когда кто то говорит о Selenium-e то имеют ввиду webDriver.

Selenium - позволяет контролировать выполнение команд в браузере автоматически.

Для поиска элементов на веб странице/приложении поможет devTools, F12, для проверки селектора (способ по которому ищется элемент:x\_path, CSS\_selector…) прямо в devTool нажать **CTRL+F** появится окно ввода.

Для того чтобы всё работало:

**First** you need to install the Selenium client library (in a simple words it is selenium itself) for your automation project. It is collection of JAR files (each jar file consist of multiple classes). Installation of the Selenium library for Java is accomplished using a build tool - Maven. <https://www.selenium.dev/downloads/> <- - link to download selenium dependency.

Specify the dependency in the project’s pom.xml file:

**<dependency>**

**<groupId>org.seleniumhq.selenium</groupId>**

**<artifactId>selenium-java</artifactId>**

**<version>4.12.1</version>**

**</dependency>**

**Second** you need to download the webDriver from the link below. It accepts commands and sends them to a browser. WebDriver is an API. The API has functions to control a browser. You use the functions in writing a script that controls a browser in the way you want.

<https://chromedriver.chromium.org/downloads/version-selection>

После того как скачал надо либо вручную его подключать либо с помощью WebDriverManager .

Любой драйвер для любого браузера (edge, chrome, fireFox…) может быть настроен.

**1**)-Способ: Скачать драйвер в какую-либо директорию и распоковать. В IDEA прописываем

System.setProperty("**webdriver.gecko.driver**", "**C:\\Program Files\\WEB\_DRIVERS\\FireFox\\geckodriver.exe**");

где первая строка это название используемого драйвера, а вторая строка путь по которому расположен сам веб дррайвер.

Вслучае с chrom или edge необходимо прописать опции, для обоих драйверов значения одинаковы:

**EdgeOptions options=new EdgeOptions();**

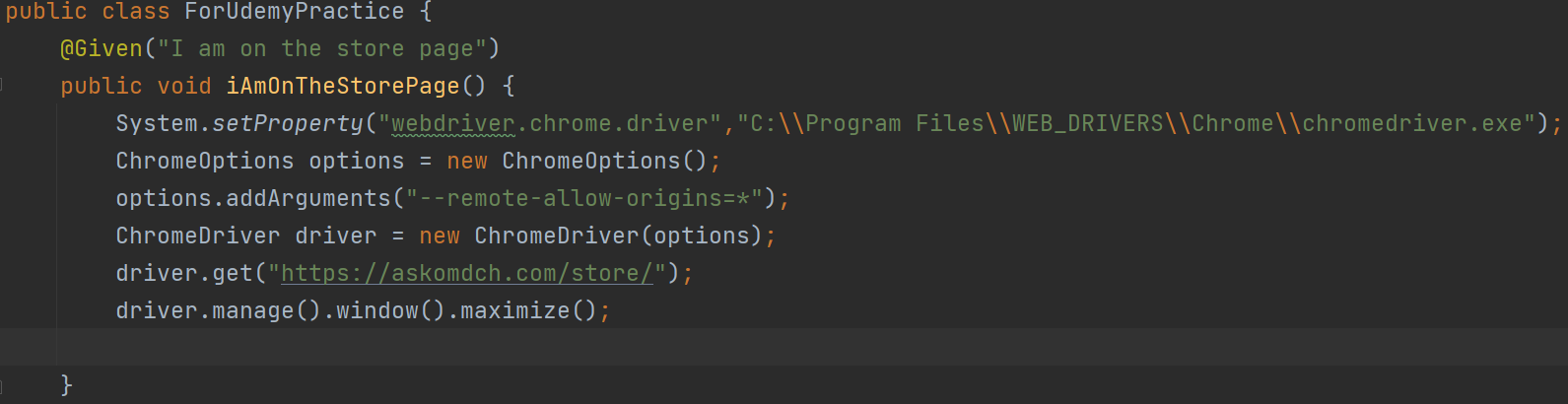
**options.addArguments("--remote-allow-origins=\*");**

**driver=new EdgeDriver(options);**

**ChromeOptions options = new ChromeOptions();**

**options.addArguments("--remote-allow-origins=\*");**

**driver = new ChromeDriver(options);**

****

**2**)-Способ воспользоватся ,библиотекой WebDriverManager. В зависимости от драйвера нужно указать его имя после

WebDriverManager. Далее пример с chromedriver, с другим драйвером соответственно другое имя.

WebDriverManager.chromedriver().setup();

Также для этого нужна зависимость в Maven:

<dependency>

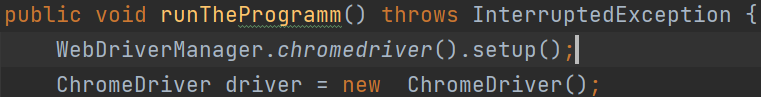
<groupId>io.github.bonigarcia</groupId>

<artifactId>webdrivermanager</artifactId>

<version>5.0.3</version>

</dependency>

Очень важно: если пользоватся WebdriverManager то обязятельно наличия зависимости + WebDriverManager.chromedriver().setup(); не посредственно где создаётся chromeDriver.



После 111 версии хрома (если ты не используешь WebDriverManager) необходимо при настройках прописывать не только путь где лежит webDriver, но и

ChromeOptions options = new ChromeOptions();

options.addArguments("--remote-allow-origins=\*");

ChromeDriver driver = new ChromeDriver(options);

**Update 28/03/2024 - Больше не надо писать WebDriverManager.chromedriver().setup();**

**достаточно: ChromeDriver driver = new ChromeDriver();**

зависимость бонигарсия не нужна, мэнеджер наверно теперь вшит в саму зависимость селениум.

**<dependency>**

**<groupId>io.github.bonigarcia</groupId> ← теперь это не надо.**

**<artifactId>webdrivermanager</artifactId>**

**<version>5.7.0</version>**

**</dependency>**

При работе используется большое количество классов:

Webdriver ←interface, позволяет работать самим браузером: переключаться по вкладкам, обращаться к URL, обновлять, находить элементы используя разные локаторы/Locator strategies:

-class name Locates elements whose class name contains the search value (compound class names are not permitted)

-css selector Locates elements matching a CSS selector

-id Locates elements whose ID attribute matches the search value

-name Locates elements whose NAME attribute matches the search value

-link text Locates anchor elements whose visible text matches the search value

-partial link text Locates anchor elements whose visible text contains the search value. If multiple elements are matching, only the first one will be selected.

-tag name Locates elements whose tag name matches the search value

-xpath Locates elements matching an XPath expression

WEBELEMENT-interface, позволяет работать с конкретным элементом: кликать по элементу, очищать поле ввода (если этот элемент input), вводить инф., проверять на конкретные атрибуты…

Actions - класс имитирует: double-click, selecting drop-down boxes, переходит/нависает над элементом, умеет изменять шрифт, copy-paste,

doubleClick(): Performs double click on the element

clickAndHold(): Performs long click on the mouse without releasing it

dragAndDrop(): Drags the element from one point and drops to another

moveToElement(): Shifts the mouse pointer to the center of the element

contextClick(): Performs right-click on the mouse

Keyboard Actions in Selenium:

sendKeys(): Sends a series of keys to the element

keyUp(): Performs key release

keyDown(): Performs keypress without release