*Selenium Testovanie*

# @Before

public void setUp(){ driver = new FirefoxDriver(); }

# @Test

public void test(){}

# @After

public void tearDown(){ driver.close();

driver.quit();}

public WebDriver driver; driver.get(BASE\_URL);

# //Najdenie elementu

driver.findElement( By.xpath(“”) By.name(“”)

By.id(“”) By.className(“”)

By. cssSelector("button.btn-success") By.tagName(“”)

//Priklady:

driver.findElement(By.xpath("//input[@type='email']")).sendKeys(email);

driver.findElement(By.xpath("//input[@type='email']")).getAttribute("value"));

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

driver.findElement(By.id("clicks")).getText();

driver.findElement(By.id("firstInput")).clear();

driver.findElement(By.xpath("//input[@value='Eva Mazikova']")).isEnabled();

//vrati true/false podla toho ci sa da nan kliknut

driver.findElement(By.xpath("//input[@value='wurst']")).isSelected();

//je zaskrnute?

driver.findElement(By.xpath("//img[@class='brano']")).isDisplayed());

//je zobrazene?

# //Najdenie zoznamu elementov

List<WebElement> tittles = driver.findElements(By….)

# //XPathy

"//img[@src='img/conchita.jpg']" "//div[contains(@class,'colours')]/h1"

"//input[@type='email']" "//\*[@id=\"clickMe\"]

# //XPath pre nejaky zoznam

List<WebElement> rows = driver.findElements(By.xpath("//table/tbody/tr"));

for (WebElement row : rows) {

Assert.assertFalse(row.findElement(By.xpath("./td[3]")).getText().isEmpty());}

//kazdy treti stlpec ci je neprazdny(AssertFalse)

# //Asserty

Assert.assertEquals(správa, element1, element2); //rovna sa? sprava byt nemusi

Assert.assertTrue(vyrok); //je pravda? true/false

Assert.assertFalse(vyrok); //nie je pravda? true/false

# //Navigacia(dozadu, dopredu, obnoviť)

driver.navigate().back(); driver.navigate().forward(); driver.navigate().refresh();

# //Manage, praca s velkostou okna

driver.manage().window().maximize(); //fullscreen

driver.manage().window().setSize(new Dimension(300,2500)); //nastavenie velkosti

driver.manage().window().setPosition(new Point(500,500)); //umiestnenie

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);

# //Cakania na nejaku udalosť

new WebDriverWait(driver,5).until(ExpectedConditions.presenceOfElementLocated(By…);

new WebDriverWait(driver,5).until(ExpectedConditions.visibilityOfElementLocated(By…);

//presence caka na objavenie v html/css kode; visibility caka na objavenie na obrazovke

new WebDriverWait(driver,5).until(ExpectedConditions.invisibilityOfElementLocated(By..);

//cakanie na neviditelnost element

new WebDriverWait(driver,10).withMessage("Ocakava sa pocet 5 mimonov")

.until(ExpectedConditions.numberOfElementsToBe(By..),numberOfMinions));

//cakanie na zobrazenie urciteho poctu elementov

new WebDriverWait(getDriver(),5).until(ExpectedConditions

.attributeToBe(By.id("waitForTextInput"),"value","dary !!!"));

//cakanie na urcity text, ktoreho artribut je value a hodnota je dary!!!

new WebDriverWait(getDriver(),10).until(ExpectedConditions

.attributeContains(By.id("startWaitForProperty"),"disabled","true"));

//cakanie na nejaku hodnotu artributu, napr. ci je nieco disabled

new WebDriverWait(getDriver(),5).until(ExpectedConditions

.numberOfWindowsToBe(numberOfPages));

//cakanie na urcity pocet otvorenych okien

new WebDriverWait(driver, 10).until(ExpectedConditions

.textToBePresentInElement(detailListTitle, sin.getTitle()));

# //Ziskanie farby

tittle.getCssValue("color") //priklad na pristup k CSS hodnotam

hexColor = Color.fromString(tittle.getCssValue("color")).asHex()

# //Akcie(posun)

Actions action = new Actions(driver);

action.clickAndHold(donald).moveByOffset(100,0).release().build().perform();

//posunutie o presnu hodnotu

action.dragAndDrop(donald,finish).build().perform(); //posun donalda na finish

//build a perform tam musia byt aby sa posun vykonal

action.moveToElement(trafficLight).build().perform();

//prejdenie mysou na trafficLight

# //ScreenShot

File screenshot = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(screenshot, new File("D://Programovanie//screenshot.png"));

# //Prepnutie na upozornenie

Alert alert = driver.switchTo().alert();

alert.getText(); alert.accept(); alert.dismiss(); alert.sendKeys(“”);

# //Prepnutie na vyskakujuce okno a rodicovsku page

for (String handle : getDriver().getWindowHandles()){

getDriver().switchTo().window(handle); } //prepinanie medzi hashkodami okien

getDriver().switchTo().window(parrentPageInfo); //prepnutie na rodicovske okno

parrentPageInfo = getDriver().getWindowHandle() //obsah premennej vyssie

# //Vyber zo zoznamu elementov

new Select(driver.findElement(By…)).selectByVisibleText("Pikachu"); //podla textu

# //Priklad Test Suite na 4 testy

import org.junit.runner.RunWith;

import org.junit.runners.Suite;

@RunWith(Suite.class)

@Suite.SuiteClasses({

TestClick.class, SemaforTest.class,

RedAlert.class, RegistraciaTest.class})

public class TestSuite {}

# //Verify test(neprerusi sa testovanie po asserte)

try { //nejaka negacia assertu, ktoreho zoznam chcem zobrazit v chybe

} catch (Error e) {verificationErrors.append(e.toString());}

@After

public void tearDown(){

String verificationErrorString = verificationErrors.toString();

if (!"".equals(verificationErrorString)) {fail(verificationErrorString);}}

# //Javascript v teste

((JavascriptExecutor)getDriver()).executeScript("arguments[0].blur()",

getDriver().findElement(By.id("waitForBlur")));

//blur je odkliknutie mysou na nejaku inu plochu, v argumente sa udava cislo elementu, pre ktore sa ma script vykonat, 0 pre prvy, 1 pre druhy…za argumentum nasleduju tieto element

"arguments[0].click()" //kliknutie v javascripte pre prvy element za nim

executeScript("arguments[0].style.border ='3px solid red'", row); //zmeni oramovanie

executeScript("arguments[0].scrollIntoView()",lastrow); //preskrolovanie nakoniec

executeScript("arguments[0].scrollIntoView(true)",fifthrow); //zoskroluje tak aby

5.riadok bol navrchu

executeScript("window.scrollBy(0,500)"); //skrolovanie o pocet pixelov

executeScript("window.scrollBy(0,"+pageHeight+")");

pageHeight = (long) js.executeScript("return document.body.scrollHeight");

//skrolovanie o vysku stranky

# //Singleton

private static WebDriverSingleton instance;

public static WebDriverSingleton getInstance() {

if (instance == null) {instance = new WebDriverSingleton();}

return instance;}

# //Chrome driver

private void initializeGoogleChrome() {

System.setProperty("webdriver.chrome.driver","src/test/resources/drivers/chromedriver.exe");

// kluc zistime,ak nastavime driver a spustime test, ktory spadne

// cesta k chromu

driver = new ChromeDriver();}

# // PhantomJS driver

private void initializePhantomJs() {

System.setProperty("phantomjs.binary.path", "src/test/resources/drivers/phantomjs.exe");

driver = new PhantomJSDriver(); }

# //Anotacie

@FindBy(name = "message")

private WebElement message;

//FindBy mi tiez najde element, netreba pouzivat findElement..., vhodne je pod nim pouzivat typ WebElement(osobna skusenost)

//musim do konstruktora ale vlozit nasledujuci kod, pre FindBy:

PageFactory.initElements(driver, this);

--------------------------------------------------------------------------------------------------------------

@RunWith(Categories.class) //toto cele je test suite

@Categories.IncludeCategory(ReleaseTest.class) //spusti danu kategoriu z test suite

@Categories.ExcludeCategory(SmokeTest.class) //spusti mimo tejto kategorie z test suite

@Suite.SuiteClasses({ DummyTest.class, WaitForItTest.class, PrimeTest.class})

@RunWith(ConcurrentTestRunner.class) //paralelne spustanie testov, pozor na zavislost

@Category({SmokeTest.class, ReleaseTest.class})//urcim kategorie nad testom, mozem aj 1

@Ignore //tato anotacia bude ignorovat test a aj sa mi zobrazi v teste ako ignorovana

-----------------------------------------------------------------------------------------------------------------

@Rule

public RepeatingRule repeatingRule = new RepeatingRule();

//opakovanie testu xkrat, testujeme tym zatazenie/ performance testy, tento kod dam mimo testu

@Repeating(repetition = 3)

//tento kod dam nad test

@RunWith(Parameterized.class)

//anotacia pre parametrizovanu triedu, tuto anotaciu (RunWith) viem pouzit len raz a pisem ju nad triedu(class) resp mimo triedy

@Parameterized.Parameters //anotacia pre parametre

public static List<Object[]> getData() {

return Arrays.asList(new Object[][]{{1, true}, {2, true}, {15, false}, {29, true},

{84, false}}); }

//toto cele uz pisem do triedy, zvycajne po definovani premennych

//este pre prehladnost pridam konstruktor:

public ParameterTest(int number, boolean expectedPrime) {

this.number = number; this.expectedPrime = expectedPrime; }

@Rule

public ExpectedException expectedException = ExpectedException.none();

//toto napisem(do triedy), ked bude ocakavat nejaku vynimku, v tom pripade test nespadne

# //Pristup k excelu(priklad)

ExcelReader primeExcelReader = new ExcelReader(TEST\_DATA\_PATH);

Sheet sheet = primeExcelReader.getSheetName(SHEET);

for (Row row : sheet) { //iteracia

int number = (int) row.getCell(0).getNumericCellValue();

boolean expectedPrime = row.getCell(1).getBooleanCellValue();}

# //Streamy

List<String> vendelinovia = new ArrayList<String>();

vendelinovia = rows.stream()

.filter(element -> getName(element).equals("Vendelin")) .map(this::getSurName) //prvy sposob

.map(element -> getSurName(element)) //druhy sposob

.collect(Collectors.toList()); //ulozi do listu

//pre rozne nastavenia firefox profile(napr test mobilnych stranok) si pozri UserAgent a WebDriverSingleton triedu v seleniumAdvanced projekte

Cucumber

**ABY BOL ZAPNUTY MAMP, INAK TO NEPOJDE!!!**

Nahranie projektu z gitu:

File -> New -> Project from version control ->git

Link: <https://github.com/skarbala/selenium_cucumber.git>

Prepinanie medzi git branche-ami:

Vpravo dole Git master -> origin/cucumber\_solution -> Checkout As

Na prepinanie spat dam master -> checkout

Kontrola Maven:

ALT+F12 -> mvn -v

BDD (Behavioral driven development)- vyvoj aplikacie zalozenom na spravani - aby scenarom pochopil aj zakaznik

-Test Driven Development - zacina sa s testami, developer na zaklade testu pise kod

-uplatnuje sa pri Unit testoch

-Specification By Example - ziva dokumentacia na zaklade ktorej sa vytvaraju/spustaju testy

Instalacia Cucumber:

1.plugin - File ->Settings -> Plugins -> Instal JetBrains plugin ->

Cucumber for Java

2. Stiahnutie cucmber - Cucumber.io -> najdeme dependency -> dame do pom

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-java</artifactId>

<version>4.0.0</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>2.4.0</version>

<scope>test</scope>

</dependency>

Cucumber sa zaobera pracou s features, struktura:

**Features**: Click me baby

As a user

I want to click a button

For no reason at all, only for study purpose

**Scenario**: Click on button once

**Given**(pripravna faza) -> pouzivatel je registrovany

**When**(akcia ktoru vykoname na stranke) -> vyplnim email a dam dalej

**Then**(Overovacia faza) -> Overim ocakavany vysledok

Iny priklad(jazyk, v ktorom je to pisane je **Gherkin**):

Scenario: Confess new valid Sin without tags

**Given** I am on sin city page

**When** I enter sin title

**And** I enter sin author

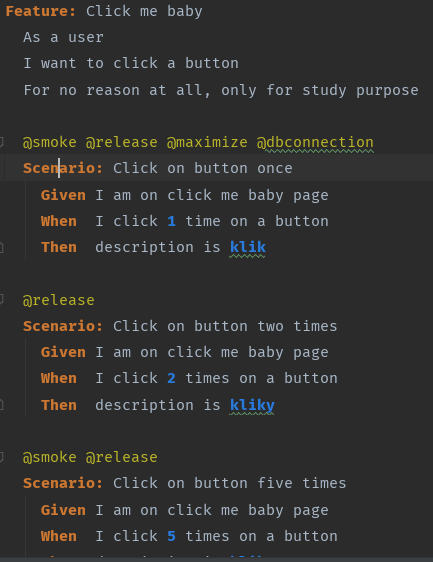
**And** I enter sin message

**When** I confess(priznat sa) my sin

**Then** my sin appears(objavit sa) in the list of all sins

**//**miesto And viem pouzit aj **But**, ktory sa dava pri nestastnych scenaroch, alebo ked neviem presne urcit o aky ide, da sa dat aj hviezdicka(\*) ale to sa neodporuca, priklad: **But** I enter password '1234'

* V jednom feature mozem viac viacero scenarov, ak kliknem pravym na feature a run, spustim vsetky scenare, ak kliknem na konkretny scenar,tak spustim len ten



* Spúšťač, kde si definujem tagy scenarov, ktore chcem spustit, miesto Suite.class dam **Cucumber.class** a musim cez mavenrepository implementovat este Cucumber Junit

Nasledne potom takto vyzera ten Spúšťač(**Runner**):

import cucumber.api.CucumberOptions;  
import cucumber.api.junit.Cucumber;  
import org.junit.runner.RunWith;  
  
@RunWith(Cucumber.class)  
@CucumberOptions(  
 features = "src/test/resources/features/ClickMe.feature",  
 glue = {"steps", "hooks"},  
 //OR  
// tags = {"@smoke or @release"},  
 //AND  
// tags = {"@smoke", "@release"},  
 tags = {"not @ignore"},  
 junit = "--step-notifications"  
)  
public class ClickMeBabyRunner {  
}

* + - V anotacii **@CucumberOptions** definujem skupiny features, ktore sa maju spustit, bud priamo pomocou ‘**features**’ alebo skupinovo pomocou tagov ‘**tags’**
    - V casti **glue**(lepidlo) definujem, kde sa casti mojho kodu nachadzaju
    - Časť **junit = “--step-notifications“** mi zaručí podrobnejší popis pri kontrole feature, viac to rozvetví(na **Given**, **When**, **Then**)
* **Cucumber JUnit**

<dependency>

<groupId>io.cucumber</groupId>

<artifactId>cucumber-junit</artifactId>

<version>2.4.0</version>

<scope>test</scope>

</dependency>

* Prepojenie medzi samotnym kodom a uzivatelskym feature je pomocou odkazu, ktory sa nachadza v anotacii napr pre given:

**@Given("^I am on click me baby page$")**

Je prepojený s nasledujúcim kódom

**Given** I am on click me baby page

* + - Tento kod za nas sam prepoji idea pomocou ALT+ENTER ak oznacim nejaky prikaz vo feature(napr text za Given), dalej dam **Create steps definition ->** vyberiem svoju triedu v Steps
* **Background:** I am on Sin City page

Given I open Sin City page

* + - * + **Background** sa spustí pred každým scenárom(Scenario)
        + **Background** by mal byt co najkratsi, aby si ho uzivatel vedel zapamätať
        + V rámci Background by malo byť všetko to, čo je všeobecné pre všetky scenáre
* **@When("^I enter number (**[**\\d+)$**](file:///\\d+)$)**")**
* Prepojenie pomocou regularneho vyrazu, ktory nam hovori, ze tam musi byt aspon jedno číslo
* Zapis scenara pomocou parametrov, aby sme sa neopakovali:
* **Scenario :** Check the number 1  
   **Given** I am on the Optimus prime page  
   **When** I enter number **1  
   And** I ask Optimus if it is prime  
   **Then** Optimus says that the number is **prime  
    
  Scenario :** Check the number 2  
   **Given** I am on the Optimus prime page  
   **When** I enter number **2  
   And** I ask Optimus if it is prime  
   **Then** Optimus says that the number is **prime  
    
  Scenario :** Check the number 3  
   **Given** I am on the Optimus prime page  
   **When** I enter number **3  
   And** I ask Optimus if it is prime  
   **Then** Optimus says that the number is **prime  
     
  Scenario :** Check the number 4  
   **Given** I am on the Optimus prime page  
   **When** I enter number **4  
   And** I ask Optimus if it is prime  
   **Then** Optimus says that the number is **not prime**

Kod hore zjednodušíme na tvar :

**Scenario Outline:** Check the number **<**number**>  
 Given** I am on the Optimus prime page  
 **When** I enter number **<number>  
 And** I ask Optimus if it is prime  
 **Then** Optimus says that the number is **<expectedResult>  
  
 Examples:  
 | number | expectedResult |  
 | 1 | prime |  
 | 5 | prime |  
 | 6 | not prime |  
 | 47 | prime |  
 | 100 | not prime |**

* Miesto Scenario použijeme **Scenario Outline**  je to akoby nejaky **pattern pre scenare**, kde sa meni len nejaky parameter, tie parametre su <number> a <expectedResult>
* Do **Examples** dam konkretne hodnoty tohto parametra, pre sformatovanie pod sebou mozno pouzit klavesovu skratku **ALT+SHIFT+L**
* Nasledujúci kód slúži na **označenie viacerých tagov(JEDNOROZMERNÁ TABULKA)**, aby som to nemusel robiť pojednom, môžem to spraviť týmto spôsobom:

**And** I select following sins  
 **| murder | robbery | car accident |**

Cucumber hore uvedeny kod zobere ako list stringov, preto pri písaní funkcionality to môžeme spraviť takto:

@And("^I select following sins$")  
public void iSelectFollowingSins(List<String> tags) { nejaky kod }

* V nasledujucom kode uvidime, ako premapovat **list stringov na list sintype-ov**:
* List<SinType> sinTypes = tags  
   .stream()  
   .map(s -> SinType.*valueOf*(s.toUpperCase().replaceAll("\\s", "\_")))  
   .collect(Collectors.*toList*());

Ten replace mi nahradi medzeru podtrznikom

* **DVOJROZMERNÁ TABULKA:**
* **Then** values in table are following  
   **| total\_price | 1.5 |  
   | number\_of\_bananas | 1 |  
   | discount | 0.00 |  
   | price\_for\_one\_banana | 1.5 |**

Kde pri definovani spravania sa pouzije miesto listu tabulka z cucumber(**DataTable**):

@Then("^values in table are following$")  
public void valuesInTableAreFollowing(DataTable table) {  
 Map<String, String> tableMap = table.asMap(String.class,String.class);  
 //vo vnutri definujem akeho typu je kluc aj hodnota, ktore maju byt navzajom previazane  
 getBananaPage().checkTableValue(tableMap.get("total\_price"), BananaTableColumn.*TOTAL\_PRICE*);  
}

* DataTable je tabulka(asi 2-rozmerna) z kniznice cucumber
* V kode nasledne zmenim tuto tabulku na mapy, kde mam kluce s hodnotami, aby sa mi nemiesali udaje v pripade ze pomiesam poradie(nech je kluc s hodnotou na seba viazany)
* V poslednom kroku skontrolujem hodnoty
* **Komentovanie** v cucumber pomocou #
* Pri debugovani mozem pouzit klavesovu skratku **ALT+F8** pre **zistenie obsahu** nejakej premennej, vo videu to bolo na zistenie DataTable tabulky

Regularne vyrazy(aj z Javy)

Zhrnutie:

? - 0-1 krat

\* - 0-x krat

+ - 1-x krat

{n} - n krat

{n,} - aspon n krat

{n,m} - aspon n krat, najviac ale az m krat

[xyz] - hladame znaky od x po z

[^xyz] - negacia znakov, vsetky znaky okrem x, y, z

[a-zA-Z] - pismena od 'a' po 'z' a od 'A' po 'Z'

[a-d[m-p]] - zjednotenie dvoch rozsahov, od 'a' po 'd' alebo od 'm' po 'p', [a-dm-p] druhy zapis

[a-z&&[def]] - prienik dvoch podmienok, od 'a' po 'z' a sucasne d, e alebo f

[a-z&&[^cd]] - odcitanie, od 'a' po 'z' okrem pismen c a d, iny zapis [a-be-z]

[a-z&&[^m-p]] - odcitanie, od 'a' po 'z' okrem pismena od 'm' po 'p', iny zapis [a-lq-z]

. - hocijaky znak

\d - cislica [0-9]

\D - okrem cisel 0-9, cize [^0-9]

\s - biely znak, cize [\t\n\x0B\f\r]

\S - nie biely znak [^\s]

\w - slovny znak, cize [a-zA-Z\_0-9]

\W - nie slovny znak, cize [^\w]

(abc+) - 'a', 'b' a 'c' 1-n krat(len 'c'), vsetko tam musi byt

(abc)+ - cely retazec je 1-n krat

[abc]+ - bud 'a', 'b' alebo 'c' je tam 1-n krat(vztahuje sa pre vsetky tri)

Greedy:

X? - 0-1 krat

X\* - 0-x krat

X+ - 1-x krat

X{n} - n krat

X{n,} - aspon n krat

X{n,m} - aspon n krat, najviac ale az m krat

Reluctant:

X?? - 0-1 krat

X\*? - 0-x krat

X+? - 1-x krat

X{n}? - n krat

X{n,}? - aspon n krat

X{n,m}? - aspon n krat, najviac ale az m krat

* greedy najde najvacsi foo...bar, cize fooxxbarxxxxfooxbar
* rozdiel medzi greedy a possesive je ten, ze greedy najde "foo",potom prejde nakoniec retazca a od konca hlada "bar"
* reluctant hlada co najmensie vyskyty, cize fooxxbar a fooxbar budu vysledky

Possesive:

X?+ - 0-1 krat

X\*+ - 0-x krat

X++ - 1-x krat

X{n}+ - n krat

X{n,}+ - aspon n krat

X{n,m}+ - aspon n krat, najviac ale az m krat

Boundary Matches:

^ - hlada na zaciatku vety(riadku), pise sa pred vyraz

$ - hlada na konci vety(riadku), pise sa za vyraz

\b - slovna hranica, oddelene medzerami

\B - neslovna hranica, neoddelene medzerami

Pri chybe sa vola PatternSyntaxException

\p{Lower} - male pismenka

\p{Upper} - velke pismenka

* @When("^I click **(\\d+)** times? on a button$")

public void clickOnButton(int **numberOfClicks**) {...}

* Parameter v zavorke(v nasom pripade hocijake cislo) sa musi zhodovat s parametrom v metode(numberOfClicks) aj poctom parametrov
* Tuto metodu potom viem pouzit pre hocijake cislo, pokial je to cislo v zatvorke tak ho bere ako parameter(ak je v anotacii when)
* Otaznik znaci, ze znak pred nim je nepovinny
* Metoda pomocou regularneho vyrazu, **prime or not prime**(tato ista metoda je pomocou tabulky zapisana hore, to je iny sposob zapisu)
* @Then("^Optimus says that the number is (prime|not prime)$")  
  public void optimusSaysThatTheNumberIsPrime(String expectedPrime) {  
   if (expectedPrime.contains("not")) {  
   getOptimusPage().checkResult(false);  
   } else {  
   getOptimusPage().checkResult(true);  
   }  
  }
* Pokial mam v kode scenaru nejaky regularny vyraz, v scenari mozem stlacit **CTRL+SPACE** a zobrazi mi ponuknute moznosti, ktore vyhovuju danemu regularnemu vyrazu
* @Given("^I (**?:**am on|open) Sin City page$")

public void iAmOnSinCityPage() {

getSinCityPage().openPage(); }

* Kombinacia ?: znamena, ze to co je za tym, tam byt nemusi ale aj moze
* **CTRL+SHIFT+SIPKA HORE/DOLE** - posun oznaceneho kodu hore/dole
* **Runner(**priklad**)**:

@RunWith(Cucumber.class)  
@CucumberOptions(  
 features = "src/test/resources/features/sinCity",  
 glue = {"steps.sinCity", "hooks"},  
 junit = "--step-notifications"  
)  
public class SinCityRunner {  
}

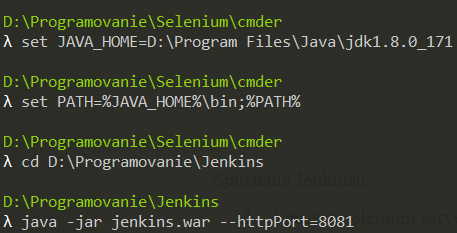
* V runneri sa môže stať, že ked pre glue zadáme len steps, môže dojsť k duplicite metod a to vypíše chybu. Tú duplicitu viem obísť práve tak, že vytvorim nový priečinok, kam jeden z tých duplikátov dám a v glue to len prepíšem napr na **steps.sinCity**
* **Tagy**:

@sin-city  
**Feature:** Confess new sin  
 As a sinner  
 I want to confess a sin  
 So that I have a clear conscience  
  
 **Background:** I am on Sin City page  
 **Given** I open Sin City page  
  
 @smoke @release  
 **Scenario:** Confess new valid Sin without tags  
 **And** I enter sin title **pocuvam one direction** and sin author **brano  
 And** I enter sin message  
 **When** I confess my sin  
 **Then** my sin appears in the list of all sins

* Tagy sa píšu za zavináč, určujú skupinu testov, ktorá sa má spustiť, tag viem písať pre samostatné scenáre, ale môžem ho dať aj nad Feature pre všetky scenáre. Jeden scenár/feature môže mať aj viac tagov
* Tagy sa následné spúšťajú cez runner:
* @RunWith(Cucumber.class)  
  @CucumberOptions(  
   features = "src/test/resources/features/ClickMe.feature",  
   glue = {"steps", "hooks"},  
   tags = {"@smoke or @release"},//OR  
   junit = "--step-notifications"  
  )
* Miesto **OR** viem použiť aj **AND** (tags = {"@smoke", "@release"})
* Pripadne **NOT** ak nechcem aby sa mi test spustil(tags = {"not @ignore"})
* **Poradie(order)** a ďalšie spúšťanie pomocou **tagov (value)**:
* @Before(value = "@dbconnection", order = 1)  
  public void dbconnect() {  
   System.*out*.println("pripajam sa na databazu");  
  }  
    
  @Before(order = 2)  
  public void setup() {  
   WebDriverSingleton.*getInstance*().initialize();  
  }  
    
  @Before(value = "@maximize", order = 3)  
  public void maximize() {  
   WebDriverSingleton.*getInstance*().getDriver().manage().window().maximize();  
  }
* Tento kód mi zaručí, že ešte pred spustením testu spustí kód(scenár) označený ako **@maximize** alebo **@dbconnection**, samotný kód je pod anotáciou Before, ide o to, že do **value** napíšem, pre aký tag to chcem použiť...ak mám napr 3x Before a chcem určiť poradie spúšťania, spravím to pomocou **order.** Toto celé môže byť dobré pri inicializácii databázy, databáza sa bude inicializovať len vtedy, ak daný scenár s databázou bude pracovať napr.
* Order a value viem použiť aj pre @After a predpokladám, že aj pre dalšie anotácie, rovnako vo value viem použiť napr NOT napr. @~dbconnection

Jenkins

Spustenie Jenkinsu pomocou cmder:



* Najprv musim prehodit javu10 na javu8(ten link udava cestu na javu 8)
* Nastavenie cesty Jenkinsu a potom jeho spustenie cez port 8081
* Do Jenkinsu sa potom dostanem cez link: http://localhost:8081/

Heslo: pri vytvarani noveho uctu v Jenkinse sa prvotne heslo nachadza v D:\programovanie\jenkins\secrets\ initialAdminPassword

Po spusteni Jenkinsu potrebujem plugin pre cucumber, cize pojdem do Manage Plugins a najdem **cucumber reports**

Nasledne ich najdem v Jenkinse Configure->post-build actions-> cucumber reports

Tu najdem zakladne nastavenia, **cucumber report** vyzaduje subor typu **JSON**, ktory vyzera podobne ako XML.

Vytvorenie **JSON** v Jave:

@CucumberOptions(  
 features = "src/test/resources/features",  
 glue = {"steps", "hooks"},  
 tags = {"@smoke"},  
 junit = "--step-notifications",  
 format = {"json:target/reports/cucumber-reports/SmokeRunner.json"}  
)

* O vytvorenie suboru typu **JSON** sa postara Runner v casti **FORMAT,** v nom za dvojbodku zadam cestu pre subor JSON(target je priecinok vytvoreny po builde, uvidis ho niekde naspodu)

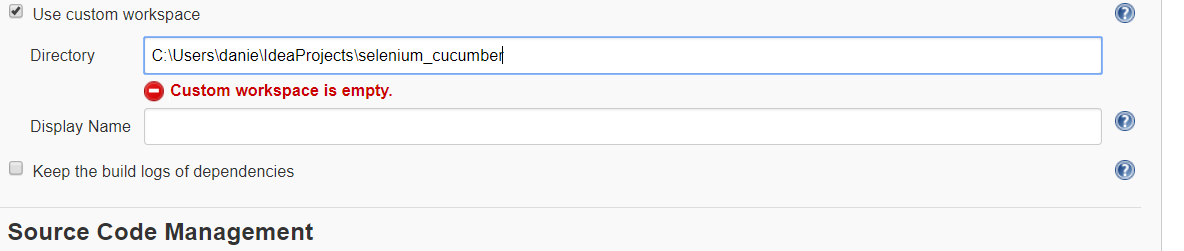
Nasledne pojdem do nastaveni jenkinsu a vyplnim nasledovne podla format:



* Toto cele s JSONom nastavujem az v nejakom Jobe, ktory predtym musim v Jenkinse vytvorit
* \*\*/\*.json znaci vsetky projekty konciace na .json

Postup nastavenia Jenkinsu:

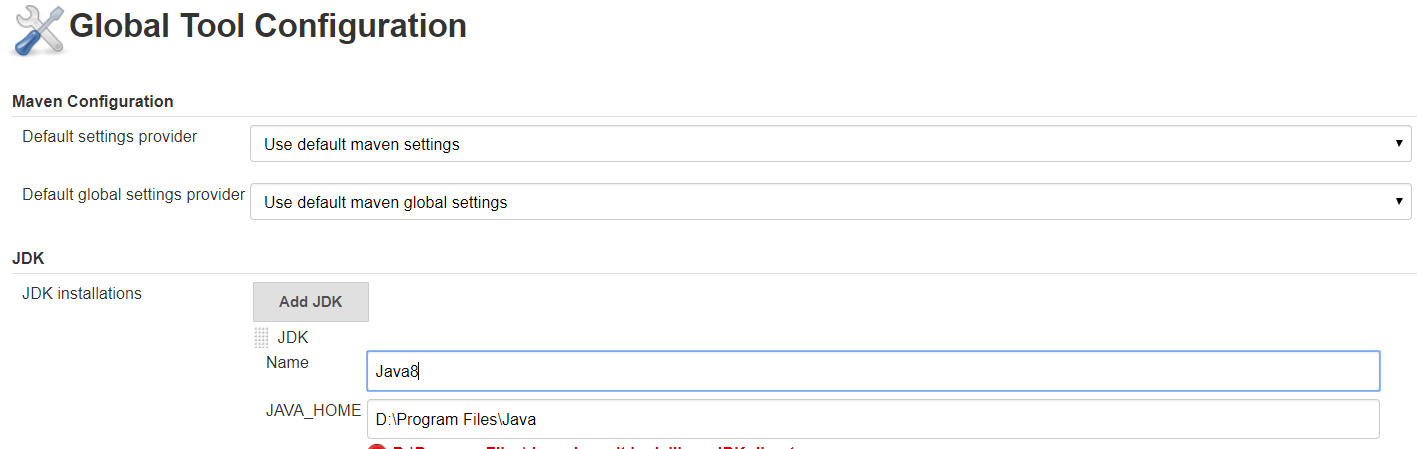
1. Nastavenie adresara v Jobe



1. Nastavenie v builde



1. Nastavenie kompilatora



* Pomocná trieda **Scenario**, ktorá slúži na logovanie do Cucumberu v Jenkinse

private static Scenario *scenario*;  
  
@Before  
public void initScenario(Scenario scenario) {  
 TaggedHooks.*scenario* = scenario;  
}

* Tento kod sa nachadza v TaggedHooks
* Pri kazdom teste sa vytvori instancia triedy Scenario, kodom vyssie si tu instanciu len ulozim do svojej premennej scenario
* V samotnom kode bol pre scenario vytvoreny aj getter, sem ho nedavam
* Samotné **logovanie**:

getSinCityPage().fillSinTitle(title);

getScenario().**write**("Title "+title+" entered");

* Samotné logovanie zabezpečí metóda triedy Scenario (write)

getScenario().write(WebDriverSingleton.getInstance().getDriver().getCurrentUrl());

* Tento kód mi vypíše URL aktuálnej stránky
* **ScreenShot**:

private void embedScreenshot() {  
 byte[] screenshot = ((TakesScreenshot)  
 WebDriverSingleton.*getInstance*().getDriver())  
 .getScreenshotAs(OutputType.*BYTES*);  
 *getScenario*().embed(screenshot, "image/png");  
}

* Screenshot sa uklada do pola byte-ov
* Nasledne scenario ma funkciu pre screenshot, kde ako druhy parameter berie ako String a tam udam ze ide o obrazok s akou koncovkou
* Screenshot sa zobrazuje v Jenkinse v cucumber reporte v casti hooks
* Celu tuto metodu davam do Hooks s anotaciou @After

@After  
public void closeScenario() {  
 if (*getScenario*().isFailed()) {  
 embedScreenshot();  
 }  
}

* **Slovencina** v cucumber :

<https://docs.cucumber.io/gherkin/reference/#spoken-languages>

* Pred Feature napisem tento kod, uplne na zaciatok:

**#language:sk**

* Takto potom mozem pisat scenare po slovensky