***Selenide***

* Na selenide.org je dependency, ktoru pridam do POM.xml a selenide sa mi sám stiahne
* Otvorenie stránky:

Selenium (pred):

driver = new ChromeDriver();

driver.get("http://localhost:82/gosslingator.php");

Selenide (v nových projektoch, kde nepotrebujem použíť starý selenium webdriver):

open("http://localhost:82/gosslingator.php");

//open v selenide otvára driver(prehliadač) aj stránku jedným príkazom

//zatial takto nefunkčné

Selenide (v starých projektoch, použijem starý webdriver zo selenia)

driver = new ChromeDriver();

WebDriverRunner.setWebDriver(driver);

open("http://localhost:82/gosslingator.php");

//v starých projektoch by zmena mohla byť náročnejšia ohladom webdrivera, preto dám selenidu najavo, aby použil starý webdriver zo selenia

* Hladanie elementu:

Selenium :

driver.findElement(By.id("addRyan")).click();

//okamžite hladá element na stránke a ak ho nenájde, tak spadne

Selenide :

$(By.id("addRyan")).click();

//v selenide čaká tento element v pozadí, až sa na stránke objaví, neočakáva ho okamžite, táto fiture rieši problém s reaktívnymi stránkami napr.

//zas aby test mohol aj spadnúť, tak selenide čaká cca 4 sekundy na svoj element, ak sa nezobrazí dovtedy, tak spadne

//pokial hladam element podla By.CssSelector tak to tam dávať netreba

* Hladanie podla ID:

Stary spôsob :

$(By.id("addRyan")).click();

Novy spôsob :

$(byid("addRyan")).click();

* Hladanie **podla xpath**:

Selenium :

driver.findelement(By.xpath("//h1[contains(text(),'" + fellowName + "')]")).click();

Selenide :

$x("//h1[contains(text(),'" + fellowName + "')]").click();

//pokial dam za dolar “x“, tak vie, ze defaultne ma ocakavat xpath a nie cssselector, nemusim mu teda pisat, ze na zaklade coho ma hladat

* Hladanie elementu **podla textu** :

Selenide (xpath):

$x("//h1[contains(text(),'" + fellowName + "')]").click();

Selenide (byText):

$(byText(fellowName)).click();

* Hladanie **podla atributu** :

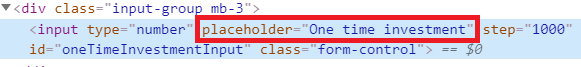
Selenium:

driver.findElement(By.xpath("//input[@placeholder='One time investment']")).sendKeys("25");

Selenide:

$(byAttribute("placeholder","One time investment")).sendKeys("20");

//pokial nemam ziadne id, alebo nemam podla coho hladat, tak mozem podla atributu



* Hladanie **elementu v tabulke (cely riadok)** :

Selenium :

driver.findElement(By.xpath("//table/tbody/tr[2]")).getText();

//druhy riadok tabulky najdem pomocou xpath

Selenide :

$("table > tbody > tr",1).getText();

//najdenie druheho elementu podla cssSelectora, hodnota jedna, lebo index zacina od nuly

* Hladanie **elementu v tabulke (konkretny riadok aj stlpec) :**

Selenium :

driver.findElement(By.xpath("//table/tbody/tr[2]/td[4]")).getText();

//druhy riadok a stvrty stlpec tabulky (bude to email) najdem pomocou xpath

Selenide :

$("table > tbody > tr",1).find("td",3).getText();

//najdenie druheho riadku a stvrteho stlpca podla cssSelectora, indexy idu od 0, pre hladanie aj stlpca musim retazit klucovym slovom find

* **Čakanie** na element :

Selenium :

new WebDriverWait(driver, 10).until(ExpectedConditions.textToBePresentInElement(

$("div.current-wait-time"),"Response time"));

Assert.assertTrue($(By.cssSelector("div.current-wait- time")).getText().contains("Response time"));

Selenide :

$("div.current-wait-time").***shouldHave***(**Condition.text**("Response time was"));

//na element čaká automaticky(písaný pomocou css selektora), akurát mu zadám podmienku, že aký text má obsahovať, keď sa objaví, dokonca môžem zmazať aj to Condition, len to musím zadefinovať ako statickú triedu

//pokial použijem **Condition.text**, tak mi stačí zadať len časť textu

//pokial použijem **Condition.exactText** tak musím zadať presne ten text ktorý očakávam

Iný príklad na AssertEquals:

Starý spôsob :

Assert.assertEquals("GOSLINGATE ME", $(".ryan-title").getText());

Selenide :

$(".ryan-title").shouldHave(Condition.text("GOSLINGATE ME"));

* Elementy should, should have, should be

//tieto elementy robia to isté, akurát jazykovo majú iný význam, použijem teda ten, ktorý sa mi tam logicky najviac hodí

* Nezobrazeny element na stranke:

Selenide :

$("img").shouldBe(Condition.hidden);

$(By.id("ryanCounter")).shouldHave(text("0"));

//pokial bude obrazok img skryty, tak hlada text “0“

* Element **value** a **cssClass**:

Selenide (value) :

$(byId("waitForTextInput")).shouldHave(value(expectedText));

Selenide (cssClass) :

$(byId("waitForProperty")).shouldHave(cssClass("error"));

* Element **objaví** sa a **zmizne**, prípadne objaví sa **prázdny** :

$("img.loading").should(appear);

$("img.loading").should(disappear);

$("p.result").should(appear);

$("p.result").shouldNotBe(empty);

//miesto should by mohlo byť aj shouldBe alebo shouldHave, tiež všetky vlastnosti (appear, disappear sú z Condition, len mám naimplementovanú statickú triedu Condition, tak to tam písať netreba)

Este mozem vyssie uvedeny kod **zretazit**:

$("img.loading").should(appear).should(disappear);

$("p.result").shouldBe(visible).shouldNotBe(empty);

//element sa objaví a zmizne

//element bude viditelný a prázdny

Dalsi sposob zretazenia:

generatedHouse = $("p.result").shouldBe(visible).shouldNotBe(empty).getText();

// po tom ako element bude viditelný a nebude prázdny, zober mi text a vlož ho do premennej generatedHouse, toto je novy spôsob zreťazenia a ide postupne a hodnotu z poslednej metódy vloží do generatedHouse

* **Čakanie** na element:

$("img.loading").should(appear);

//defaultne čaká 4 sekundy na element, potom test spadne

$("img.loading").waitUntil(appears,15000);

//pokial chcem čakať dlhšie ako 4 sekundy, tak použijem waitUntil s dvoma parametrami, prvy je podmienka a druhy je čas v ms

//appear a appears je to isté, len aby to gramaticky pri čítaní sedelo, tak je použitá tretia osoba

* Hladanie **viacerych** elementov :

Selenium :

driver.findElements(By.cssSelector("ul.spells li"))

Selenide :

$$(By.cssSelector("ul.spells li"))

//jednym dolarom hladam jeden element, dvoma dolarmi hladam viacero elementov

* **Pocet** elementov **vacsi ako 0** :

$$("ul.spells li").shouldHave(CollectionCondition.sizeGreaterThan(0))

//CollectionCondition mozem vyhodit a dat ho ako staticku triedu

* **Konkretny pocet** elementov :

$$("ul.spells li").shouldHave(CollectionCondition.size(2))

//CollectionCondition mozem vyhodit a dat ho ako staticku triedu

* Namiesto WebElement pouzijeme **SelenideElement :**

Selenium :

.map(WebElement::getText)

Selenide :

.map(SelenideElement::getText)

* **Kolekcie**:

Selenium :

List<WebElement>

Selenide :

ElementsColection

* Hladanie **jedneho elementu** v ramci jedneho elementu :

$("ul.spells").find("li")

* Hladanie **viacerych elementov** v ramci jedneho elementu :

$("ul.spells").findAll("li")

* Hladanie **textov** v elementoch

List<String> displayedSpells = $("ul.spells")

.findAll("li")

.shouldHave(sizeGreaterThan(1))

.shouldHave(texts(expectedSpells))

//hlada nejake ocakavane slova z textoveho suboru, staci zadat iba cast textu

//ak zadam **exactTexts** tak musi byt text presne rovnaky

* Hladanie textu v kolekcii

Selenium :

ElementsCollection spellElements = $("ul.spells")

.findAll("li")

.shouldHave(CollectionCondition.sizeGreaterThan(1));

for (SelenideElement spellElement : spellElements) {

if (spellElement.getText().equals("tortures a person")) {

spellElement.click();

}

}

//takyto foreach cyklus mozem v Selenide nahradit jednoducho

Selenide :

$("ul.spells")

.findAll("li")

.shouldHave(CollectionCondition.sizeGreaterThan(1))

.find(Condition.text("tortures a person"))

.click();

//Najdem kolekciu spellov(li elementov) pomocou findAll, a ked sa mi jeden zobrazi tak bude hladat text „tortures a person“ a nakoniec klikne na tento element

* Hladanie **aktivnych** elementov :

$("ul.list-of-fellows")

.findAll("li > div")

.filterBy(Condition.cssClass("active"))

.shouldHave(CollectionCondition.textsInAnyOrder(fellowsToSelect));

//najde element podla cssSelectora, je to ul element s triedou list-of-fellow, dalej najde viacero elementov obsahujucich li element a pod nim este div element a nakoniec odfiltruje aktivne a hlada texty ulozene v premennej fellowsToSelect

//pokial by som dal len texts, sice by hladalo spravne nazvy postav, ale boli v zlom poradi a test by spadol, pokial nezalezi na poradi, tak miesto texts zmenim **textsInAnyOrder**

//**filterBy** samo o sebe nevykona ziadnu akciu, akciu vykona az prikaz za nim, napr shouldhave alebo click, cize filtrovanie nastane az v kombinacii s inou akciou, inak to bude mat pocitac len pripravene (Lazy evaluation)

//**find** vracia jeden element a **filterBy** vracia kolekciu elementov podla nejakej podmienky

* **Exclude** (vyhodenie) z kolekcie :

$("ul.spells")

.findAll("li")

.shouldHave(sizeGreaterThan(1))

.filterBy(**matchesText**("^shoots.\*"))

**.**exclude**(readonly)**

.exclude(matchesText("^opens.\*"))

.exclude(matchesText("^[a|b|c|d].\*"))

.forEach(selenideElement -> System.out.println(selenideElement.getText()));

//vyhadzovat zo zoznamu sa da podla regularneho vyrazu (**matchesText**) alebo podla citatelnosti (readonly), filtrovat je tiez mozne podla regularneho vyrazu, vyssie je priklad

* Zobrazenie poctu bodov :

Selenium :

ElementsCollection displayedFellows = getFellowElements();

for (WebElement displayedFellow : displayedFellows){

String actualPoints = displayedFellow.findElement(By.cssSelector("div.fellow- points h2")).getText();

Assert.assertFalse(actualPoints.isEmpty());

}

Selenide :

getFellowElements().forEach(selenideElement -> selenideElement.find("div.fellow- points h2").shouldNotBe(empty));

//zapis pomocou streamu, pre kazdy selenide element, ktory ma h2 cssSelector nemoze byt prazdna hodnota

* **Obalenie** Selenium WebElementu a metoda **val**:

Definícia WebElementu v Seleniu:

@FindBy(id = "oneTimeInvestmentInput")

private WebElement oneTimeInvestmentInput;

Použitie metódy so zadefinovaným WebElementom:

public void enterOneTimeInvestment(String amount) {

oneTimeInvestmentInput.clear();

oneTimeInvestmentInput.sendKeys(amount);

}

Obalenie WebElementu do Selenide elementu (Selenide rozsiruje Selenium WebElement) :

$(oneTimeInvestmentInput).val(amount);

//staci WebElement **obalit** dolarom a mame z toho Selenide Element

//metoda **val** robi clear aj sendKeys v jednom kroku

Obalenie do Selenidu, zmazanie a zadanie hodnoty pridanie tabulatora:

$(emailInput).val(email).pressTab();

//pressTab() pridá tabulátor za email

* **Vyberanie** **z** **comboBoxu** :

Selenium :

new Select(fundSelect).selectByVisibleText(fundToSelect);

Selenide :

$(fundSelect).selectOption(2);

//obalenie elementu do Selenide a nasledne vyber z comboBoxu na zaklade stringu, id, obsahujuceho textu

* **Obalenie** metody :

Selenium :

public WebElement getApplyButton() {

return applyButton;

}

//applyButton je typu WebElement, preto ho chcem obalit do Selenidu

Selenide :

public SelenideElement getApplyButton() {

return $(applyButton);

}

* Nahradenie **AssertTrue a AssertFalse** v suvislosti s predoslou metodou:

Selenium :

assertTrue(savingsCalculatorPage.getApplyButton().isEnabled());

assertFalse(savingsCalculatorPage.getCalculatedTotalIncomeElement().getText().isEm pty());

savingsCalculatorPage.getCalculatedTotalIncomeElement().shouldNotBe(Condition.empty);

Selenide :

savingsCalculatorPage.getApplyButton().shouldBe(Condition.enabled);

savingsCalculatorPage.getCalculatedTotalIncomeElement().shouldNotBe(Condition.e mpty) .shouldHave(Condition.***text***("kr"));

//nema byt prazdny a ma obsahovat menu kr, je to len cast textu, ak by som chcel cely text, tak dam exactText

* Obalenie viacerych metod do Selenidu (pouzitim xpath a cssSelectora) :

Selenium :

public SelenideElement getCalculatedTotalIncomeElement() {

return resultElement.findElement(By.xpath("./div[1]/p"));

}

public SelenideElement getCalculatedInterestIncomeElement() {

return pageDriver.findElement(By.cssSelector("div.result > div:nth-child(2) p"));

}

public SelenideElement getCalculatedRiskElement() {

return resultElement.findElement(By.xpath("./div[3]/p"));

}

Selenide :

public SelenideElement getCalculatedTotalIncomeElement() {

return $(resultElement).find(By.xpath("./div[1]/p"));

//hlada prvy div v ramci lubovolneho elementu a v nom p element

}

public SelenideElement getCalculatedInterestIncomeElement() {

return $(resultElement).find(By.cssSelector("div.result > div:nth-child(2) p"));

//hladanie pomocou cssSelectora, v div elemente s triedou result hlada dalsi div element a jeho druheho potomka

}

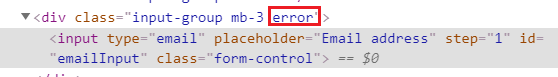
public SelenideElement getCalculatedRiskElement() {

return *$*(resultElement).find("div",2).find("p");

//hlada treti div v ramci lubovolneho elementu a v nom p element, v cssSelectoroch(Javascripte zacina indexovanie od nuly)

}

* Vratenie **rodiča** v ktorom sa nachadza error :



Selenium :

return pageDriver.findElement(By.xpath("//input[@id='emailInput']/.."));

//v Seleniu sa rodic definuje dvoma bodkami cez xpath

Selenide :

return $(emailInput).parent();

//toto je prvy sposob, parentov mozno retazit, tak sa dostanem v strukture vyssie, napr. return $(emailInput).parent().parent();

return $(emailInput).closest(“div”);

//toto je druhy sposob, closest najde najblizsieho rodica na zaklade tagu

* **Hover** a **Scroll** v Selenide :

Selenium (**hover**) :

Actions action = new Actions(driver);  
 WebElement we = *$*("div.saving-detail");  
 action.moveToElement(we).build().perform();  
 Thread.*sleep*(300);  
 *assertEquals*("rgba(4, 102, 156, 1)", we.getCssValue("background-color"));

//ten threadsleep je tam kvoli tomu, lebo pri prejdeni mysou na element sa mu farba nezmeni okamzite, ale postupne, ide o hover, ked po prejdeni mysou na element sa nieco vysvieti, pripadne zmeni kurzor na ruku

Selenide (**hover**):

*$*("div.saving-detail")  
 .hover()  
 .shouldHave(*cssValue*("background-color","rgba(4, 102, 156, 1)"));

//na element div s classou saving-detail prejdem mysou a ocakavam farbu zadanu v rgb, farba sa nachadza v artribute backgroung color, neni potrebne cakat ako v pripade selenia, selenide si na to pocka sam, jak je zvykom

Selenium (scroll) :

WebElement lastRow = *$*("table > tbody > tr:last-child");  
 ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", lastRow);

Selenide (scroll) :

$("table > tbody > tr:last-child").scrollIntoView(false);

//preskrolujem sa na posledny element v tabulke(last-child), parameter boolean v scrollIntoView urcuje, ci hladana hodnota sa ma preskrolovat na vrch (top) stranky

* Metoda **because** v selenide :

savingsCalculatorPage  
 .getRecentRequestDetail()  
 .find("p.fund-description")  
 .shouldHave(*exactText*(fundToSelect).because("Preco toto slovicko hladam"));

//because pouzijem, pokial pri hladani nejakeho textu chcem zadat dovod hladania, je to hlaska, pomocou ktorej viem lepsie lokalizovat chybu, v priklade to bolo pouzite pre string pri exactText

* Inicializacia elementov v Selenide, použitie **page** :

private SelenideElement oneTimeInvestmentInput= $(byId("oneTimeInvestmentInput"));

private SelenideElement fundSelect= $(byId("fundSelect"));

private SelenideElement applyButton= $("button.btn");

private SelenideElement resultElement= $("div.result");

private SelenideElement mostRecentSavingsDetail= $("ul.saving-list").find("div.saving- detail");

public SavingsCalculatorPage() {

page(this);

//pomocou page sa inicializuju elementy na stranke, ktore su definovane vyssie

}

//pri selenide nie je potrebny **page factory pattern**(FindBy elementov)

* **Statický blok**, vykoná sa raz pri každom vzniku novej inštancie, vhodný pre konfiguračné nastavenia:

public class TestBase {

protected WebDriver driver;

static{

Configuration.baseUrl = <http://localhost:82>;

}

//nejaky dalsi kod

}

Pri spustani stranky uz potom **nepouzijem** baseUrl :

@Before

public void openPage() {

open("/fellowship.php");

//postaci mu relativna cesta, cize to co je za baseUrl, lebo ma baseUrl definovany v konfiguracii

}

Ak by som chcel zavolat baseUrl, tak to zavolam cez **prazdny** string:

@Before

public void openPage() {

open("");

}

* Ďalšie **konfiguračné nastavenia** :

Nastavenie **defaultného čakania** (klasicky sú to 4 sekundy) :

static{

Configuration.timeout = [4000](http://localhost:82);

//nastavenie defaultneho casu v konfiguracii v milisekundach

Configuration.pollingInterval = 200;

//interval v ktorom sa na stranku pozera, ci nastala nejaka podmienka ohladom zobrazenia elementu napr, vsetkych ukonov, ktore sa na webe deju, defaultne je tato hodnota nastavena na 200

}

**Driver** v Selenide :

//zakomentovaný **driver** v TestBase v časti Before, pretože Selenide to spraví za nás a inicializaciu drivera nepotrebuje

//tiez zakomentované (**nepotrebne**) ukoncenie a uzavretie drivera v časti After, ide o **close()** a **quit(),** aj toto spravi Selenide

**Headless** mod :

static{

Configuration.headless = true;

//pri tomto mode nebude otvarat stranky vizualne, len na pozadi, nastavuje sa to v konfiguraku definovanej v TestBase v statickej metode

}

Ostatne konfiguracie :

static{

Configuration.startMaximized = true;

//otvori stranku v maximalnom mode(asi sa mysli **full screen**)

Configuration.holdBrowserOpen = true;

//po skonceni testu necha **otvoreny** **prehliadac**

Configuration.browser = Browsers.FIREFOX;

//**nastavenie** ineho **weboveho** **prehliadaca**, nie je potrebny ziadny geckodriver

Configuration.remote = “localhost:4444/wd/hub”;

//**nastavenie** **remote** **servera**, napr selenium server pokial pouzivam

Configuration.clickViaJs = true;

//pokial mam viacero elementov na sebe, napriklad obrazky sa **prekryvaju**, tak touto konfiguraciou dokazem **zacielit** aj na **obrazok**, ktory normalne nevidim, lebo sa nachadza **za niecim inym**

Configuration.reportsFolder = "src\\test\\resources\\reporty";

//**cesta** kam sa ukladaju **screenshoty** **a html stranka** pred spadnutim, defaultna adresa je Nazov projektu -> build -> reports -> tests

//toto je cesta pre **windows**, pre mac by to bolo “src/test/resources/reporty“

Configuration.reportsUrl = “cesta na web napr pre Jenkins”

//cesta pre **reporty** **na nejake webove ulozisko**, pre lokalne ulozisko sluzi reportsFolder

}

* **Hromadne klikanie** na konkretne body:

$$("img").forEach(SelenideElement::click);

//klikni mi na kazdy element obsahujuci cssSelector img

$$("").shouldHave(CollectionCondition.size(0));

//skontroluj, ci velkost kolekcie je 0 (neostal ziadny img na obrazovke)

* **Reporty a screenshoty** :

//pri lubovolnej chybe robi Selenide automaticky screenshot a tiez uklada stranku v html kode pred spadnutim, vsetky tieto informacie najdem vo svojom projekte tu:

Nazov projektu -> build -> reports -> tests

//v konfiguracii mozem nastavit aj inu cestu pomocou **reportsFolder**

* **Nedostatky pri testoch :**

**Zastavenie dynamickej webovej stránky :**

//počas testov ked chcem zistit informácie o elemente, ktorý rýchlo zmizne na dynamickej webovej stranke, tak ju potrebujem zastaviť, na čo mi slúži príkaz debugger

//F12 v chrome -> Console -> napisem debugger a stlacim enter

//stranku viem manazovat a zastavovat pomocou F8

**Vypnutie webDrivera, pripadne len okna:**

//niekedy sa moze hodit vypnutie okna alebo rovno webdrivera, je dobre davat nasledujuci prikaz do sekcie @After

@After  
 public void closeBrowser(){  
 WebDriverRunner.*closeWebDriver*();  
 WebDriverRunner.*closeWindow*();  
 }