Auriga, Inc.

Selenium

Arkadiy.hachikyan@auriga.com

Elite Software R&D Services

Since 1990

Формат занятий

- Мало теории
- Много практики
- Минимально необходимые навыки
- Обратная связь приветствуется
- Домашние задания

Необходимые знания в программировании

- Основные типы переменных
- Циклы
- Условные операторы
- Статические методы
- Простейшее наследование (без инкапсуляции и полиморфизма)

Необходимый инстументарий

- Firefox
  - FireBug
  - FirePath
  - Selenium IDE
    https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/
- IntelliJIDEA Community
  https://www.jetbrains.com/idea/download/#tabs_1=windows
- Библиотека Selenium Standalone Server
  http://www.seleniumhq.org/download/

План

Часть 1

- Selenium теория
- Идентификация элементов при помощи XPath
- Selenium IDE
  - Рекордер
  - Основные команды
  - Локаторы
  - Проверки
  - Ожидания
  - Задание

Часть 2

- Selenium WebDriver
  - Создание (импорт) проекта
  - Основные части автотеста. JUnit
  - Основные команды
  - Ожидание элементов, асинхронные действия
  - Проверки
  - Создание простого фреймворка
  - Задание

Selenium

Selenium - среда для тестирования web-приложений, выполняющая проверки средствами браузера

Selenium automates browsers.

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

Принцип работы Selenium

Selenium RC - использовал ядро, которое передавало браузеру на выполнение необходимые JavaScript-команды. Это обеспечивало некоторую кросбраузерность поскольку JavaScript выполняется, в значительной мере, одинаково в разных браузерах.

Selenium WebDriver – общается нативными средствами с каждым браузером (AndroidDriver, ChromeDriver, EventFiringWebDriver, FirefoxDriver, HtmlUnitDriver, InternetExplorerDriver, IPhoneDriver, PhantomJSDriver, RemoteWebDriver, SafariDriver)

Идентификация элементов

Хорошее решение для Firefox (FireBug + FirePath):

- FireBug https://getfirebug.com/

  Просмотр и редактирование HTML с применением изменений «на лету»

  Дебаггер JavaScript

- FirePath https://addons.mozilla.org/ru/firefox/addon/firepath/

  Поиск на странице и автоматическое построение XPath

  Подсветка самописных Xpath

- “Инструменты разработчика” в Chrome

XPath

XPath. Важнейшие запросы

- ***** — обозначает *любое* имя или набор символов, @* — любой атрибут
- **[]** — дополнительные условия выборки
- **/** — определяет уровень дерева
- **text**() - Возвращает набор текстовых узлов;
- **contains**(string, string) - Возвращает истину, если первая строка содержит вторую, иначе возвращает ложь.
- **or** — логическое «или»
- **and** — логическое «и»
- **=** — логическое «равно»

XPath. Примеры

С применением индексов:

html/body/div[4]/div[1]/div[3]/div[1]/ul/li**[2]**/span/a

Внимание! Распространенная ошибка

Одновременно к нескольким элементам:

html/body/div[4]/div[1]/div[3]/div[1]/ul/li/span/**a**

(html/body/div[4]/div[1]/div[3]/div[1]/ul/li/span/a)**[2]** – можно выбрать нужный по индексу, начиная с [1]

По значению атрибута определенного тега:

html/body/div[4]/div[1]/div[3]/div[1]/ul/li**[@class='collapsible']**/span/a

XPath. Примеры

По значению атрибута неопределенного тега:

//span/*[@title=**'Эта страница защищена от изменений, но вы можете посмотреть и скопировать её исходный текст [Alt+Shift+e]'**]

По тексту:

//a[**text()='Добро пожаловать'**]

По части текста :

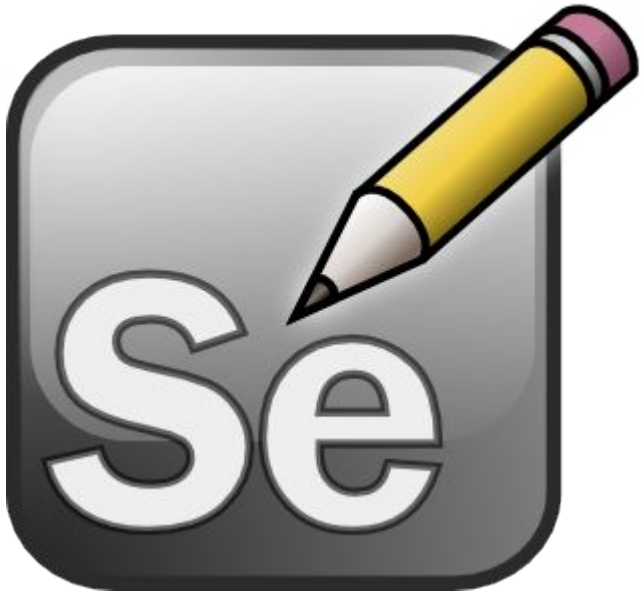
//a[**contains(text(),'Добро')**]

С применением логики:

//a[contains(text(),'Добро') **and** contains(@title,'Википедия')]

Selenium IDE

Selenium IDE – плагин к Firefox, позволяющий осуществлять запись, редактирование и воспроизведение действий в браузере.



Применимость Selenium IDE

- Простые тесты
- Простые сайты
- Не долгосрочная перспектива
- Рутина
- Тренировка, обучение

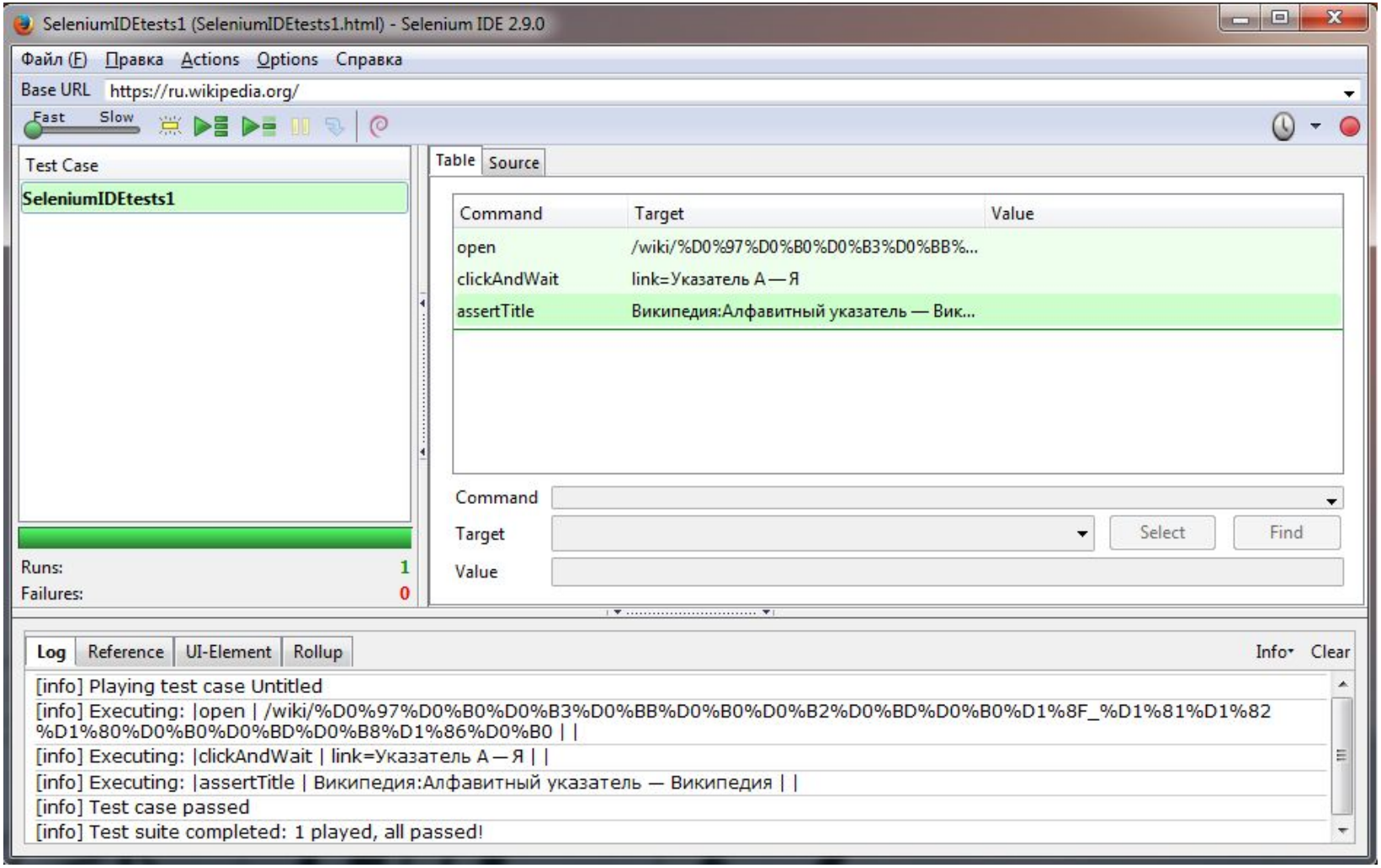
Что понадобится для работы с Selenium IDE

- Mozilla Firefox
- Selenium IDE
- Firebug + Firepath
- Умение работать с XPath

Тест кейс

№	Шаг	Проверка
1	Перейти на «http://ru.wikipedia.org»	
2	Открыть «Указатель А — Я»	На открытой странице виден текст «Википедия: Алфавитный указатель»

Записанный рекордером тест



Основные действия Selenium IDE (Command)

- **open(URL)** – открыть страницу в браузере по определенному адресу
- **click(locator)–** клик по элементу
- **type(locator, value)**– ввести значение в поле
- **select(selectLocator, optionLocator)** – выбрать значение из выпадающего списка

- **selectWindow(windowID)**– переключить фокус на другое окно
- **goBack() -** вернуться на предыдущую страницу
- **close()**– закрыть текущее окно
- **dragAndDropToObject(locatorOfObjectToBeDragged, locatorOfDragDestinationObject) –** перемещение одного элемента на другой

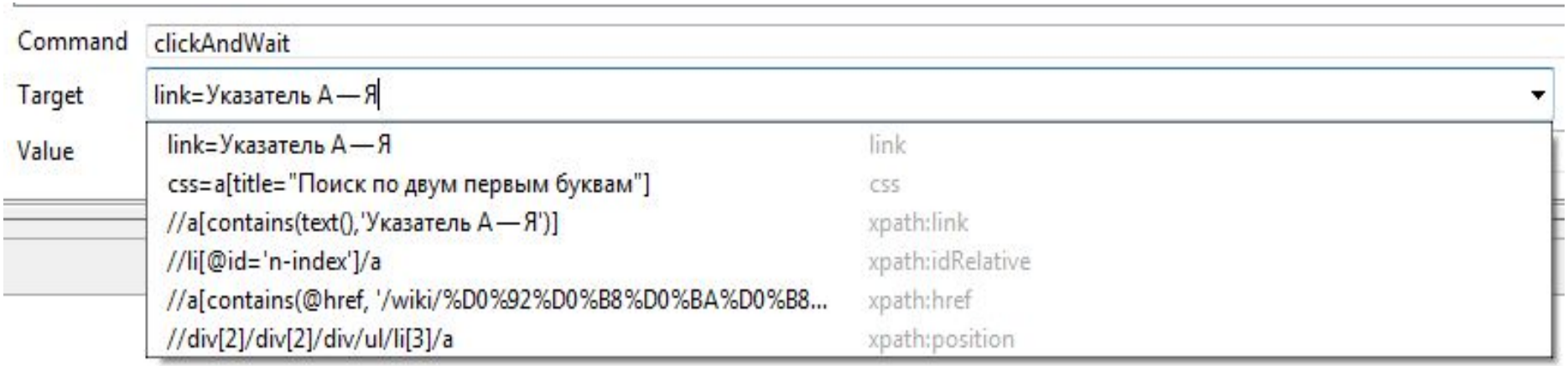
Локаторы Selenium IDE (Target)

- **id** –id элемента страницы;
- **name** –атрибут name элемента страницы;
- **xpath** –XPath выражение;
- **link** – текст ссылки;

- **identifier** –id элемента. Если по id элемент не найден, то поиск будет вестись по атрибуту name;
- **css** – данный тип локаторов основан на описаниях таблиц стилей (CSS).
- **dom** –DOM выражение;

Локаторы Selenium IDE (Target)

НАПРИМЕР:



Проверки Selenium IDE

Verify – просто проверка. Ошибка будет отмечена, но тест продолжит выполняться

Assert – заваливает и останавливает тест

Command	Target	Value
open	/wiki/%D0%97%D0%B0%D0%B3%D0%BB...	
clickAndWait	link=Указатель А — Я	
verifyTextPresent	Википедия:Алфавитный указатель22	
click	link=Заглавная страница	

Проверки Selenium IDE

- **verifyLocation(pattern)/ assertLocation(pattern)** – проверить адрес текущей страницы.
- **verifyTitle (pattern)/ assertTitle (pattern)**– проверить значение Title страницы.
- **verifyValue(locator, pattern) / assertValue (locator, pattern)**– проверить значение элемента страницы.
- **verifyTextPresent(pattern) / assertTextPresent(pattern)** – проверить, что страница содержит указанный в текст.
- **verifyElementPresent(locator) / assertElementPresent(locator)** – проверить, есть ли на странице указанный элемент.
- **verifyVisible(locator)/ assertVisible(locator)**– проверить видимость элемента
- **verifyAttribute(attributeLocator, pattern) / assertAttribute(attributeLocator, pattern)** – проверить значение указанного атрибута

Ожидания Selenium IDE

- **waitForElementPresent(locator)** – ожидание появления элемента на странице
- **waitForTextPresent(pattern) -** ожидание появления текста на странице
- **waitForVisible(locator) -** ожидание видимости элемента

Промежуточное задание

Автоматизировать тест кейс:

№	Шаг	Проверка
1	Перейти на «http://ru.wikipedia.org»	
2	Перейти по ссылке «Указатель А — Я»	На открытой странице виден текст «Википедия: Алфавитный указатель»
3	Перейти по ссылке «АА—АЯ»	В текстбоксе «Вывести страницы, начинающиеся» текст равен «АА»

Необходимый инстументарий для следующей части

- Firefox
  - FireBug
  - FirePath
  - Selenium IDE
    https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/
- IntelliJIDEA Community
  https://www.jetbrains.com/idea/download/#tabs_1=windows
- Библиотека Selenium Standalone Server
  http://www.seleniumhq.org/download/

Вопросы?

Selenium WebDriver

Selenium WebDriver

План:

- Настройка окружения. **IDEA IDE**
- Основные части автотеста. **JUnit**
- Простой тесткейс
- Основные команды
- Проверки
- Ожидание элементов явные и неявные
- Фреймворк. Паттерн **PageObject**
- Итоговое задание

Selenium WebDriver

Selenium WebDriver– позволяет взаимодействовать с браузерами их нативными средствами. Для каждого браузера свой WebDriver (AndroidDriver, ChromeDriver, EventFiringWebDriver, FirefoxDriver, HtmlUnitDriver, InternetExplorerDriver, IPhoneDriver, PhantomJSDriver, RemoteWebDriver, SafariDriver)

Окружение для работы с WebDriver

- IDEA IDE
- Firefox
  - FireBug
  - FirePath

Почему IDEA IDE?

- Java
- Бесплатный
- Удобный
- Некоторые плагины работают лучше, чем, например, в Eclipse (Maven, Cucumber)

Создание тестового проекта в IDEA

Используя экспортированный проект:

1. Экспорт теста из Selenium IDE в WebDriver в Java/JUnit4/WebDriver
2. Создание нового проекта в IDEA
3. Добавление результатов экспорта
4. Подключение библиотек

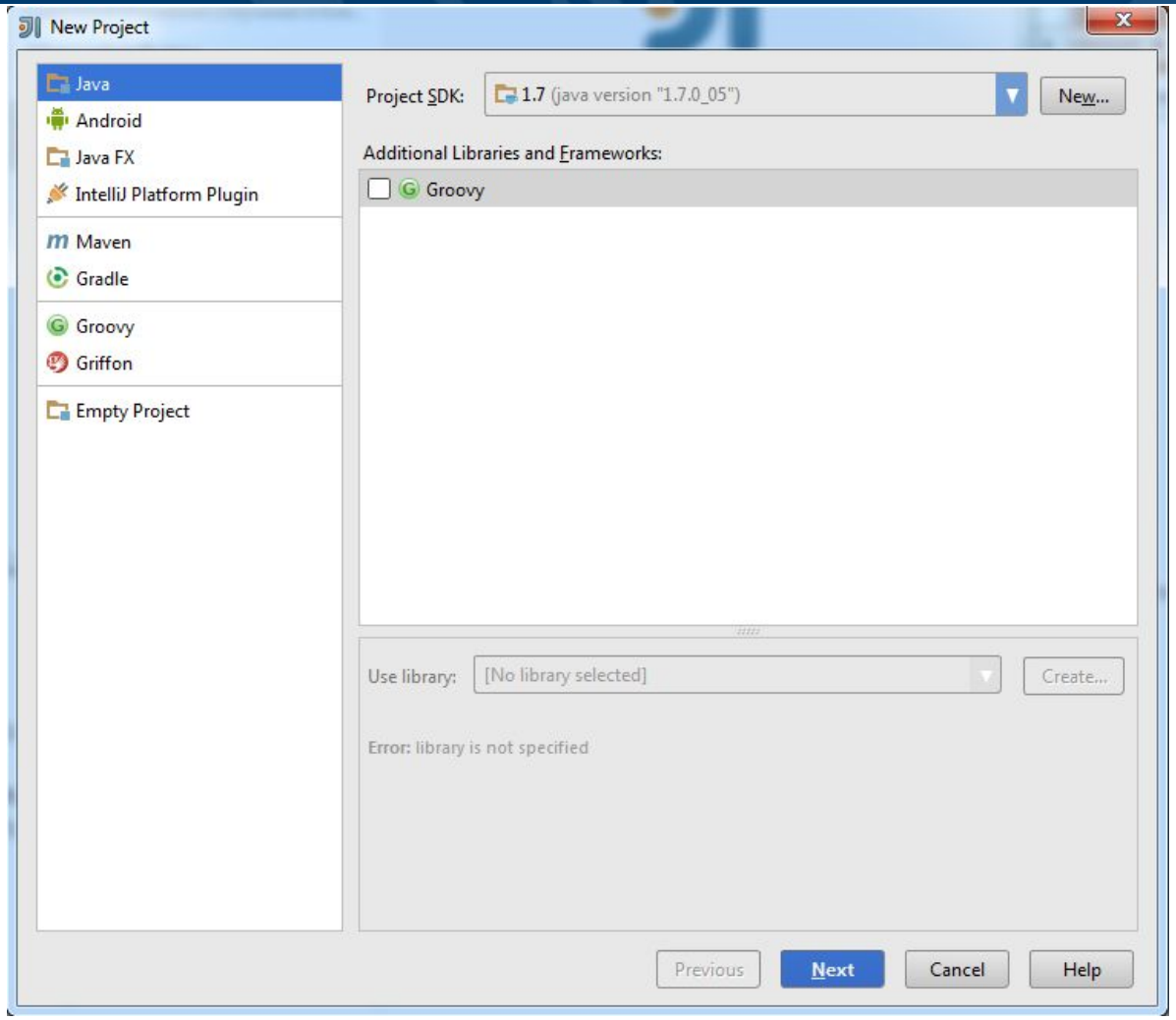
С нуля:

1. Создание нового проекта в IDEA
2. Создание структуры теста вручную (@Before, @After, @Test)
3. Подключение библиотек

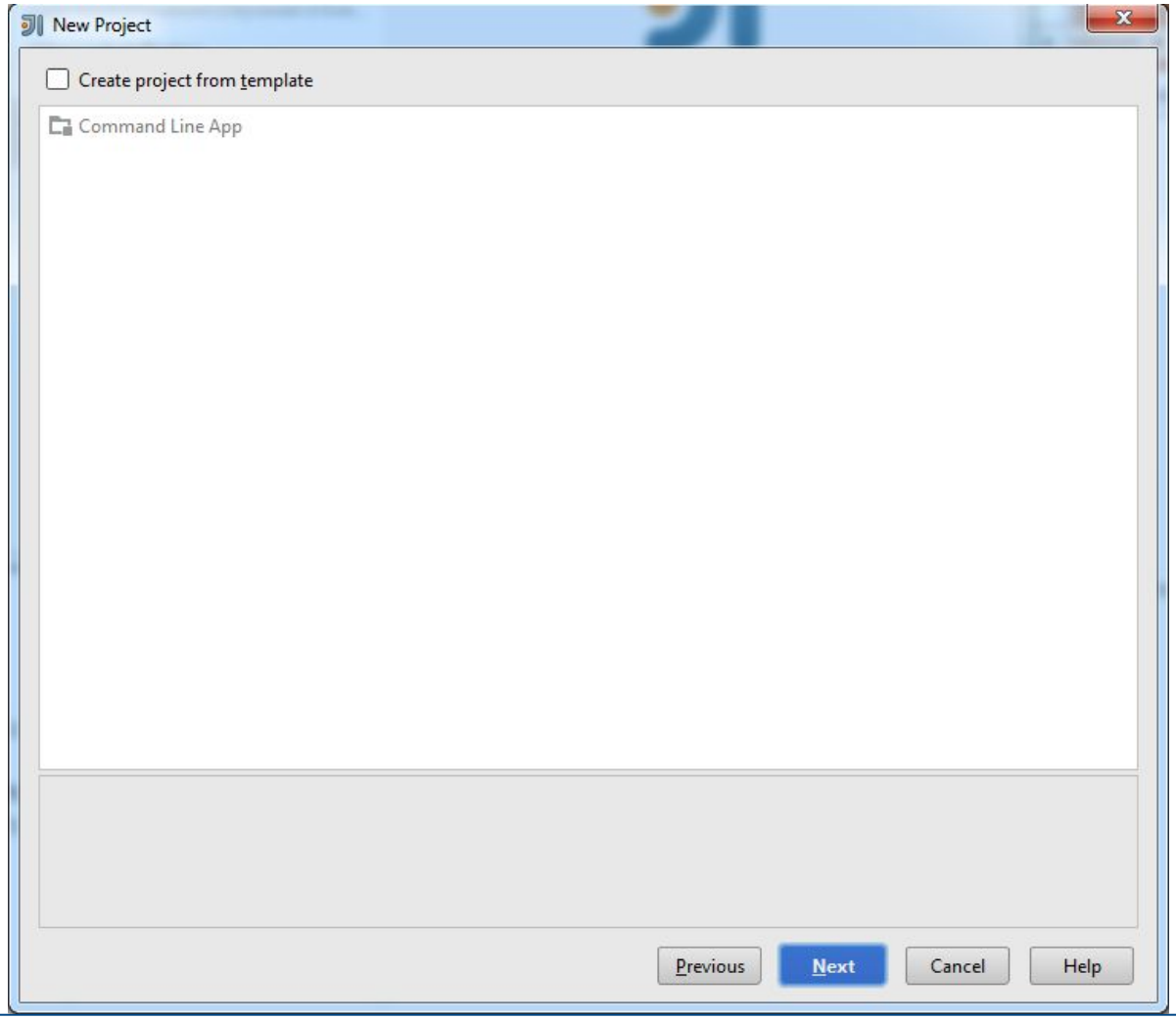
Используя один из внутренних шаблонных проектов Ауриги:

1. Web + xUnit
2. Web + BDD

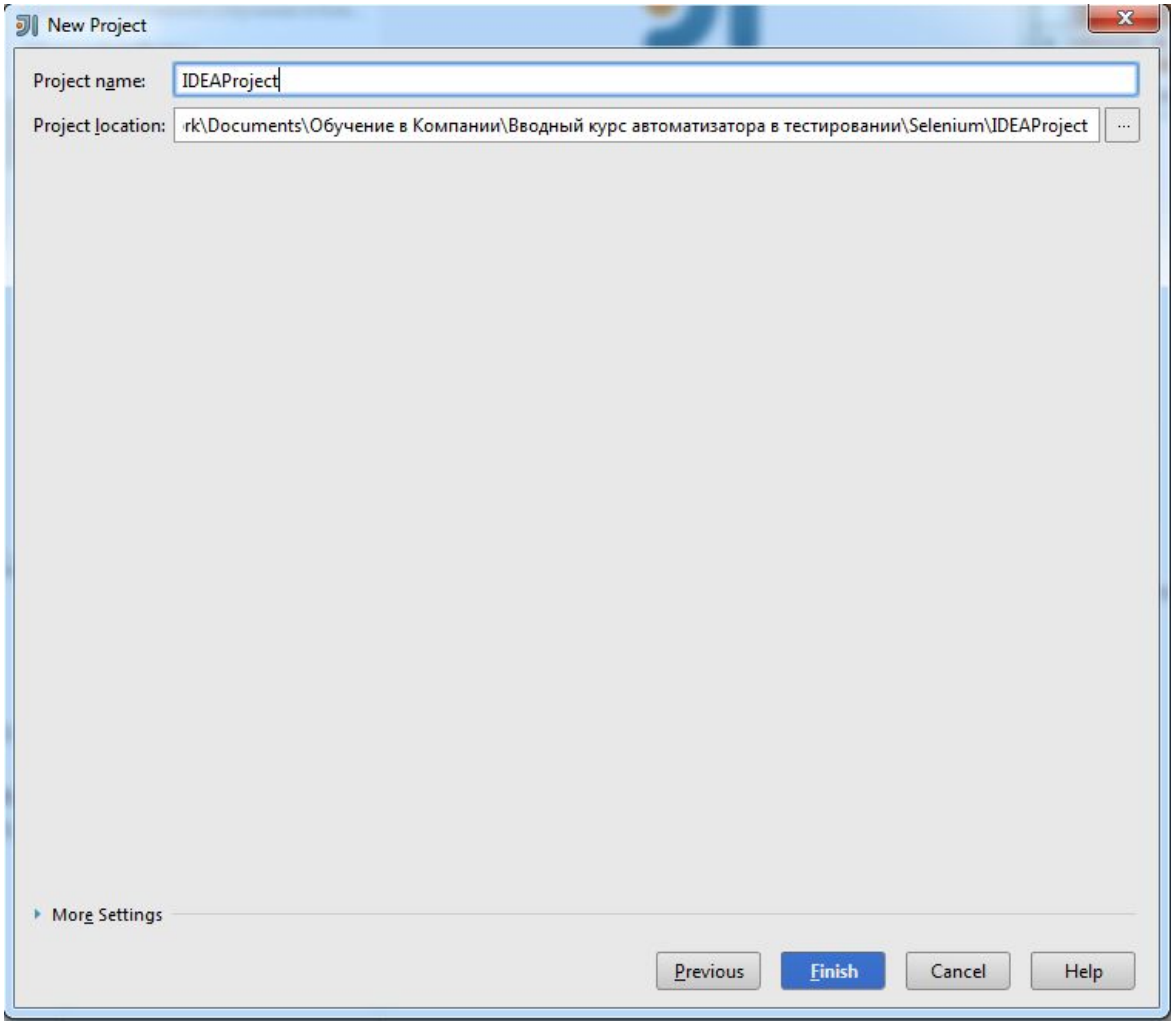
Шаг 1


New Project
Java
Android
Java FX
IntelliJ Platform Plugin
Maven
Gradle
Groovy
Griffon
Empty Project
Project SDK:
1.7 (java version "1.7.0_05")
New...
Additional Libraries and Frameworks:
Groovy
Use library:
[No library selected]
Create...
Error: library is not specified
Previous
Next
Cancel
Help

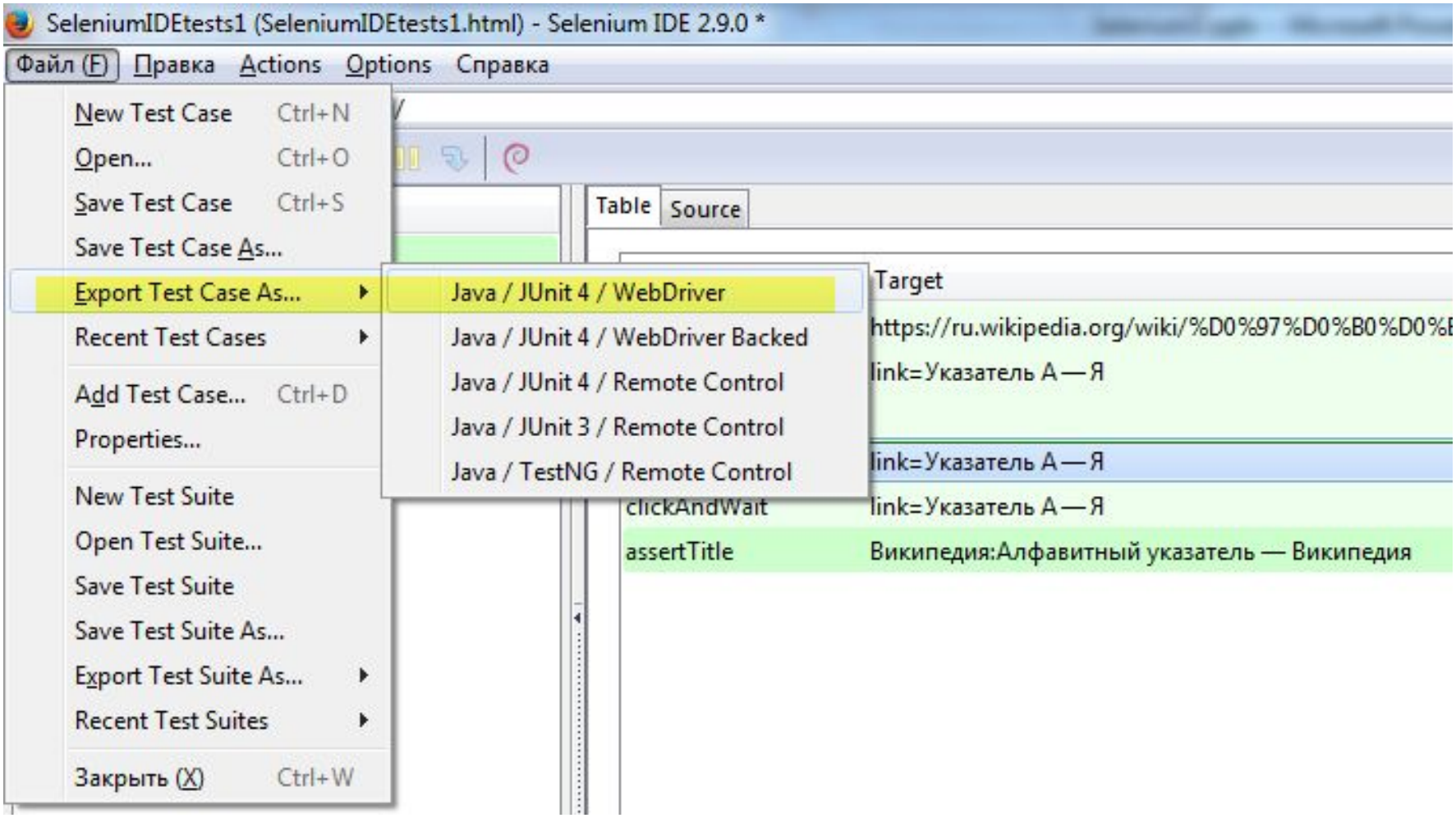
Шаг 2


New Project
Create project from template
Command Line App
Previous
Next
Cancel
Help

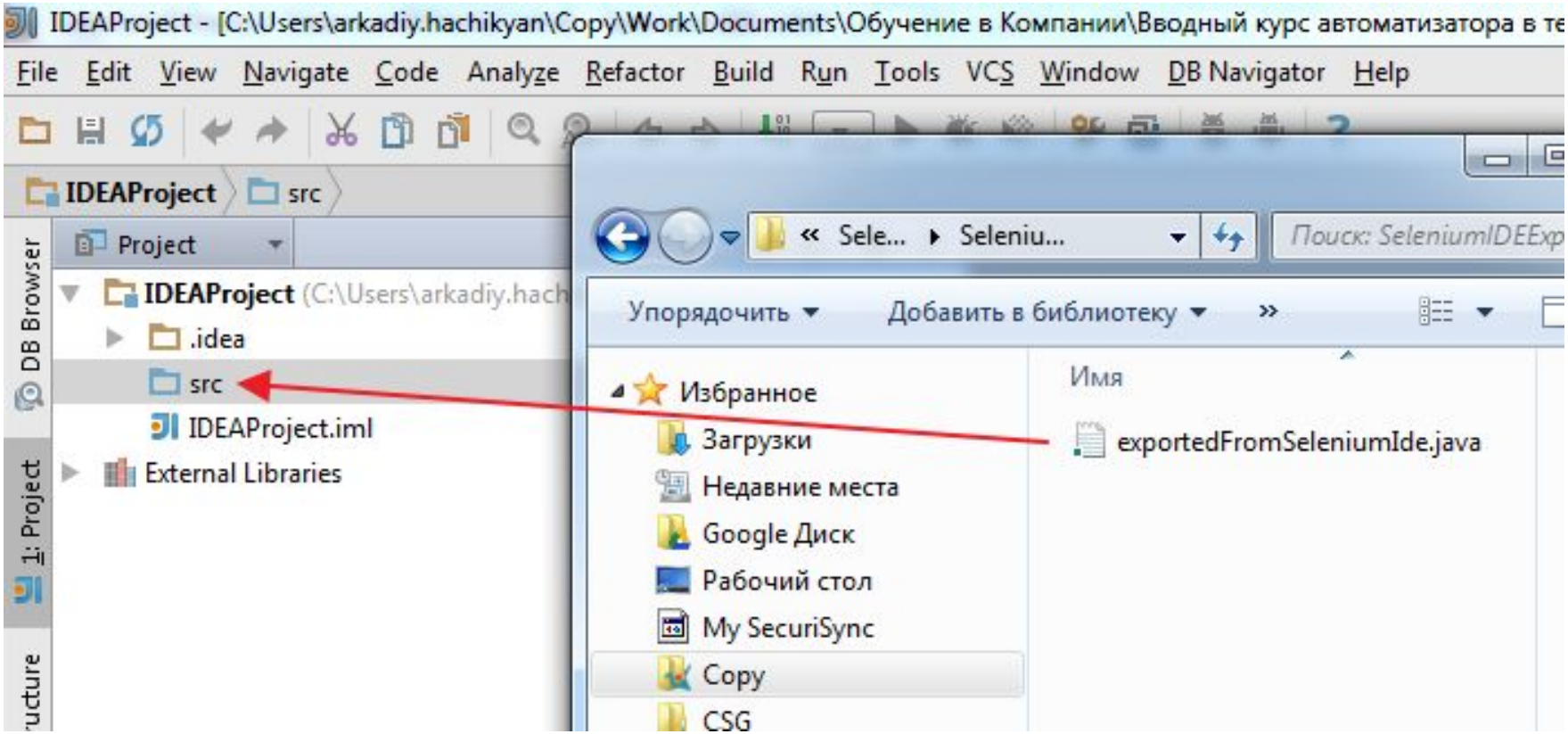
Шаг 3


New Project
Project name:
IDEAProject
Project location:
rk\Documents\Обучение в Компании\Вводный курс автоматизатора в тестировании\Selenium\IDEAProject
...
More Settings
Previous
Finish
Cancel
Help

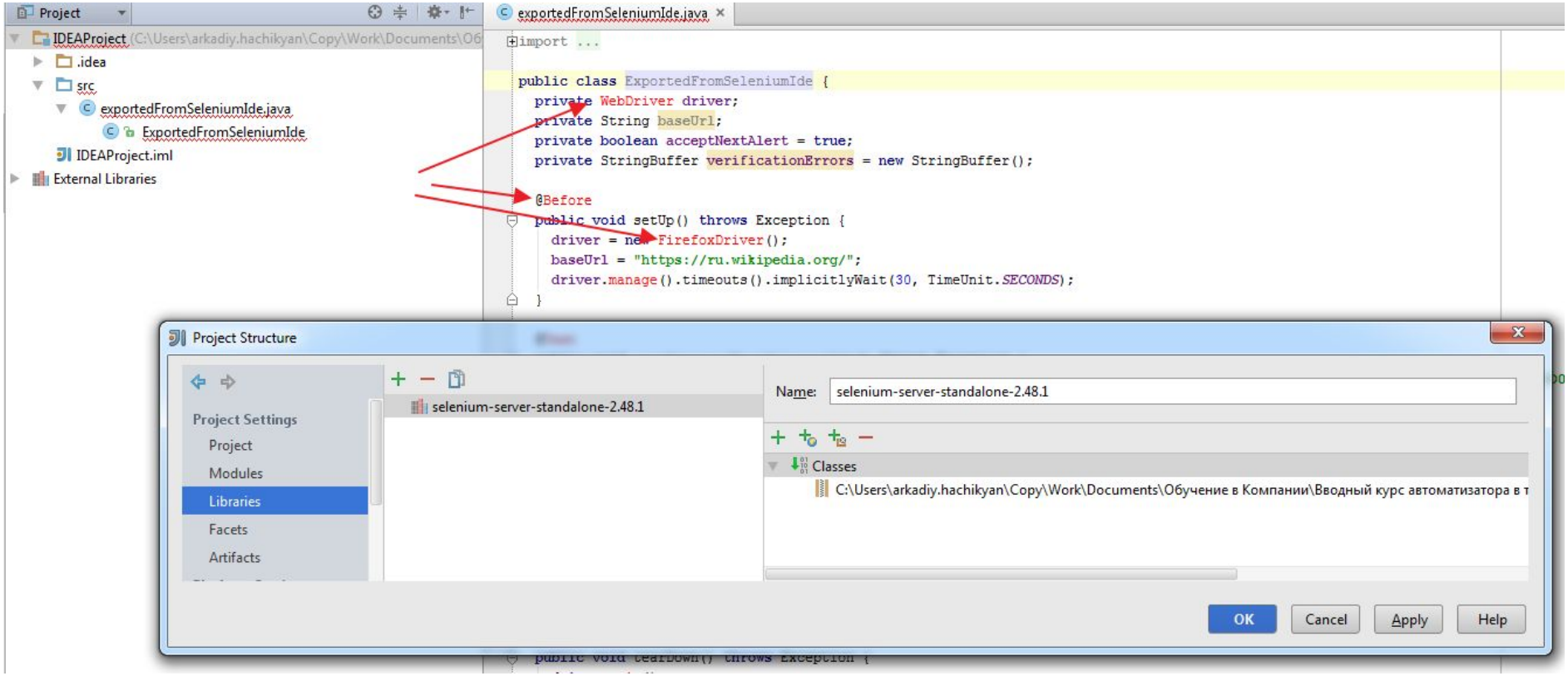
Шаг 4


SeleniumIDEtests1 (SeleniumIDEtests1.html) - Selenium IDE 2.9.0 *
Файл (F)
Правка
Actions
Options
Справка
New Test Case Ctrl+N
Open... Ctrl+O
Save Test Case Ctrl+S
Save Test Case As...
Export Test Case As...
Recent Test Cases
Add Test Case... Ctrl+D
Properties...
New Test Suite
Open Test Suite...
Save Test Suite
Save Test Suite As...
Export Test Suite As...
Recent Test Suites
Закрыть (X) Ctrl+W
Java / JUnit 4 / WebDriver
Java / JUnit 4 / WebDriver Backed
Java / JUnit 4 / Remote Control
Java / JUnit 3 / Remote Control
Java / TestNG / Remote Control
Table
Source
Target
https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%
link=Указатель А — Я
link=Указатель А — Я
clickAndWait
link=Указатель А — Я
assertTitle
Википедия:Алфавитный указатель — Википедия

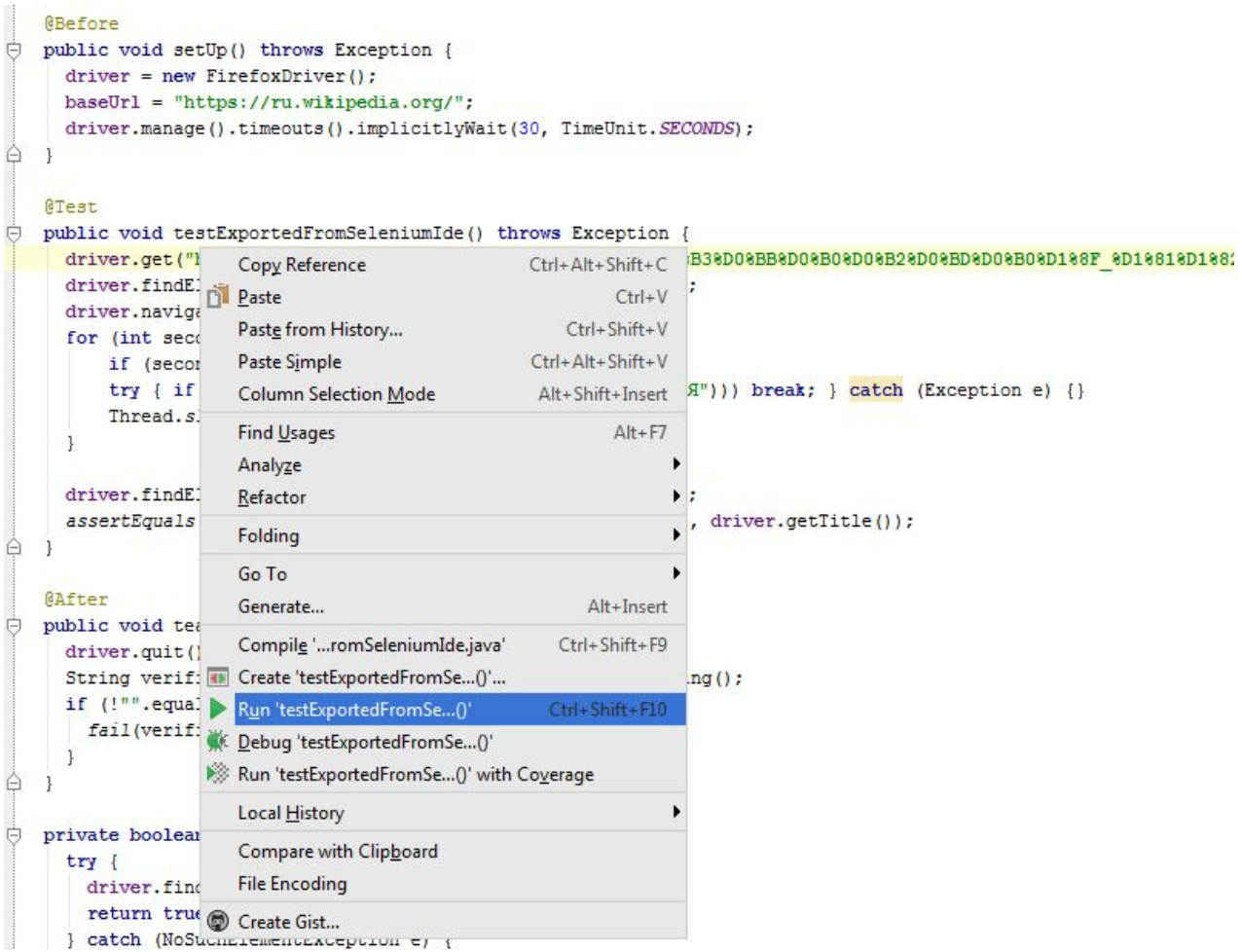
Шаг 5


IDEAProject - [C:\Users\arkadiy.hachikyan\Copy\Work\Documents\Обучение в Компании\Вводный курс автоматизатора в те
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window DB Navigator Help
IDEAProject
src
Project
IDEAProject (C:\Users\arkadiy.hach
.idea
src
IDEAProject.iml
External Libraries
DB Browser
1: Project
« Sele... ▸ Seleniu...
Упорядочить
Добавить в библиотеку
Избранное
Загрузки
Недавние места
Google Диск
Рабочий стол
My SecuriSync
Copy
CSG
Имя
exportedFromSeleniumIde.java

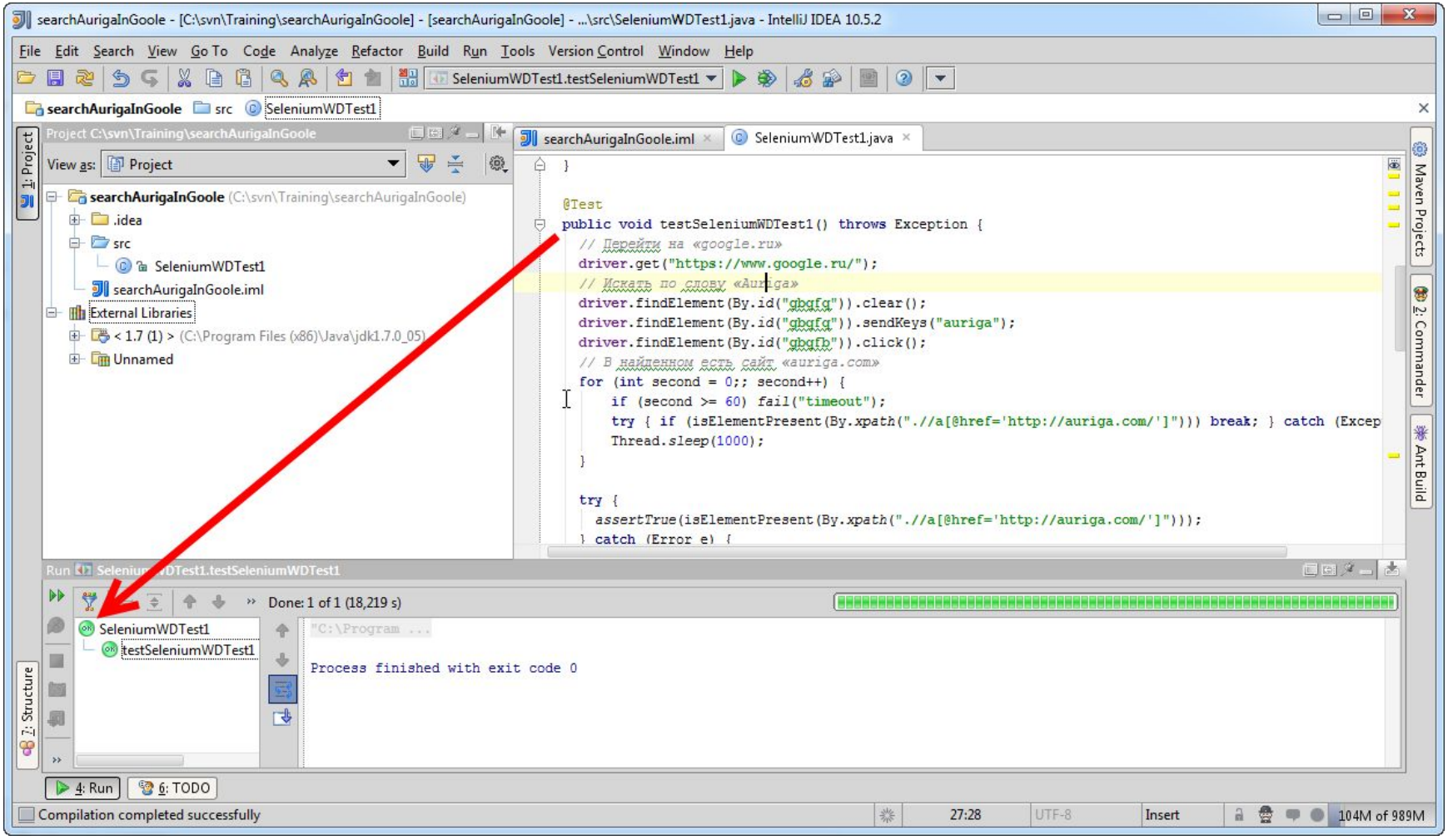
Шаг 6


Project
IDEAProject (C:\Users\arkadiy.hachikyan\Copy\Work\Documents\O6
.idea
src
exportedFromSeleniumIde.java
ExportedFromSeleniumIde
IDEAProject.iml
External Libraries
exportedFromSeleniumIde.java
import ...
public class ExportedFromSeleniumIde {
private WebDriver driver;
private String baseUrl;
private boolean acceptNextAlert = true;
private StringBuffer verificationErrors = new StringBuffer();
@Before
public void setUp() throws Exception {
driver = new FirefoxDriver();
baseUrl = "https://ru.wikipedia.org/";
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
}
Project Structure
Project Settings
Project
Modules
Libraries
Facets
Artifacts
selenium-server-standalone-2.48.1
Name:
selenium-server-standalone-2.48.1
Classes
C:\Users\arkadiy.hachikyan\Copy\Work\Documents\Обучение в Компании\Вводный курс автоматизатора в т
OK
Cancel
Apply
Help

Шаг 7


@Before
public void setUp() throws Exception {
driver = new FirefoxDriver();
baseUrl = "https://ru.wikipedia.org/";
driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
}
@Test
public void testExportedFromSeleniumIde() throws Exception {
driver.get("
driver.findE
driver.naviga
for (int seco
if (secon
try { if
Thread.s
}
driver.findE
assertEquals
}
@After
public void te
driver.quit()
String verif
if (!"".equa
fail(verif
}
}
private boolean
try {
driver.find
return true
} catch (NoSuchElementException e) {
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste Simple Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Analyze
Refactor
Folding
Go To
Generate... Alt+Insert
Compile '...romSeleniumIde.java' Ctrl+Shift+F9
Create 'testExportedFromSe...()'...
Run 'testExportedFromSe...()' Ctrl+Shift+F10
Debug 'testExportedFromSe...()'
Run 'testExportedFromSe...()' with Coverage
Local History
Compare with Clipboard
File Encoding
Create Gist...
B3%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D1%81%D1%82
;
Я"))) break; } catch (Exception e) {}
;
, driver.getTitle());
ng();

Шаг9


searchAurigaInGoole - [C:\svn\Training\searchAurigaInGoole] - [searchAurigaInGoole] - ...\src\SeleniumWDTest1.java - IntelliJ IDEA 10.5.2
File Edit Search View Go To Code Analyze Refactor Build Run Tools Version Control Window Help
searchAurigaInGoole
src
SeleniumWDTest1
Project C:\svn\Training\searchAurigaInGoole
View as: Project
searchAurigaInGoole (C:\svn\Training\searchAurigaInGoole)
.idea
src
SeleniumWDTest1
searchAurigaInGoole.iml
External Libraries
< 1.7 (1) > (C:\Program Files (x86)\Java\jdk1.7.0_05)
Unnamed
searchAurigaInGoole.iml
SeleniumWDTest1.java
}
@Test
public void testSeleniumWDTest1() throws Exception {
// Перейти на «google.ru»
driver.get("https://www.google.ru/");
// Искать по слову «Auriga»
driver.findElement(By.id("gbqfq")).clear();
driver.findElement(By.id("gbqfq")).sendKeys("auriga");
driver.findElement(By.id("gbqfb")).click();
// В найденном есть сайт «auriga.com»
for (int second = 0;; second++) {
if (second >= 60) fail("timeout");
try { if (isElementPresent(By.xpath(".//a[@href='http://auriga.com/']"))) break; } catch (Excep
Thread.sleep(1000);
}
try {
assertTrue(isElementPresent(By.xpath(".//a[@href='http://auriga.com/']")));
} catch (Error e) {
Run SeleniumWDTest1.testSeleniumWDTest1
Done: 1 of 1 (18,219 s)
SeleniumWDTest1
testSeleniumWDTest1
"C:\Program ...
Process finished with exit code 0
4: Run
6: TODO
Compilation completed successfully
27:28
UTF-8
Insert
104M of 989M
Maven Projects
2: Commander
Ant Build
1: Project
7: Structure

Структура типичного теста с JUnit

- Инициализация (@Before). Действия, выполняемые перед каждым тестом. Например, старт браузера, открытие домашней страницы, логин
- Тело (@Test). Сам тест
- Завершение (@After). Действия, выполняемые после каждого теста. Например, закрытие браузера

Основные команды WebDriver

- Объявление драйвера
- Поиск элементов
- Действия
    - Навигация
    - Действия с браузером
    - Действия с элементами
- Проверки
- Ожидания
    - Implicit Waits
    - Explicit Waits
    - Кастомные

Объявление драйвера

- WebDriver driver = new FirefoxDriver();
- WebDriver driver = new ChromeDriver();
- WebDriver driver = new InternetExplorerDriver();

Поиск элементов

WebElement element = driver.findElement(By.<критерий поиска>) - Возвращает первый найденный элемент, удовлетворяющий, условию поиска

driver.findElements(By.<критерий поиска>) - Возвращает все элементы, удовлетворяющие условию поиска

Основные методы поиска элементов:

- **driver.findElement(By.xpath(<XPath>))** – поиск элемента по Xpath
- **driver.findElement(By.id(<ID>))** – поиск элемента по ID
- **driver.findElement(By.name(<имя>))** – поиск элемента по значению атрибута name
- **driver.findElement(By.linkText(<текст ссылки>))** – поиск элемента по тексту ссылки
- …

Действия. Навигация

- **driver.navigate().to(<URL>)** - переход по URL
- **driver.navigate().back()** - переход назад
- **driver.navigate(). forward()** - переход вперед
- **driver.navigate(). refresh()** - обновление
- …

Действия. Действия с браузером

- **driver.close()** - закрыть текущее окно. Закрывает браузер, если нету больше открытых окон;
- **driver.quit()** - выход из драйвера, закрытие всех окон связанных с ним
- **driver.getTitle()** - возвращает Title текущей страницы;
- **driver.getCurrentUrl()** - возвращает URL текущего окна
- **driver.getPageSource()** - возвращает содержимое последней загруженной страницы
- …

Действия. Действия с элементами

- **element.click()** - одиночное нажатие по элементу
- **element.getText()** - возвращает текст элемента
- **element.getAttribute(<название атрибута>)** - возвращает значение указанного атрибута
- **element.isDisplayed()** - является ли элемент видимым
- **element.isEnabled()** - является ли элемент доступным
- **element.isSelected()** - является ли элемент выбранным (чекбокс, радиобаттон)
- **element.sendKeys(<последовательность символов>)** - послать элементу последовательность символов. Текстовых или функциональных клавиш
- …

Пример:

element.sendKeys (“abcd” + Keys.TAB + “efgh”);

Проверки

- **Assert.fail(<необязательный текст ошибки>)** - завалить тест
- **Assert.assertEquals(<необязательный текст ошибки>, <ожидаемое значение>, <фактическое значение>)** - сравнивает 2 объекта, 2 строки или 2 числа. Срабатывает при несовпадении
- **Assert.assertNotEquals(<необязательный текст ошибки>, <ожидаемое значение>, <фактическое значение>)** - срабатывает при совпадении
- **Assert.assertTrue(<необязательный текст ошибки>, <выражение или булевое значение>)** - срабатывает на ложном значении
- **Assert.assertFalse(<необязательный текст ошибки>, <выражение или булевое значение>)** - срабатывает на правдивом значении
- …

Пример:

Assert.assertEquals("Текущее значение " + cur + " не равно фактическому" + fact, cur, fact);

Ожидания. Explicit Waits

Explicit Waits (явное ожидание) – готовое решение с использованием класса WebDriverWait. Представляет собой фактически циклическое **ожидание указанного события**. По умолчанию "WebDriverWait" вызывает "ExpectedCondition" каждые 500 миллисекунд до тех пор, пока условие не будет удовлетворено.

Ожидаемые события:

- **ExpectedConditions.visibilityOf(<элемент>)** - видимость элемента
- **ExpectedConditions.presenceOfElementLocated(<локатор>)** - наличие элемента
- **ExpectedConditions.textToBePresentInElement (<локатор>, <текст>)** - текст элемента
- …

Пример:

```
WebDriverWait wait = new WebDriverWait(driver, 10);
wait.until(ExpectedConditions.visibilityOf(element));
```

// Данный код будет либо выдаст исключение "TimeoutException" по прошествии 10 секунд, либо вернет найденный в течении 10 секунд элемент.

Фреймворк. Паттерн PageObject

Идея:

- Класс страницы представляет собой интерфейс, модель настоящей страницы

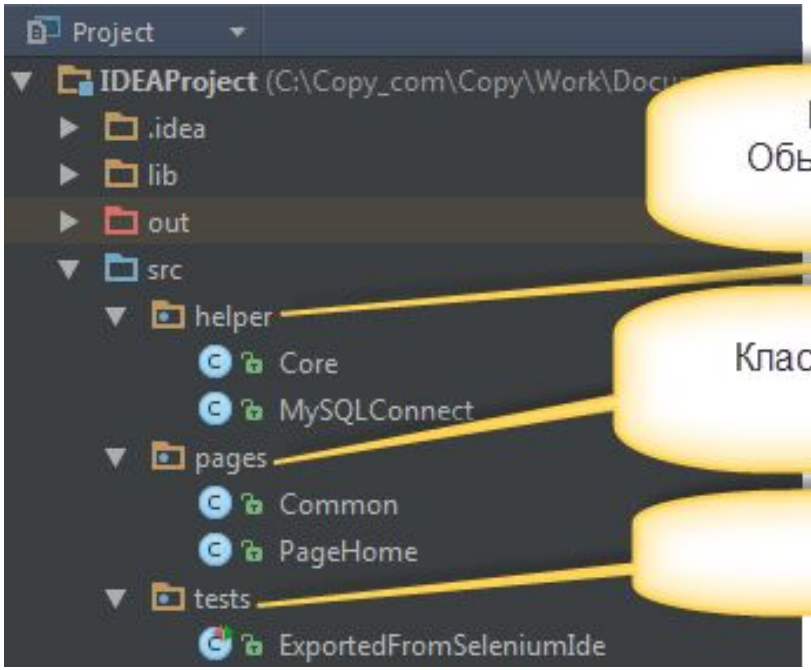
Профит:

- Читабельность
- Сопровождаемость. Изменения интерфейса отражаются только в одном месте
- Уменьшение дублирования

Правила:

- В классе страницы содержатся ее контролы с локаторами и только типичные действия, осуществляемые на странице
- Если страница большая, то ее можно логически разделить на разные классы
- Избегать проверок внутри объекта страницы
- ~~Методы возвращают объекты других страниц~~

Пример организации простого фреймворка



Классы, не связанные непосредственно с тестами. Обычно, реализация @Before, @After, скриншотирования, логгирования, работы с БД и др.

Классы страниц. Один класс с общими для всех страниц действиями и классы индивидуальных страниц

Сами тесты. Помечены @Test

Разное

Проблемы тестирования асинхронных приложений

Проблема: в AJAX-ориентированных приложениях данные, передаваемые сервером, затем отображаются на странице без ее перезагрузки, обновляется только ее часть или же непосредственно измененный элемент.

Решение: активно использовать ожидания. При необходимости писать обертки. Не использовать тупо слипы, а ждать каких-то событий (видимости элемента, исчезновения элемента и т.п.)

Итоговое задание

- Автоматизировать 5 тесткейсов:
  1. Клик по мозаичному шару «ВикипедиЯ» возвращает на главную страницу
  2. Максимальное количество предложений предикативного ввода в поиске составляет 10 значений (проверить на примере ввода «а»)
  3. Содержимое предложений предикативного ввода в поиске при вводе «аур» включает (могут быть и другие, но надо проверить, что есть хотя бы эти): Аур, Аура, Аурих, Аурано, Ауриго, Ауро, Ауреа

Полезные сылки

- http://selenium2.ru/docs/selenium-ide.html
- http://docs.seleniumhq.org/docs/03_webdriver.jsp
- http://docs.seleniumhq.org/docs/04_webdriver_advanced.jsp
- http://automated-testing.info/knowledgebase/article/webdriver-osnovnye-komandy
- https://code.google.com/p/selenium/wiki/PageObjects

Contacts

Thank You and
We Look Forward to Working with You

Auriga, USA
92 Potter Rd, Ste 1
Wilton, NH 03086, USA
Phone: +1 (866) 645-1119
Fax: +1 (603) 386-6097
info@auriga.com
www.auriga.com

Auriga, Russia
125 Varshavskoe Shosse, Unit 16A
Moscow, 117587
Tel:+7 (495) 713-9900
Fax:+7 (495) 939-0300
info@auriga.com
www.auriga.com