

# EP119 automātisko testu izstrādes instrukcijas

## Konfigurācija

---

Testu izstrādei nepieciešamā programmatūra

Nepieciešams *Git*: <https://git-scm.com/download/win>

Nepieciešams *JDK 8*: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Testu izstrādei izmanto *Eclipse* vidi.

Lejupielāde: <https://www.eclipse.org/downloads/> - jāizvēlas *Eclipse IDE for Java Developers*.

Priekš *Eclipse* nepieciešams spraudnis *EGit*. Visvienkāršāk to iespējams instalēt no *Eclipse* vides: *Help -> Install New Software..* -> Logā *Work with* iekopē adresi <http://download.eclipse.org/egit/updates/> -> *Select All* -> *Next* -> *Next* -> *Next* -> "I accept the terms" -> *Finish*

Nepieciešams arī *TestNG* spraudnis: <https://marketplace.eclipse.org/content/testng-eclipse>

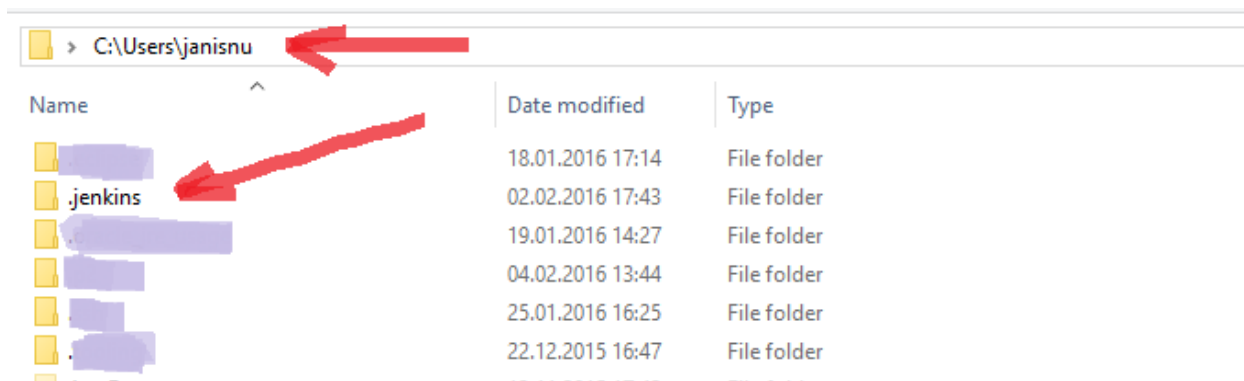
## Servera konfigurācija

Serverim nepieciešams *Firefox* pārlūks: <https://www.mozilla.org/lv/firefox/new/>

Serverim nepieciešams *JDK 8*: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Nepieciešams instalēt arī *Git*: <https://git-scm.com/download/win>. Instalējot *Git* jāatzīmē iespēja izmantot *Git* no *Windows command line*.

Mapi *.jenkins* (pielikumā) ievieto servera mapē *Users/[Lietotājs]/*. Mapē *.jenkins* glabājas *Jenkins* konfigurācija. Failus *jenkins.war* un *runJenkins.bat* novieto jebkurā servera mapē (abus vienuviet). Pēc tam *Jenkins* var palaist atverot failu *runJenkins.bat*.



Jāatver *Jenkins* kontroles panelis (interneta pārlūkā ierakstot *[servera ip]:8080*). Jāielogojas *Jenkins* sistēmā ar lietotājevārdu “admin” un paroli “zupa”. Jādodas uz *Manage Jenkins->Konfigurēt sistēmu* un jānorāda *JDK* un *Git* instalāciju mapes. Sadaļā *Extended E-mail notifications* laukā *Default recipients* jānorāda e-pasta adrese, uz kuru tiks nosūtīti testu rezultāti.

### Pirms testu izstrādes..

Nepieciešams izveidot *GitHub* kontu, caur kuru modificēt *GitHub* projektu *ep119* (<https://github.com/>). Tad jāprasa man, lai pievienoju tiesības augšuplādēt izmaiņas projektā *ep119*.

Nepieciešams *Eclipse* savienot ar *ep119* repozitoriju. To dara: *File->Import->Git->Projects from Git(with smart import)->Next->Clone URI->Next->URI:<https://github.com/JanisNulle/ep119.git>* , user un password – sava *GitHub* konta lietotājs un parole->*Next ->Next->Next*.

Pēc šī *ep119* repozitorijam uz datora jābūt kā *Eclipse* projektam.

Jauniegtajā projektā, mapē *UserFiles* faila *email* saturs jāizmaina uz e-pastu, kurš tiks izmantots testos, kad testos tiek prasīts lietotāja e-pasts.

Kad projektā veiktas kādas izmaiņas, tas jānosūta uz *GitHub* repozitoriju: Labais peles klikšķis uz *Eclipse* projekta->*Team->Commit* un : Labais peles klikšķis uz *Eclipse* projekta->*Team->Push to upstream*.

## Izstrāde

### Projekta uzbūve

Projekts sastāv no:

- *src* mapes, kurā glabājas testpiemēru kods
- *bin* mapes, kurā glabājas nokompilētie testpiemēri
- *lib* mapes, kas satur visas projektam nepieciešamās bibliotēkas
- Mapes *UserFiles* kur glabājas faili, kas nepieciešami testu izpildes gaitā

- Faila *testng.xml* kurā glabājas *TestNG* konfigurācija
- Faila *.gitignore*, kurā norādīti faili, kas netiek kopīgoti ar *GitHub* repozitoriju (šajā gadījumā fails *user.pass*, kas satur testa lietotāja paroli)
- Pēc lokālas testu palaišanas, *TestNG* projektā izveido mapi *test-output*, kurā var apskatīt pēdējā testa rezultātus
- Citi *Eclipse* palīgfaili, kas tiek izmantoti *Eclipse* uzstādījumu glabāšanai

Mapē *UserFiles* atrodas šādi faili:

- *document.edoc* šo failu testos izmanto, kad nepieciešams augšuplādēt parakstītu e-dokumentu.
- *email* fails, kurā rakstīts e-pasts, kuru testos izmanto kā lietotāja e-pastu.
- *objectmap.properties* fails, kurā tiek saglabāti visi izmantojamie *web elementi* un to atribūti
- *user.pass* fails, kurā glabājas testa lietotāja parole. Šis fails automātiski netiek sūtīts uz *ep119* repozitoriju iekš *GitHub*, tādēļ, lai palaistu testus lokāli, šis fails jāizveido pašiem. Faila saturs ir tikai parole. Arī *Jenkins* uz servera izveido šo failu, lai tam nebūtu nepieciešamības atrasties publiskajā repozitorijā.

## Testpiemēri

Testpiemēru organizācijai tiek izmantots *TestNG* ietvars. Viegļākai pārskatāmībai testpiemēri tiek veidoti katrs savā klasē (un tātad failā). Ir izveidota klase *Base* kas satur visas palīgfunkcijas, taču pati nesatur nevienu testpiemēru. Visiem testpiemēriem jābūt klases *Base* pēctečiem, jebšu *jāextendo* šī klase.

Piemērs testpiemēra klasei *VelviensPiemers* kura ir *Base* pēctecis.

```
package TestCases;

import org.testng.annotations.Test;

public class VelviensPiemers extends Base{

    @Test
    public void Kodols() throws Exception{
        throw new Exception("Neveiksme");
    }
}
```

Svarīgi atcerēties pirms metodes, kurā atrodas testpiemēra kods, pievienot anotāciju *@Test*, lai *TestNG* ietvars to atpazītu un palaistu testu izpildes laikā.

## Faila *testng.xml* konfigurēšana

Lai pievienotu jaunus testus svarīgi ir saprast faila *testng.xml* uzbūvi. *TestNG* ietvars atsevišķus testus (*test case*) grupē kopās (*suite*). Failā *testng.xml* visi testpiemēri tiek iekļauti `<suite>` `</suite>` tagos. Pēc tam katram atsevišķam piemēram ir savi `<test>` `</test>` tagi. `<test>` tagu saturs ir konkrētās klases un metodes, kuras iekļaujas testpiemērā. Kad izveidota jauna testpiemēru klase, svarīgi failā *testng.xml* izveidot jaunu testpiemēru, un tajā norādīt klasi, kā to, kura jāizpilda veicot šo testu.

Piemērs – faila *testng.xml* šī brīža saturs:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">

<suite name="Piemera testu kopa" parallel="none">

    <test name="EP219 Piemers">
        <classes>
            <class name="TestCases.EP219Piemers"/>
        </classes>
    </test>

    <test name="Velviens Piemers">
        <classes>
            <class name="TestCases.VelviensPiemers"/>
        </classes>
    </test>

</suite>
```

## Elementu meklēšana

Izstrādei tiek izmantota *Selenium* bibliotēka. *Selenium* meklē *web elementus* un veic darbības uz tiem, piemēram, `.click()`, `.sendKeys(String)`, `.getText()` utt. *Web elementi* ir pogas, izvēles rūtiņas (*checkbox*), teksta lauki, saites u.c.

Piemēram var skatīt failu *EP219Piemers.java* kurā ir pirmā testa paraugs.

Lai veiktu darbības ar kādu elementu, vispirms nepieciešams šo elementu pievienot failam *objectmap.properties* (Fails atrodas projekta mapē *UserFiles*). Šajā failā ierakstīti elementi šādā formātā:

Elementa\_nosaukums=atribūta\_tips:atribūta\_vērtība

Piemērs:

```
piekritu_CheckBox=id:ctl00_content_ctl00_uxAgreementCheckBox
nosaukums_Field=id:ctl00_content_ctl00_MyWizard_uxNameInput
```

Elementu atribūti tiek saglabāti failā, jo šādā veidā, ja tīmekļa pakalpojumā notiek kādas izmaiņas, vajadzības gadījumā, elementa atribūtus jāmaina tikai failā, nevis katrā vietā kodā, kur šis elements izmantots.

Kad elements pievienots sarakstam, to testpiemērā var atrast vienkārši izmantojot metodi

```
webElement("Elementa nosaukums").
```

Piemērs:

```
webElement("piekritu_CheckBox").click();  
webElement("nosaukums_Field").sendKeys("SIA");
```

Elementu atribūtus un to vērtības var atrast dažādi. Piemēram, interneta pārlūkā nospiežot F12 var atvērt rīku, kas ļauj atrast informāciju par lapas elementiem. Atribūtu *xpath* iespējams atrast izmantojot *Firefox* ar instalētiem spraudņiem *FireBug* un “FirePath”. Pieejami šādi atribūti, pēc kuriem var atrast elementus:

- *Id*
- *Name*
- *Classname*
- *Tagname*
- *Linktext*
- *partialLinkText*
- *cssSelector*
- *xpath*

Dažreiz var gadīties, ka nav pieejams neviens cits atribūts kā vienīgi *xpath*. Vienīgais veids, ka no tā izvairīties – pieprasīt izstrādātājiem izmantot viegli saprotamus *Id* vai *Name* atribūtus elementiem, ar kuriem strādā lietotājs. Šajā ziņā ep119 ir nepateicīgs, pie tam ir daudz automātiski ģenerētu elementu, tādēļ grūti atrast viegli saprotamas atribūtu vērtības.

Metode `webElement("Elementa nosaukums")` ir izveidota koda saīsināšanai. Var izsaukt arī elementus, kas nav pievienoti failam *objectmap.properties* izmantojot `driver.findElement(By)`.

`By` ir klase, kuru izmanto elementu atribūtu specificēšanai.

Piemērs:

```
driver.findElement(By.name("Nosaukums"));
```

Atradīs elementu, kura nosaukums ir *Nosaukums*.

## Citas metodes

Metode `driver.get("web lapas URL")` ļauj atvērt kādu konkrētu adresi.

Metode `Assert.assertEquals(actual, expected)` pārbauda, vai divas vērtības sakrīt. Gadījumā, ja tās nesakrīt, tiks izmests *exception* - tests tiks atzīmēts kā neveiksmīgs un pārtrauksies.

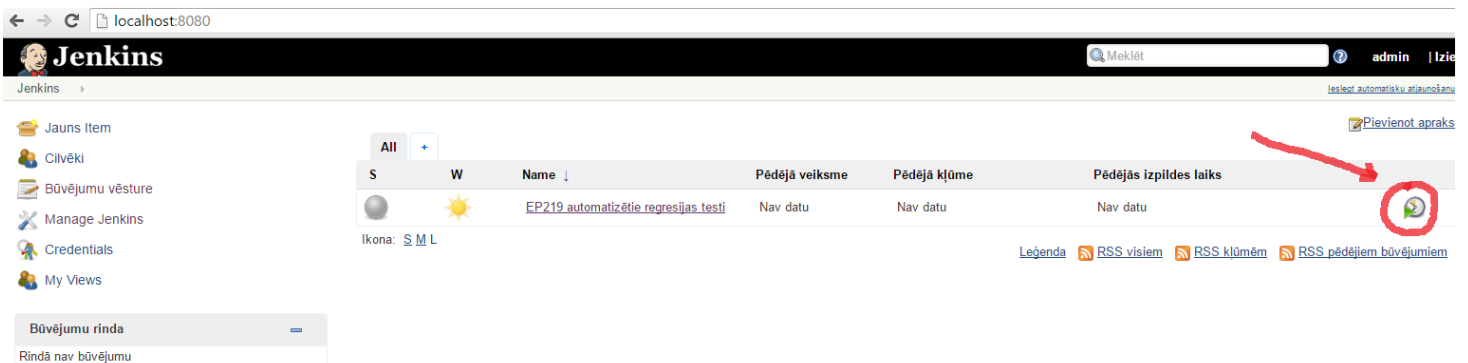
Klasē *TestCases.java* iekļautas palīgmetodes, lai atvieglotu testu izstrādi:

- Metode `LogIn()` - jāizsauc mirklī, kad portāls *latvija.lv* piedāvā izvēlēties autentificēšanās veidu. Šī metode autentificēsies sistēmā ar testa lietotāju.
- Metode `salidzinatMaksu(valstsNodeva, publikacija)` - jāizsauc mirklī, kad EP219 parāda nepieciešamo valsts nodevu un publikācijas maksu. Šī metode salīdzina, vai EP219 redzamā



samaksa atbilst gaidāmajai. Metode salīdzina tiešo tekstu, tāpēc argumentos jānorāda gaidāmās summu vērtības *string* formā.

## Testu palaišana

Testus palaiž caur *Jenkins* kontroles paneli un nospiežot pogu *Izplānot būvējumu*.



The screenshot shows the Jenkins web interface at localhost:8080. The top navigation bar includes the Jenkins logo, a search bar, and user information (admin). The left sidebar contains links for 'Jauns Item', 'Cilvēki', 'Būvējumu vēsture', 'Manage Jenkins', 'Credentials', and 'My Views'. The main content area displays a table of builds. The table has columns for 'S' (Status), 'W' (Workspace), 'Name', 'Pēdējā veiksmē' (Last Success), 'Pēdējā kļūme' (Last Failure), and 'Pēdējās izpildes laiks' (Last Build Time). The first row shows a build named 'EP219 automatizētie regresijas testi' with a status of 'S' and a workspace of 'W'. A red arrow points to the 'Izplānot būvējumu' button in the top right corner of the build list.

S	W	Name ↓	Pēdējā veiksmē	Pēdējā kļūme	Pēdējās izpildes laiks
		<a href="#">EP219 automatizētie regresijas testi</a>	Nav datu	Nav datu	Nav datu

Ikona: [S](#) [M](#) [L](#)

[Legenda](#) [RSS visiem](#) [RSS kļūmēm](#) [RSS pēdējiem būvējumiem](#)

Veiktiem testiem rezultāti tiks nosūtīti uz *Jenkins* konfigurācijā norādīto e-pastu. Gadījumā, ja kāds tests ir neveiksmīgs, e-pasta pielikumā būs attēls ar interneta pārlūka saturu brīdī, kad tests apstājies.

Lokāli testus var palaist arī ar *Eclipse*. Lai to izdarītu, jābūt instalētam *TestNG* spraudnim. Pēc tam – labais klikšķis uz faila *testng.xml* -> *Run As* -> *TestNG suite*.