

Software Project Management - Laboratory

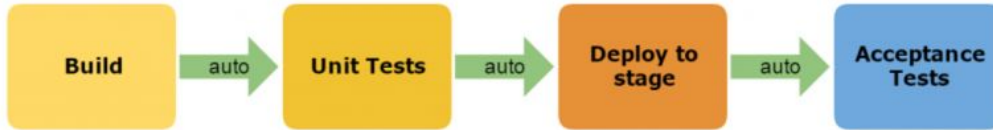
Lecture n° 15
A.Y. 2021-2022

Prof. Fabrizio Fornari
fabrizio.fornari@unicam.it

The Product Pipeline

© Ankesh K,
www.linuxnix.com

Continuous Integration



Continuous Delivery



Continuous Deployment



Environments

NOTE: Referred also as Development, Testing, Acceptance and Production (DTAP)

Development

Development and Unit testing for the developed feature are done on the individual developer's laptop or desktop system with a proper version control system in place.

For web based applications, at a minimum, it requires:

- The same web server used in production.
- The same database used in production.
- The same language being used in production.



Build/Test

The build/test server should automatically check out all the code, refresh the database and then execute tests.

All unit tests are run, then integration and regression testing are performed to make sure that all the pieces fit together and nothing previously working was broken.



Staging

The staging site is used to assemble, test and review new versions of a web app before it goes into production.

It is often used to present the client with the final project for them to perform **Acceptance testing**

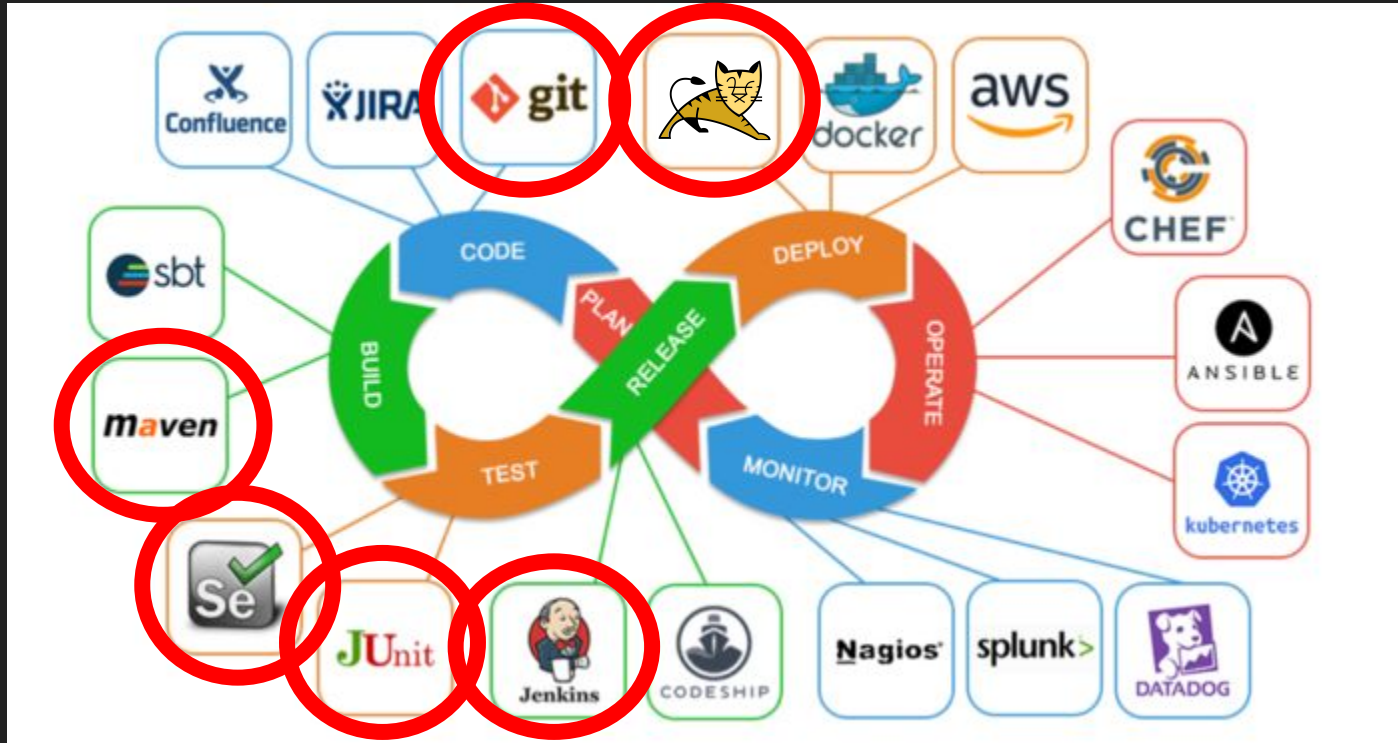


Production

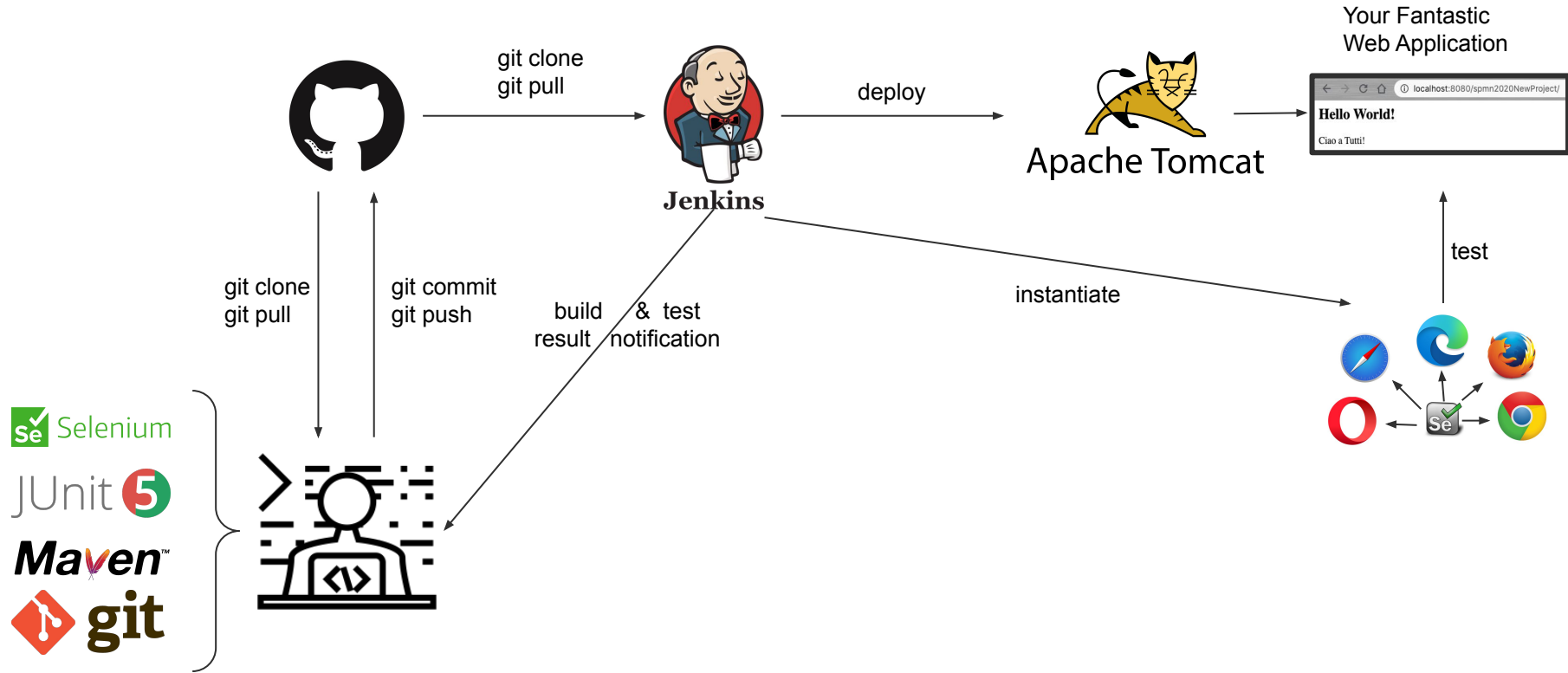
The accepted product, is deployed to a Production environment, making it available to all users of the system.



DevOps

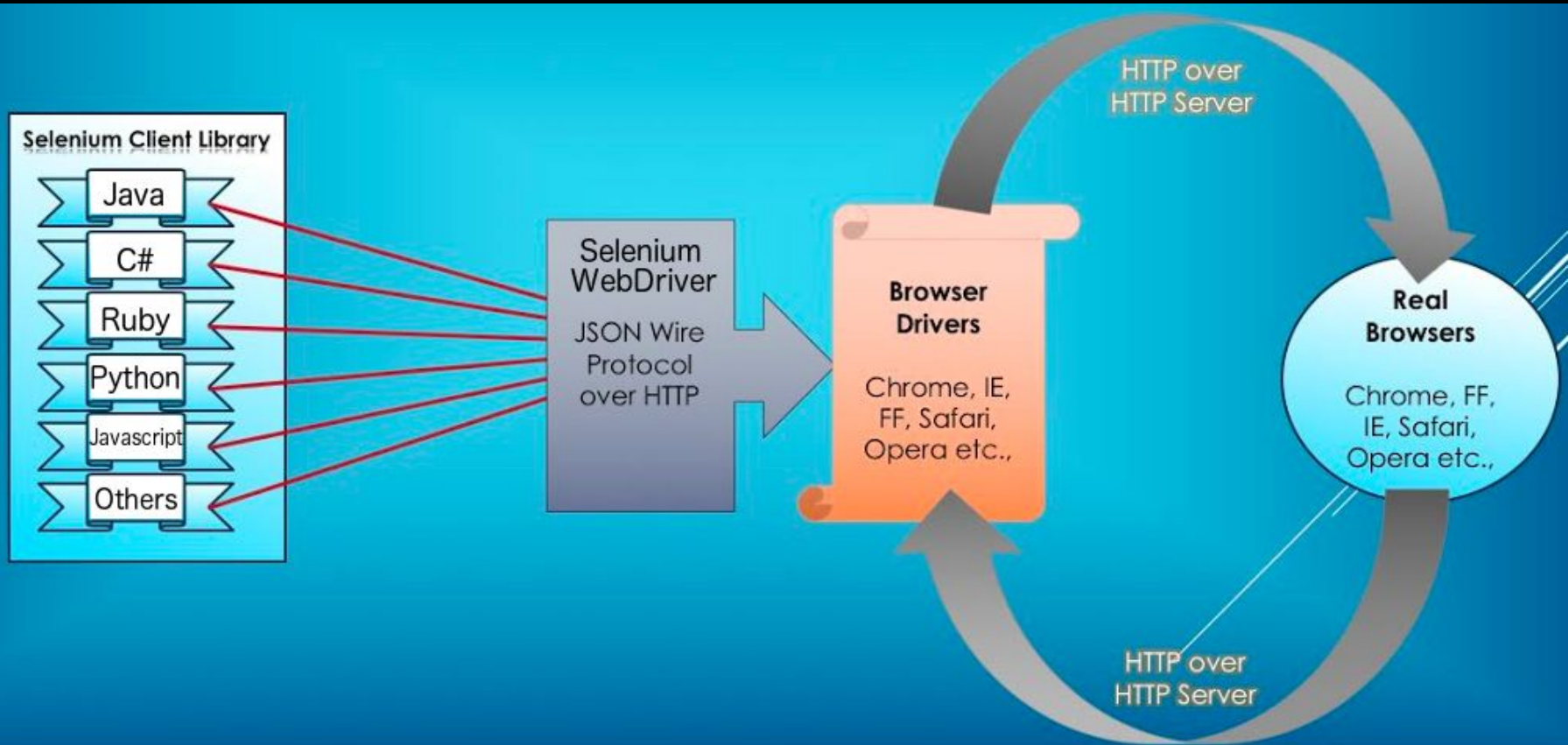


Our Toolchain





Selenium Architecture



Selenium

Something more to say about it...



WebDriver
Wait
for Page



Selenium

To make Selenium tests resilient, we need to make them wait for certain elements to load. Elements that we want to interact with. This is especially true with JavaScript heavy pages.

Implicit waits vs Explicit waits

And the standard advice from the Selenium Core Committers is to use explicit waits.

Note: Explore `AdvancedSeleniumTest.java`

Implicit Wait

An implicit wait requires setting a default amount of time for Selenium to wait if it can't perform an action immediately, and/or setting static sleeps.

Static sleeps:

```
Thread.sleep(ms);           // To avoid!
```

It forces your tests to wait a hard-coded amount of time to perform an action

Implicit sleeps:

```
driver.manage().timeouts().implicitlyWait(TimeOut, TimeUnit.SECONDS);
```

By implicitly waiting, WebDriver polls the DOM for a certain duration when trying to find any element. It means that if the element is not located on the web page within that time frame, it will throw an exception.

Explicit Waits

An explicit wait is code you define to **wait for a certain condition to occur** before proceeding further in the code.

The **condition** is called with a certain frequency until the timeout of the wait is elapsed. This means that for as long as the condition returns a falsy value, it will keep trying and waiting.

Since explicit waits allow you to wait for a condition to occur, they make a good fit for synchronising the state between the browser and its DOM, and your WebDriver script.

```
import org.openqa.selenium.support.ui.ExpectedConditions;
import org.openqa.selenium.support.ui.WebDriverWait;
...
WebDriver driver = new FirefoxDriver();
driver.get("http://somedomain/url_that_delays_loading");
...
WebDriverWait wait = (new WebDriverWait(driver, NumberOfSeconds));
wait.until(ExpectedConditions.presenceOfElementLocated(By.id("ElementId")));
```

Screenshot

```
public void takeSnapShot(WebDriver webdriver,String filePath){  
  
    TakesScreenshot scrShot =((TakesScreenshot)webdriver);  
  
    //Call getScreenshotAs method to create image file  
    File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);  
  
    //Move image file to new destination  
    File DestFile=new File(filePath);  
    Files.copy(SrcFile.toPath(), DestFile.toPath(),StandardCopyOption.REPLACE_EXISTING);  
  
}
```

How to check the status of HTTP Request?

We can use Rest Assured

REST Assured is a Java DSL for simplifying testing of REST based services built on top of HTTP Builder.

It supports POST, GET, PUT, DELETE, OPTIONS, PATCH and HEAD requests and can be used to validate and verify the response of these requests.



<https://rest-assured.io/>

<https://github.com/rest-assured/rest-assured/wiki/Usage>

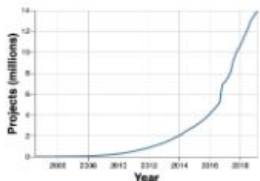
Rest Assured

MVNREPOSITORY

Search for groups, artifacts, categories

Search

Indexed Artifacts (18.4M)



Popular Categories

- Aspect Oriented
- Actor Frameworks
- Application Metrics
- Build Tools
- Bytecode Libraries
- Command Line Parsers
- Cache Implementations
- Cloud Computing
- Code Analyzers
- Collections
- Configuration Libraries
- Core Utilities
- Date and Time Utilities

Home » io.rest-assured

Group: REST Assured

Sort: **popular** | newest



1. REST Assured

1,079 usages

io.rest-assured » rest-assured

Apache

Java DSL for easy testing of REST services

Last Release on Nov 8, 2020



2. JSON Path

39 usages

io.rest-assured » json-path

Apache

JSON Path

Last Release on Nov 8, 2020



3. JSON Schema Validator

35 usages

io.rest-assured » json-schema-validator

Apache

JSON Schema Validator

Last Release on Nov 8, 2020

Home » io.rest-assured » rest-assured » 4.3.2



REST Assured » 4.3.2

Java DSL for easy testing of REST services

License	Apache 2.0
HomePage	http://code.google.com/p/rest-assured
Date	(Nov 08, 2020)
Files	bundle (683 KB) View All
Repositories	Central
Used By	1,079 artifacts

Maven [Gradle](#) [SBT](#) [Ivy](#) [Grape](#) [Leiningen](#) [Buildr](#)

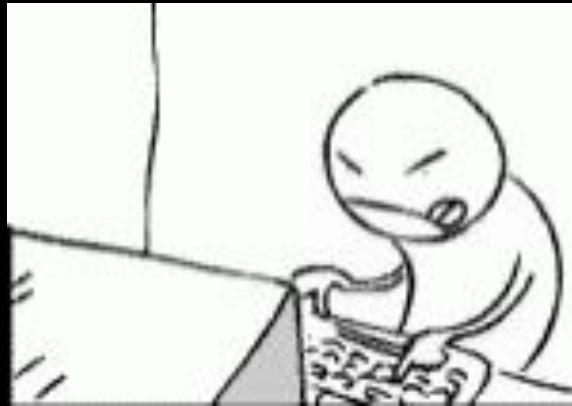
```
<!-- https://mvnrepository.com/artifact/io.rest-assured/rest-assured -->
<dependency>
  <groupId>io.rest-assured</groupId>
  <artifactId>rest-assured</artifactId>
  <version>4.3.2</version>
  <scope>test</scope>
</dependency>
```

Include comment with link to declaration

Copied to clipboard!

What about complex tests...?

Do we have to write them entirely from scratch?



Fortunately No!

Selenium IDE

Download it from:

<https://www.seleniumhq.org/selenium-ide/>

and let us see what we can do with it...

However we cannot export tests in a format that we can use for writing tests in our preferred programming language



Katalon Recorder

Katalon Automation Recorder it is an automation recorder that helps to export Selenium WebDriver code.

Download the extension for the browser you want to use



Katalon

<https://www.katalon.com/>

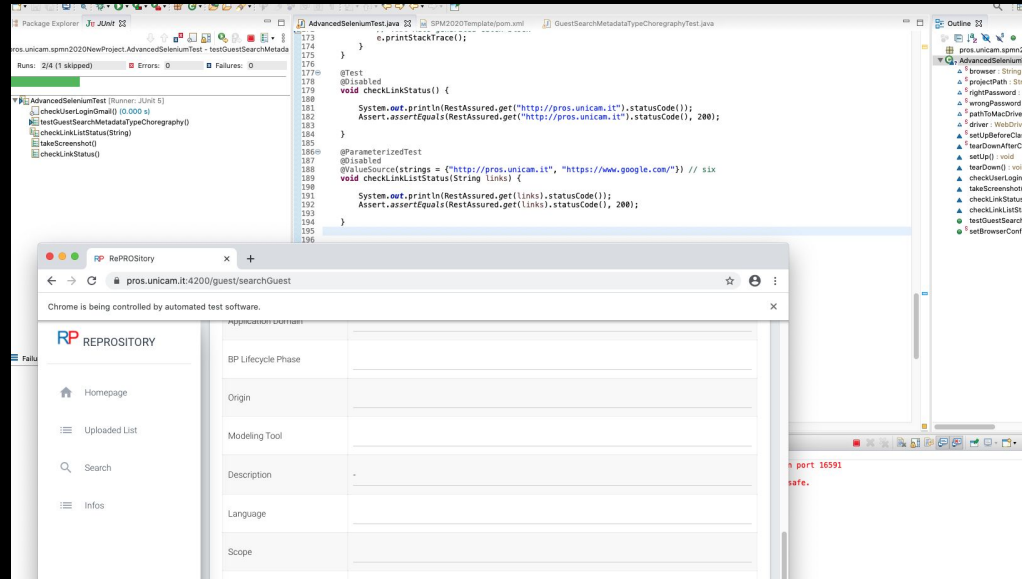
Try to record some tests

Try to find any difference

Do we really need a browser...?

Or better...do we really need a graphical interface?

Every time we run a test, an instance of a browser is created and the graphical user interface of the chosen browser appears...do we really need it?



Headless Browser...



Headless Browser...

- It is a browser without graphical interface
- What is it for?

Headless Browser...

It is a browser without graphical interface

Headless browsers are commonly used for:

- Website and application testing
- JavaScript library testing
- JavaScript simulation and interactions
- Running one or more automated UI tests in the background

Headless Browser...

In a headless testing environment, you can write and execute scripts to:

- Test basic and alternative flows
- Simulate clicks on links and buttons
- Automate form filling and submission
- Test SSL performance
- Experiment with various server loads
- Get reports on page response times
- Scrape useful website code
- Take screenshots of results

Testing these use cases provides you with a solid overview of how a site's UI performs and gives you essential information for making changes before deployment.

Which Headless Browser...?

Can you name one Headless Browser?

Which Headless Browser...?

- Firefox Headless Mode
- Headless Chrome
- PhantomJS
- Zombie JS
- HtmlUnit
- Splash

Headless Chrome



Headless Chrome

The biggest downside is that you need to be able to install Chrome. You don't need a UI, but installing software is not always possible.

Chrome Driver also requires an executable to be downloaded.

I keep the executable in the same directory as the project (or in a binary repository and copy it to the workspace.)

It still requires Chrome itself to be installed.



Html Unit



<https://htmlunit.sourceforge.io/>

Html Unit

In the past, Selenium came with a built in headless driver called HtmlUnitDriver.

While this driver is still supported, it is now a separate dependency and, unsurprisingly, uses the Html Unit framework.

Prior to Single Page Applications and largely AJAX based pages, this driver was an excellent choice. You have the ability to choose whether to run the page JavaScript, it runs in memory and is very fast. It's still a good choice for web pages with a good amount of HTML data on them.



HtmlUnit Driver

The screenshot displays the Maven Repository interface for the HtmlUnit Driver. The top navigation bar includes the Maven Repository logo and a search box. The main content area shows the breadcrumb path: Home > org.seleniumhq.selenium > htmlunit-driver. The page title is "HtmlUnit Driver" with a Selenium logo icon, and the description is "WebDriver compatible driver for HtmlUnit headless browser". Below this, there are sections for "Licer" (Maven Repository), "Tags" (2.4), and "Used" (18.4M). A "Popular Categories" sidebar lists various tool categories. The main content area also includes a "Popular Categories" section and a "Used By" section with a list of build tools: Maven, Gradle, SBT, Ivy, Grape, Leiningen, and Buildr. A code block shows the Maven dependency declaration for the driver, and a checkbox is checked for "Include comment with link to declaration".

Indexed Artifacts (18.4M)

Home > org.seleniumhq.selenium > htmlunit-driver

HtmlUnit Driver
WebDriver compatible driver for HtmlUnit headless browser

Licer **MVNREPOSITORY** Search for groups, artifacts, categories Search

Tags **2.4**

Used **Indexed Artifacts (18.4M)**

Home > org.seleniumhq.selenium > htmlunit-driver > 2.45.0

HtmlUnit Driver >> 2.45.0
WebDriver compatible driver for HtmlUnit headless browser

License	Apache 2.0
HomePage	https://github.com/SeleniumHQ/htmlunit-driver
Date	(Nov 13, 2020)
Files	jar (73 KB) View All
Repositories	Central
Used By	133 artifacts

Maven Gradle SBT Ivy Grape Leiningen Buildr

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/htmlunit-driver -->
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>htmlunit-driver</artifactId>
  <version>2.45.0</version>
</dependency>
```

Include comment with link to declaration



<https://htmlunit.sourceforge.io/gettingStarted.html>

Phantom JS

PhantomJS is a headless web browser scriptable with JavaScript.
It runs on Windows, macOS, Linux, and FreeBSD.

<https://phantomjs.org/>

<https://github.com/ariya/phantomjs/>

Project Suspended - <https://github.com/ariya/phantomjs/issues/15344>
<https://groups.google.com/g/phantomjs/c/9a15d-LDuNE?pli=1>



Running Acceptance Tests

1. Undeploy web application

Which issue do we had last time?

A possible solution is to exclude Acceptance tests from our first Jenkins Job and include them into a second Jenkins Job

One way is to assign them a Tag and specify the maven goals:

To skip the acceptance tests: `clean install -DexcludedGroups=AcceptanceTest surefire:test`

To run the acceptance tests: `test -Dgroups=AcceptanceTest`



Modify the First Job

The screenshot shows the Jenkins configuration interface for a job named 'spmProject2020'. The 'Build' tab is selected, and the 'Invoke top-level Maven targets' step is visible. The 'Maven Version' is set to 'Maven' and the 'Goals' are 'clean install -DexcludedGroups=AcceptanceTest surefire:test'. There is a red 'X' icon and a help icon in the top right of the step configuration area. An 'Advanced...' button is located at the bottom right of the step configuration. At the bottom left of the build step area, there is an 'Add build step' button.

Jenkins > spmProject2020 >

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions


Build


Invoke top-level Maven targets X ?

Maven Version

Goals

Create the Second Job

 Back to Dashboard

 Status

 Changes

 Workspace


 Build Now

 Configure

 Delete Project

 GitHub

 Rename

 Build History trend ^

find

Project spmProject2020AcceptanceTest

This job runs the AcceptanceTest with Selenium for the spm2020 project

 edit description

[Disable Project](#)

 Workspace

 Recent Changes

Upstream Projects

 spmProject2020

Permalinks

- Last build (#47), 4 days 5 hr ago
- Last stable build (#47), 4 days 5 hr ago
- Last successful build (#47), 4 days 5 hr ago
- Last failed build (#39), 5 days 18 hr ago
- Last unsuccessful build (#39), 5 days 18 hr ago
- Last completed build (#47), 4 days 5 hr ago

Configure the Second Job

Jenkins ▾ ▶ spmProject2020AcceptanceTest ▶

General Source Code Management Build Triggers Build Environment **Build** Post-build Actions

With Ant ?

Build

Invoke top-level Maven targets X ?

Maven Version ▾

Goals ▾

...and now?

Modify the First Job

Jenkins > spmProject2020 >

General Source Code Management Build Triggers Build Environment Build **Post-build Actions**

Build other projects X ?

Projects to build

Trigger only if build is stable

Trigger even if the build is unstable

Trigger even if the build fails

Add post-build action ▾

The image shows the Jenkins web interface for configuring a job. The breadcrumb navigation shows 'Jenkins > spmProject2020 >'. The 'Post-build Actions' tab is selected, and the 'Build other projects' section is expanded. A text input field contains 'spmProject2020AcceptanceTest'. Three radio buttons are visible, with the first one selected. A red 'X' icon and a blue question mark icon are in the top right corner of the section. At the bottom, there is a button labeled 'Add post-build action' with a dropdown arrow.

Downstream/Upstream

Project spmProject2020

This job is related to the spm2020 project



Downstream Projects

 spmProject2020AcceptanceTest

Project spmProject2020AcceptanceTest

This job runs the AcceptanceTest with Selenium for the spm2020 project



Upstream Projects

 spmProject2020