

# Software Project Management - Laboratory

Lecture n° 9  
A.Y. 2021-2022

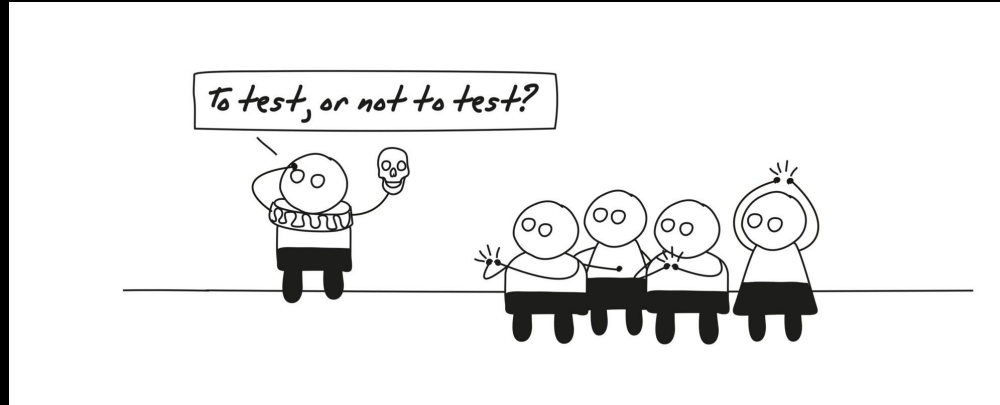
Prof. Fabrizio Fornari  
[fabrizio.fornari@unicam.it](mailto:fabrizio.fornari@unicam.it)

# Unit Testing - Recap

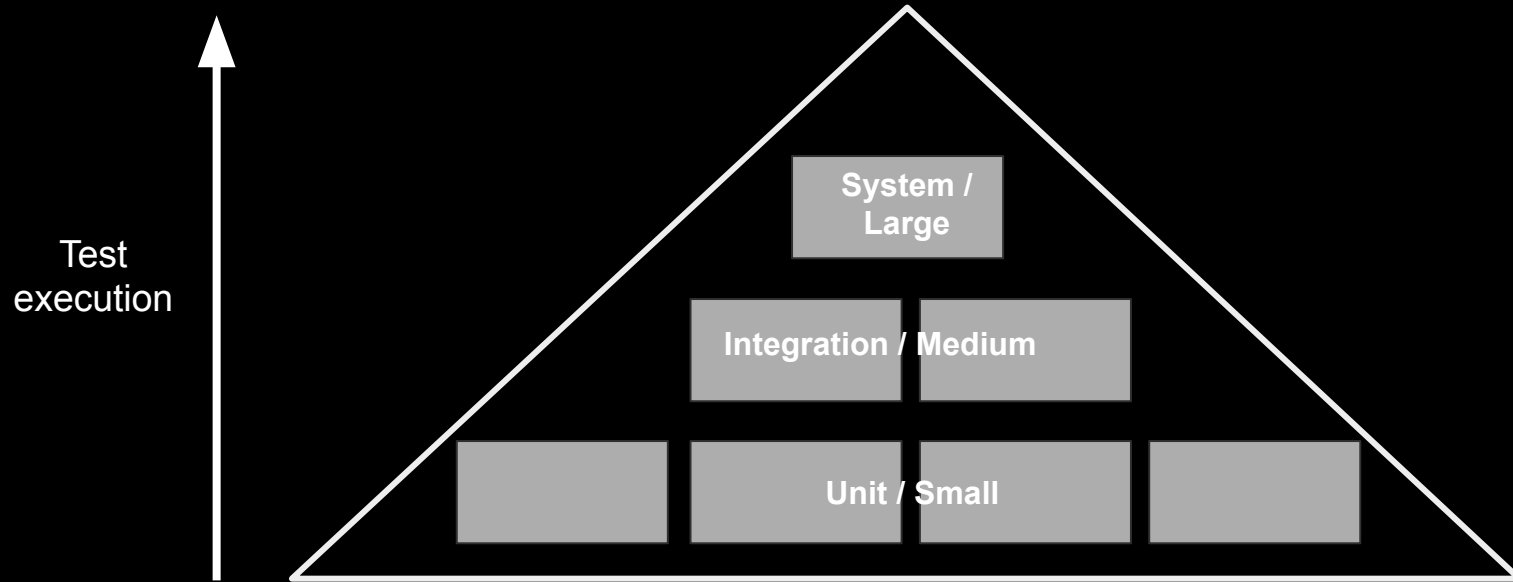
with JUnit

# Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently analysed for proper operation.



# The Test Stack



# Types of Testing

- Unit Testing
- Integration Testing
- Regression Testing
- ...



# Integration Testing

Individual modules are combined and tested as a group.  
Data transfer between the modules is tested as well.

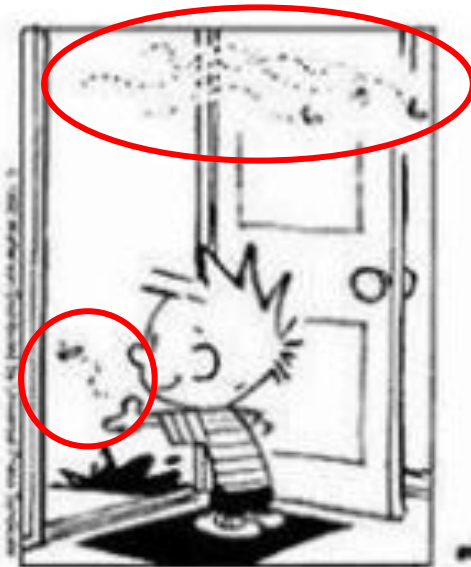


# Regression Testing

Regression means retesting the unchanged parts of the application.

# Regression:

"when you fix one bug, you introduce several newer bugs."





# Regression Testing

**Test cases are re-executed** in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.

This test can be **performed on a new build** when there is a significant change in the original functionality, even in correspondence of a single bug fix.

# Manual Testing

The oldest type of software testing.

It requires a tester to perform manual test operations on the test software without automation scripts.

The tester choose which tests to run, when to run them, and how many times.



# Manual Testing in Eclipse

The screenshot displays the Eclipse IDE interface during a manual test run. The Package Explorer on the left shows a test suite named 'HelloWorldTest' with several test methods, including 'testHelloShouldReturnAString()'. The main editor shows the source code of 'App.java' with a red circle around the '@Disabled' annotation on line 57. A context menu is open over the '@Disabled' annotation, listing various actions like 'Undo Typing', 'Copy', 'Run As', and 'Debug As'. The Console at the bottom shows the execution output of the test suite, including the message 'HelloWorldTest (1) [JUnit] /Library/Java/JavaVirtualMachines/...'. The console output shows that the test suite 'HelloWorldTest' was executed successfully, with all tests passing.

```
48
49 @AfterEach
50 void tearDown() throws Exception {
51     hW = null;
52     // Thread.sleep(1000);
53     LOGGER.info("@AfterEach - executes");
54 }
55
56 @Test
57 @Disabled
58 void testHelloShouldReturnAString() {
59     assertEquals("HelloWorld.hello()");
60 }
61
62 @Test
63 @Tag("display")
64 @DisplayName("Custom test name containing spaces")
65 void testWithDisplayNameContainingSpaces() {
66 }
67
68 @Test
69 @Tag("display")
70 @DisplayName(" ")
71 void testWithDisplayNameContainingSpaces() {
72 }
73
74 @Test
75 @Tag("display")
```

Package Explorer: HelloWorldTest [Runner: JUnit 5] (0.154 s)

- onlyOnWindowsOs() (0.000 s)
- testName() (0.000 s)
- Custom testNumberFizz (0.051 s)
- onlyOnLinuxOs() (0.004 s)
- testingTaxCalculation() (0.005 s)
- testHelloShouldReturnAString() (0.004 s)
- Custom test name containing spaces (0.016 s)
- onlyOnMacOs() (0.005 s)
- Custom testNumber (0.050 s)

Console:

```
<terminated> HelloWorldTest (1) [JUnit] /Library/Java/JavaVirtualMachines/...
Nov 10, 2020 11:34:48 AM pros.unicam.spmm2020Newl
INFO: @BeforeEach - executes before each test me
Nov 10, 2020 11:34:48 AM pros.unicam.spmm2020Newl
INFO: @AfterEach - executes after each test metho
Nov 10, 2020 11:34:48 AM pros.unicam.spmm2020Newl
INFO: @AfterAll - executes once after all test mi
```

# JUnit5

Unlike previous versions of JUnit, JUnit 5 is composed of several different modules from three different sub-projects.

**JUnit 5 = JUnit Platform + JUnit Jupiter + JUnit Vintage**

With the objective of separating "JUnit the tool" (which we use to write tests) and "JUnit the platform" (which tools use to run our tests) the JUnit team decided to split JUnit 5 into three sub-projects:

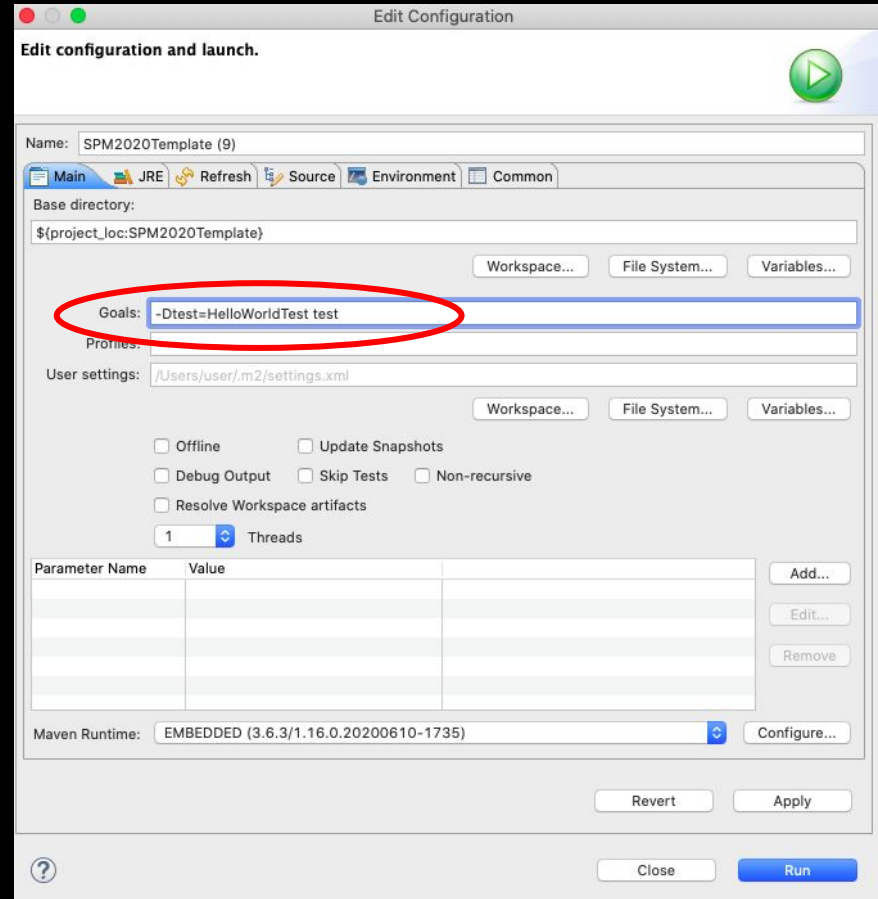
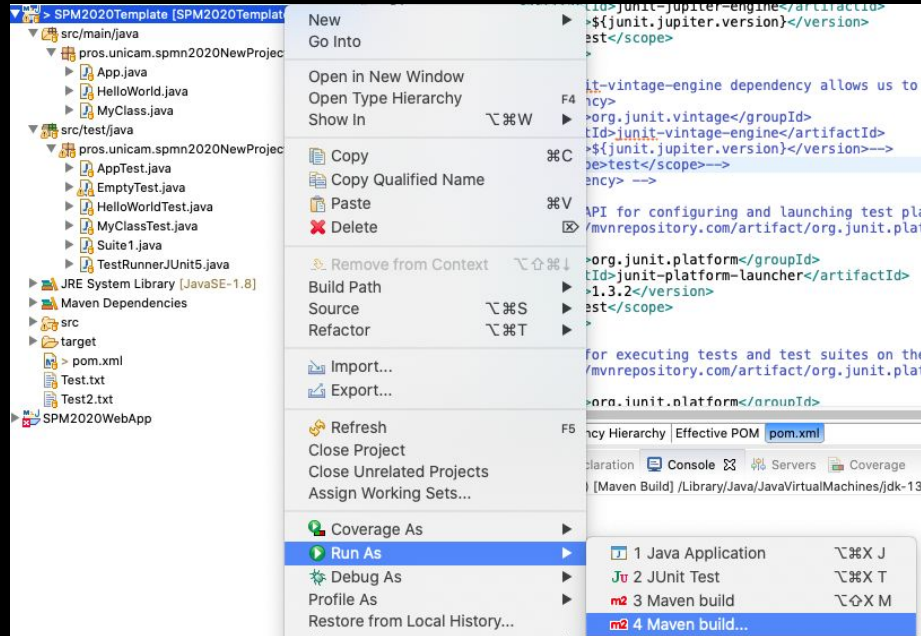
- **JUnit Platform:** Contains the engine API and provides a uniform API to tools, so they can run tests
- **JUnit Jupiter:** The API against which we write tests and the engine that understands it.
- **JUnit Vintage:** An engine that allows to run tests written in JUnit 3 and 4 with JUnit 5

# JUnit5

- The *junit-jupiter-api* dependency provides the public API that allows us to write tests and extensions which use JUnit 5.
- The *junit-jupiter-engine* dependency allows us to run tests which use JUnit 5.
- The Maven Surefire Plugin 2.22.1 provides native support for JUnit 5.
- If we want to use the native JUnit 5 support of the Maven Surefire Plugin, we must ensure that at least one test engine implementation is found from the classpath.

# Manual Testing with Maven

- Run a single test class:  
-Dtest=<NameOfTheTestClass> test



# Manual Testing with Maven

- Run a single test method from a test class:  
**-Dtest=<NameOfTheTestClass>#<NameOfTheTestMethod> test**

The screenshot displays an IDE environment with three main components:

- Project Explorer (Left):** Shows the project structure for 'SPM2020Template'. The 'src/test/java' directory contains a package 'pros.unicam.spmn2020NewProject' with files 'AppTest.java', 'EmptyTest.java', 'HelloWorldTest.java', 'MyClassTest.java', 'Suite1.java', and 'TestRunnerJUnit5.java'.
- Code Editor (Middle):** Shows the source code for 'HelloWorldTest.java'. The code includes:

```
41  
42 @BeforeEach  
43 void setUp() throws Exception {  
44     hw = new HelloWorld();  
45     Thread.sleep(1000);  
46     LOGGER.info("@BeforeEach - executes before each test");  
47 }  
48  
49 @AfterEach  
50 void tearDown() throws Exception {  
51     hw = null;  
52     Thread.sleep(1000);  
53     LOGGER.info("@AfterEach - executes after each test");  
54 }  
55  
56 @Test  
57 void testHelloShouldReturnAString() {  
58     assertNotNull(HelloWorld.hello());  
59 }
```
- Console (Bottom):** Shows the Maven build output for the test. The output includes:

```
<terminated> SPM2020Template (12) [Maven Build] /Library/Java/JavaVirtualMachines/...  
INFO: @AfterAll - executes once after all test methods in this suite  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.257 s  
[INFO] Results:  
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0  
[INFO] BUILD SUCCESS  
[INFO] Total time: 3.257 s  
[INFO] Finished at: 2020-11-10T11:55:02+01:00
```
- Edit configuration and launch (Right):** Shows the 'Goals' field in the Maven configuration dialog, where the goal '-Dtest=HelloWorldTest#testHelloShouldReturnAString test' is entered and highlighted with a red circle.

# Manual Testing with Maven

- Run a single test class:  
`-Dtest=<NameOfTheTestClass> test`
- Run a single test method from a test class:  
`-Dtest=<NameOfTheTestClass>#<NameOfTheTestMethod> test`
- Run multiple test classes:  
`-Dtest=<NameOfTheTestClass>,<NameOfTheTestClass> test`
- Run all test methods that match pattern 'testMethod1\*' from a test class:  
`-Dtest=TestApp1#testMethod1* test`
- Run all test methods match pattern 'testMethod1\*' and 'testMethod2\*' from a test class:  
`-Dtest=TestApp1#testMethod1*+testMethod2* test`
- Run all tests tagged with a specific tag eg. "display":  
`test -Dgroups=display`



# Automated Testing

# Automated testing

To automatically verify main functionality, ensure new version does not cause new defects, provide regression testing and help the teams to run a large number of tests in a short period of time.

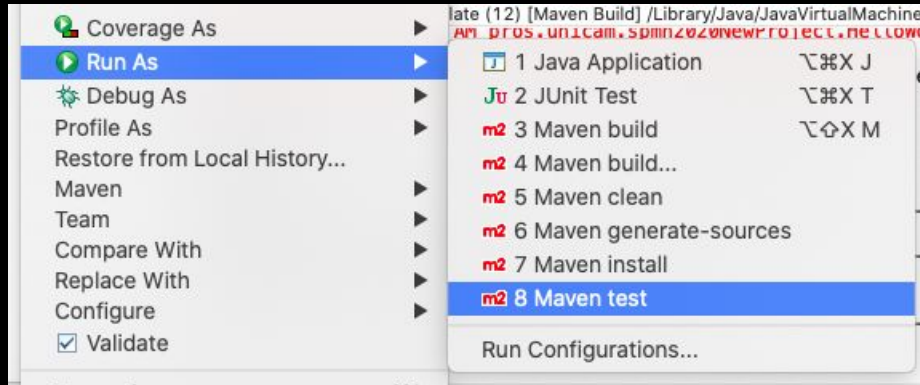
Companies having great number of projects are looking for specialists in the field of automated testing.

# Automated Testing with Maven

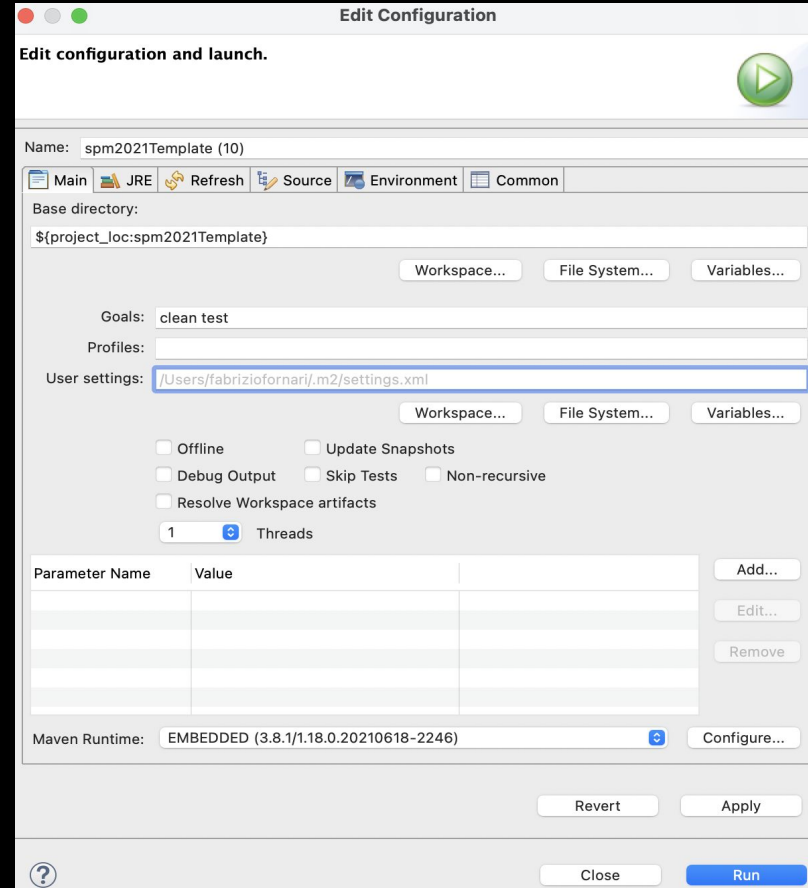
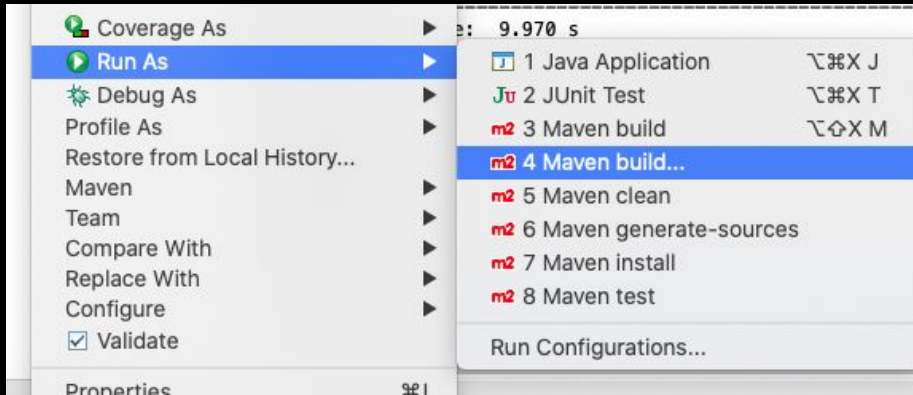
Use the "Test" suffix at the end of test classes names.

ClassName**Test** es. MyClassTest or HelloWorldTest...

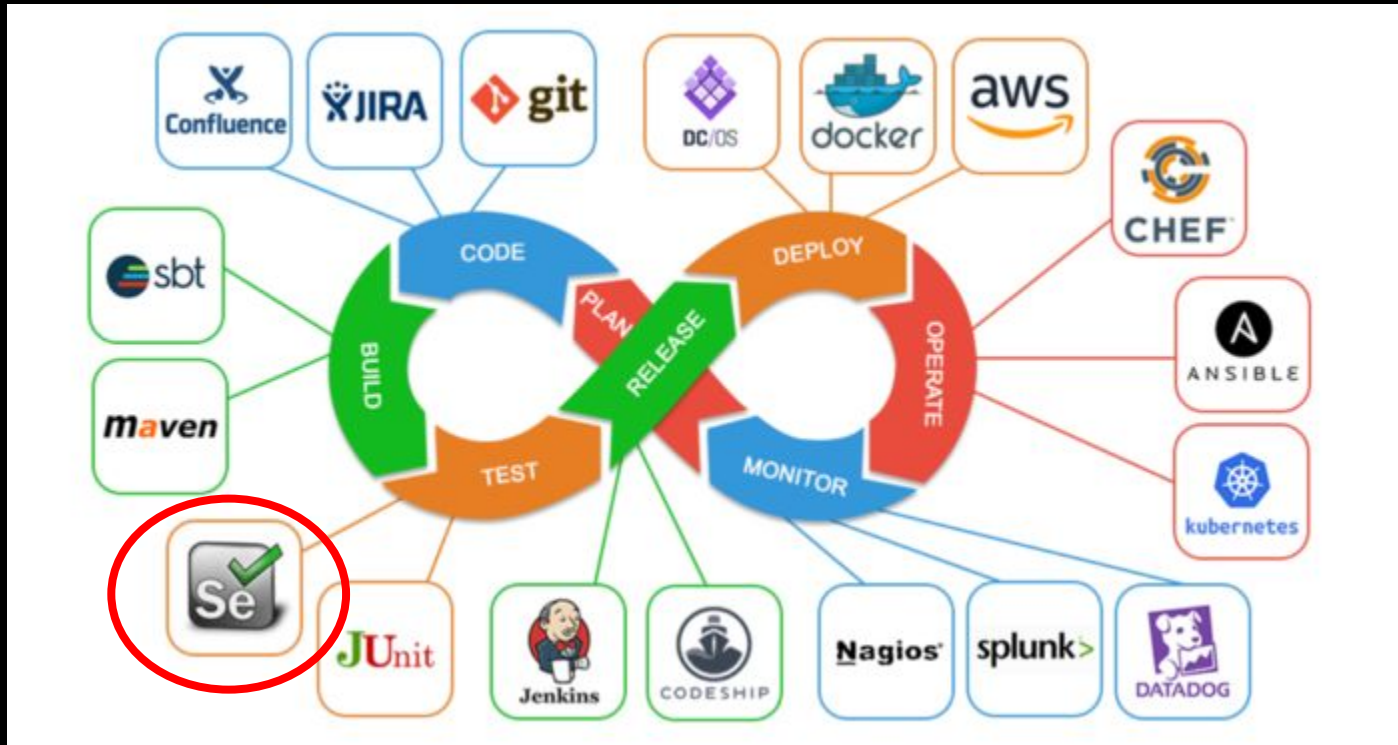
The Maven build system automatically includes such classes in its test scope.



# Automated Testing with Maven



# DevOps



# Automated web testing

with Selenium

# What is Web Testing?



# What is Selenium?





# Selenium

is a chemical element with symbol **Se** and atomic number **34**

group	1*	2											13	14	15	16	17	18			
1*	Ia	IIa											IIIb	IVb	Vb	VIb	VIIb	VIIIb			
1	Ia	IIa											IIIa	IVa	Va	VIa	VIIa	0			
1	H																	He			
2	Li	Be											B	C	N	O	F	Ne			
3	Na	Mg	IIIa**	IVa	Va	VIa	VIIa	VIIIa	IXa	Xa	XIa	XIIa	Al	Si	P	S	Cl	Ar			
			IIIb***	IVb	Vb	VIb	VIIb	VIIIb	IXb	Xb	XIb	XIIb									
4	K	Ca	Sc	Ti	V	Cr	Mn	Fe	Co	Ni	Cu	Zn	Ga	Ge	As	Se	Br	Kr			
5	Rb	Sr	Y	Zr	Nb	Mo	Tc	Ru	Rh	Pd	Ag	Cd	In	Sn	Sb	Te	I	Xe			
6	Cs	Ba	La	Hf	Ta	W	Re	Os	Ir	Pt							Bi	Po	At	Rn	
7	Fr	Ra	Ac	****	****	****	****	****	****	****	****	****									
			6	58	59	60	61	62	63	64							69	70	71		
			7	90	91	92	93	94	95	96	97	98	99	100	101	102	103				
				Th	Pa	U	Np	Pu	Am	Cm	Bk	Cf	Es	Fm	Md	No	Lr				

34  
**Se**

\* Numbering system recommended by the International Union of Pure and Applied Chemistry (IUPAC)  
 \*\* Previous IUPAC numbering system  
 \*\*\* Numbering system recommended by the Chemical Abstracts Service

# What is Selenium?

Selenium is a suite of tools **to automate web browsers** across many platforms.

We will use it **for automating tests of web applications**

Why is it called Selenium then?

# Why is it called Selenium then?

During the development of selenium core 2004 there was another competitive product developed by a company called **Mercury Interactive**.

A joke by Jason Huggins (co-creator of Selenium) saying that “*mercury poisoning can be cured by taking selenium supplements*”.

Selenium is used to remove the toxic content mercury from the human body, so Jason coined the term Selenium for their creative open-source project.

# Which is the idea?

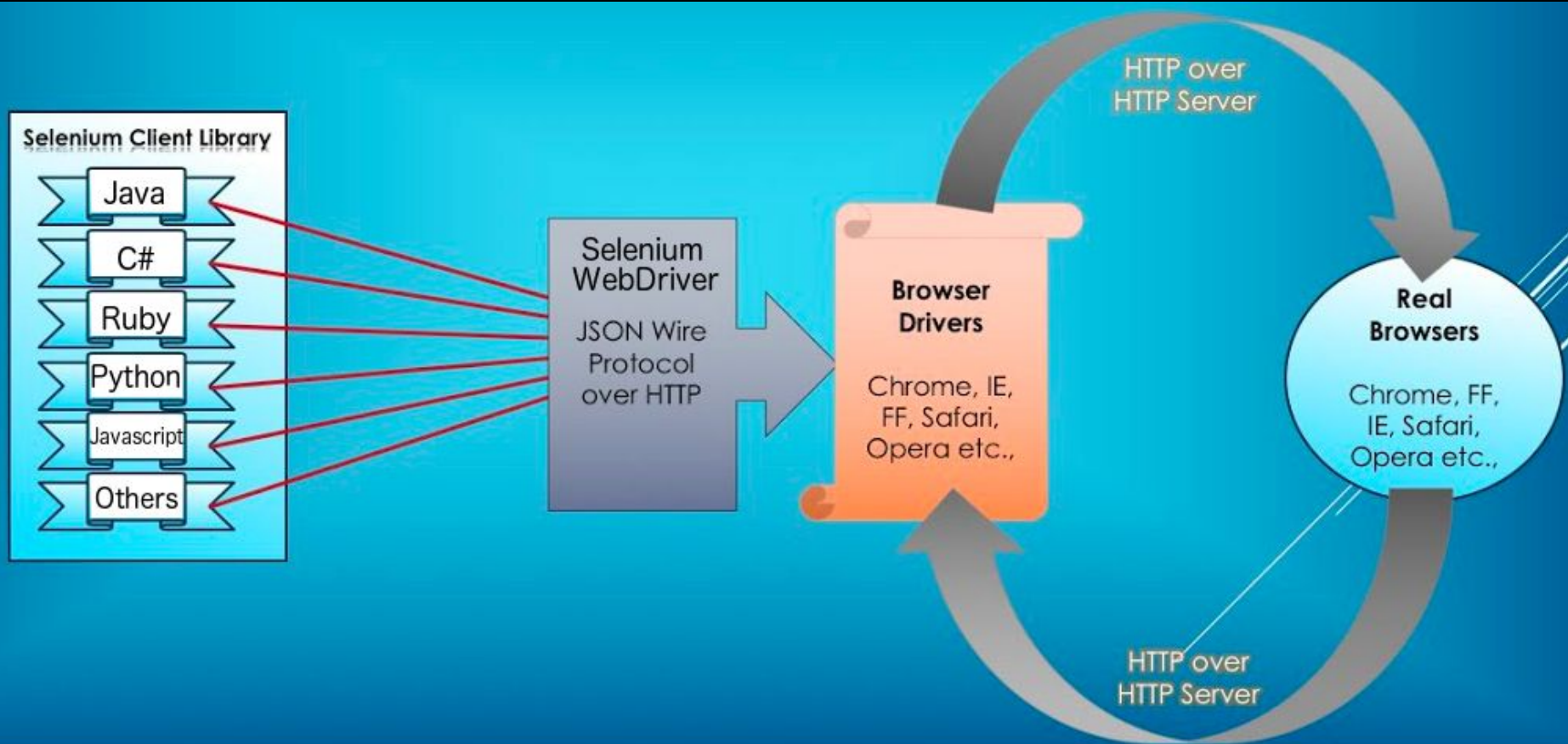
To **remotely control browsers** so that we can do things like write automated tests for the content they run or tests for the browser UI itself.

Should we write a test in a different way per each browser that is out there?

No, to this end, a group of people from several organizations is working on the WebDriver Specification.

<https://w3c.github.io/webdriver/>

# Selenium Architecture



# Selenium WebDriver

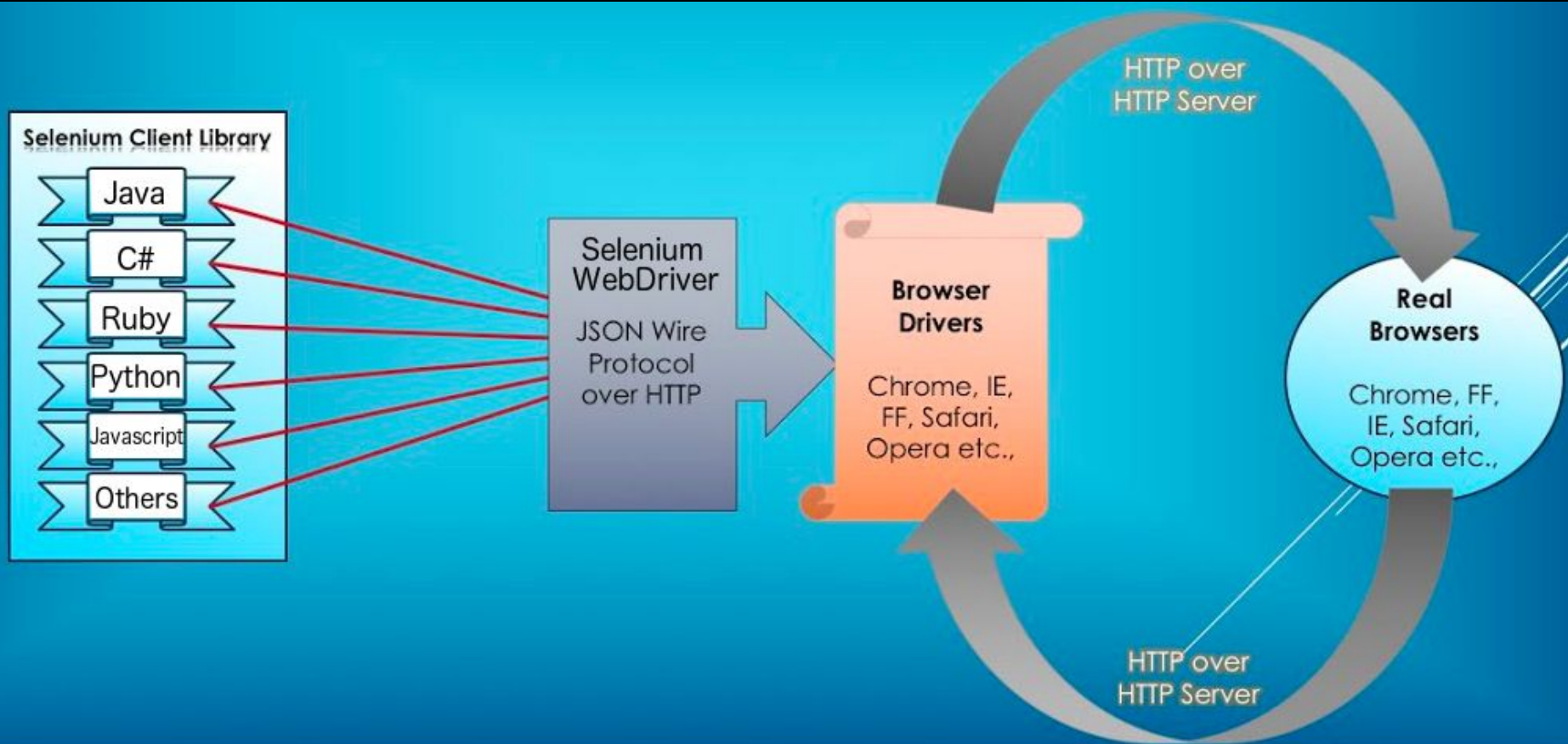
Selenium WebDriver provides APIs so that you can write code in your favourite language to simulate user actions like this:

```
client.get("http://pros.unicam.it/")  
link = client.find_element_by_id("participate")  
link.click()
```

Underneath that API, commands are transmitted via JSON over HTTP.

For example,  
to tell the browser to navigate to a url, a client sends a POST request to the endpoint */session/{session id of the browser instance you're talking to}/url* with body *{"url": "http://pros.unicam.it/"}*.

# Selenium Architecture





# Browser Driver

At the beginning Selenium had to develop drivers for some browser they wanted to interact with.

Then, browser vendors started implementing the Selenium JSON Wire Protocol themselves!

This makes a lot of sense: they're in the best position to maintain the server side and they can build the necessary behaviour directly into the browser.

It started with OperaDriver in 2009-2011, and then others followed such as ChromeDriver and Mozilla's geckodriver with Marionette. This is where the motivation for a WebDriver standard comes from.

# Selenium

GO TO: <https://www.seleniumhq.org/>

User Guide:

[https://www.selenium.dev/documentation/en/getting\\_started/](https://www.selenium.dev/documentation/en/getting_started/)

# Common Steps to follow

Step 0 - Open Eclipse

Step 1 - Create a new Maven Project or  
use the one you have created during previous lessons

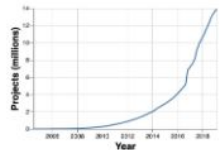
Step 2 - Add Selenium Maven Dependency

Step 3 - Download Third Party Browser Drivers

# Add Selenium Maven Dependency

Selenium-java  
Selenium-api  
Selenium-server  
Selenium-firefox-driver  
Selenium-chrome-driver

Indexed Artifacts (18.3M)



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
- Dependency Injection
- Embedded SQL Databases
- HTML Parsers
- HTTP Clients
- I/O Utilities
- JDBC Extensions
- JDBC Pools
- JPA Implementations
- JSON Libraries
- JVM Languages

Home » org.seleniumhq » selenium

## Group: Seleniumhq Selenium

Sort: **popular** | newest



### 1. Selenium Java

org.seleniumhq.selenium » selenium-java

1,275 usages

Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Last Release on May 29, 2020



### 2. Selenium API

org.seleniumhq.selenium » selenium-api

339 usages

Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Last Release on May 29, 2020



### 3. Selenium Server

org.seleniumhq.selenium » selenium-server

296 usages

Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Last Release on Jul 2, 2019



### 4. Selenium Support

org.seleniumhq.selenium » selenium-support

276 usages

Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Last Release on May 29, 2020



### 5. Selenium Firefox Driver

org.seleniumhq.selenium » selenium-firefox-driver

273 usages

Apache

Selenium automates browsers. That's it! What you do with that power is entirely up to you.

Last Release on May 29, 2020

# Download Third Party Browser Drivers

## Third party drivers and plugins

Selenium can be extended through the use of plugins. Here are a number of plugins created and maintained by third parties. For more information on how to create your own plugin or have it listed, consult the docs.

Please note that these plugins are not supported, maintained, hosted, or endorsed by the Selenium project. In addition, be advised that the plugins listed below are not necessarily licensed under the Apache License v.2.0. Some of the plugins are available under another free and open source software license; others are only available under a proprietary license. Any questions about plugins and their license of distribution need to be raised with their respective developer(s).

Browser	Latest	Change log	Issue Tracker
<a href="#">Mozilla GeckoDriver</a>	<a href="#">latest</a>	<a href="#">change log</a>	<a href="#">issue tracker</a>
<a href="#">Google Chrome Driver</a>	<a href="#">latest</a>	<a href="#">change log</a>	<a href="#">issue tracker</a>
<a href="#">Opera</a>	<a href="#">latest</a>	-	<a href="#">issue tracker</a>
<a href="#">Microsoft Edge Driver</a>	<a href="#">latest</a>	-	<a href="#">issue tracker</a>
<a href="#">SafariDriver</a>	Built in	-	<a href="#">issue tracker</a>

# I Chose Google Chrome Driver

## ChromeDriver

WebDriver is an open source tool for automated testing of webapps across many browsers. It provides capabilities for navigating to web pages, user input, JavaScript execution, and more. ChromeDriver is a standalone server that implements the [W3C WebDriver standard](#). ChromeDriver is available for Chrome on Android and Chrome on Desktop (Mac, Linux, Windows and ChromeOS).

You can view the current implementation status of the WebDriver standard [here](#).

### All versions available in **Downloads**

- Latest stable release: [ChromeDriver 86.0.4240.22](#)
- Latest beta release: [ChromeDriver 87.0.4280.20](#)

My version of Chrome  
86.0.4240.111

# Selenium Locators



Preferred selector order : id > name > css > xpath

# HTML

- Html is a standard markup language for creating Web pages. Html elements are the building blocks of HTML pages. HTML tags label pieces of content, such as “heading”, “paragraph”, “table”, and so on.
- HTML elements usually consists of start tag and end tag with content inserted between them.
- For example:
  - `<h1>` An example for an HTML title `</h1>`
  - `<p>` An example for an HTML paragraph `</p>`



# CSS

- CSS is a language that describes the style of an HTML document.
- CSS describes how HTML elements should be displayed.
- A CSS rule-set consists of a selector and a declaration block. For example: selector - h1, declaration – {color:blue;}.

```
<style>

body {
  background-color: lightblue;
}

</style>
```

# CSS

- CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute, and more.
- Some Selectors:
  - Element selector : The element selector selects elements based on the element name. for example: p, h1 etc.
  - ID selector: The id selector uses the id attribute of an HTML element to select a specific element. For example: #id1, #para2 etc.
- A CSS rule-set consists of a selector and a declaration block. For example: selector - h1, declaration – {color:blue;}.

# XPath

- XPath stands for XML Path Language, it can be used to navigate through elements and attributes in an XML document.
- XPath uses path expressions to select nodes or node-sets in an XML document

## Syntax:

- // - Selects nodes in the document from the current node that match the selection no matter where they are.
- @ - Selects attributes

# XPath

- Example

```
<!DOCTYPE html>
<html>
<head>
<title> Page Title</title>
</head>
<body>
<h1 id="h1_id"> This is part of the presentation title</h1>
<p>This is the paragraph.</p>
</body>
</html>
```

The query: `//h1[@id='h1_id']` will get the h1 element

# Selenium WebDriver

- A Selenium Web driver must be created
- For using Chrome:

```
System.setProperty("webdriver.chrome.driver",projectPath+"/drivers/chromedriver");*  
WebDriver driver = new ChromeDriver();
```

- Interaction with the Chrome instance will be made in the code on the driver.

**\*Note:** you need to specify, before instantiating the `WebDriver`, the path to the actual driver that you downloaded following instructions from the selenium website <https://www.seleniumhq.org/download/>.

# Selenium WebDriver

- Navigation using a Selenium WebDriver is very simple, given a defined URL. It can be done in two ways, `driver.get(...)` or `driver.navigate().to(...)`
  - `driver.get("https://www.google.com/");`
  - `driver.navigate().to("https://www.google.com/");`
- *The `driver.get(...)` and `driver.navigate().to(...)` do exactly the same thing. `driver.navigate()` supports also `driver.navigate().forward()` and `driver.navigate().backward()`*

# Finding Web Elements

- Web elements can be defined as each opening and closing tags in the web page. For example: `<button>Click Me</button>` - a web element.
- Finding a web element and interacting with it can be done in several ways: - ID. - Class. - Name. - Xpath. - Css Selector, etc.

# Finding Web Elements

- An example:
  - Assuming that we have the following web page:

```
<html>  
  <body>  
    <button id= "my_button"> Click Me</button>  
  </body>  
</html>
```
  - The following lines of code will be used for clicking the button:  
*WebElement button = driver.findElement(By.id(" my\_button "));  
button.click();*



# Selenium Example

Pull the <https://github.com/FabrizioFornari/SPM2021Template> repository for the updates related to Selenium.

Run the Test Class SeleniumTest.java

Complete the test checkProsSiteSearch to test if the search functionality on the pros.unicam.it website returns what expected. We expect to search for “bprove” and to have only results that include in the title the “bprove” term.

# Selenium Example

Complete the test checkProsSiteSearch to test if the search functionality on the pros.unicam.it website returns what expected. We expect to search for “bprove” and to have only results that include in the title the “bprove” term.

