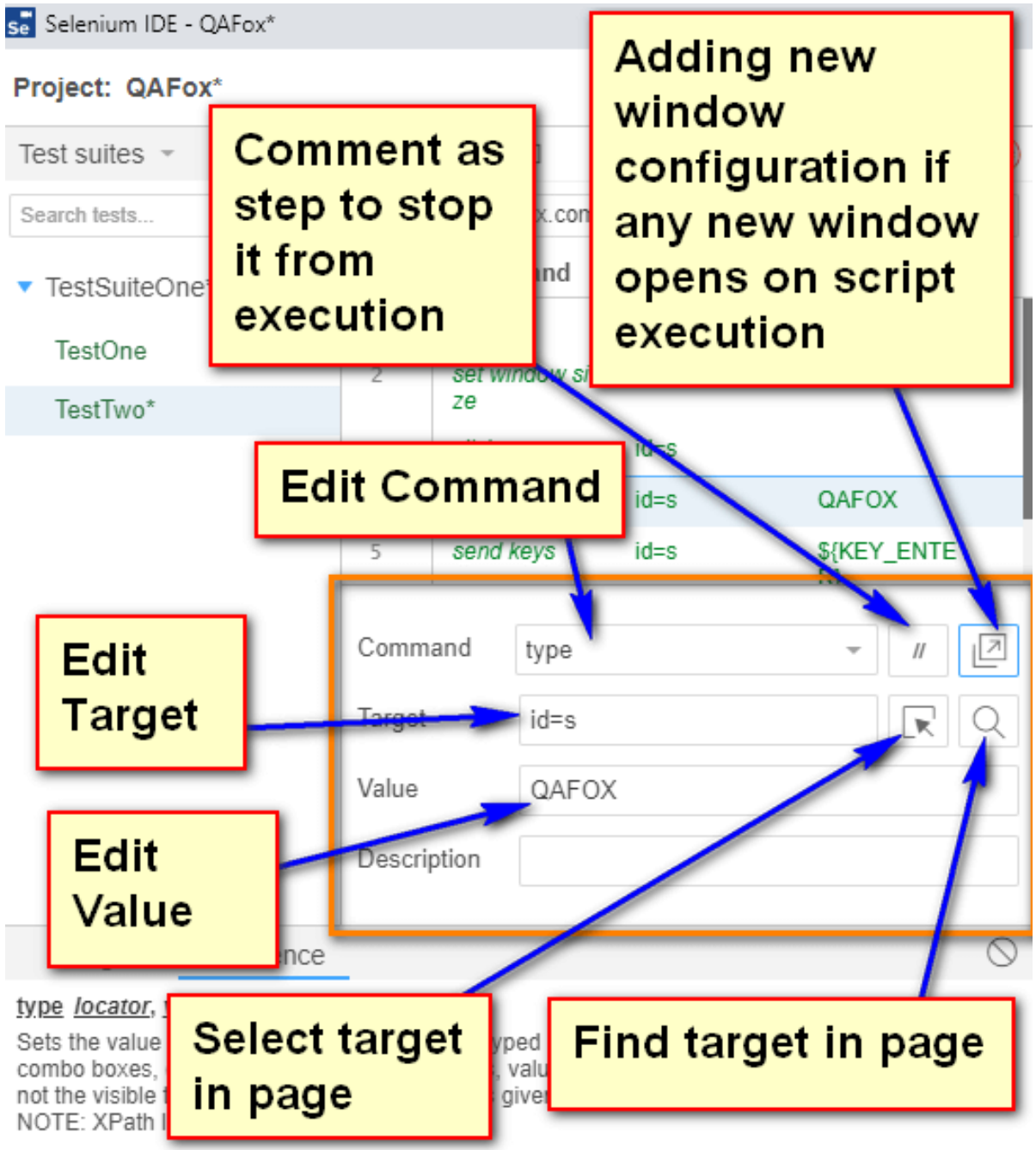


Можно добавить несколько наборов тестов и несколько тестов в разделе «Наборы тестов» в этой панели тестовых сценариев.
Можно выбрать тесты для выполнения на панели Test Case.
Таким образом, панель Test Case содержит список записанных тестов, организованных в разные наборы тестов.

Selenium IDE. Редактор тестовых сценариев



Окно редактора тестовых сценариев содержит действия пользователя, которые записываются в виде шагов тестирования в Selenium IDE. Каждый записанный шаг теста содержит значения в трех столбцах, т. е. Command, Target и Value.


Selenium IDE - QAFox*
Project: QAFox*
Test suites
Search tests...
TestSuiteOne
TestOne
TestTwo*
Comment as step to stop it from execution
Adding new window configuration if any new window opens on script execution
Edit Command
Edit Target
Edit Value
Select target in page
Find target in page
Command
type
Target
id=s
Value
QAFOX
Description
send keys
id=s
${KEY_ENTE
arguments:

Selenium IDE. Редактор тестовых сценариев

Включить/отключить комментарий: можно закомментировать выбранный шаг, чтобы остановить его выполнение, включив эту опцию.
Добавить новую конфигурацию окна: если при выполнении текущего выбранного шага открывается какое-либо новое окно, можно предоставить более подробную информацию для настройки нового окна с помощью этой опции.
Редактировать команду: можно редактировать команду, отображаемую в поле «Команда».
Редактировать цель: можно редактировать текст локатора, отображаемый в поле цели.
Изменить значение: можно изменить значение, отображаемое в этом поле значения.
Выберите цель на странице: можно выбрать любой элемент пользовательского интерфейса на странице, используя эту опцию, чтобы сгенерировать текст локатора в поле «Редактировать цель».
Найти цель на странице: можно выбрать любой элемент пользовательского интерфейса на странице, используя эту опцию, чтобы сгенерировать текст локатора в поле «Редактировать цель».

Консольная панель

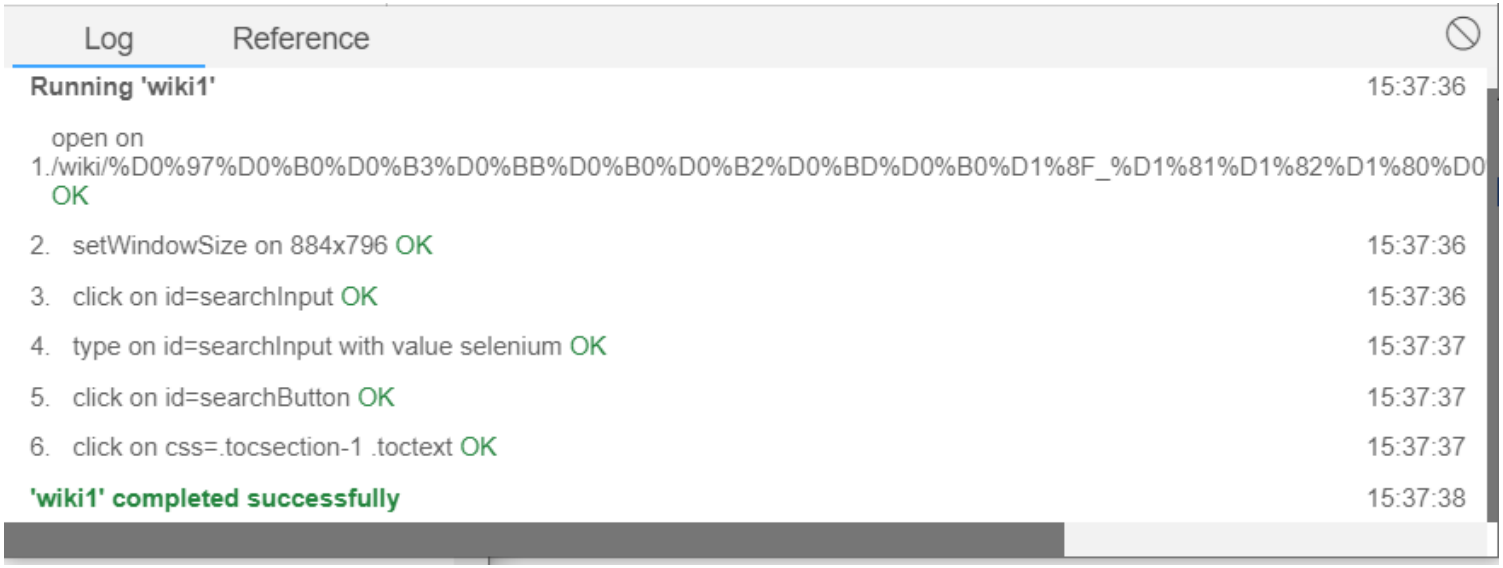
Консольной панелью используется для отображения служебной информации:

- Журнал (**Log**),
- Справочник (**Reference**)

в зависимости от того, какая вкладка выбрана.

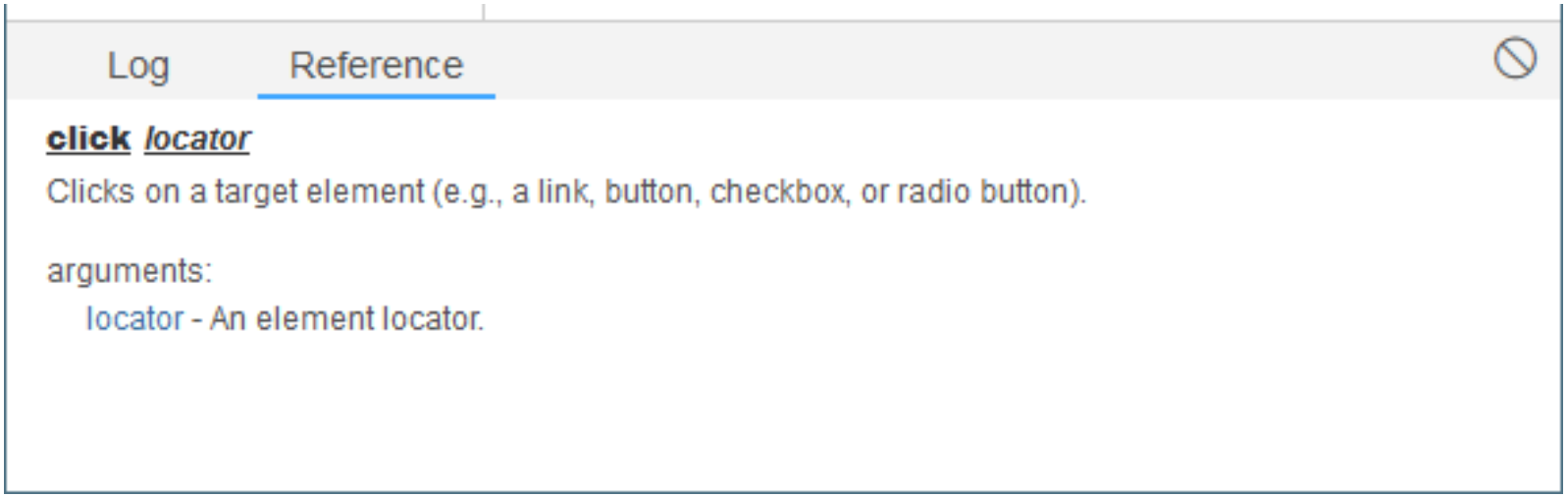
Log

Когда запускается тестовый сценарий, сообщения об ошибках и информационные сообщения показывают ход выполнения кейса и отображаются на этой панели автоматически, даже если сначала не выбрана вкладка «Журнал» (Log). Эти сообщения часто полезны для отладки тестовых случаев. Справа находится кнопка «Очистить» для очистки журнала.



Reference

При добавлении, редактировании или просмотре шагов теста на этой панели отображается информация о команде, которая в данный момент находится в фокусе. Детали, такие как имя, описание, какие аргументы он принимает, и детали этих аргументах.



Сценарий для записи действий

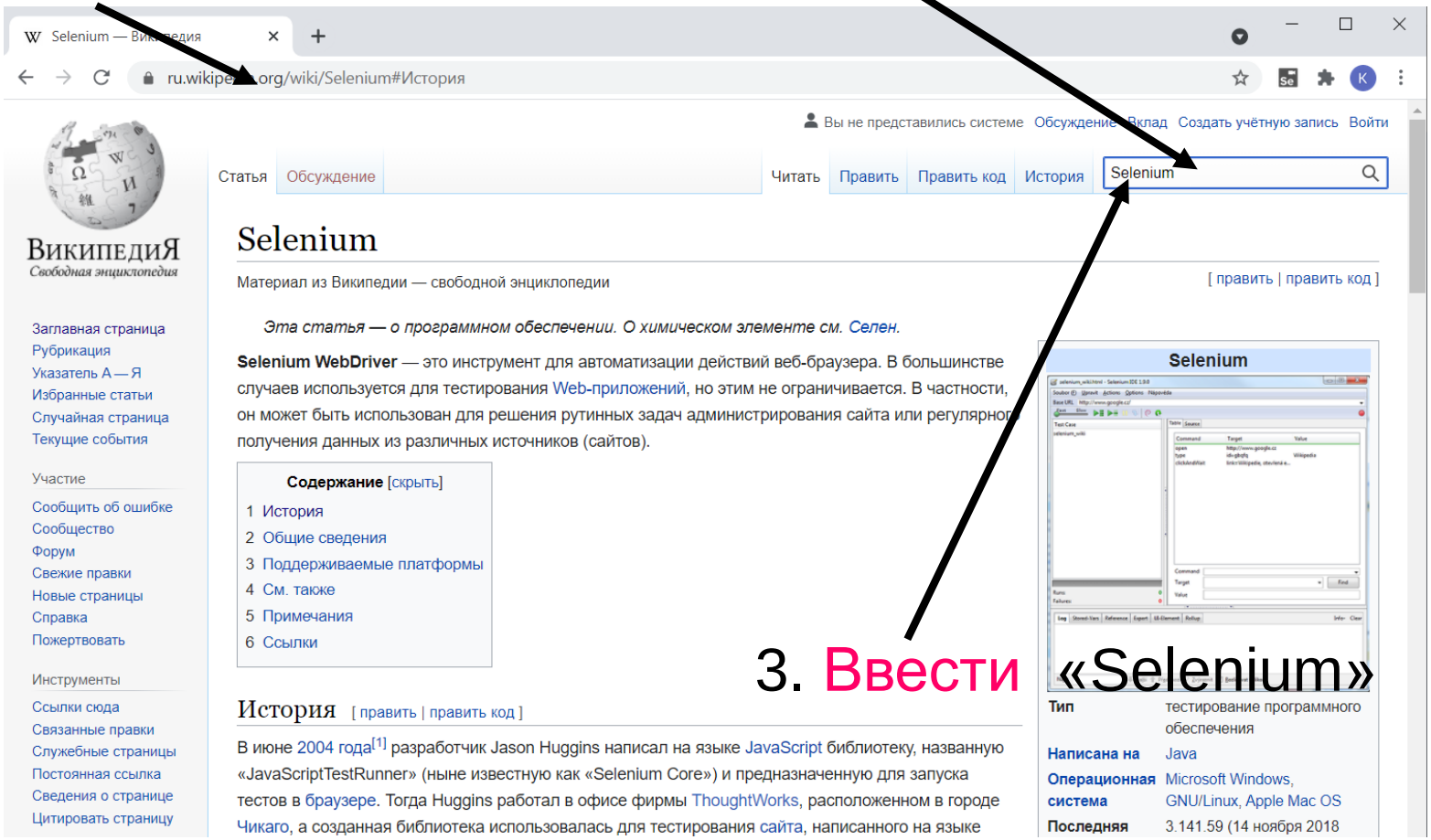
Для организации записи тест кейса в автоматическом режиме, необходимо выполнить действия по добавлению нового тестового набора и нового тест-кейса.

Далее нажать на панели инструментов кнопку **Record/Stop**

Сценарий для записи действий

1. Открыть https://ru.wikipedia.org/

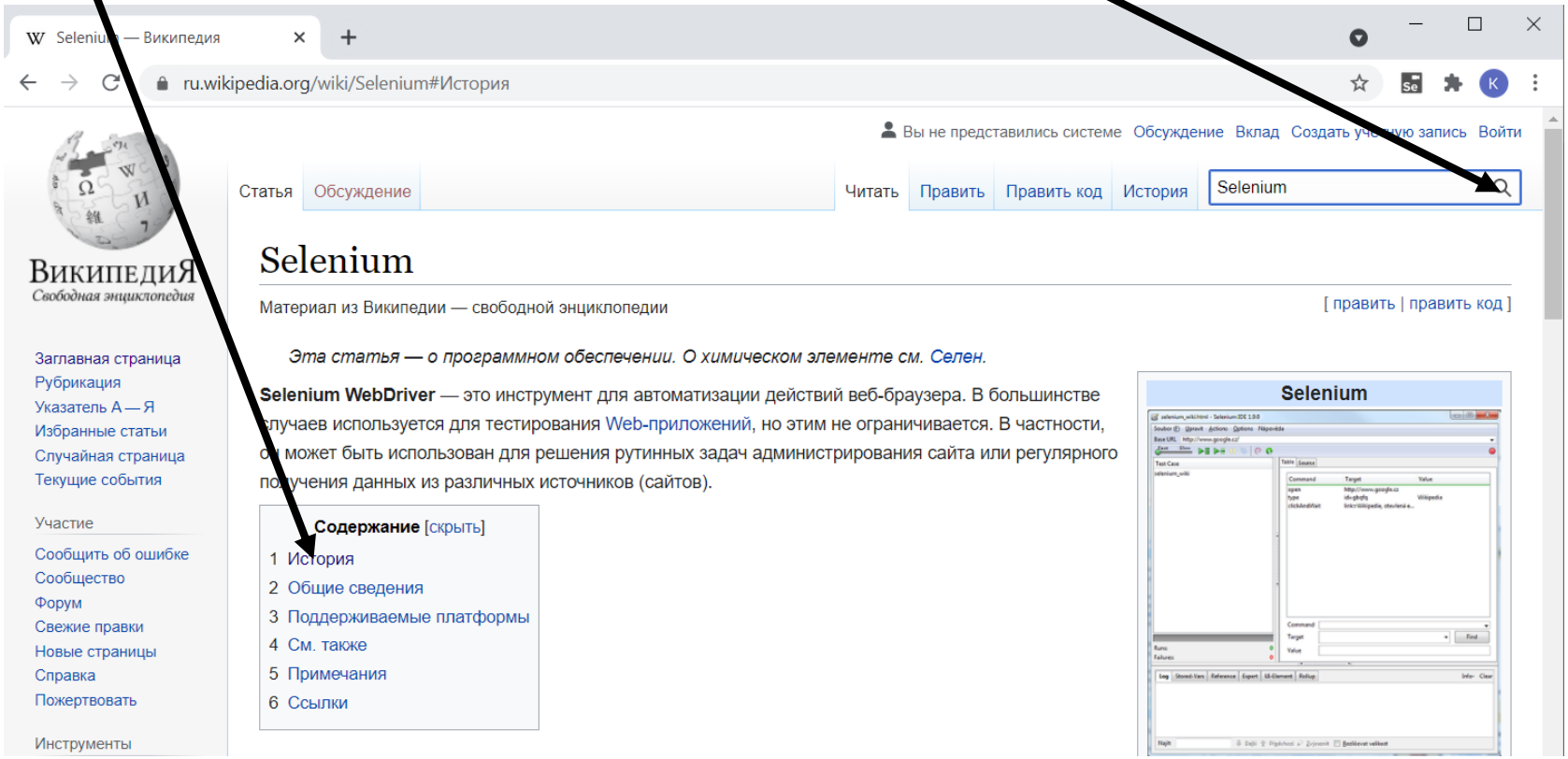
2. Кликнуть окно поиска



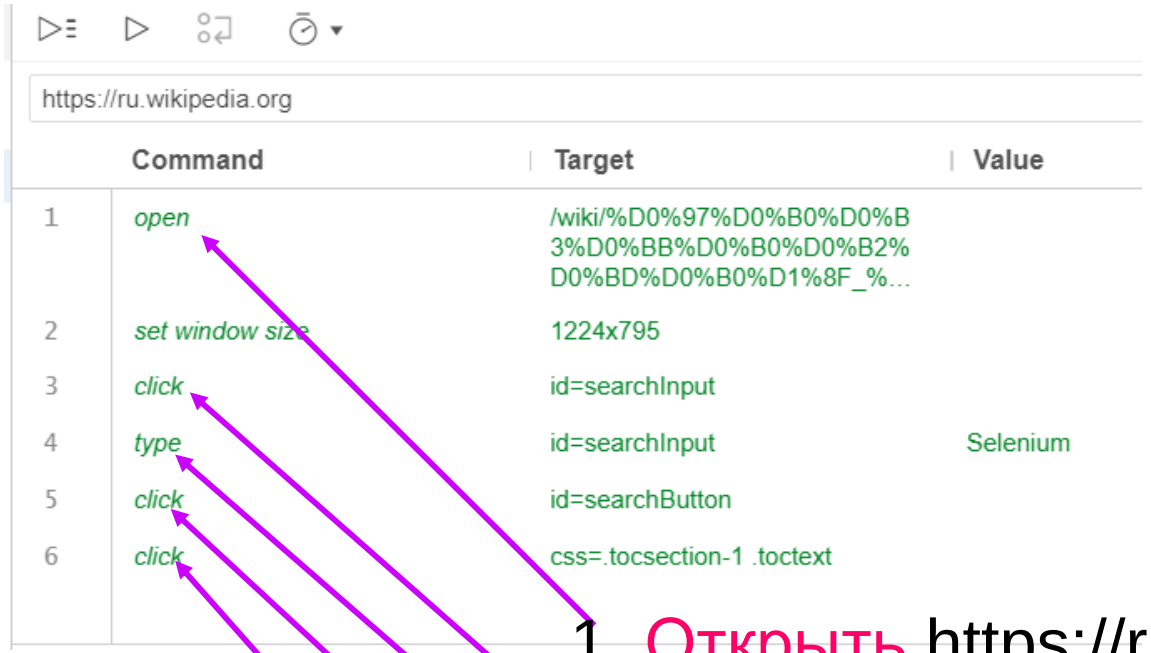
3. Ввести «Selenium»

Сценарий для записи действий

4. Нажать кнопку поиска
5. Перейти к «Истории»



Записанный скрипт

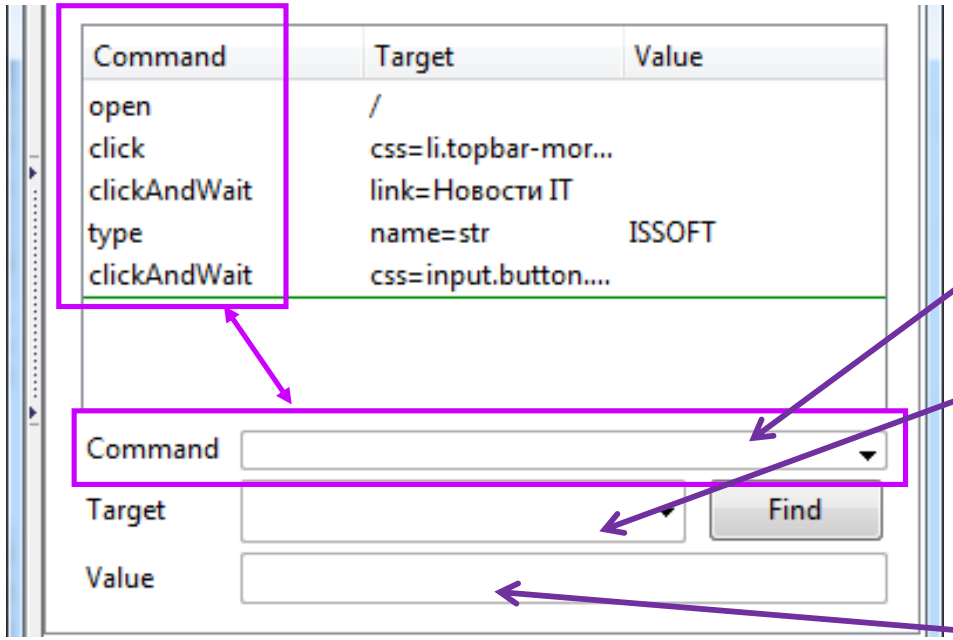


https://ru.wikipedia.org

	Command	Target	Value
1	*open*	/wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%...	
2	*set window size*	1224x795	
3	*click*	id=searchInput	
4	*type*	id=searchInput	Selenium
5	*click*	id=searchButton	
6	*click*	css=.tocsection-1 .toctext	

1. Открыть https://ru.wikipedia.org/
2. Кликнуть окно поиска
3. Ввести «Selenium»
4. Нажать кнопку поиска
5. Перейти к «Истории»

Структура команды



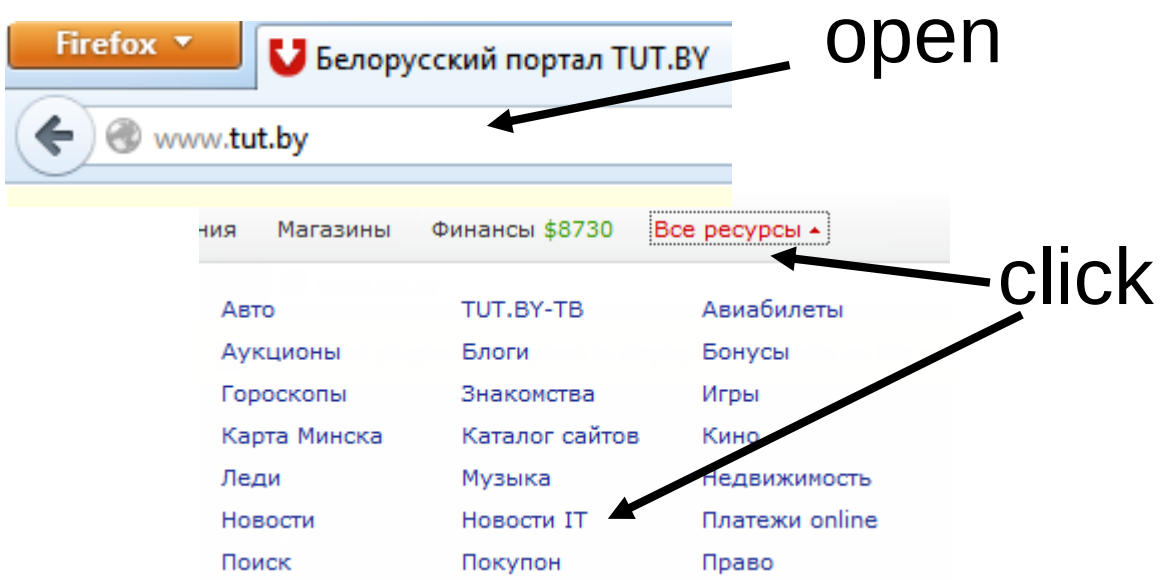
Поле «Command» содержит указание того, что необходимо выполнить на данном шаге теста.

Команда состоит из трех частей:

- ✓Какое **действие** необходимо сделать
- ✓Над каким **объектом** необходимо провести это действие
- ✓Нужны ли для этого **дополнительные данные**

Ключевое поле: **command** (действия)

Примеры действий (actions):



Ошибка (невозможность) выполнения любой из этих команд приводит к остановке теста!

Ключевое поле: **command** (действия)

Действия *(actions)* - это команды, которые управляют состоянием приложения. После выполнения, если действие не выполняется, выполнение текущего теста прекращается.

Команда	Описание
click (locator)	Щелчок по ссылке, кнопке, флажку или переключателю
clickAt (locator,coordString)	Щелчок по элементу с помощью локатора и координат
close()	Имитирует пользователя, нажимая кнопку «закрыть» в строке заголовка всплывающего окна или вкладки.
Double Click (locator)	Двойной щелчок на веб-элементе на основе указанного элемента.
dragAndDropToObject (Dragobject,dropobject)	Перетаскивает элемент и отбрасывает его на другой элемент.
echo (message)	Распечатывает указанное сообщение на консоли, которое используется для отладки.

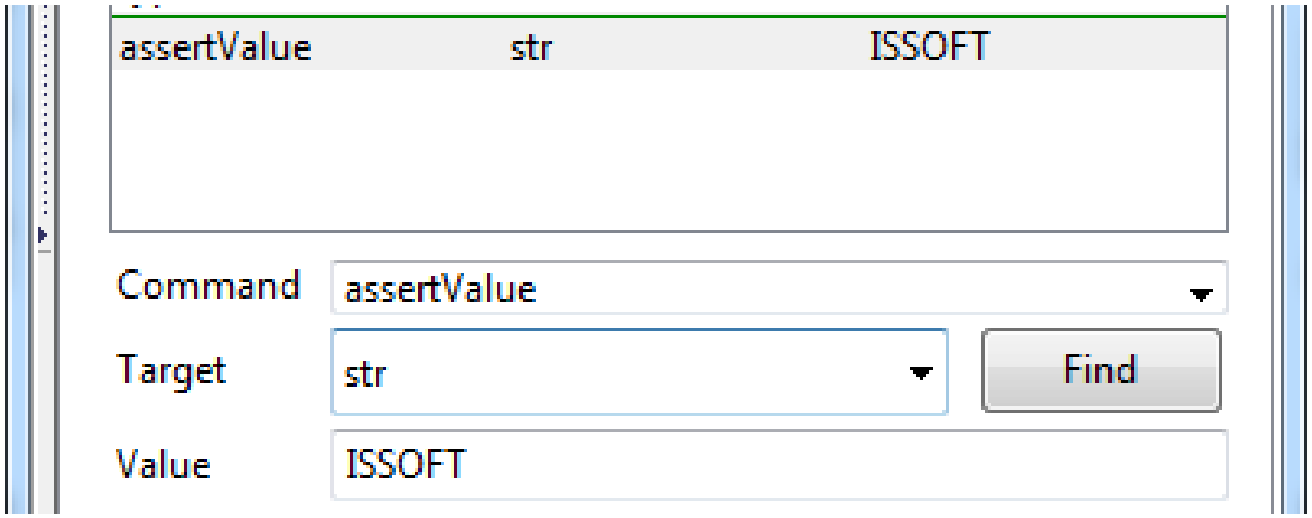
Ключевое поле: **command** (действия)

Команда	Описание
mouseDown (locator)	Имитирует пользователя, нажимая левую кнопку мыши на указанном элементе.
mouseDownAt (locator,coordString)	Имитирует пользователя, нажимая левую кнопку мыши в указанном месте на указанном элементе.
mouseUp (locator)	Имитирует событие, которое происходит, когда пользователь отпускает кнопку мыши
mouseUpAt (locator,coordString)	Имитирует событие, которое происходит, когда пользователь отпускает кнопку мыши в указанном месте.
open (url)	Открывает URL-адрес в указанном браузере и принимает как относительные, так и абсолютные URL-адреса.
pause (waitTime)	Ожидает заданное время (в миллисекундах)
select (selectLocator,optionLocator)	Выберите опцию из раскрывающегося списка с помощью локатора параметров.
selectWindow (windowID)	Выбирает всплывающее окно с помощью локатора окон; как только всплывающее окно выбрано, все фокусы сдвигаются в это окно.
store (expression, variableName)	Имя переменной, в которой должен быть сохранен результат, и выражение - это значение для хранения
type (locator,value)	Устанавливает значение поля ввода, аналогичное действию ввода пользователя.

Ключевое поле: **command (проверки)**

Проверки используются для анализа состояния веб-ориентированного приложения.

Например, можно проверять наличие того или иного элемента страницы, значение того или иного поля и т.п.



Утверждения позволяют проверять состояние приложения и сравнивать с ожидаемым. Они используются в трех режимах, а именно: - утверждать (**assert**), проверять (**verify**) и ждать (**waitfor**)

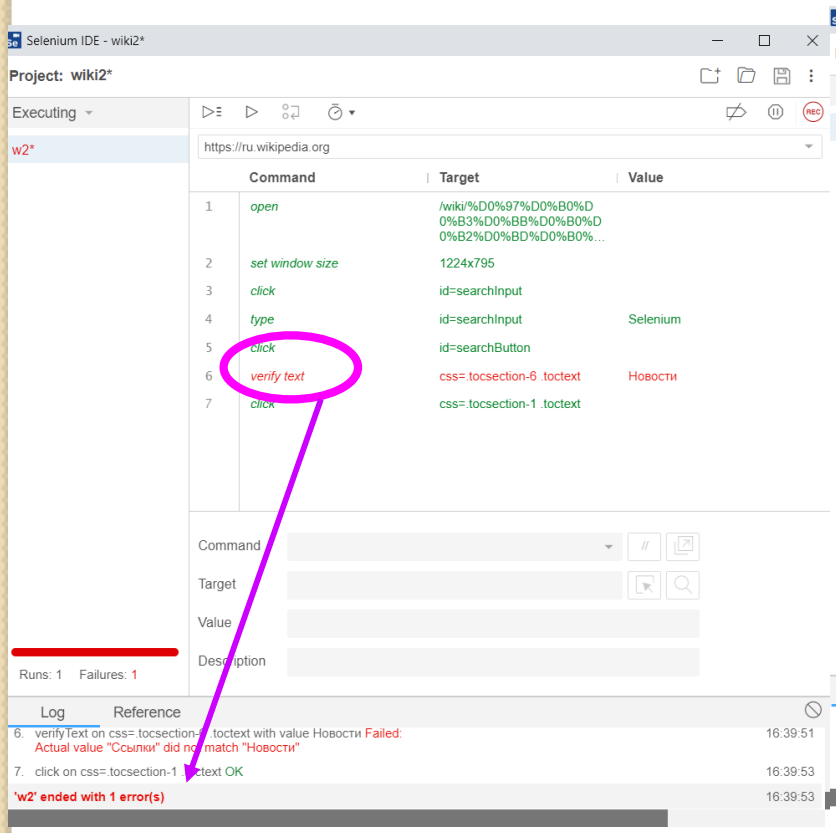
Ключевое поле: **command (проверки)**

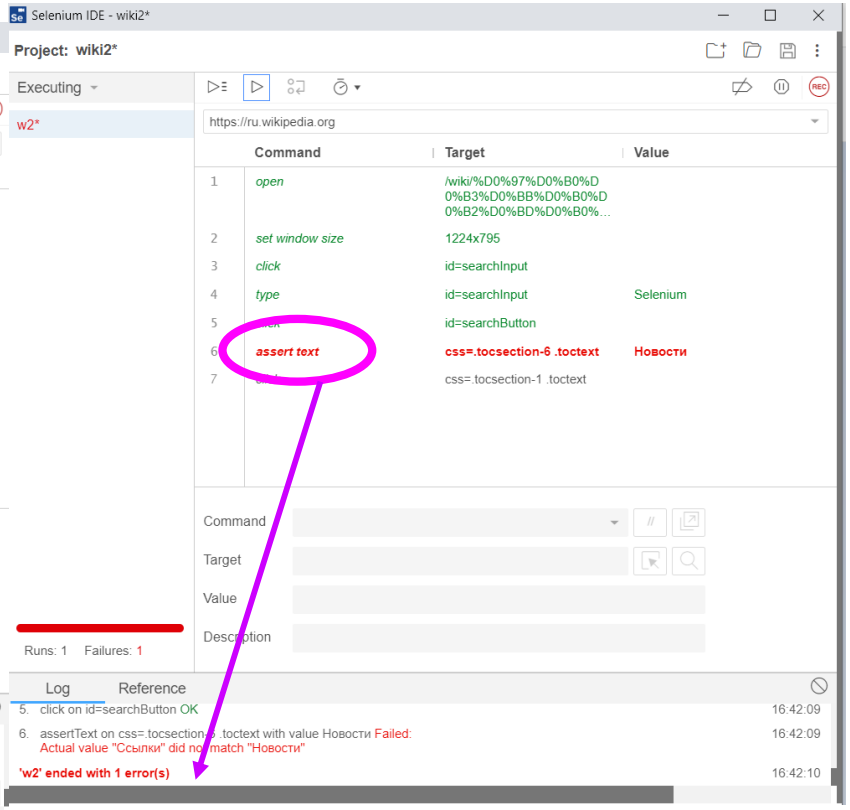
Команды, выполняющие проверку, делятся на два больших класса:

- начинающиеся с **assert**;
- начинающиеся с **verify**.

Ошибка	Ошибка
assert =	**verify** =
тест останавливается.	протоколируется факт наличия проблемы и тест продолжается.

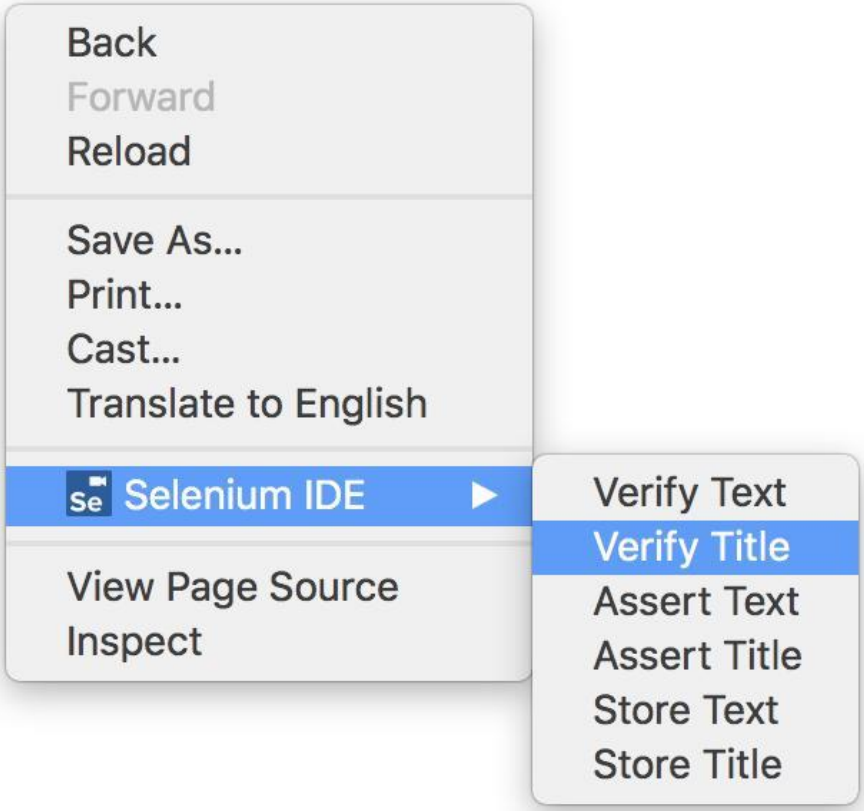
Пример


Selenium IDE - wiki2*
Project: wiki2*
Executing
w2*
https://ru.wikipedia.org
Command
Target
Value
1 open /wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%...
2 set window size 1224x795
3 click id=searchInput
4 type id=searchInput Selenium
5 click id=searchButton
6 verify text css=.tocsection-6 .toctext Новости
7 click css=.tocsection-1 .toctext
Runs: 1 Failures: 1
Log Reference
6. verifyText on css=.tocsection-6 .toctext with value Новости Failed: Actual value "Ссылки" did not match "Новости" 16:39:51
7. click on css=.tocsection-1 .toctext OK 16:39:53
'w2' ended with 1 error(s) 16:39:53


Selenium IDE - wiki2*
Project: wiki2*
Executing
w2*
https://ru.wikipedia.org
Command
Target
Value
1 open /wiki/%D0%97%D0%B0%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%...
2 set window size 1224x795
3 click id=searchInput
4 type id=searchInput Selenium
5 id=searchButton
6 assert text css=.tocsection-6 .toctext Новости
7 css=.tocsection-1 .toctext
Runs: 1 Failures: 1
Log Reference
5. click on id=searchButton OK 16:42:09
6. assertText on css=.tocsection-6 .toctext with value Новости Failed: Actual value "Ссылки" did not match "Новости" 16:42:09
'w2' ended with 1 error(s) 16:42:10

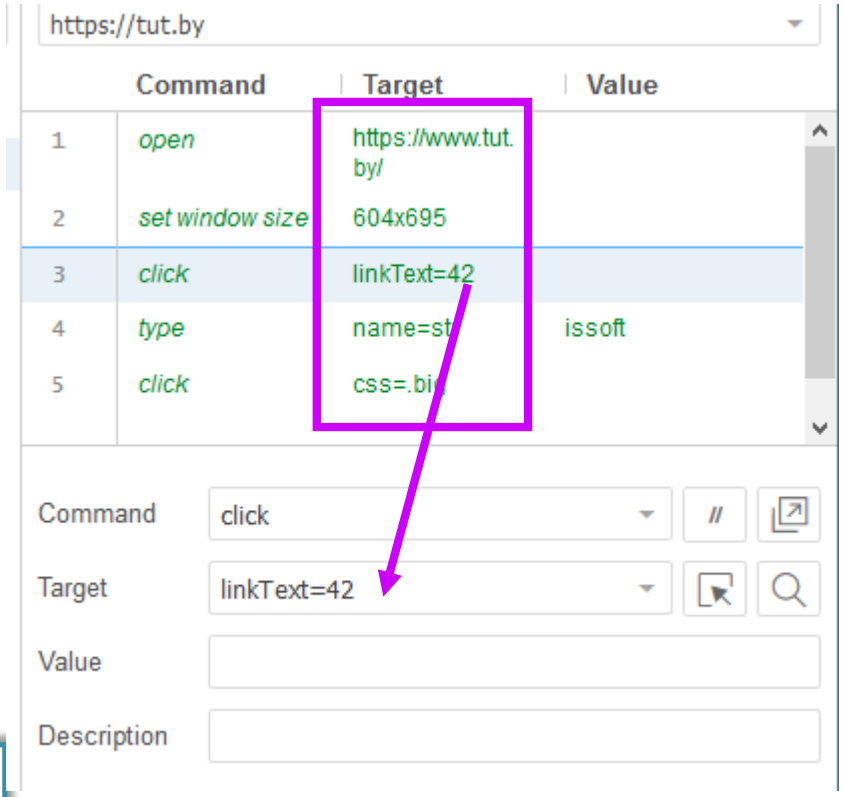
Добавление проверок

Пока записывается сценарий, щелкните правой кнопкой мыши в любом месте страницы. В результате откроется контекстное меню, показывающее команды проверки и / или подтверждения.



Ключевое поле: **target** (локатор)

Поле **target** указывает, с каким элементом следует выполнить действие.

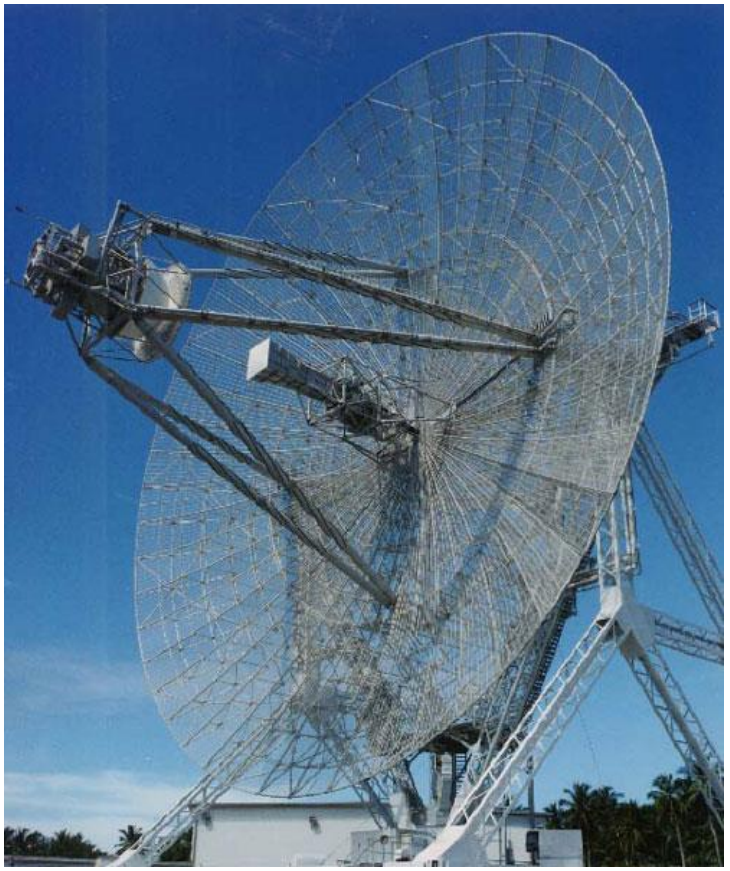


Формат локатора таков:
LocatorType = Argument

Ключевое поле: **target** (локатор)

Локаторы бывают следующих типов:

- **identifier = ID** или **NAME**
- **id = ID**
- **name = NAME**
- **DOM = JS Expression**
- **CSS = CSS Selector**
- **link = Link Text**
- **XPath = Xpath**



identifier - локаторы

Это наиболее универсальная стратегия поиска.
Selenium IDE ищет:

- Первый элемент с таким ID.
  - *(и если не находит)*
- Первый элемент с таким именем (name).

Пример: метод поиска вернет такие элементы (в скобках указаны номера строк):

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
  </form>
 </body>
<html>
```

✓identifier=loginForm (3)
✓identifier=password (5)
✓identifier=continue (6)
✓continue (6)

Так как тип локатора **identifier** является типом по умолчанию, то **identifier**= в первых трех примерах писать не обязательно.

id - локаторы

Такие локаторы позволяют однозначно определить элемент, если известен его **id**.

Selenium IDE ищет:

- Первый элемент с таким ID.
  - *(и если не находит)*
- Тест считается проваленным.

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
  </form>
 </body>
<html>
```

✓id=loginForm (3)

Имя элемента нужно смотреть в HTML-коде страницы.

name - локаторы

Такие локаторы позволяют определить элемент, если известно значение его атрибута **name** и/или **value** .

Selenium IDE ищет:

- Первый элемент с **name**.
- ИЛИ первый элемент с таким **name** и **value**.

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
<html>
```

✓name=username (4)
✓name=continue value=Clear (7)
✓name=continue Clear (7)
✓name=continue type=button (7)

Если у нескольких элементов одинаковое значение атрибута “**name**”, то тогда можно использовать фильтры, чтобы отсеять ненужные результаты. Тип фильтра по умолчанию – это значение атрибута “**value**”.

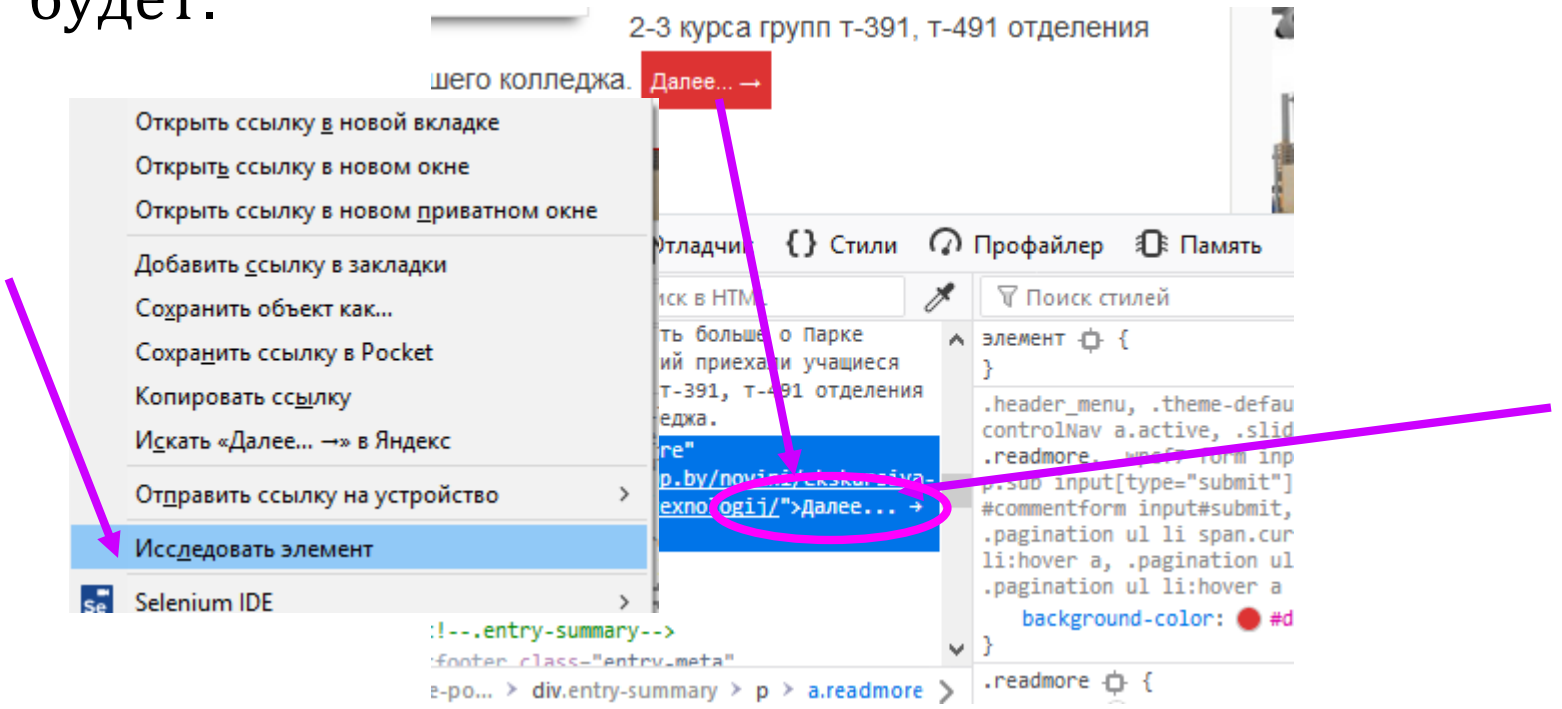
Локатор link = LinkText

Локатор вида

link = LinkText

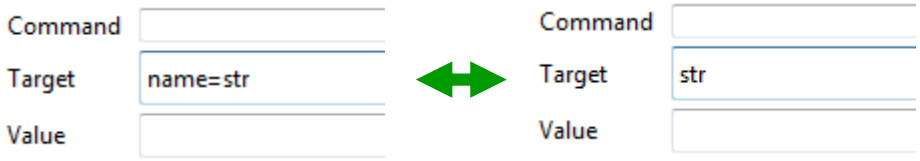
НЕЛЬЗЯ записывать просто в виде LinkText.

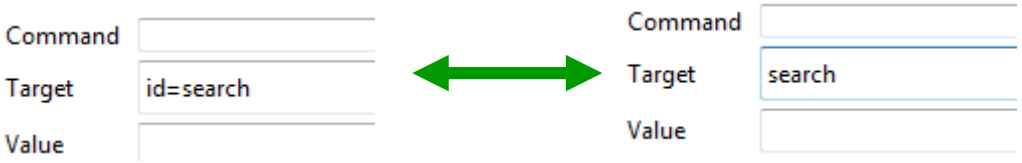
Текст ссылки чаще всего виден «невооружённым глазом», но если посмотреть его в коде – хуже не будет.



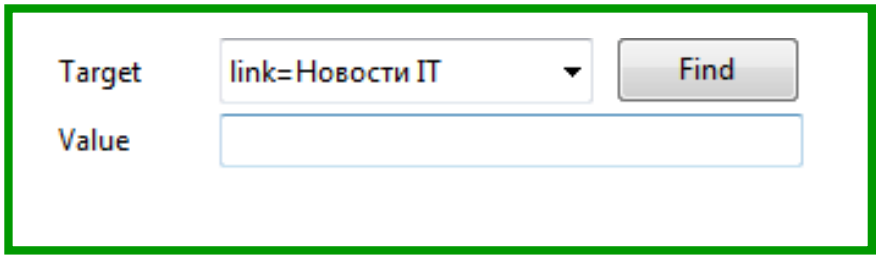
Локаторы

Лучше НЕ использовать сокращённую форму записи локаторов (это может привести к неоднозначности определения элемента), но если очень хочется…





Можно сокращать локаторы типа id или name, но НЕЛЬЗЯ сокращать локаторы типа link.



Локатор xpath = XPath

Является самым универсальным, но требует понимания XML и непосредственно **XPath**.

- **XPath** – это язык, который используется для нахождения элементов в XML документах.
- Одна из главных причин использования **XPath** – это отсутствие подходящего атрибута “id” или “name” для элемента, который нужно найти.
- Абсолютное выражение содержит в себе путь **XPath** начиная от корневого элемента (html), поэтому при малейшем изменении в приложении высока вероятность сбоя.

Нахождение с помощью XPath

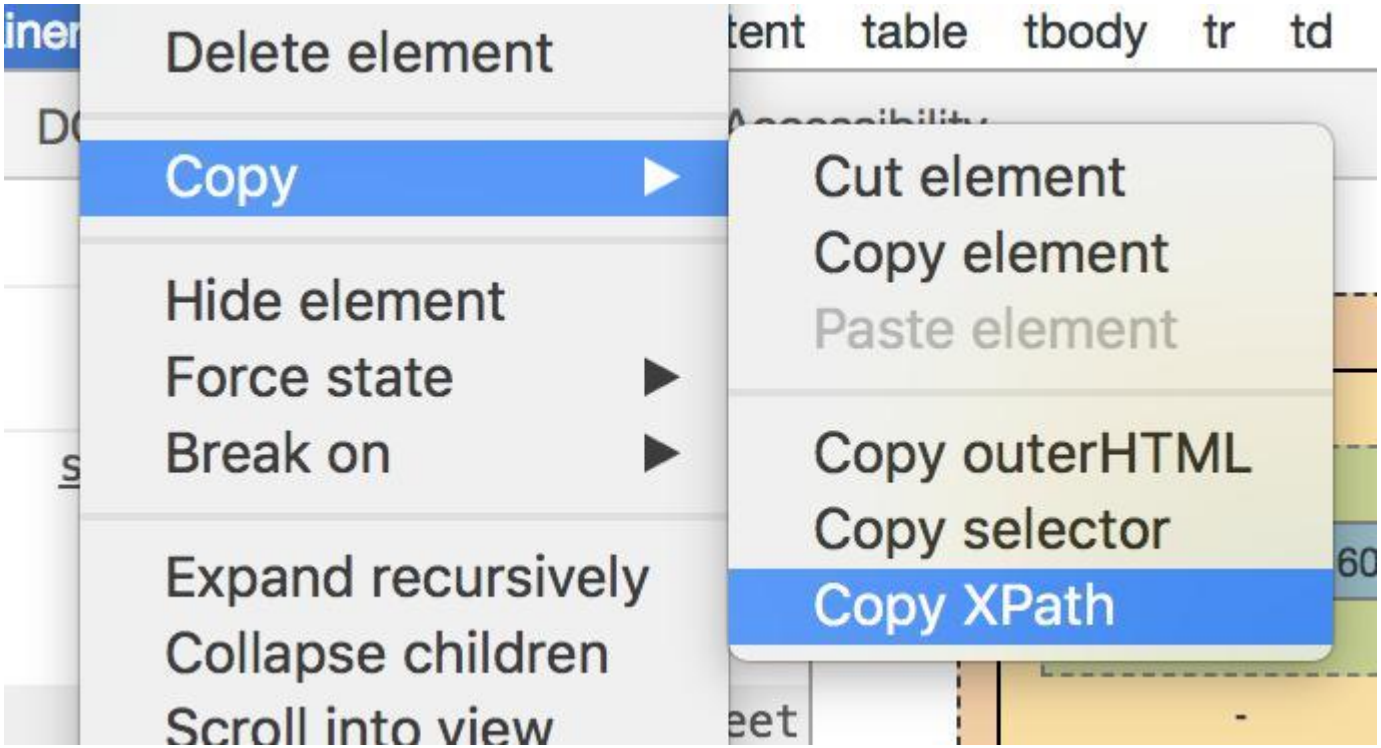
```
1
2
3
4
5
6
7
8
9
10
```

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
</html>
```

- `xpath=/html/body/form[1]` (3)
- `//form[1]` (3) – *Первая форма на странице*
- `xpath=//form[@id='loginForm']` (3) – *Форма с атрибутом “id”, имеющим значение “loginForm”*
- `xpath=//form[input/\@name='username']` (4) – *Форма, в которой есть поле ввода с атрибутом “name”, имеющим значение “username”*
- `//input[@name='username']` (4) – *Поле ввода с атрибутом “name”, имеющим значение “username”*
- `//form[@id='loginForm']/input[1]` (4) – *Первое поле ввода в форме с атрибутом “id”, имеющим значение “loginForm”*
- `//input[@name='continue'][@type='button']` (7) – *Поле ввода с атрибутом “name”, имеющим значение “continue”, и с атрибутом “type”, имеющим значение “button”*
- `//form[@id='loginForm']/input[4]` (7) – *Четвертое поле ввода в формес атрибутом “id”, имеюшим значение “loginForm”*

Нахождение с помощью XPath

Можно скопировать **XPath**, используя инструменты разработчика браузера:



Нахождение с помощью CSS

CSS (Cascading Style Sheets, каскадные таблицы стилей) – это язык, используемый для описания правил визуализации HTML и XML документов. Для привязки стилей к элементам документа, в CSS используются селекторы. Эти селекторы могут быть использованы Selenium в качестве еще одного метода

```
1
2
3
4
5
6
7
8
9
10
```

```
<html>
 <body>
  <form id="loginForm">
   <input class="required" name="username" type="text" />
   <input class="required passfield" name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
</html>
```

- `css=form#loginForm` (3)
- `css=input[name="username"]` (4)
- `css=input.required[type="text"]` (4)
- `css=input.passfield` (5)
- `css=#loginForm input[type="button"]` (7)

Контрольные вопросы

1. Что представляет собой автоматизированное тестирование?
2. Области применения автоматизированного тестирования.
3. Достоинства и недостатки автоматизированного тестирования.
4. Что представляет собой Selenium IDE. Описать процесс создания нового теста.
5. Описать основные панели Selenium IDE.
6. Что собой представляют команды Selenium IDE.
7. Описать типы используемых локаторов.