

Functional and Security Testing Course

2023-2024

Introduction – lesson 1

Filippo Ricca
DIBRIS, University of Genova

Whoami?

- **Filippo Ricca**, associate professor
 - DIBRIS, University of Genova
 - Previously FBK, Trento (Italy)
- Research interests:
 - **Software engineering**
 - **Software testing**
 - Web/Mobile apps
 - IoT systems
- Teacher of SE and SSGS
 - Bachelor program
- Use of Selenium since 2005/2006
 - Selenium = testing framework
- Participation in industrial projects (also using Selenium)
- “Hands-On Selenium WebDriver with Java” Book revision
 - Prof. Boni García



WHOAMI

Teachers

- **Filippo Ricca**
 - Associate Prof. @ DIBRIS, University of Genova
 - Teacher of SEand SSGS
 - Bachelor program
 - Research interest: Software Testing
- **Enrico Cambiaso**
 - Technologist @ IEIIT institute of Consiglio Nazionale delle Ricerche (CNR)
 - Expert in Cyber-security and network security



Content of the Course and Practical Info



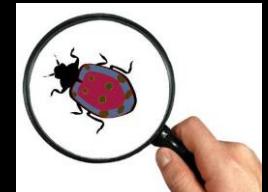
Course Overview

Students will learn the fundamentals in **functional** and **security testing** of software systems, with special emphasis on challenges posed by **Web and Mobile applications** and using **automated testing tools**

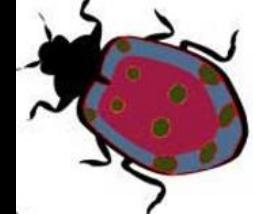
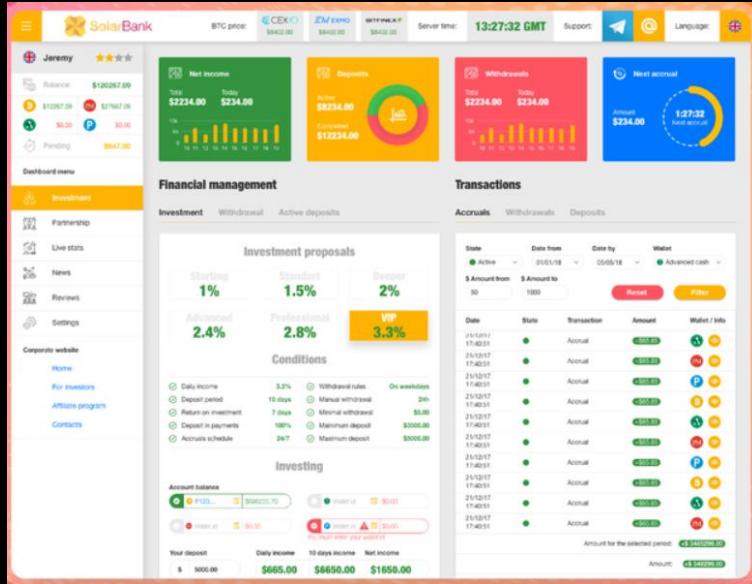


Functional vs. Security testing

- Functional and Security testing have **two different goals**:
 - **Functional testing** points to detect software defects (i.e., unexpected behaviors w.r.t requirements or user's needs)
 - **Security testing** checks whether the application or the product is secured or not. It checks if the application is vulnerable to attacks



Online Banking Web Application



Now, let's suppose the web app has a bug in the make investments functionality. This bug could cause very serious economic damages to the bank and its customers

Let's consider an online Banking Web application where it is possible move funds to other accounts, get mortgages and make investments (e.g., buy bank stock)

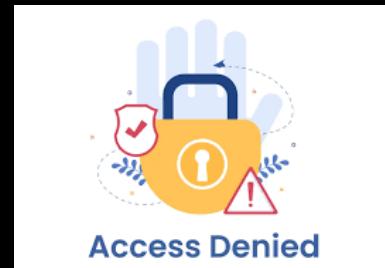


Now let's suppose that the bug is fixed but in the web application is also present a security problem in which the password of a client can be stolen. An event of this type could have catastrophic implications for the client. She could lose all her money in seconds!

97% of global banks are susceptible to web and mobile attacks

A whopping 97 out of 100 banks worldwide are vulnerable to mobile and web attacks, leaving them open to infiltration from hackers, a study has found.

The study, carried out by the security and automation testing website, ImmuniWeb, found that of 100 of the world's largest banks (according to S&P global list for 2019), 97 of them are not cyber-secure





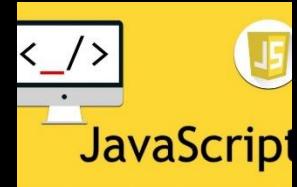
Course Goals

- After the completion of the course, the participants would be able to:
 1. Understand **the differences** between functionality and security testing
 2. Understand fundamental concepts of software testing
 1. e.g., **manual vs automated testing**
 3. Use **techniques/tools** for designing and executing functional tests
 4. Learn how attackers succeed in **breaking applications**
 5. Understand **the attack target possibilities** of web apps (e.g., SQL injection)
 6. Understand the 'Top Ten' vulnerabilities proposed by **OWASP**
 1. Open Worldwide Application Security Project (OWASP)
 7. **Get hands on Web and Mobile application testing techniques** (both functional and security), using, e.g. Selenium framework and other automated testing tools
 1. The course is practical!
 8. Incorporate software testing as a **continuous process**

Prerequisites

1. Object Oriented and procedural programming fundamentals

- Basic knowledge of
 - Java, JavaScript, PHP and SQL languages



2. Basic knowledge of Web and Mobile applications

1. A little of HTML, JavaScript and PHP





Contents (only Functional part)

- **Basics of Web applications**
- **Functional testing**
 - Manual vs automated testing
 - Software testing fundamentals (approaches and techniques)
 - Unit testing tools (e.g., JUnit/TestNG)
 - Automated testing approaches and tools for Web/Mobile apps
 - Ex. **Selenium WebDriver** (used also in the Security testing part)
- **Continuous Integration** (a little)
 - Tool: **Jenkins** and/or **GitHub actions**
- **API Testing** (if there is time)
 - Tool: **Postman**
- **Mobile Testing** (if there is time)
 - Testing framework: **Appium**



Get hands on Web (and Mobile application) testing techniques! A practical course with a lot of labs exercises!

Example of Lab task

Laboratory



- **Web application testing** (design and implementation of test scripts)

Simple Web app for Veterinarians



Using Selenium WebDriver

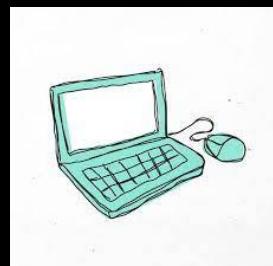
| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
|-----|------------------|-------------------|---------------|------------------|-------------------|-----|
| W5 | 28 | 29 | 30 | 31 | 1 | 2 |
| | 9a Cubone | 8:30a Houndour | 7:30a Starmie | 8a Bulbasaur | 8a Murkrow | |
| | 9:15a Smogon | 10:15a Forretress | 8:30a Staryu | 8:45a Starmie | 8:45a Charizard | |
| | 9:30a Goldeen | 10:15a Forretress | 8:30a Crobat | 10:30a Bulbasaur | 9:45a Houndour | |
| | 10:30a Crobat | 11:15a Murkrow | 11:30a Horsea | 11a Kingler | 11a Staryu | |
| | 11:15a Staryu | 12p Rowlet | 12p Steelix | 12p Starmie | 11a Dewgong | |
| | 2:15p Kingler | 1:30p Rowlet | 12p Bulbasaur | 12p Drowzee | 11:45a Forretress | |
| | 2:30p Rattata | 1:45p Steelix | 1:45p Psyduck | 12:30p Cubone | 12:15p Cubone | |
| | 2:45p Staryu | 3:15p Goldeen | 2p Houndour | 1:15p Staryu | 2p Starmie | |
| | 3:30p Rowlet | 3:15p Houndour | 3p Goldeen | 1:15p Forretress | 3p Crobat | |
| | 3:45p Forretress | 3:45p Sandshrew | 3:15p Crobat | 3:45p Pikachu | 3:30p Psyduck | |
| W6 | 4 | 5 | 6 | 7 | 8 | 9 |
| | | | | | | 10 |

<https://demo.jmix.io/petclinic/#main>

<https://github.com/jmix-framework/jmix-petclinic-online-demo>

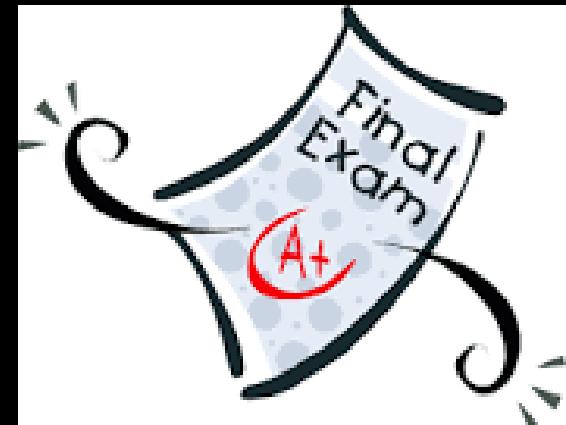
Practical Info

- **Six credits**, approximately **48 hours** (24 lessons), SOFTWARE SECURITY & ENGINEERING curriculum, 2° semester
- **Aulaweb** (e-learning system) “**mandatory**”
 - <https://2023.aulaweb.unige.it/course/view.php?id=6650>
- **Participation in the course “is assumed”**
 - Lessons will take place in person, I will provide ‘old podcasts’ and video-tutorial
- Not only lessons but **above all lab sessions!**
 - It would be convenient to work on your laptop. Is anyone without it?



Exam

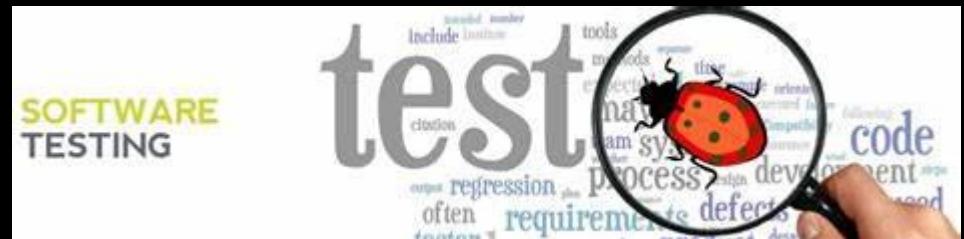
- The exam (oral or written) will primarily **focus on** the **mandatory assignments** completed throughout the year, but there may also be questions related to the topics covered in the course
- **It is important to do and complete assignments!**



Questions?



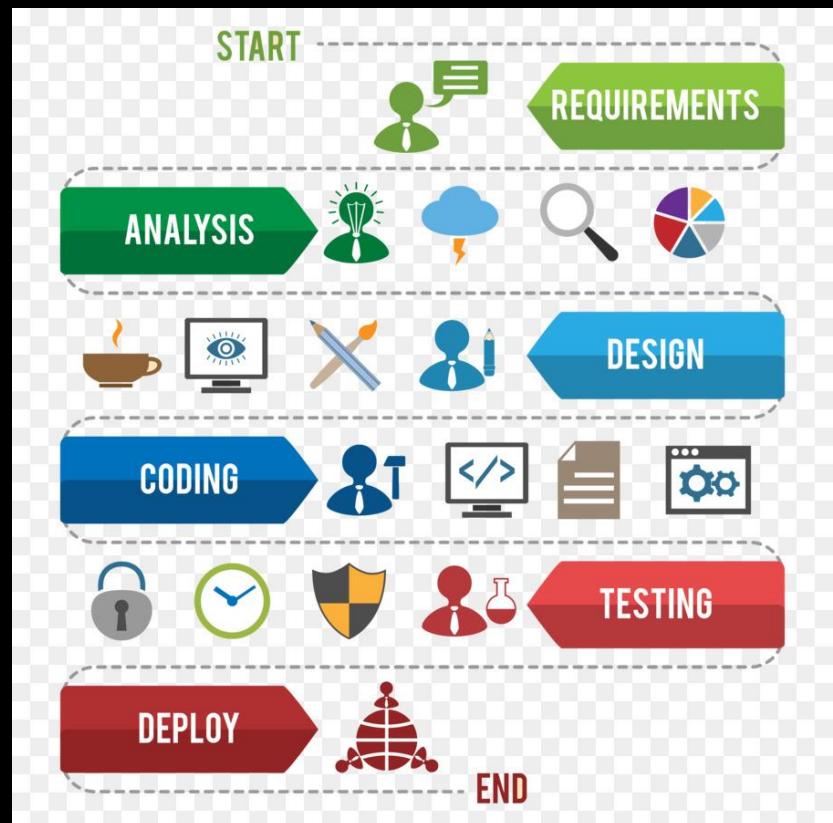
Introduction to important concepts



Software engineering

GOAL: Producing ‘good quality’ software systems reducing Costs and Time!

Development process

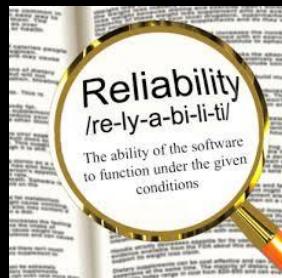


SE give a lot of importance to:

To build a system/app of good quality different factors must be considered

'External' Software quality factors

Works for 24x7x365
days a year



Correctness

Simple to use

Reliability

Usability

Software
Quality
Factors

Efficiency

Interoperability

Security

Not waste memory
and CPU time

Able to work with
other products or systems



The ability of software to interact with
one or more specified components
or systems.



Each software factor can be checked applying a specific type of testing

Functional vs non-Functional testing

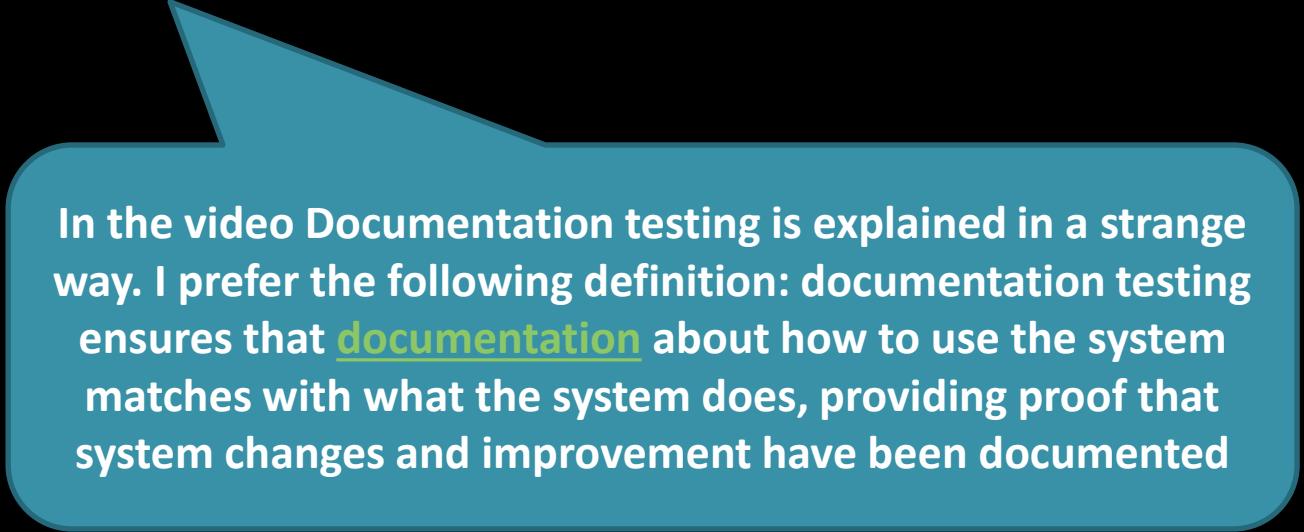


NON-FUNCTIONAL TESTING check non-functional aspects (efficiency, usability, reliability, security, etc) of a software application

Difference between Functional and Non-Functional Testing

(also levels of software testing and introduction of test case)

https://www.youtube.com/watch?v=j_79AXkG4PY



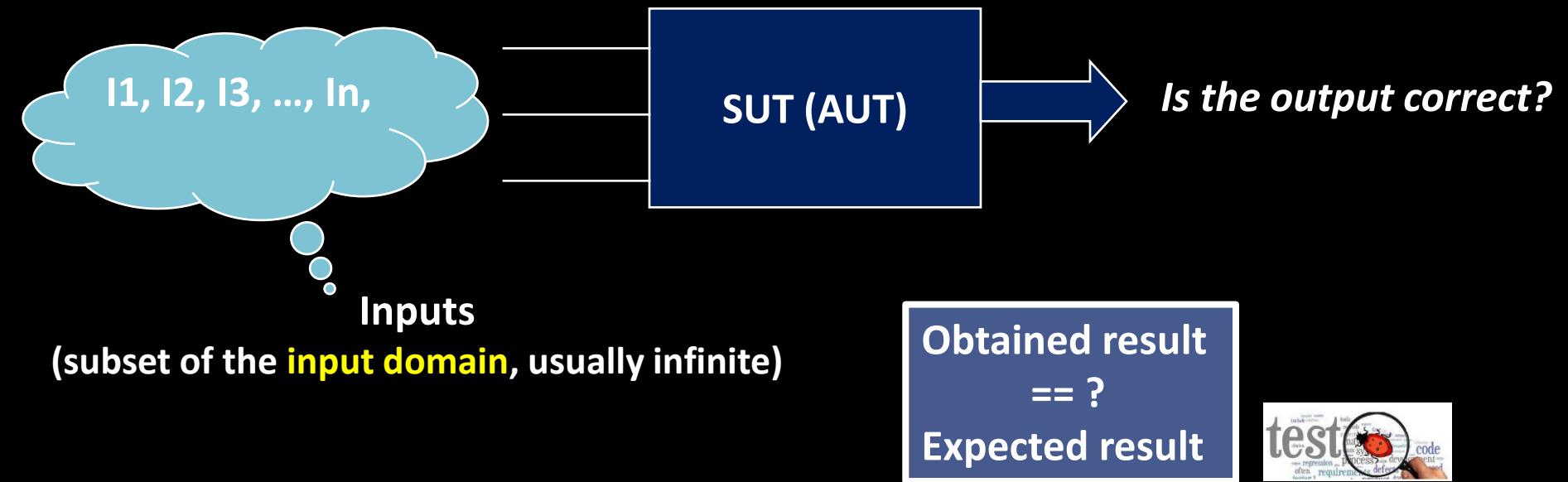
In the video Documentation testing is explained in a strange way. I prefer the following definition: documentation testing ensures that documentation about how to use the system matches with what the system does, providing proof that system changes and improvement have been documented

(Functional) Software testing

(in general, for all apps/systems)

- **(Functional) Software testing** is a systematic approach attempting to find defects, i.e., bugs in a software system

Sut = system under test



Goals of Software testing

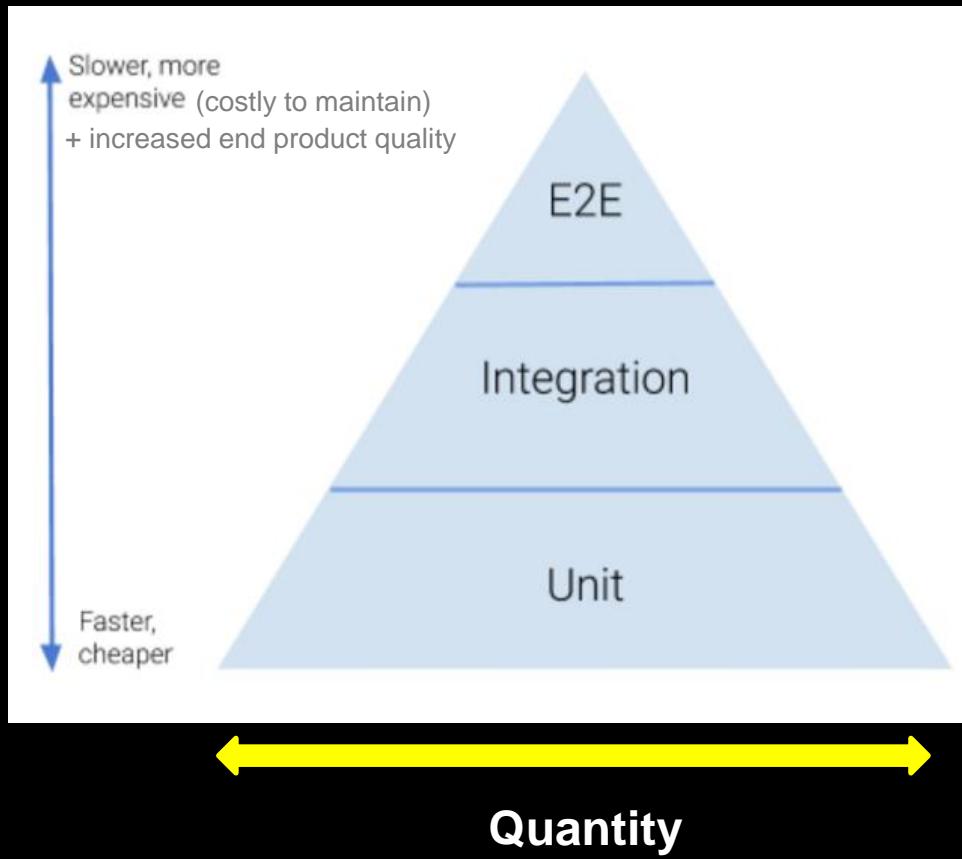
1. Expose and **eliminate bugs** as early as possible
2. Increase **confidence** in software correctness
3. Increase **software reliability** (24/7/365)



- NOT a goal of testing: **prove correctness**, because **exhaustive testing** is unaffordable (input domain is huge!)
 - Other techniques to do this, e.g. Model checking

Levels of Software Testing

Test Pyramid Rule



Testing the system as a whole in real conditions (using the GUI)



Individual units are combined and tested as a group



Testing of a single function/class



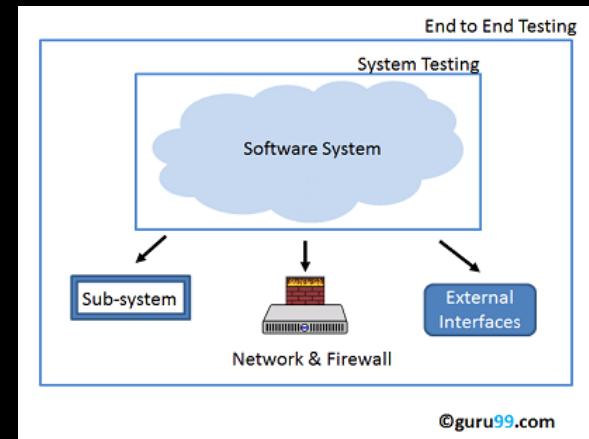
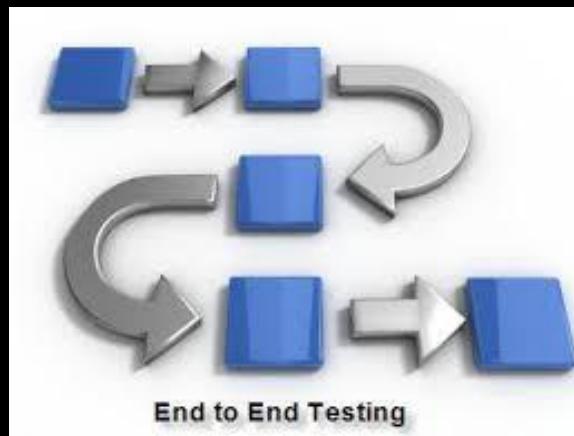
In theory, the distribution of test cases in a project should resemble a pyramid

End-to-End (E2E) Testing

Testing the system as a whole through the GUI in real conditions

- considering network and external systems

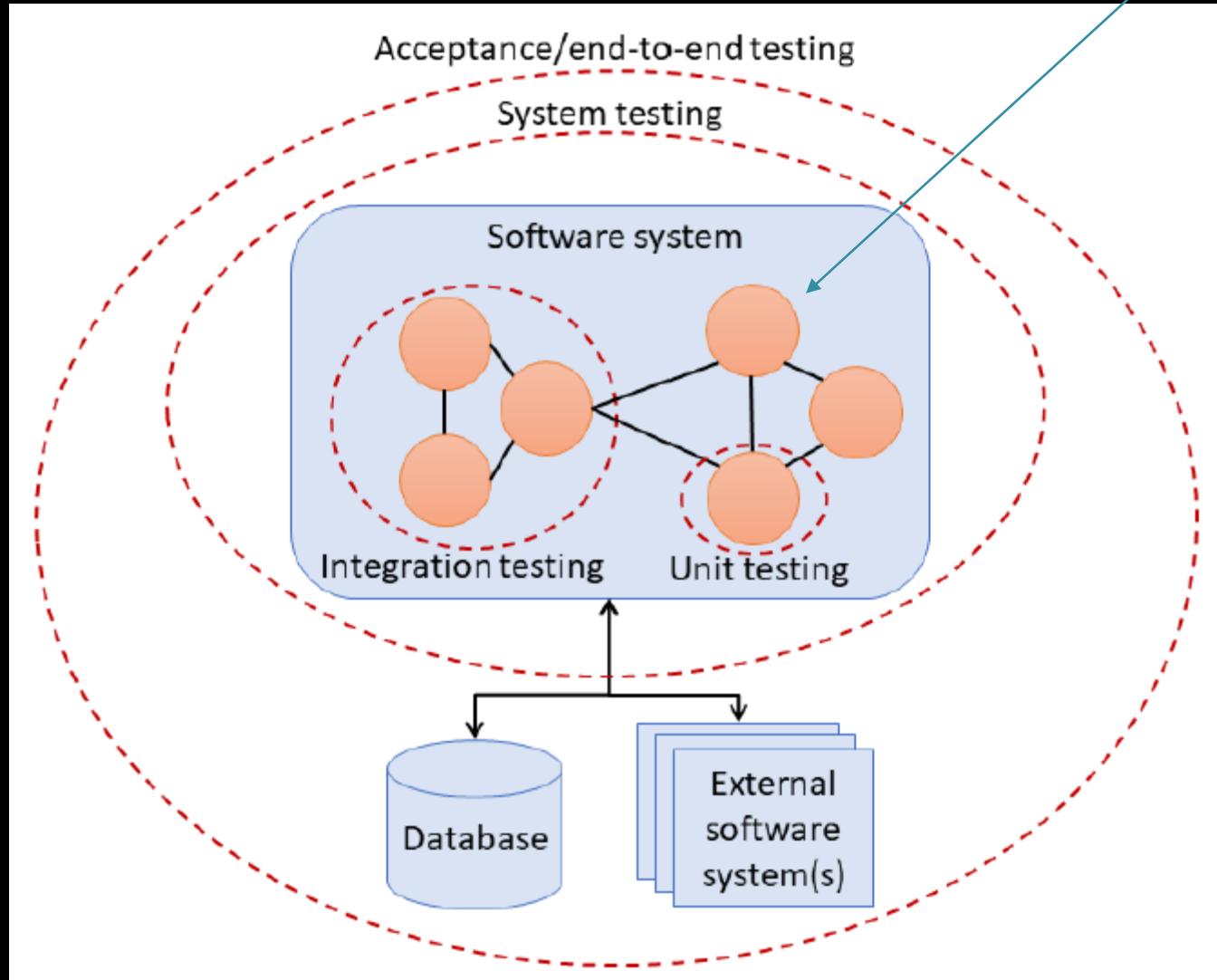
**Features
are tested
in real conditions**



It is performed **from start to finish under real world scenarios** like communication of the application with hardware, network, database and other applications ...

To Summarize

Component



Pros and Cons of E2E Testing

- **Good things**
 - Hide the complexity of the (backend) system
 - Uniformed interface (the GUI) – only one
 - A testing framework at the front-end can automate the tests
 - Particularly useful (when automated) for regression testing
(next slide)
- **Bad things**
 - Time consuming to run
 - Prone to non-determinism problems
 - Flaky tests, e.g., no internet connection
 - Prone to the fragility problem (expensive to maintain)
 - Cannot pinpoint the root cause of failure
 - Anything in the entire flow could have contributed to the failure



find regressions in the code during software evolution

Regression testing

- The goal of regression testing is to ensure that previously developed and tested code **still functions as expected after** new changes



Four versions of the
Same software system

Regression Testing

<https://www.youtube.com/watch?v=kL-imb40IQw>

Test case and Test suite

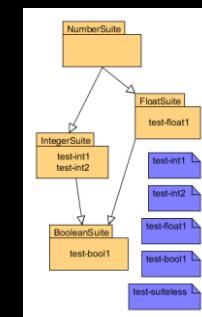
- There are different definitions of **Test case**

IEEE Standard Glossary of Software Engineering Terminology

http://www.mit.jyu.fi/ope/kurssit/TIES462/Materiaalit/IEEE_SoftwareEngGlossary.pdf

- A set of **inputs** (**test data**), **execution preconditions**, and **expected outcomes** developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement [IEEE, Std 610.121990]

- **Test suite** = collection of test cases



Examples (definition must be instantiated!)

What is really a test case depends on the context:

Sorting algorithm

Test case for desktop app “Sort”:

- Input: <“I”, 12, -29, 32, > I=Increasing order
- Expected Output: -29 12 32

input (test data)
+
expected output

Web application

Test case for Shopping cart Web Apps

Input:

1. login to <<http://www.abc.it/home.html>>
2. go to page “Cart”
3. select product X
4. add the product to the shopping cart

Expected Output:

Product X has been added to the cart

sequence of test steps
+
input (test data)
+
expected output

How to find good inputs (test data)

- To create effective test cases we have to find 'good' inputs (test data)
 - We can not select them randomly!
- There are two families/categories of approaches:



Each category contains different approaches!

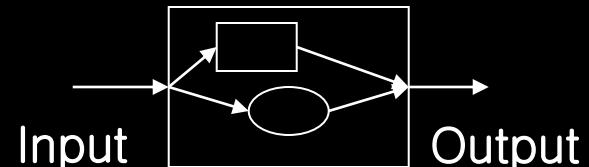
White vs. Black box testing

White Box Testing is a software testing family in which the internal structure of the software being tested **is known** to the Software Tester

- Usually **based on code/models**
 - ex. Code (or Line) coverage

Black Box Testing is a software testing family in which the internal structure of the software being tested **is not known** to the Software Tester

- Usually based on requirements or user's expectations



Internal structure of the code is used to create TCs



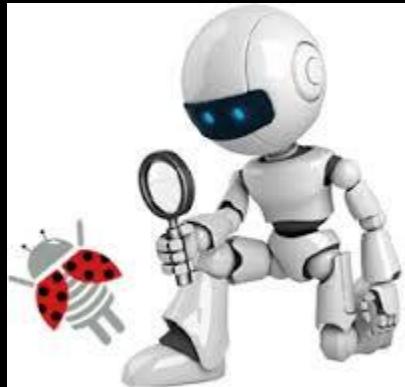
Internal structure of the code is NOT used to create TCs

Manual vs Automated Software Testing



Testers **manually** execute the test cases

VS.



A testing framework **automatically** execute test scripts (implementation of test cases) that are produced by Testers

Difference between Manual and Automated Testing

<https://www.youtube.com/watch?v=RbSIW8jZFe8&list=PLXXvO4OXeJrfbPrI0CV-re2VkdiV1dC7X&index=4>

Manual testing for a Web app

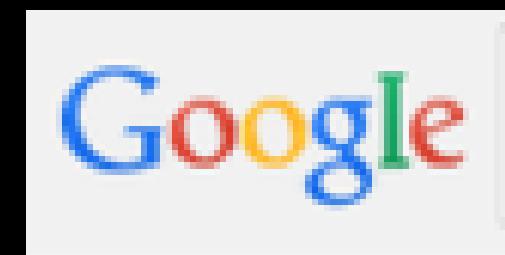
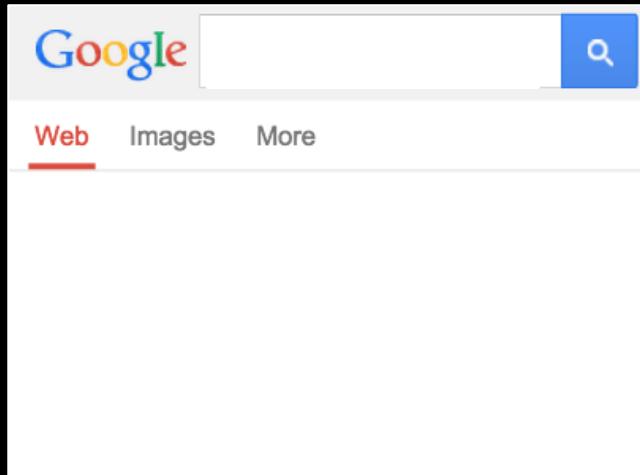
- Process in which the human tester **create and execute test cases** on the Application Under Test (AUT)
 - Two phases: creation (design) and execution
- A test case **for a Web app** is a set of actions (**Test steps**) + inputs (**Test data**) done on a AUT to verify that it works as expected
 - Expected result



Note: old google.com version ...

Testing the Google Search page

Web App under test



(E2E) Test case example

Test Step

Test case (can be executed manually)

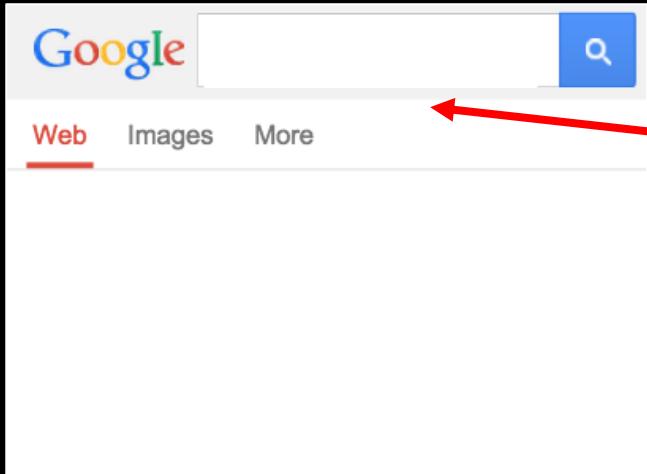
1. Open www.google.com
2. Insert “ERC” in input field
3. Click submit button
4. Check first result = “ERC: European Research Council”

Test data



(E2E) Test case example

Web App under test



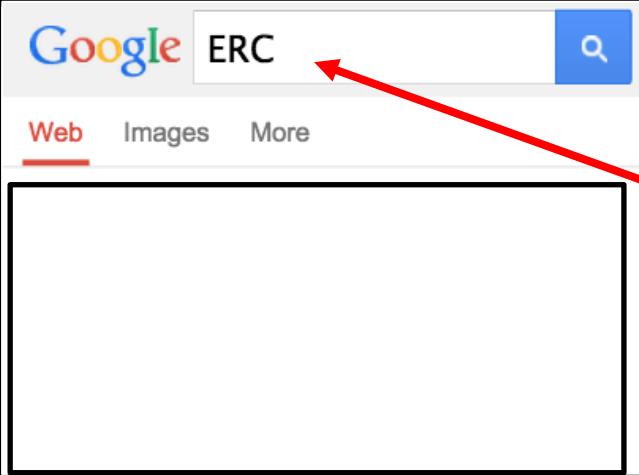
Test case (can be executed manually)

1. Open www.google.com
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "*ERC: European Research Council*"



(E2E) Test case example

Web App under test



Test case (can be executed manually)

1. Open www.google.com
2. Insert “ERC” in input field
3. Click submit button
4. Check first result = “ERC: European Research Council”



(E2E) Test case example

Web App under test

A screenshot of a Google search results page. The search bar at the top contains the query "ERC". Below the search bar, there are three navigation links: "Web" (underlined in red), "Images", and "More". The first search result is a link to the European Research Council's website, titled "ERC: European Research Council" and located at "erc.europa.eu/". The snippet below the title describes the ERC as a funding body set up to stimulate scientific excellence. At the bottom of the snippet, there are links to "Funding and Grants", "About ERC", "Contact us", and "Projects and Results". A red arrow points from the "Test case" section to the "Submit" button in the search bar.

Test case (can be executed manually)

1. Open www.google.com
2. Insert “ERC” in input field
3. Click submit button
4. Check first result = “ERC: European Research Council”



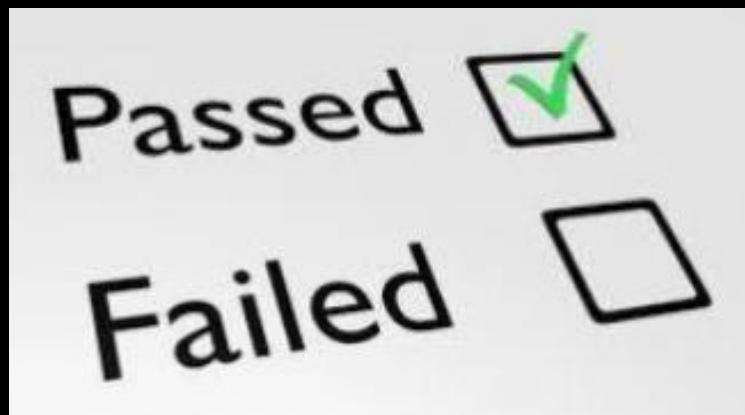
(E2E) Test case example

Web App under test

A screenshot of a Google search results page. The search bar at the top contains the query "ERC". Below the search bar, there are three navigation links: "Web", "Images", and "More". The first search result is highlighted with a red box and a red arrow points from the "Test case" section below to this result. The result title is "ERC: European Research Council" with the URL "erc.europa.eu/". The snippet below the title reads: "Funding body set up to stimulate scientific excellence by supporting and encouraging scientists, engineers and scholars ...". At the bottom of the snippet, there are links: "Funding and Grants - About ERC - Contact us - Projects and Results".

Test case (can be executed manually)

1. Open www.google.com
2. Insert “ERC” in input field
3. Click submit button
4. Check first result = “ERC: European Research Council”



'Table' of test cases for Login

UserID

Password

Typically Testers produce a **table of test cases** and then manually execute the tests

TABLE of test cases

| Test Case ID | Test Case Description | Test Steps | Test Data | Expected Results |
|--------------|--|---|---------------------------------------|---|
| TU01 | Check Customer Login with valid Data | <ol style="list-style-type: none">1. Go to site http://demo.guru99.com2. Enter UserId3. Enter Password4. Click Submit | Userid = guru99 Password = pass99 | User should Login into an application |
| TU02 | Check Customer Login with invalid Data | <ol style="list-style-type: none">1. Go to site http://demo.guru99.com2. Enter UserId3. Enter Password4. Click Submit | Userid = guru99 Password = glass99 | User should not Login into an application |

How ‘to write’ a test case

<https://www.youtube.com/watch?v=BBmA5Qp6Ghk>

Test script

A **test script** is a set of commands in a programming language that will be executed by a **Testing Framework** on the AUT to test that the app/system works as expected

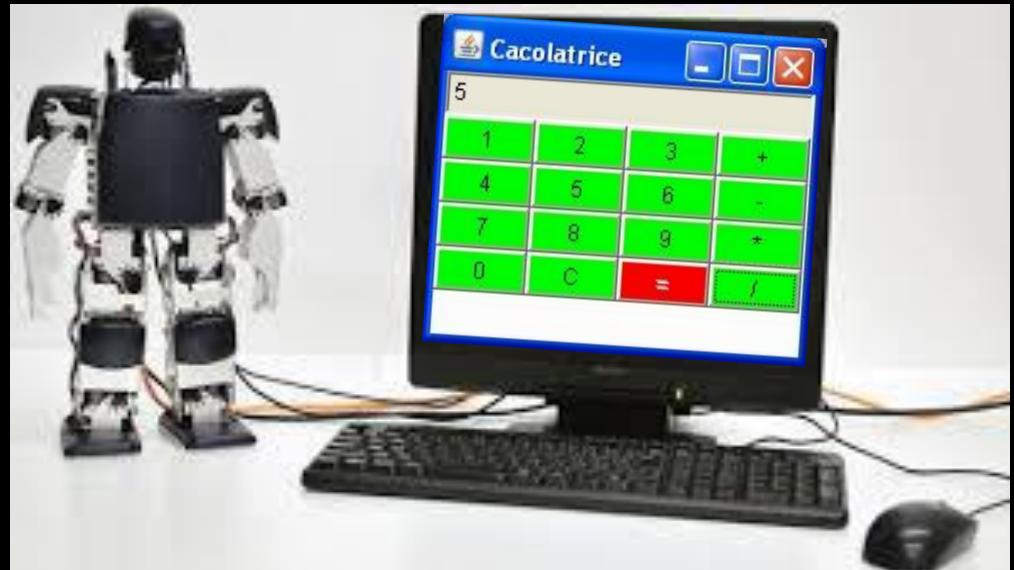
Test Script = test case that can be executed by a Testing Framework!

Test script

```
Press(3);  
Press('+');  
Press(2);  
Press('=')  
Assert(total==5)
```

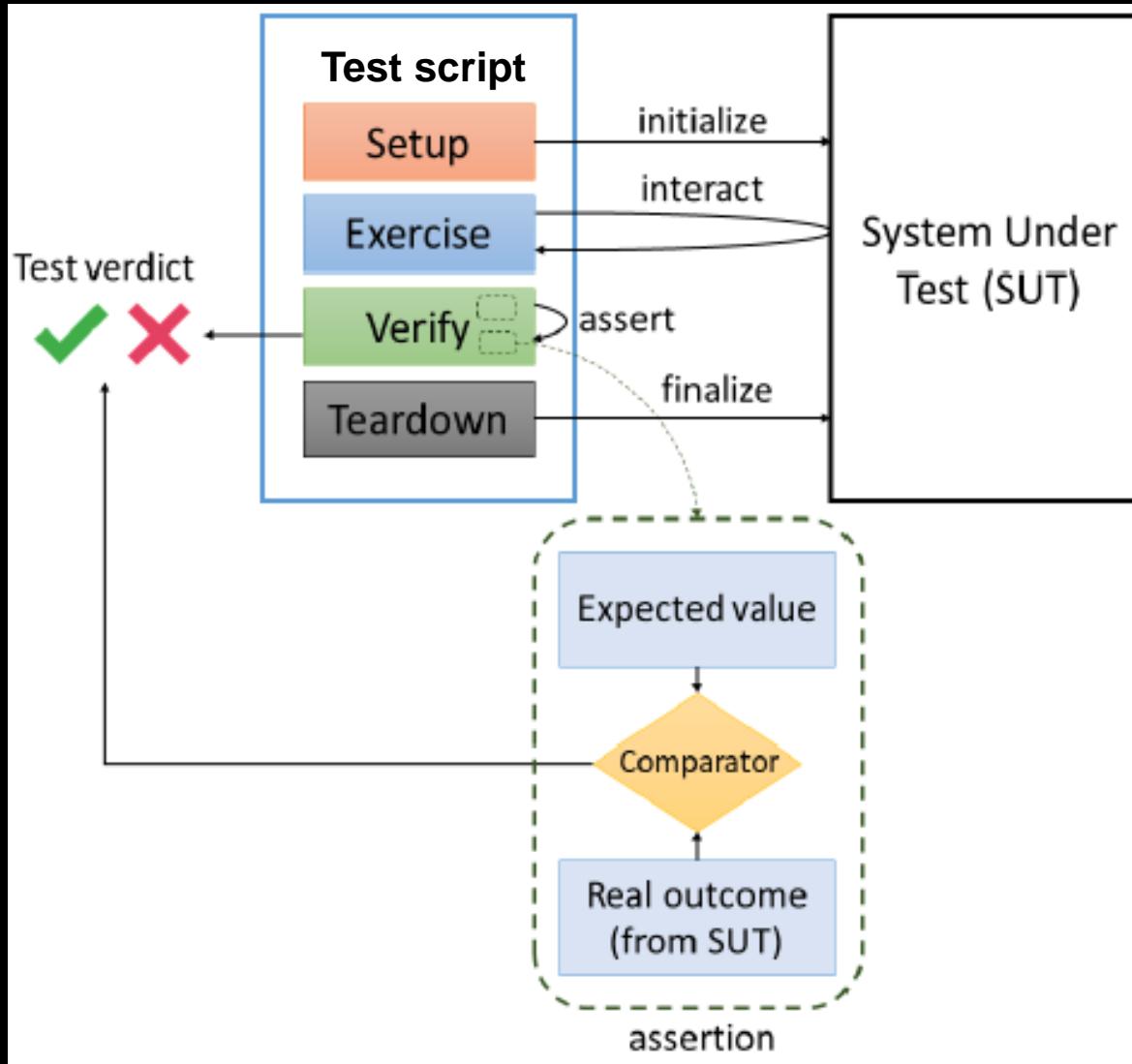
Actual result

Expected result



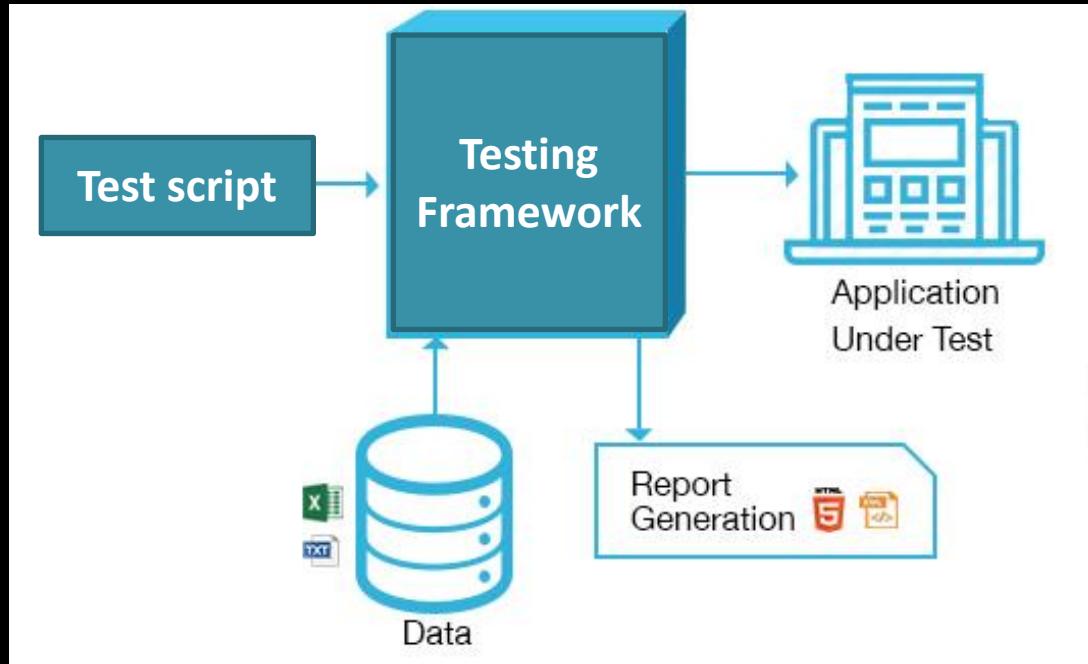
Assertion = is the mechanism used in software testing to compare the actual behavior of the software with what is expected

Test script structure



Testing Framework

'Data driven' test scripts



A Testing Framework is a software tool/component:

- able to 'interpret' and execute **Test scripts** (and Test suite)
- able to exercise **Application Under test (AUT)**
- able to read **test data** from a repository (data driven / parametric test scripts)
- able to generate a **Test report** (pass/fail)

A Unit Testing Framework example: JUnit 5

Test Class

```
import static org.junit.jupiter.api.Assertions.assertEquals;  
  
import org.junit.jupiter.api.BeforeEach;  
import org.junit.jupiter.api.DisplayName;  
import org.junit.jupiter.api.RepeatedTest;  
import org.junit.jupiter.api.Test;  
  
class CalculatorTest {  
  
    Calculator calculator;  
  
    @BeforeEach  
    void setUp() {  
        calculator = new Calculator();  
    }  
  
    @Test  
    @DisplayName("Simple multiplication should work")  
    void testMultiply() {  
        assertEquals(20, calculator.multiply(4, 5),  
                    "Regular multiplication should work");  
    }  
  
    @RepeatedTest(5)  
    @DisplayName("Ensure correct handling of zero")  
    void testMultiplyWithZero() {  
        assertEquals(0, calculator.multiply(0, 5), "Multiple with zero should be zero");  
        assertEquals(0, calculator.multiply(5, 0), "Multiple with zero should be zero");  
    }  
}
```

① Import libraries

② Create and setup of the object

③ First Test

④ ⑤ ⑥

Class To be tested

```
package com.vogella.junit5;  
  
public class Calculator {  
  
    public int multiply(int a, int b) {  
        return a * b;  
    }  
}
```

We will
introduce JUnit 5
in the next
lessons

Java annotations (ex. @Test) are used 'to instrument' the Testing framework

Why automated software testing?



More quality

Cost reduction

Reuse

Manual testing is
laborious, boring
and time
consuming

Test scripts can
be run more
often and faster

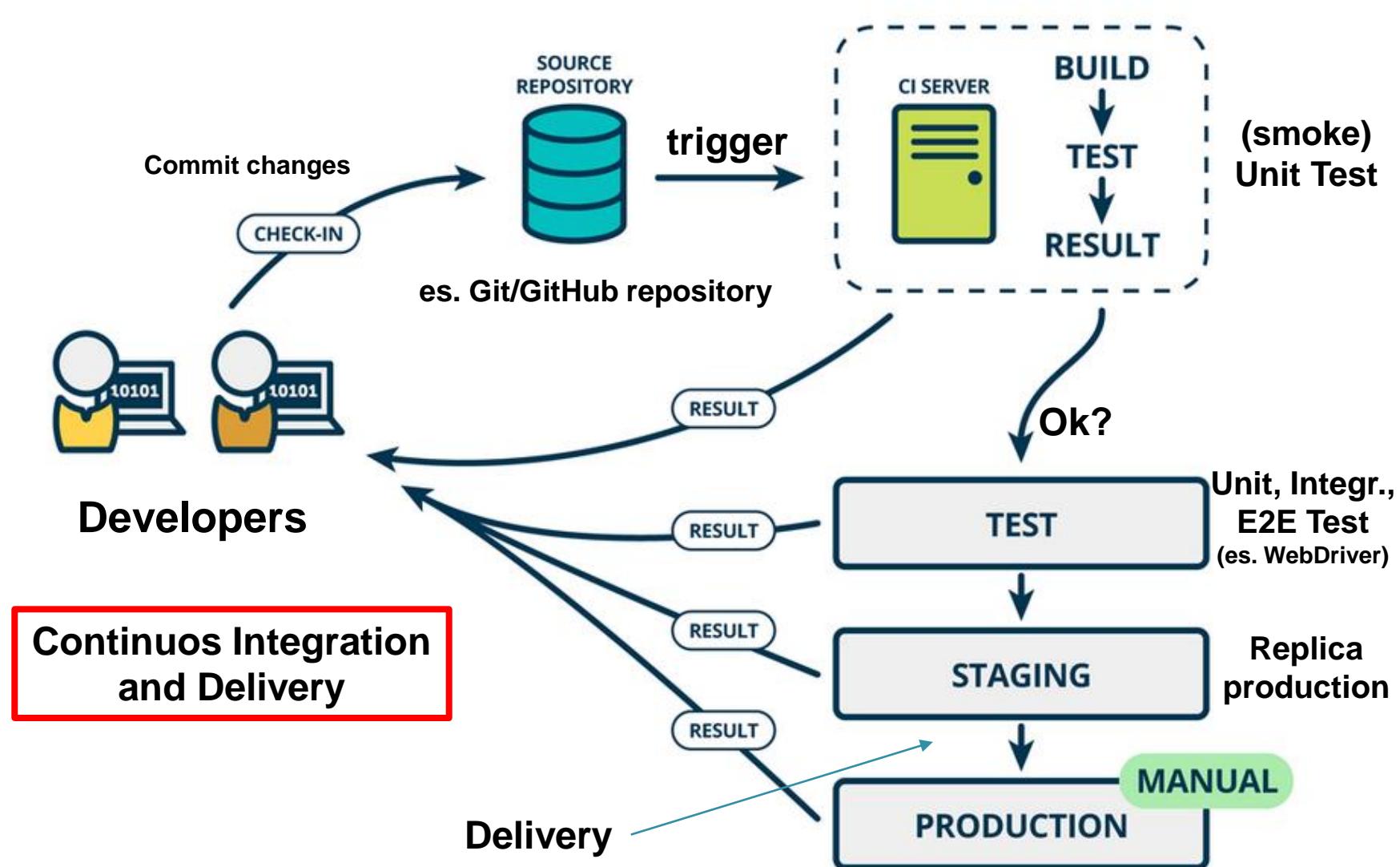
Test scripts
can be
reused

Automated software testing

<https://www.youtube.com/watch?v=HtXJe1rAzal>

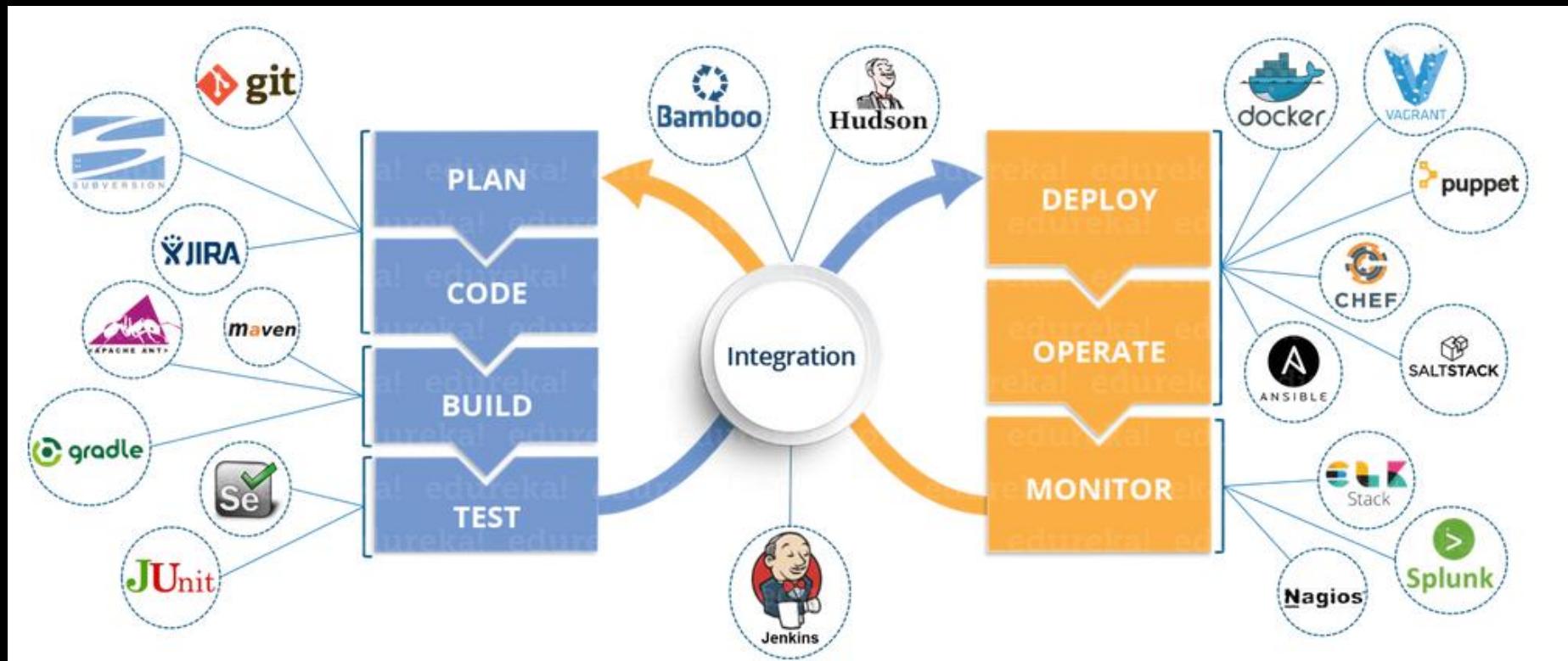
Build produces the executables

Modern development process

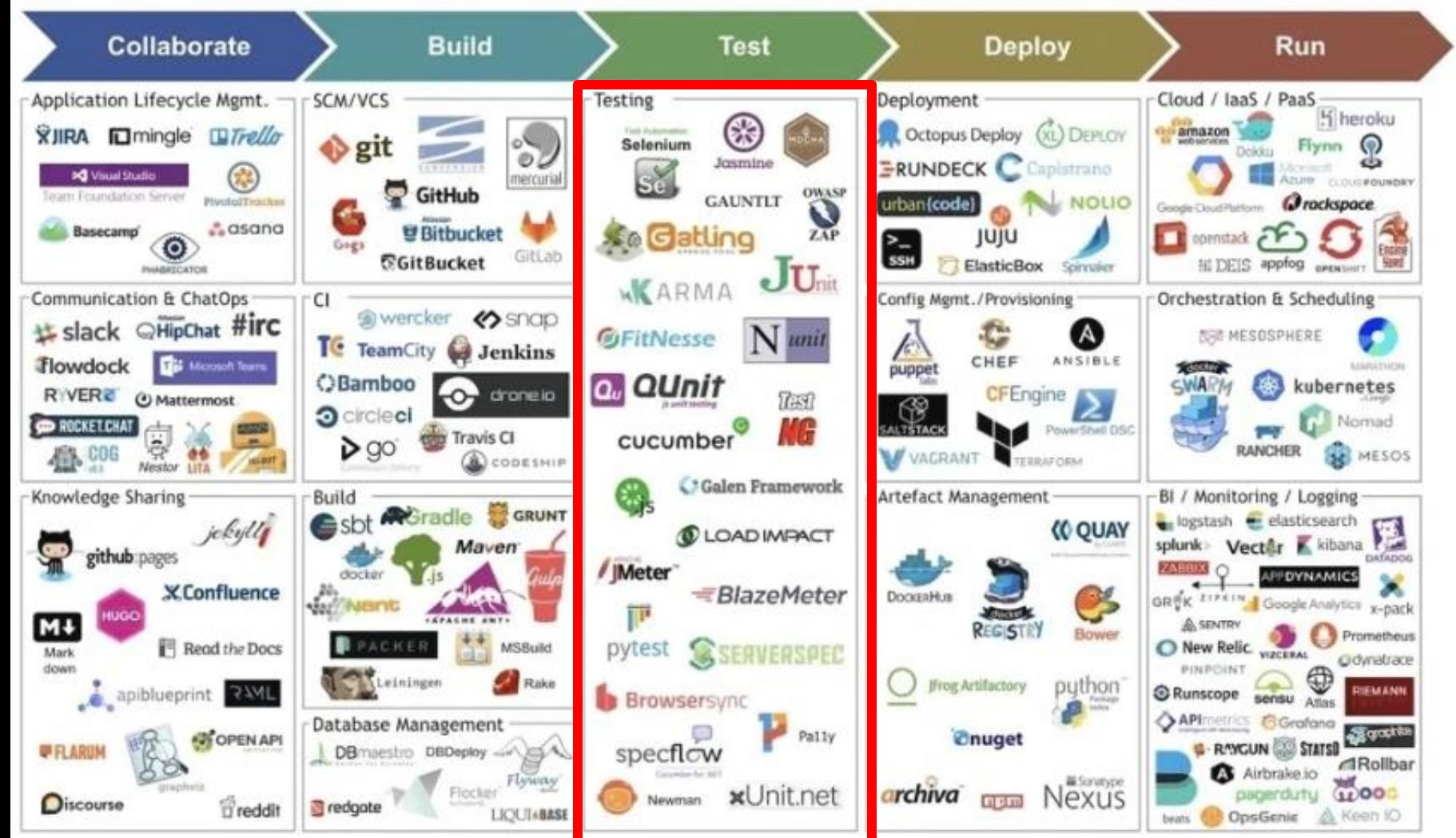


Name and number of environments/servers vary by the team, product and organization

Modern development process (Tools and Frameworks)

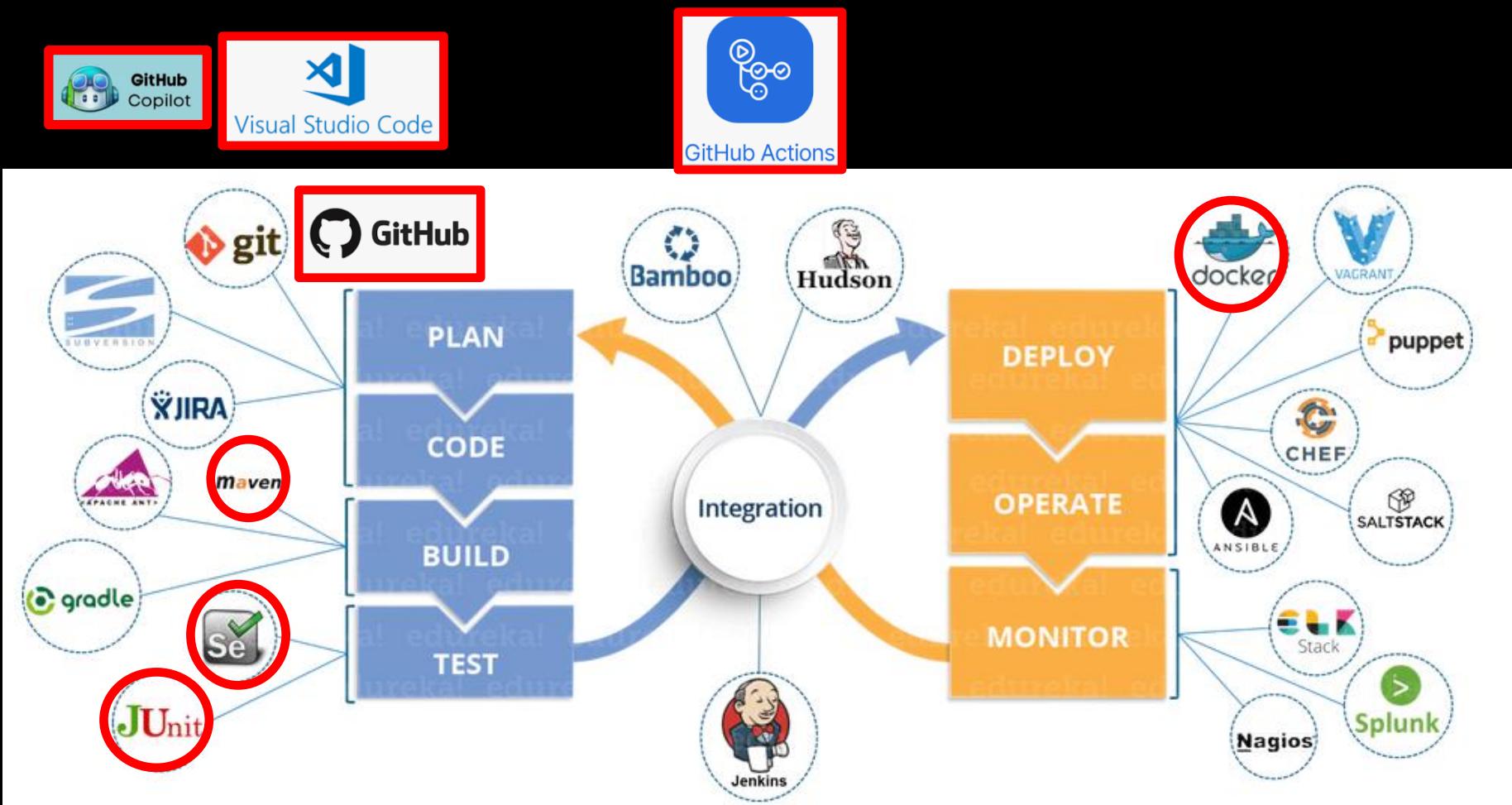


- This development process must be repeatable and reliable
 - For this reason during development a lot of tools are used!



- The problem is that available tools are very numerous!!
- As a consequence, it is difficult to select a subset of them

Tools/Frameworks we will use in FSTT (in red)



CI/CD

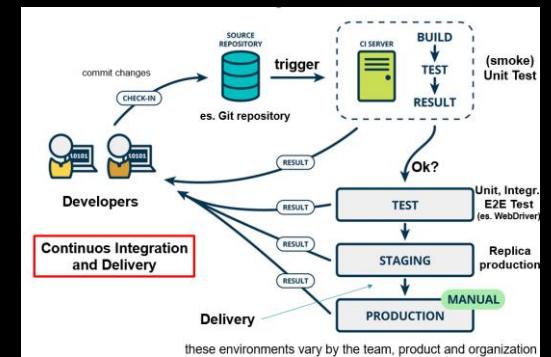
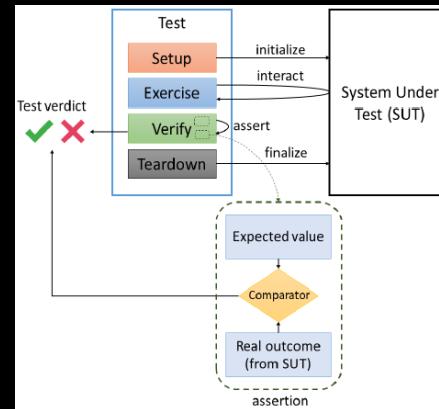
<https://www.youtube.com/watch?v=k2aNsQKwyOo>

Modern development process (DevOps) and tools

<https://www.youtube.com/watch?v=vkxk-fq8T6U>

To Recap

- Course Overview
 - Goals, Prerequisites, Content
- Software engineering
- External Software quality factors
- Functional vs. non-functional testing
 - E.g., non-functional: security testing
- Levels of Software testing
 - Unit, Integration and E2E
- White and Black box testing
- Manual vs Automated Software testing
- Test cases, Test scripts and Testing Framework
- Modern development process



Functional and Security Testing Course

2023-2024

Basics of Web apps and VS CODE – lesson 2

Filippo Ricca
DIBRIS, University of Genova

Course Overview

Students will learn the fundamentals in **functional** and **security testing** of software systems, with special emphasis on challenges posed by **Web** and Mobile **applications** and using automated testing tools



BASICS OF WEB APPLICATIONS



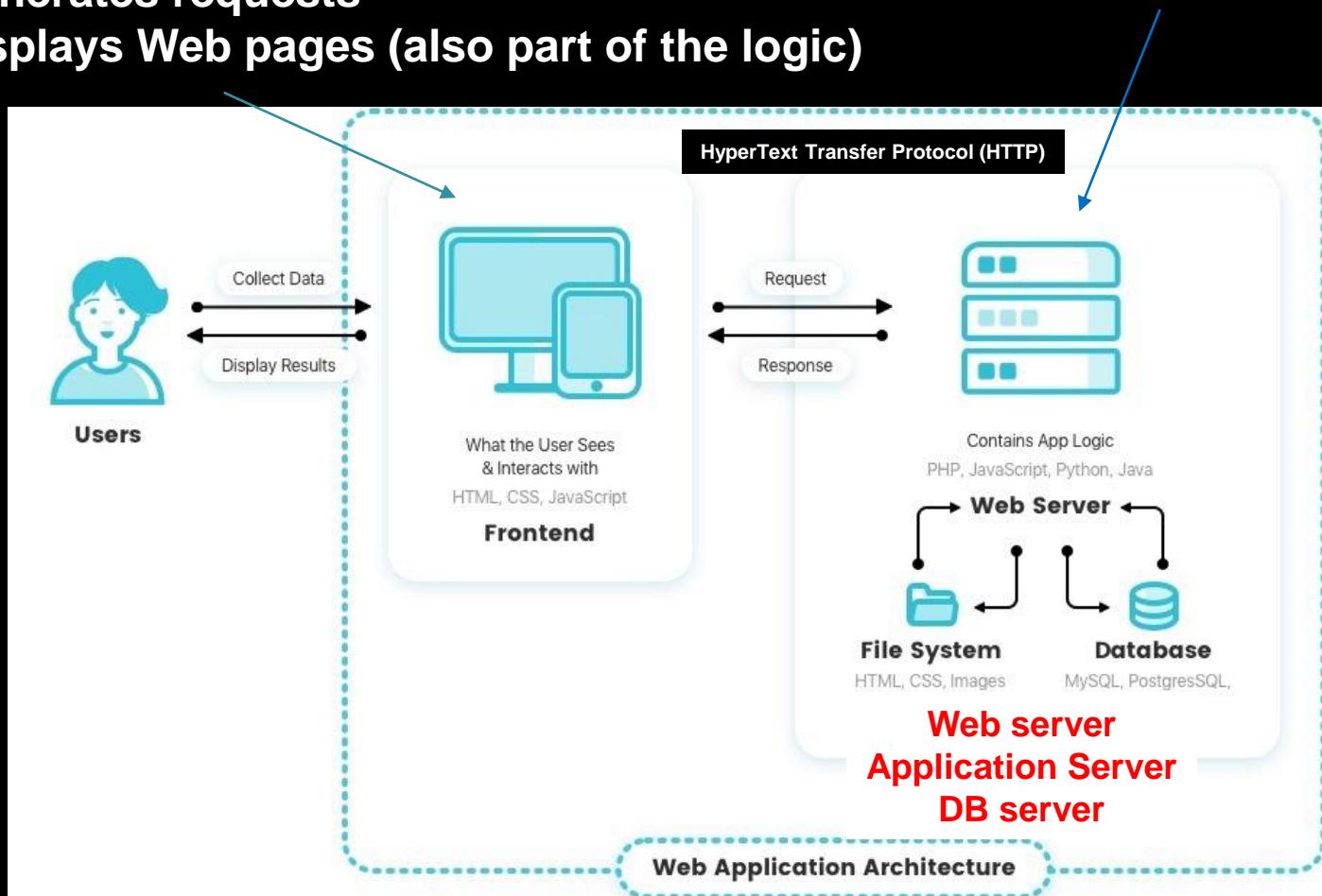
Web applications

Client:

- Hosts only a component (browser)
- Generates requests
- Displays Web pages (also part of the logic)

Server (Web, Application, DB):

- Generates Web pages
- Exposes the business logic

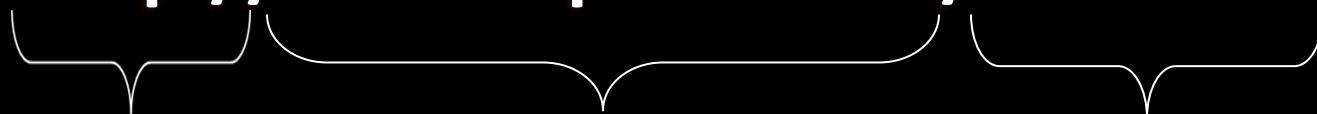


HTTP URL

- A browser requires a **Web page** by means of a **HTTP URL**
 - Uniform Resource Locator
- An URL identifies in unique way **a resource** specifying:
 - the **name of the server** that contains it
 - the **name of the file** on that server

Example:

http://www.repubblica.it/index.html



Protocol

Domain name

File name

Web page (Client-side languages)



Web apps are heterogeneous systems!

```
<html lang="en" id="facebook" class="no_js">
<head><meta charset="utf-8" /><script>function envFlush(a){function b(c){for(var d in a)c[d]=a[d];}f
envFlush({"ffid1":"AcEhdku5-jWdFFPkkIjOWbBGHn8S8s8ufqddxwgALSi8Ws7KKbRXQpmA96UUq5cwbs","ffid2":"AcIm0
<link rel="stylesheet" href="https://s-static.ak.fcdn.net/rsr...</link>

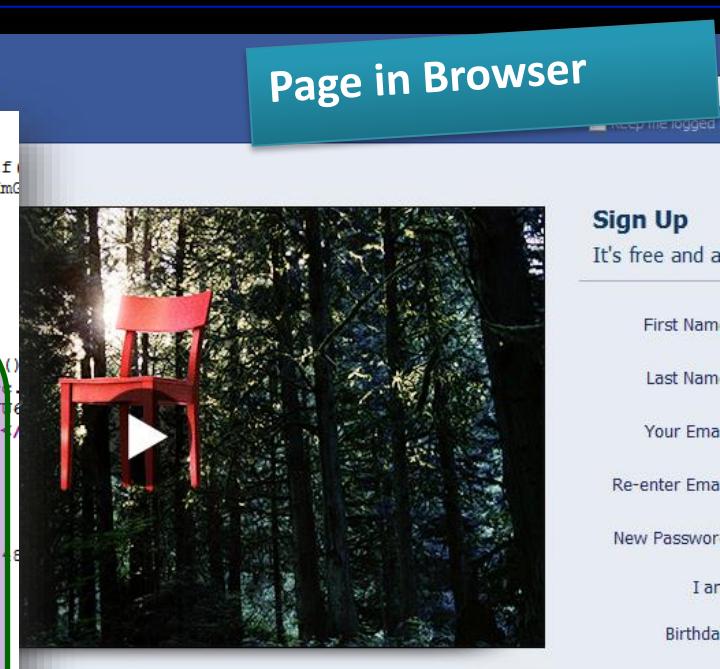
<script src="https://s-static.ak.fcdn.net/rsr...</script>
<script>window.Bootloader && Bootloader.done(["j2Y1E"]);</script><script>new (require("ServerJS"))()
<script>Bootloader.setResourceMap({"zmU4p":{"type":"css","src":https://s-static.ak.fcdn.net/rsr...
Bootloader.enableBootload("legacy:async-signal":{"resources":["JU6Q1"]}, "AsyncDOM":{"resources":["JU6Q1"]})
<script>require("InitialJSLoader").loadOnDOMContentLoaded(["JU6Q1","MX2xp","T8gbI","Abmov","UjXTQ"]);
<script>
Bootloader.configurePage(["zmU4p","M7bAm","jg0IV","D721J","VuNPD"]);
Bootloader.done(["jDr+c"]);

new (require("ServerJS"))().handle({"instances":[[{"m187486_2","Autosize","m187486_1"}, {"__m": "m187486_1"}]}, {
onloadRegister_DEPRECATED(function (){useragentcm()});
onloadRegister_DEPRECATED(function (){
    if (typeof RegUtil == 'undefined') {
        RegUtil = function() {};
    }
    RegUtil.captcha_class="ReCaptchaCaptcha";
    RegUtil.recaptcha_public_key="6LfDxsYSAAAAAGGLBGaRu
});;
onloadRegister_DEPRECATED(function (){RegistrationBootloa
onloadRegister_DEPRECATED(function (){startFormTypeABTest();
onloadRegister_DEPRECATED(function (){$("#upfspt3").value = tz_calculate(1352931874)});
```

Source behind

HTML

JavaScript



Page in Browser

Sign Up

It's free and a

First Name

Last Name

Your Email

Re-enter Email

New Password

Forgot

Birthda

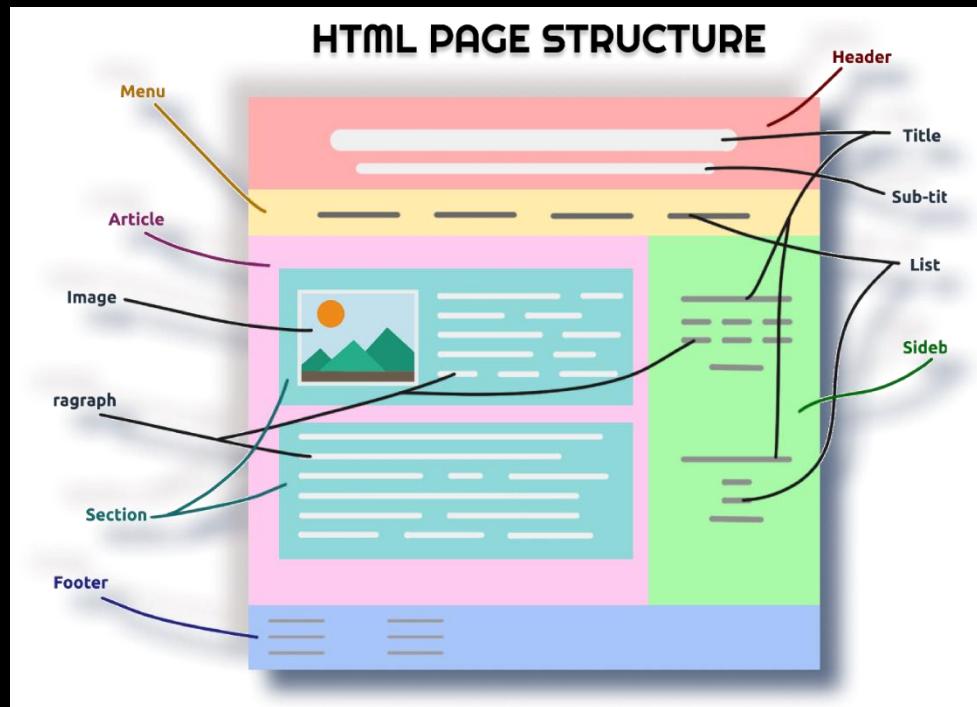
Mix of three languages



- Logic
- Style
- Structure

HTML

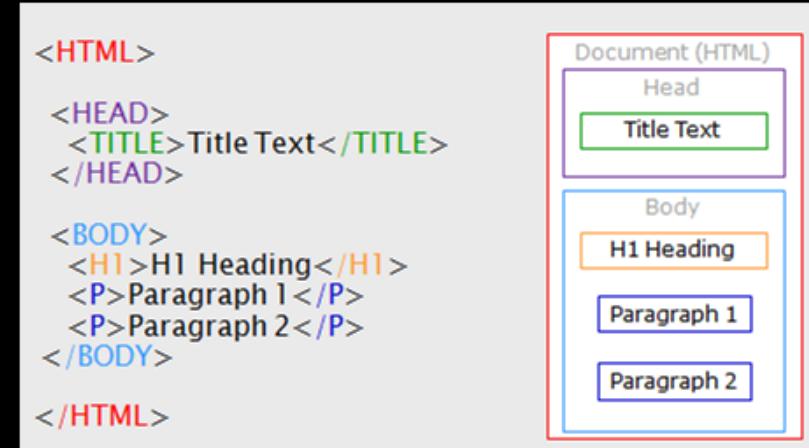
- HTML stands for **HyperText Markup Language**. It contains **Tags** and **Attributes** that are used to design the structure of the web pages.
- Also, we can link multiple pages using **Hyperlinks**



```
<HTML>
<HEAD>
  <TITLE>Title Text</TITLE>
</HEAD>

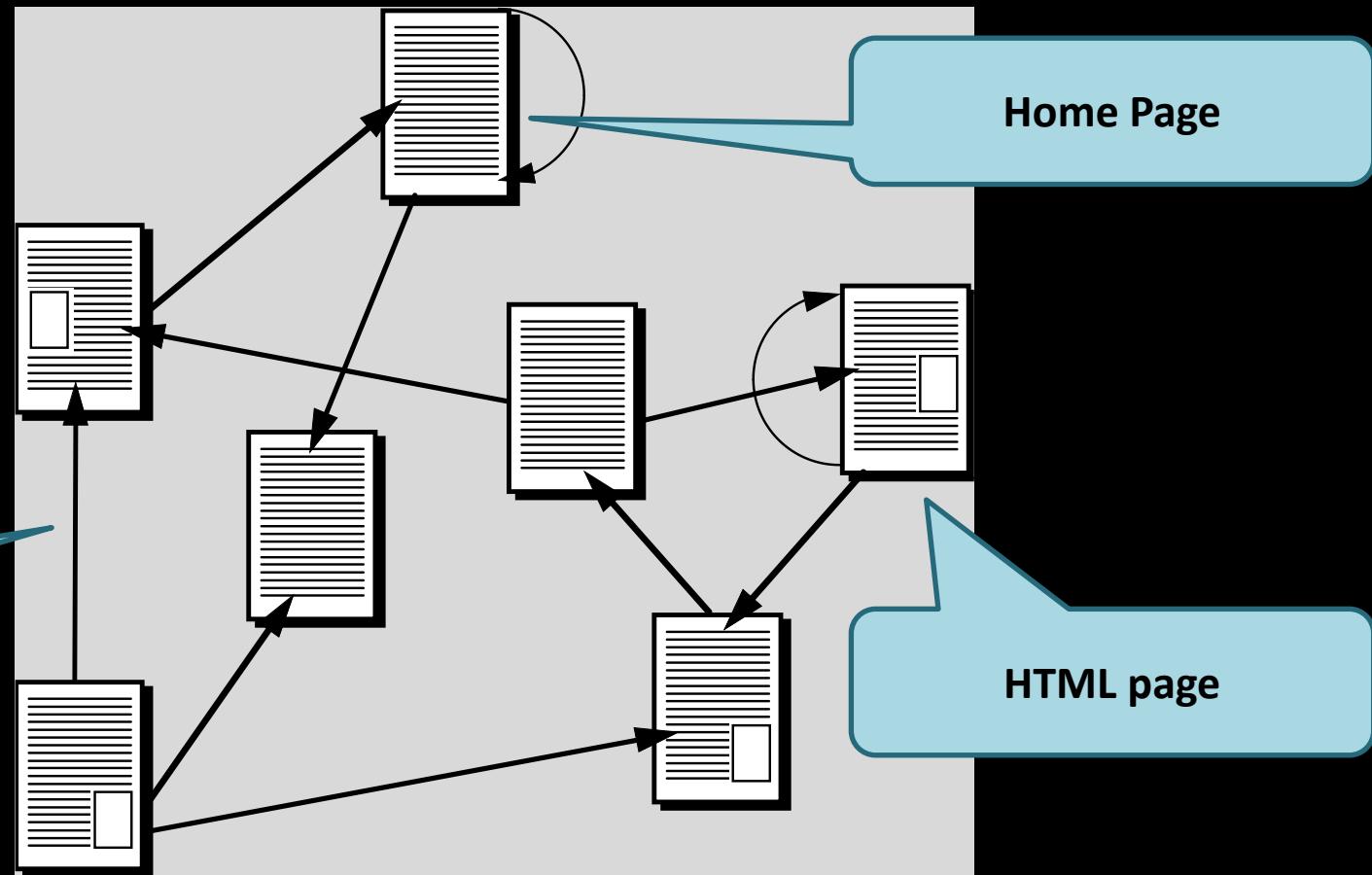
<BODY>
  <H1>H1 Heading</H1>
  <P>Paragraph 1</P>
  <P>Paragraph 2</P>
</BODY>

</HTML>
```



HyperLinks

Used to move from one page to another



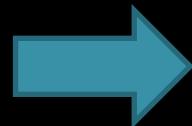
Basics of Web applications

<https://www.youtube.com/watch?v=RsQ1tFLwIdY>

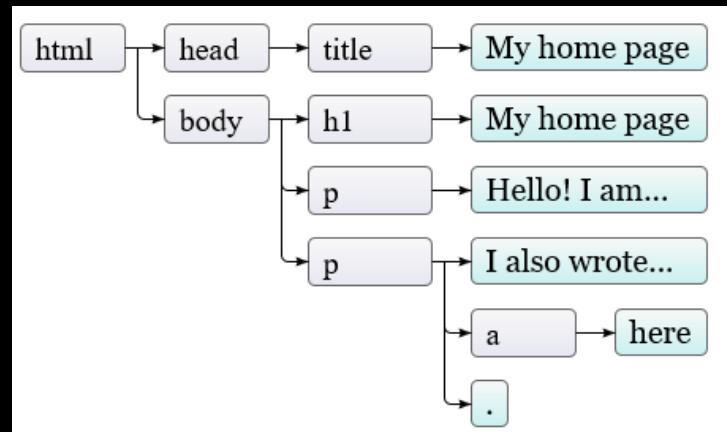
Document Object Model (DOM)

- When a web page is loaded in a browser, **the browser**:
 - retrieves the page's HTML text
 - parses it
 - builds up a **model of the document (DOM)**, used to draw the page

```
<!doctype html>
<html>
  <head>
    <title>My home page</title>
  </head>
  <body>
    <h1>My home page</h1>
    <p>Hello, I am...</p>
    <p>I also wrote...
      <a href="http://...">here</a>
    </p>
  </body>
</html>
```



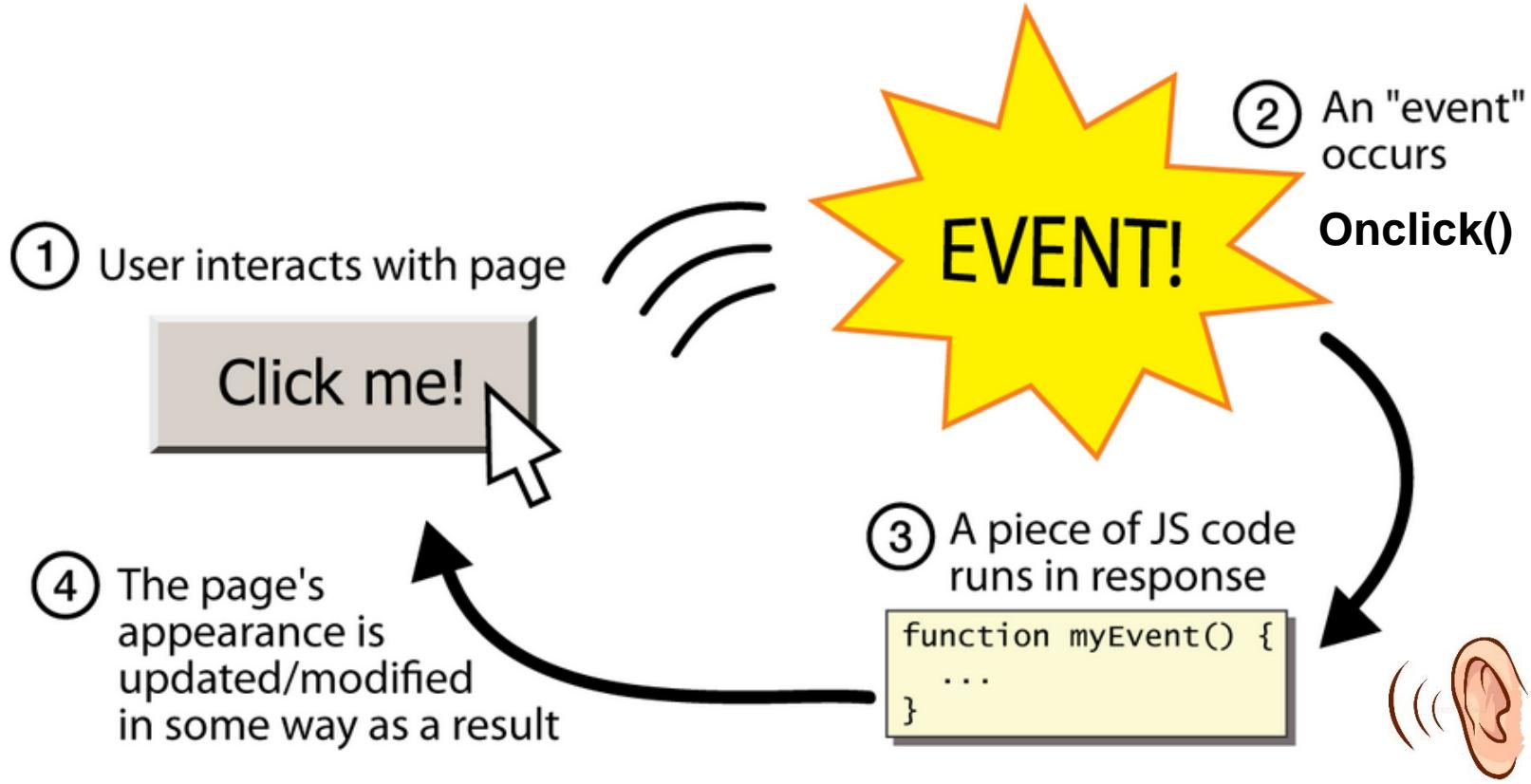
DOM tree representation



‘Each node is an object’

JavaScript (contained in the web page) can read the DOM and change it at run time!

'Event driven programming' paradigm



Mechanism used by JavaScript to modify a page is based on
'Event Driven Programming' paradigm

Clickable element

DOM manipulation (simple example)

JavaScript function

```
<div id="mydiv" onClick="addtext('*')">  
    Click here:  
</div>  
  
<script type="text/javascript">  
function addtext(what){  
    var mytext=document.createTextNode(what)  
    document.getElementById("mydiv").appendChild(mytext)  
}  
</script>
```

Simple to use JS editor
<https://js.do/>

DOM manipulation

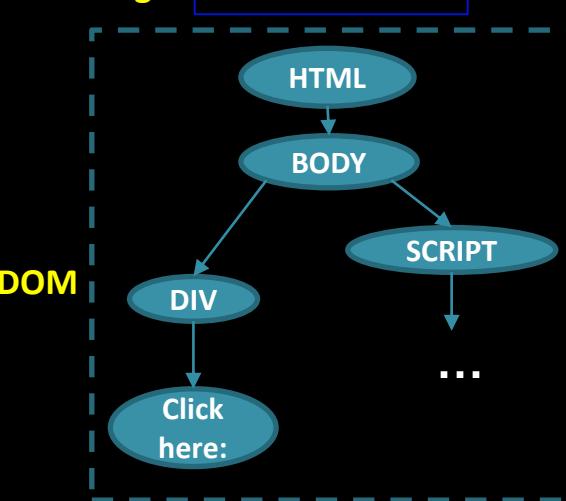
(simple example)

JavaScript function

```
<div id="mydiv" onClick="addtext('*')">  
    Click here:  
</div>  
  
<script type="text/javascript">  
function addtext(what){  
    var mytext=document.createTextNode(what)  
    document.getElementById("mydiv").appendChild(mytext)  
}  
</script>
```

Simple to use JS editor
<https://js.do/>

HTML Page Click here:



DOM manipulation

(simple example)

JavaScript function

```
<div id="mydiv" onClick="addtext('*')">  
    Click here:  
</div>  
  
<script type="text/javascript">  
function addtext(what){  
    var mytext=document.createTextNode(what)  
    document.getElementById("mydiv").appendChild(mytext)  
}  
</script>
```

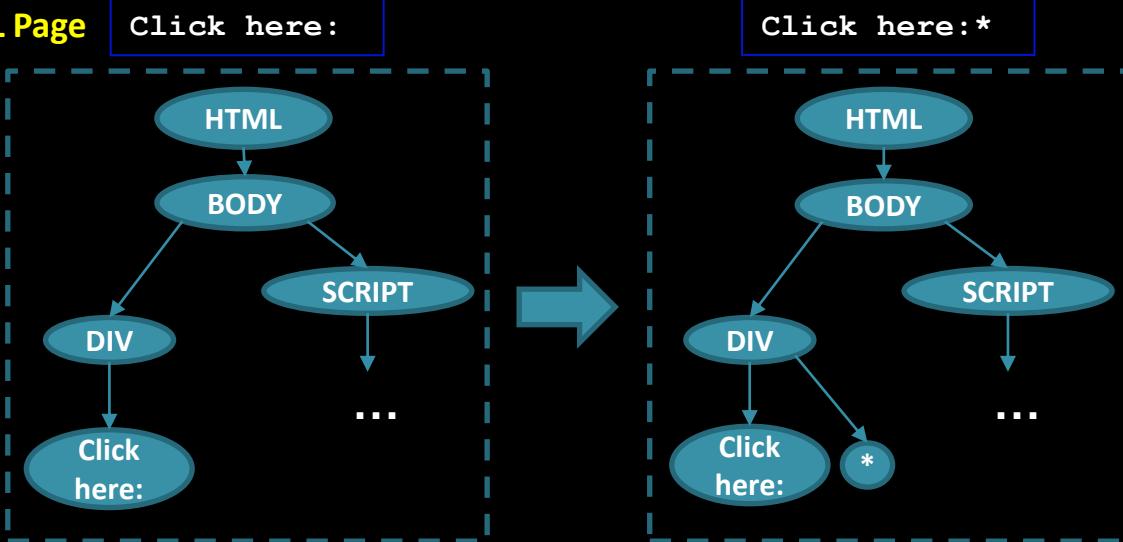
Simple to use JS editor
<https://js.do/>

HTML Page

Click here:

Click here:*

DOM



DOM manipulation

(simple example)

JavaScript function

```
<div id="mydiv" onClick="addtext('*')">  
    Click here:  
</div>  
  
<script type="text/javascript">  
function addtext(what){  
    var mytext=document.createTextNode(what)  
    document.getElementById("mydiv").appendChild(mytext)  
}  
</script>
```

Simple to use JS editor
<https://js.do/>

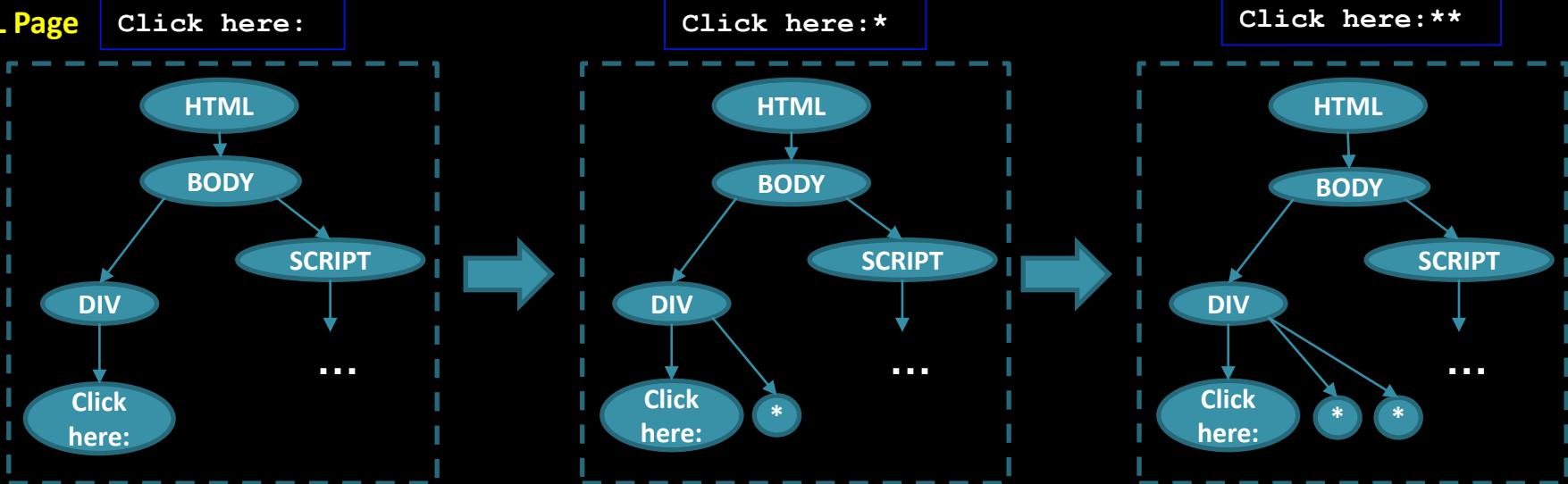
HTML Page

Click here:

Click here:*

Click here:**

DOM



Server side languages

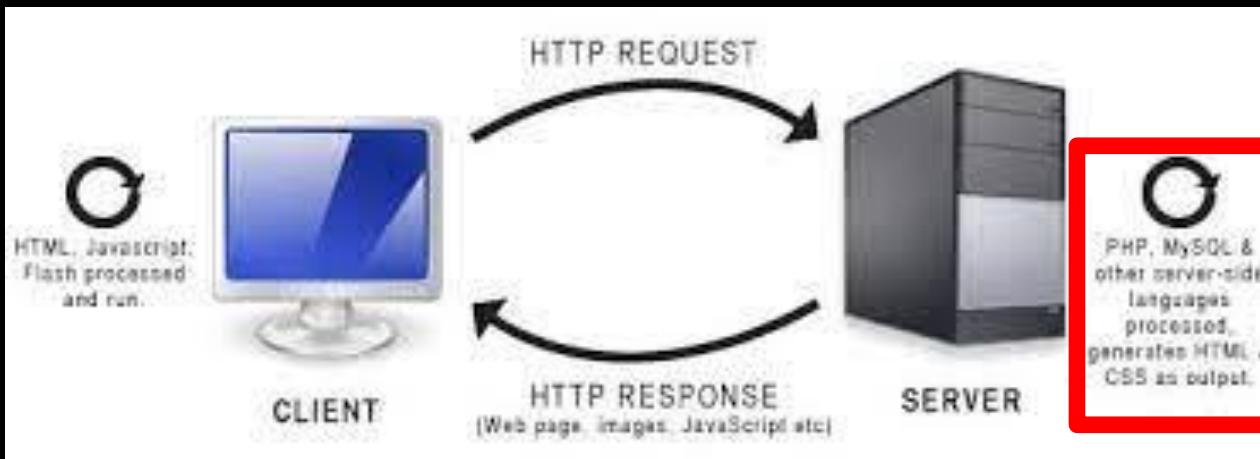
- Server side
 - Scripts can be written in **several different languages**
 - Main language: PHP, Java, C# ...
 - HTML is used to give the structure of the page
 - SQL is used for communicating with DB server

```
1.<?php
2.
3. // Add the new user if one is being added //
4. if($_POST['username'] == 1 && $_POST['password'] != "" && $_POST['type'] != "") {
5.     $query = mysql_query(
6.         "SELECT username FROM users"
7.     );
8.
9.     or die("ManageUsers.php: unable to get list of users - ".mysql_error());
10.
11. // Make sure that the supplied username does not already exist in the database //
12. $userList = mysql_fetch_row($query);
13.
14. if($_POST['username'] == $userList[0]) {
15.     die("ManageUsers.php: Username already exists!");
16.
17. }
18.
19. // If all is good, insert the new user into the database //
20. $query = mysql_query(
21.     "INSERT INTO users VALUES('".$_POST['username']."', '".$_POST['password']."', '".$_POST['type']."')"
22. );
23.
24. or die("ManageUsers.php: Unable to insert new user - ".mysql_error());
25.
26. }
27.
28.
29. // If all is good, insert the new user into the database //
30. $query = mysql_query(
31.     "SELECT * FROM users WHERE id='$_POST[id]'"
32. );
33.
34. or die("ManageUsers.php: Unable to insert new user - ".mysql_error());
35. }
```

PHP

SQL

HTML



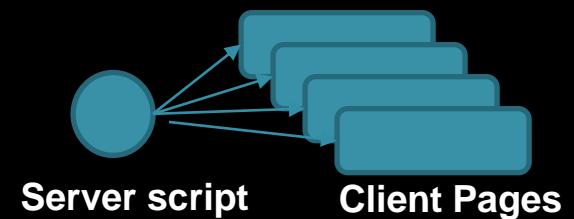
Web application frameworks

- Really web apps are developed using Web app frameworks
- A Web application framework is a software platform that provides developers with pre-built tools and libraries to simplify the development of web applications
 - for both front-end and back-end!
 - improve efficiency in web development!

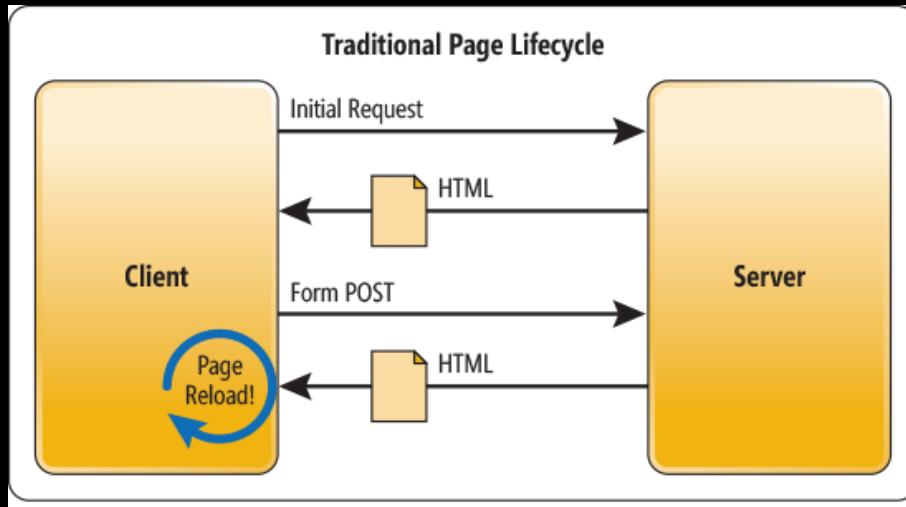


Dynamic pages

- **Client page is dynamic**
 - It can change itself at runtime
 - HTML can be modified by JavaScript
- **Server page is dynamic**
 - Client pages are constructed at runtime by a Server script
 - A same ‘server script’ can produce completely different client pages



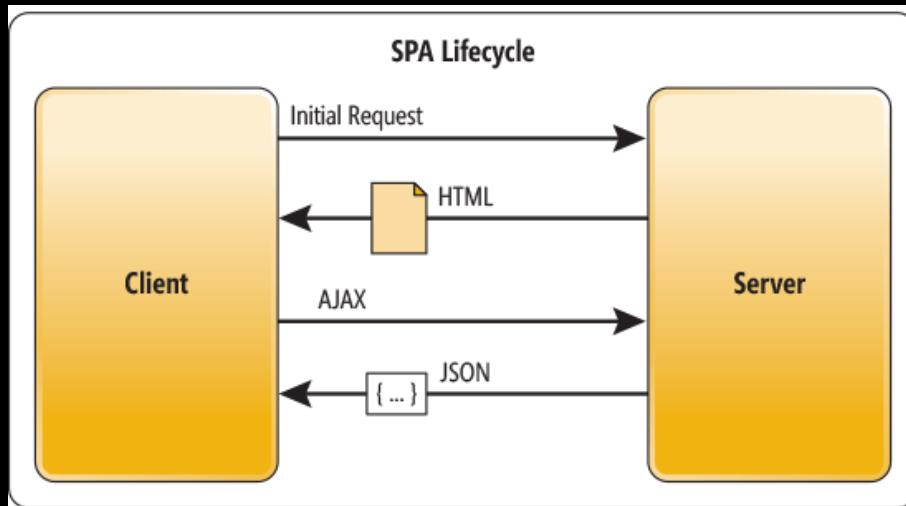
Multi-page vs. Single-page apps



Multi page Web app

Every time the client calls the server, the server returns a new HTML page. This triggers a page refresh in the browser

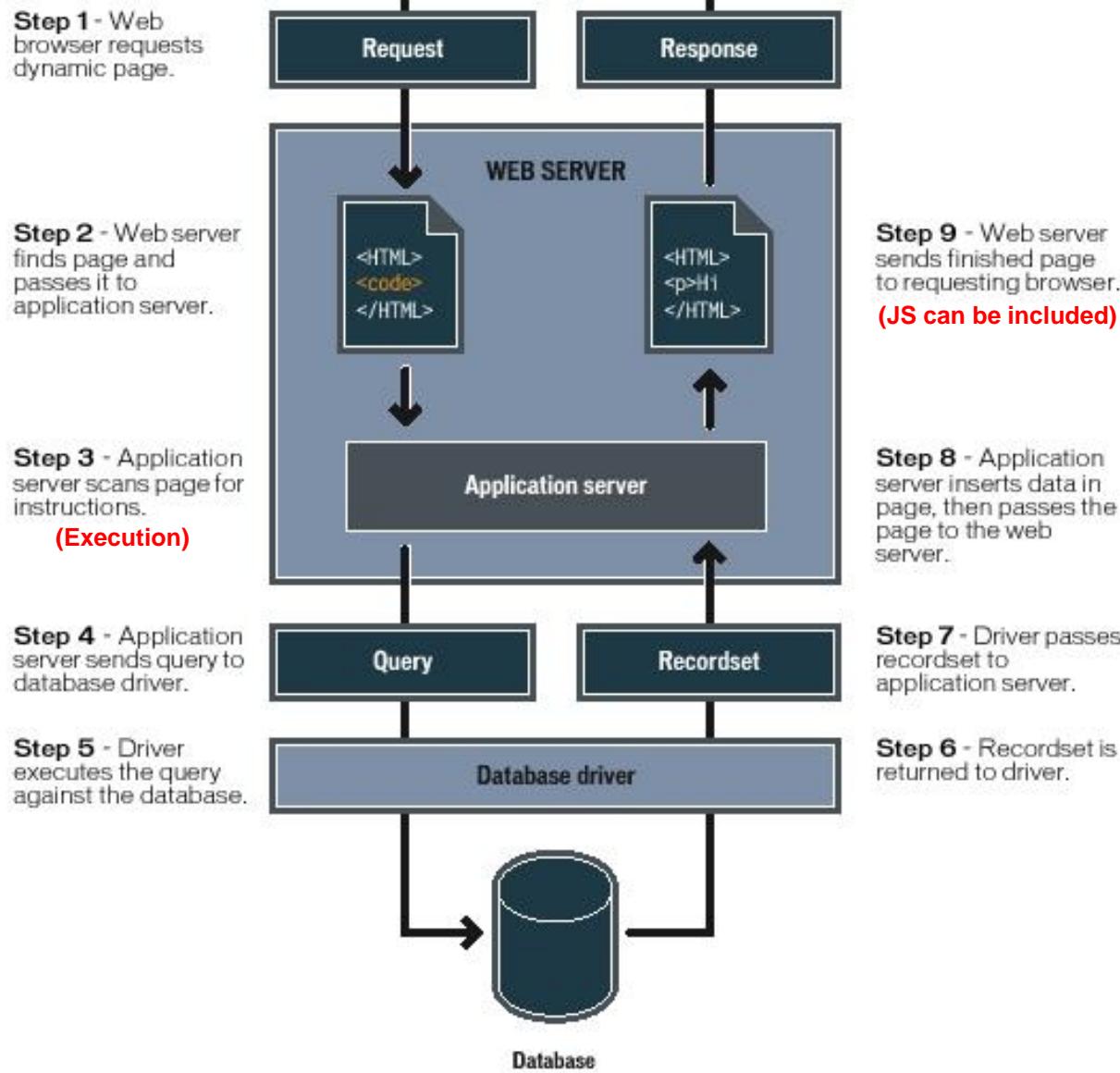
old-fashioned web apps!



Single-page Web app

After the first page loads, all interaction with the server happens through **AJAX calls** (next slide). The app uses JSON/XML data to update the page dynamically, without **reloading the page**

Multi-page Web app

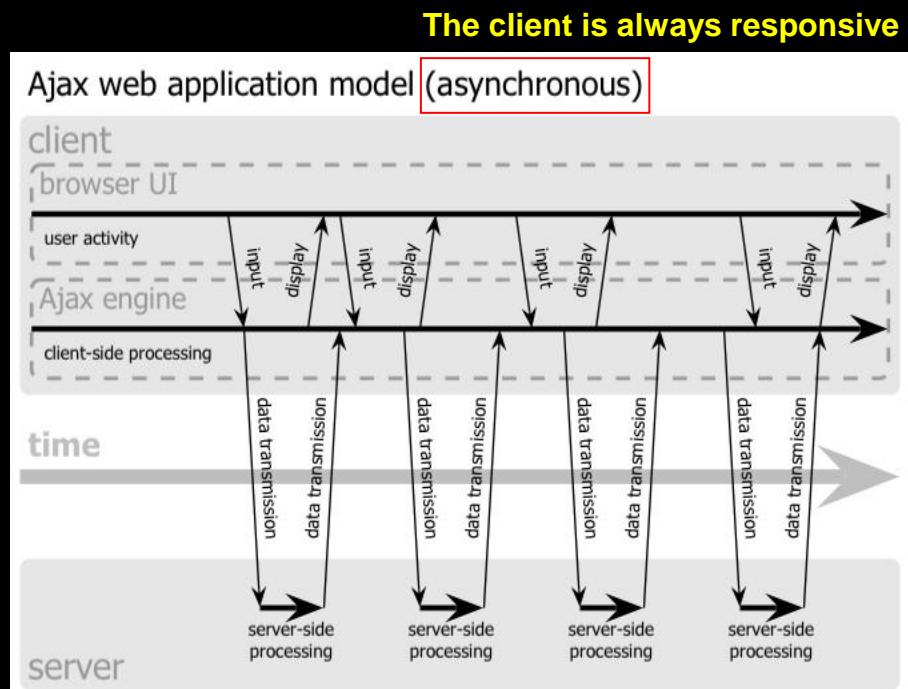
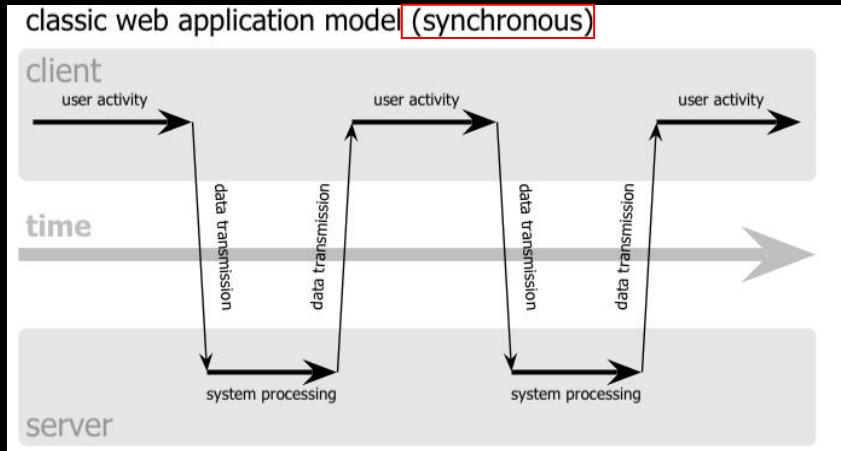


SPA app: AJAX

Ajax is the most prominent technique used for building SPA!

"Asynchronous JavaScript and XML" (AJAX)

- it is a bundle of existing technologies (HTML, JavaScript, XML, JSON, ...)
- it implements **asynchronous communication**
 - using, e.g., the XMLHttpRequest object
- it lets us update fragments (pieces) of a Web page



Browser doesn't interrupt the user interaction when is waiting the server's data

AJAX: How it works

by http://www.denverjug.org/meetings/files/200510_AJAX.ppt

Client page

Server Page

Name Validator

```
<script type="text/javascript">
```

XMLHttpRequest

```
function handleResponse() {  
...read response from server  
...get a handle to the DIV  
...replace DIV contents with msg  
}
```

```
function checkName(name) {  
...initialize request url, parms  
...register callback handleResponse()  
...build XMLHttpRequest + call server  
}
```

steve

Save

Not a Valid Name

Message:
{"name": "steve"}
Or
<name>Steve</name>

Validation Server Script

```
ValidateName(req, res) {  
parse request  
lookup name in database  
is it valid?  
if so, return OK  
otherwise  
return error message
```

Message:
msg=Not a valid name

Database

Web Page rendered

Client (Browser)

Server

SPA vs MPA

https://www.youtube.com/watch?v=Kg0Q_YaQ3Gk

(To summarize) Peculiarities

- **Heterogeneous and complex systems**
 - Web applications require use of **multiple languages, technologies** and **Web apps Frameworks**



- **Dynamism**
 - **highly dynamic and compelling UI!**
 - fetching server data without page refreshes
 - animating and manipulating page contents on-the-fly
 - the line between Web and “Desktop” apps is blurry ...



SPAs have advanced, sophisticated, and interactive GUIs!

User vs. Software Tester

SPAs have advanced and responsive UI!

User



Very good ... Happy!
(rich user experience)

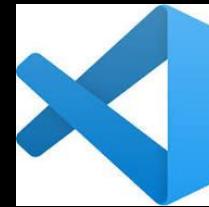
Software Tester



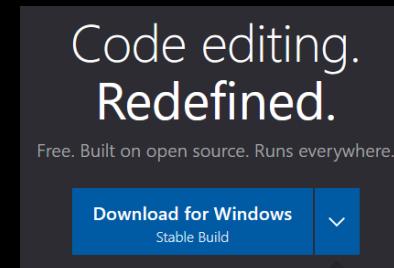
Problems!!

GETTING STARTED WITH JAVA IN VS CODE

VS Code



- *Visual Studio Code* is a **code editor** developed by **Microsoft** optimized for building and debugging modern **web and cloud applications**
- Cross-platform and language-agnostic
- It takes back the potential of Visual Studio by making it **lightweight**
- Visual Studio Code was born in 2015, bringing Microsoft closer to the **open-source** world
- Code available at this link
 - <https://github.com/microsoft/vscode>



'Offered' features

- **Syntax Highlight**
 - Code in 'different colours'
- **IntelliSense**
 - Intelligent code completion, parameter info, quick info
- **GIT/GITHUB integrated**
 - Control Version system integrated
- **Integrated Debugger**
 - Error detection and correction with breakpoint definition
- **Extensions (plug-in)**
 - Ability to install additional extensions easily and quickly

A screenshot of a Java code editor window titled "untitled". The code is as follows:

```
1 public class Main {  
2     // with/without access modifier:  
3     public  
4         Set<Map.Entry<K, V>> entrySet0;  
5         Set<Map.Entry<K, V>> entrySet1;  
6     }  
7     public static void main(String[] args) {  
8         Set<Map.Entry<K, V>> entrySet0;  
9             Node<K, V> m = someNode;  
10            Node<K, V> first = (Node<K, V>) someValue;  
11        }  
12        public static void method() {  
13            final int MY_CONSTANT = 0;  
14            int a = MY_CONSTANT + 1;  
15            int A = 1, B = 2;  
16            boolean less = A < B;  
17        }  
18    }  
19}  
20}
```

The code editor shows syntax highlighting for Java keywords (e.g., `public`, `class`, `void`) and variable types (e.g., `String`, `Set`, `Map`). It also highlights comments and strings. The status bar at the bottom indicates "Line 7, Column 17", "Spaces: 4", and "Java".

Operating Systems (OSs)

- Visual Studio Code is natively supported on **Linux**, **Microsoft**, and **macOS** environments

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

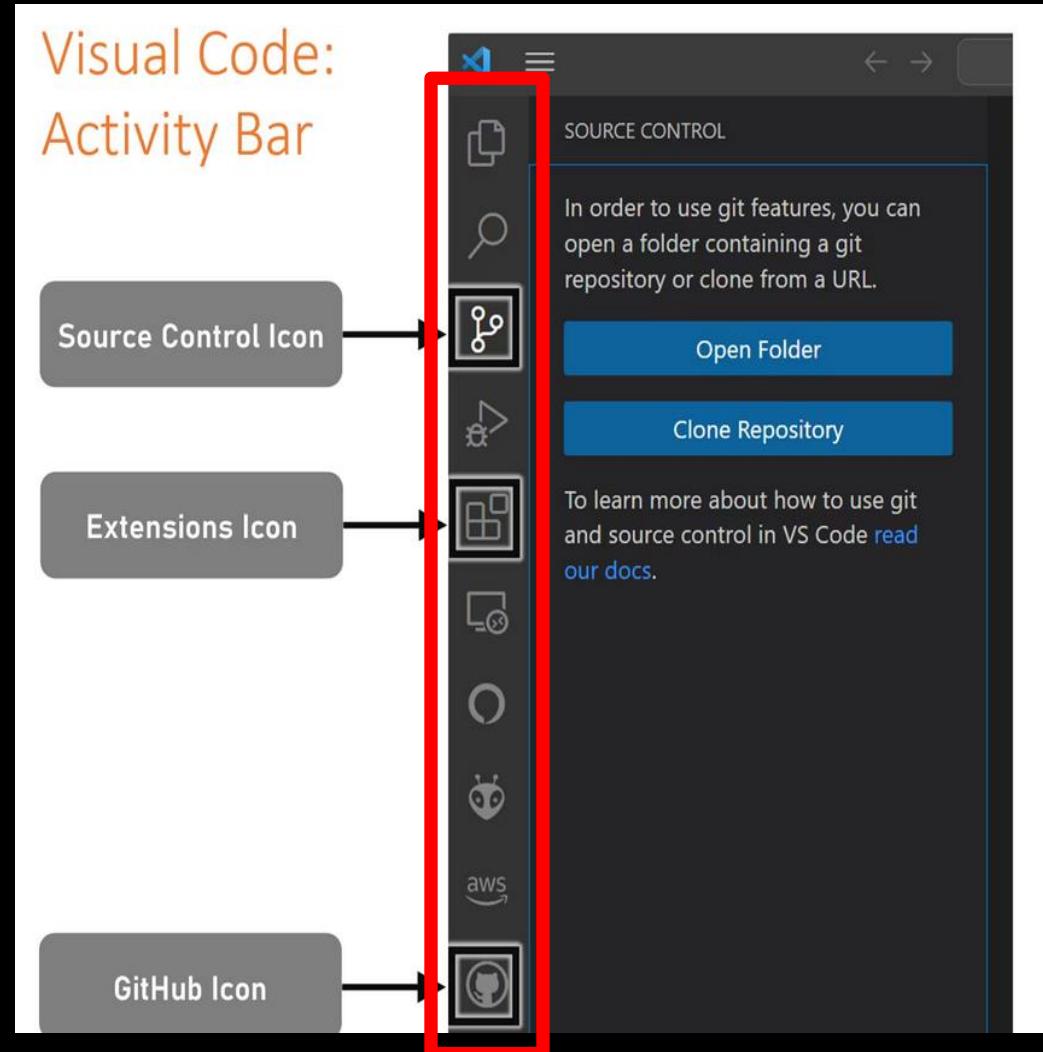
The screenshot shows the official Visual Studio Code download page. It features three main download sections: Windows (with icons for 64-bit, 32-bit, and ARM), Linux (with icons for .deb, .rpm, and .tar.gz), and macOS (with icons for .zip and Intel Chip/Apple Silicon). Each section includes a 'Snap Store' link at the bottom.

| Platform | File Types | Architectures |
|----------|--|--------------------------------------|
| Windows | User Installer, System Installer, .zip | 64 bit, 32 bit, ARM |
| .deb | .deb, .rpm, .tar.gz | 64 bit, ARM, ARM 64 |
| .rpm | .deb, .rpm, .tar.gz | 64 bit, ARM, ARM 64 |
| .tar.gz | .deb, .rpm, .tar.gz | 64 bit, ARM, ARM 64 |
| Mac | .zip | Universal, Intel Chip, Apple Silicon |

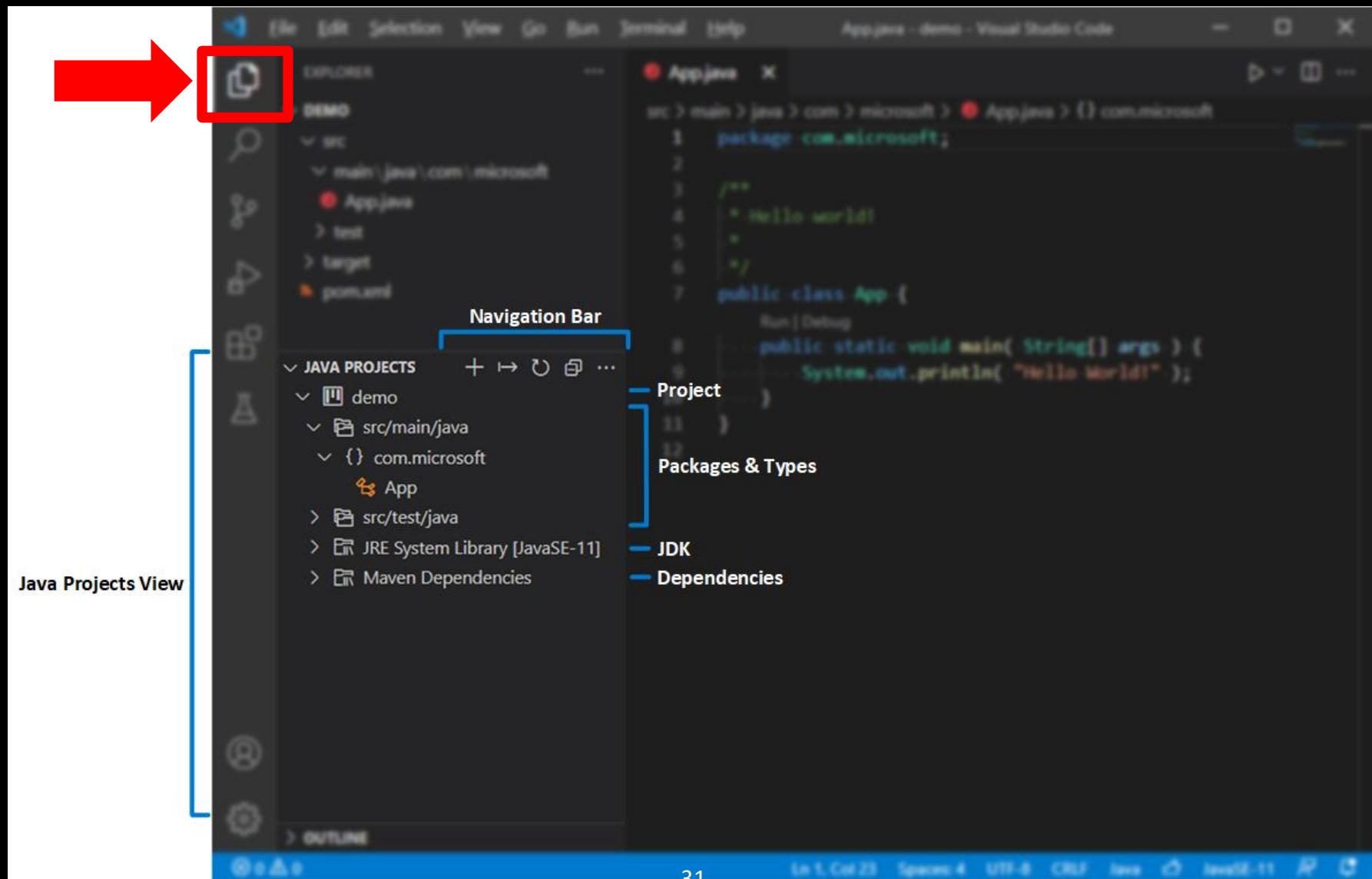
<https://code.visualstudio.com/Download>

Activity Bar

- The Activity bar is the **main navigation area** in VS Code
 - On the left when VS Code is open
- It determines the actions that can be carried out
- Extensions can contribute to the Activity bar

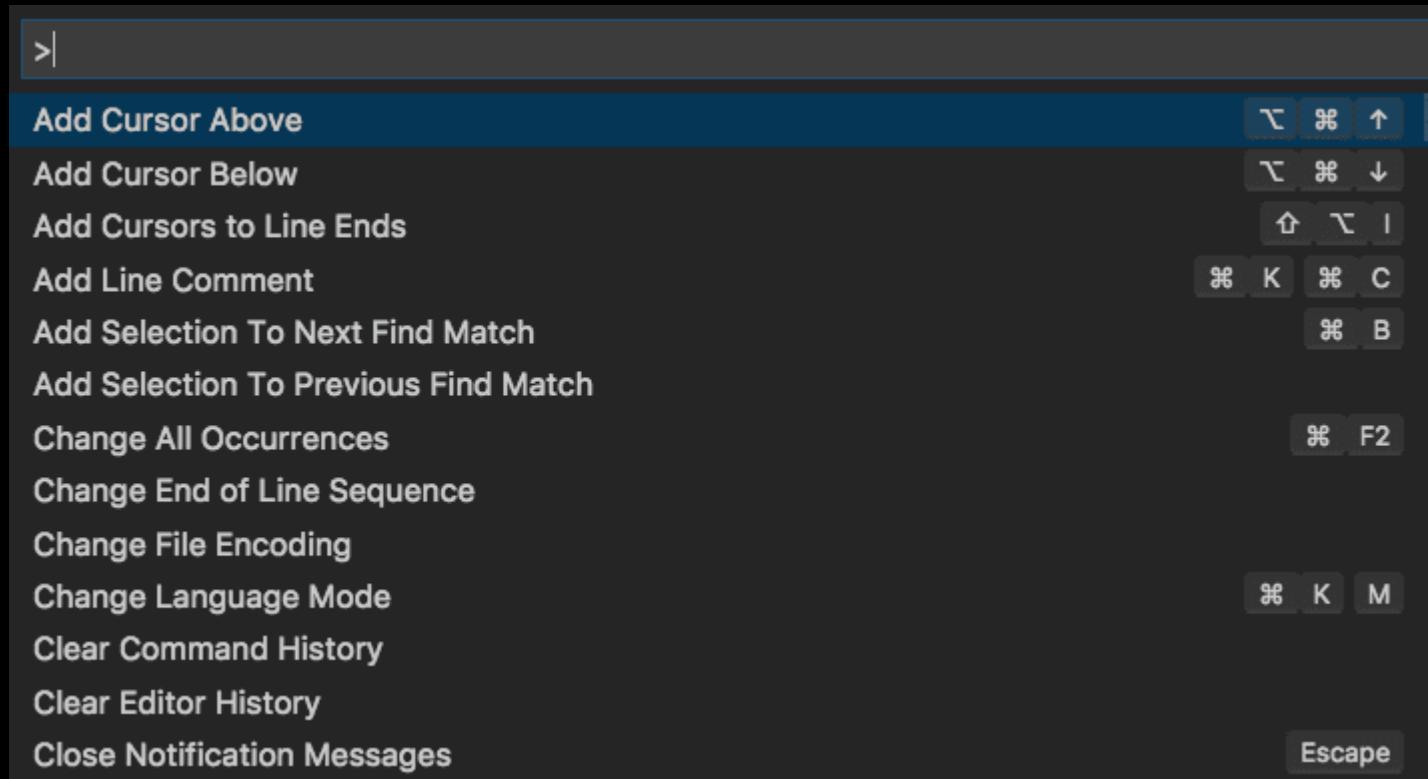


Projects view (Java)



Command Palette

- The most important key combination to know is **Ctrl+Shift+P**, which brings up the **Command Palette**
- From here, you have **access to all of the functionality of VS Code**, including keyboard shortcuts for the most common operations

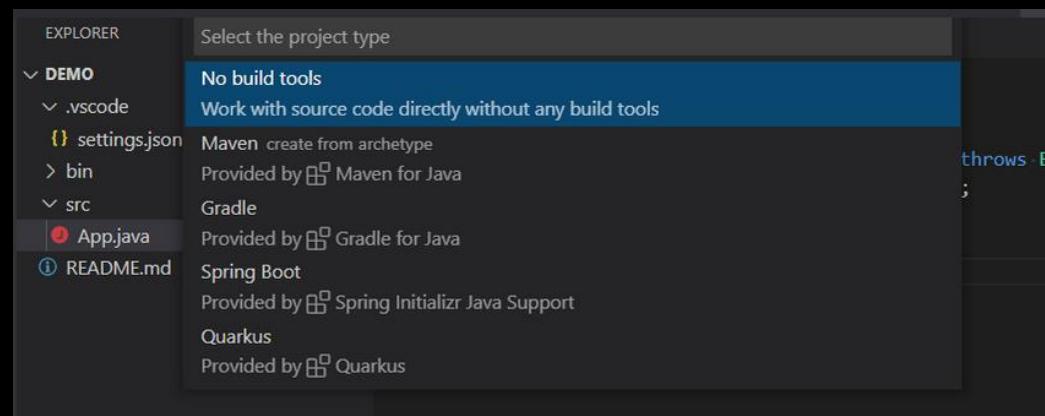


Download and install VS CODE

- Go to: <https://code.visualstudio.com/download>
 - Select OS and download executable
 - Double click
 - Select all default settings
 - Select look&feel GUI
 - VS code should appear
- Goto Java-tutorial page
 - <https://code.visualstudio.com/docs/java/java-tutorial>
- Install **Extension pack for Java**
 - If not already present, it Installs new JDK
 - download JDK 17.0.4 (contains also JRE)
- Check that Java runtime is correctly installed
 - Open a novel terminal (Terminal --> new terminal)
 - Type 'Java –version'
- To change the language:
 - Ctrl+shift+p --> Configure Display language

First project: HelloWorld

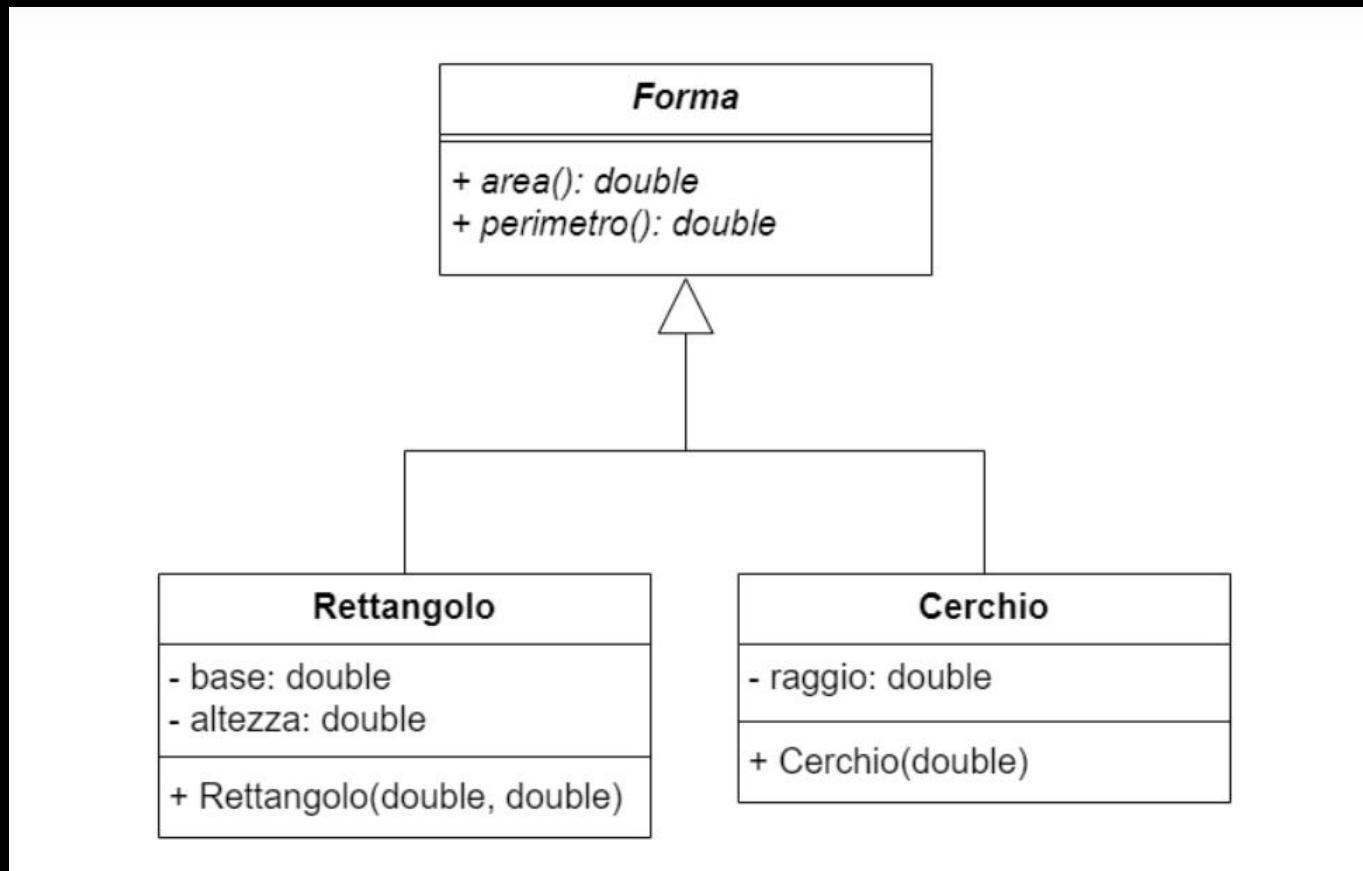
- **Ctrl+Shift+P** (Command palette)
- **Select ‘Java: Create Java Project’**
- Select a folder on your laptop
 - Where you want to insert the project
- Insert the name of the project
 - es. HelloWorldProject
- Select ‘No build tools’



- Open in VS code the folder ‘src’, you should find App.java
- Execute App.java with ‘Run without Debugging’ (CTRL+F5)

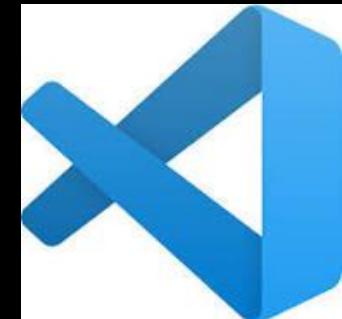
Demo

- I will write the code and you could try to replicate it on your laptop!



To Recap

- **Basics of Web applications**
 - Architecture: client-server
 - DOM manipulation
 - Multi-page apps vs Single-page apps
 - Ajax
 - Very dynamic apps: difficult to test!
- **VS CODE**
 - Code editor for Web development
 - Installation
 - Features
 - Simple demo



Functional and Security Testing Course

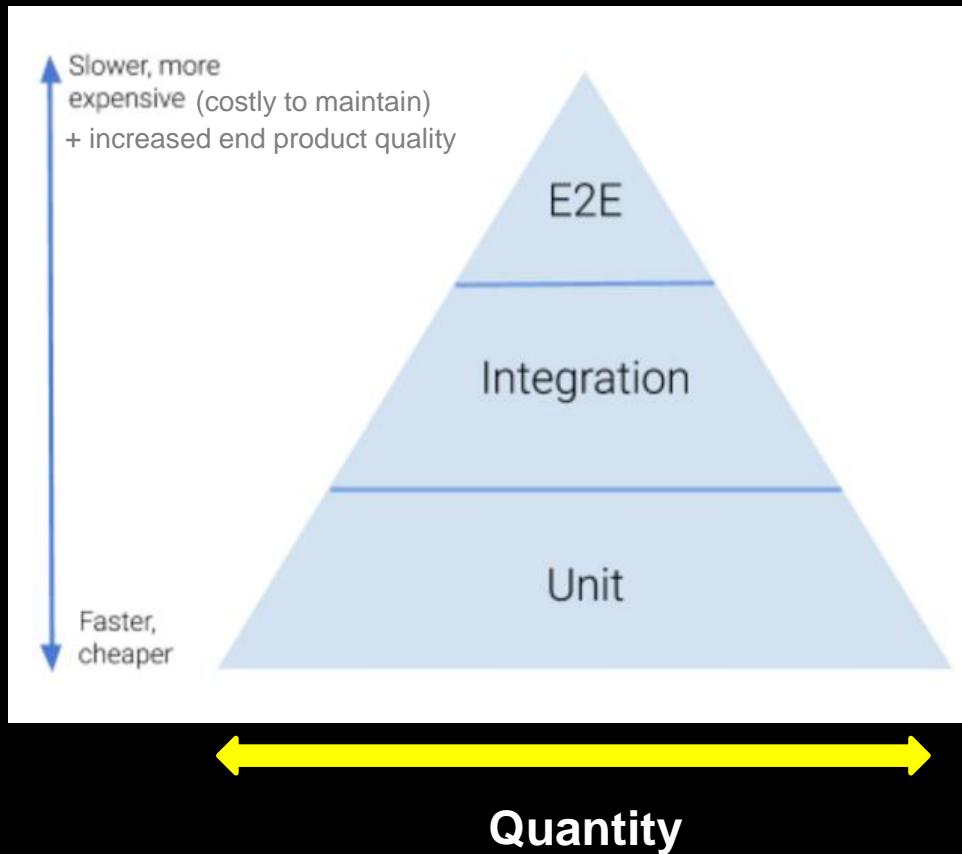
2023-2024

Unit Testing and JUnit – lesson 3

Filippo Ricca
DIBRIS, University of Genova

Levels of Software Testing

Test Pyramid Rule



Testing the system as a whole (GUI)



Individual units are combined and tested as a group



Testing of a single function/class



In theory, the distribution of test cases in a project should resemble a pyramid

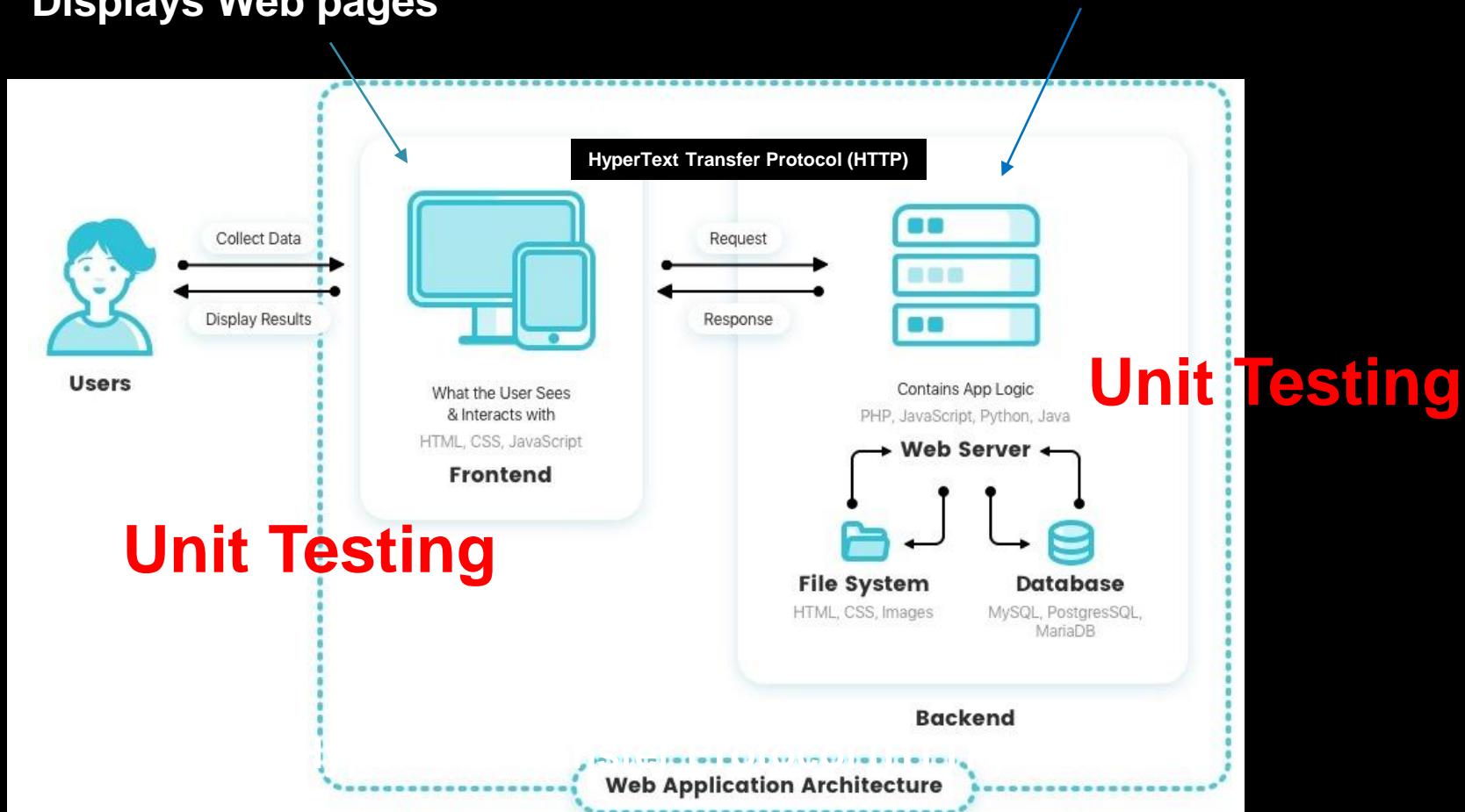
Have a Client-server architecture

Web applications and Unit Testing

- Hosts a browser
- Generates requests
- Displays Web pages

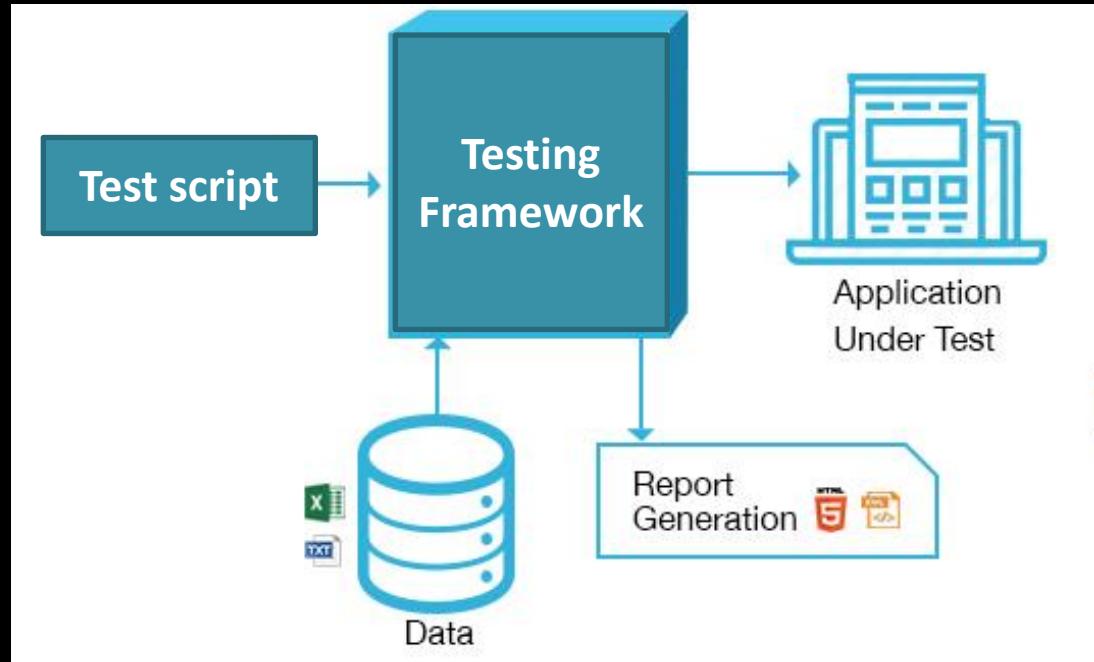


Generates Web pages



Testing Framework

'Data driven' test scripts

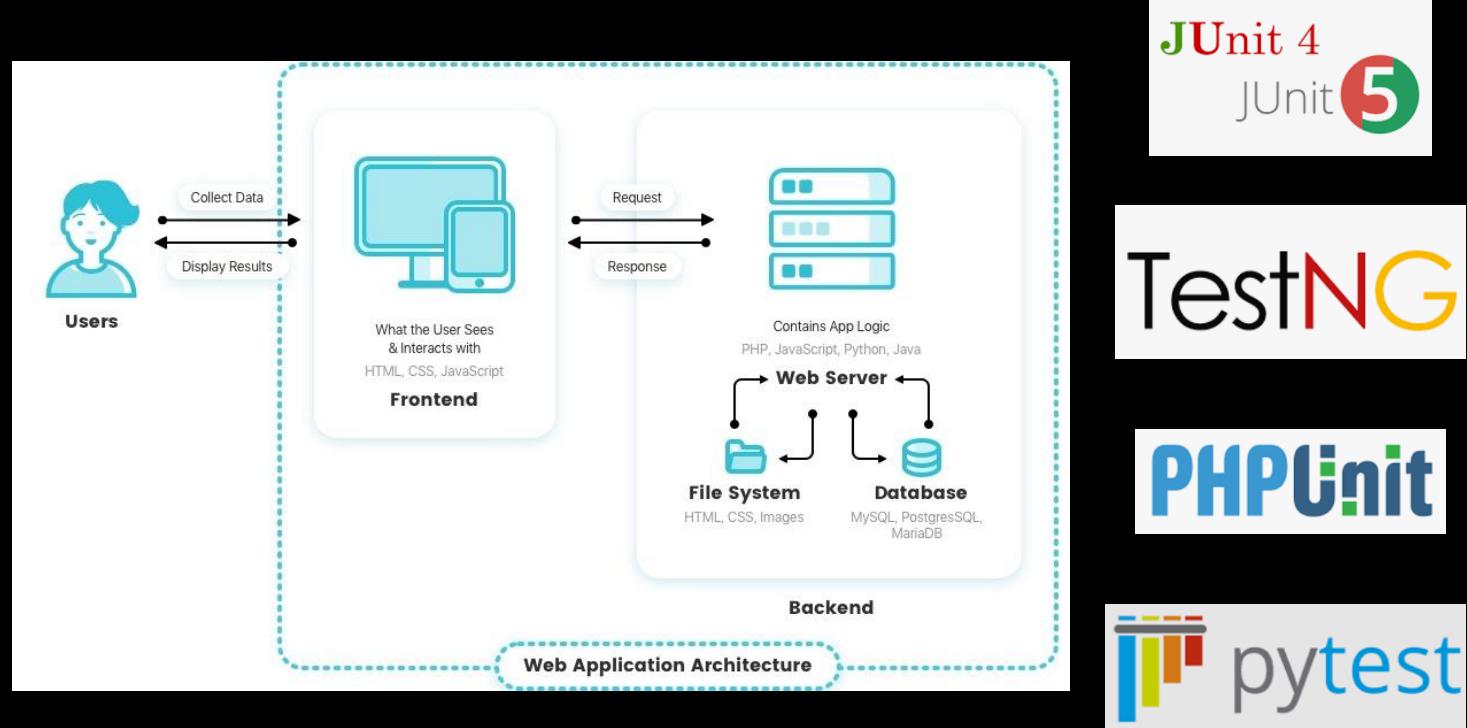


Testing Framework:

- able to 'interpret' and execute Test scripts (and Test suite)
- able to exercise AUT
- able to read data from a repository (data driven / parametric tests)
- able to generate a Test report (pass/fail)

Web applications

For each language we need a specific Testing Framework!!



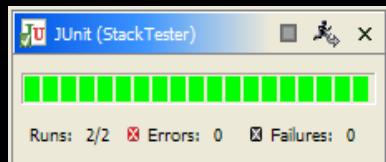
Fortunately, these frameworks are all similar...

...

BASICS OF JUNIT 5 (ALSO JUNIT 4)

What is JUnit?

- JUnit is a **testing framework** for Java code
 - *Unit testing*
 - Framework Java = set of classes and convention to use them
 - JUnit uses annotations (@Test)
- Developed by:
 - *Erich Gamma*
 - *Kent Beck*
- Integrated in **Eclipse**, **VS code** e **IntelliJ IDEA** through graphical plug-ins

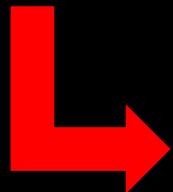


```
7  
8  public class SumTest {  
9  
10     @Test  
11     public void testSum() {  
12         assertEquals(expected: 5, Ca  
13     }
```



Simple Calculator project

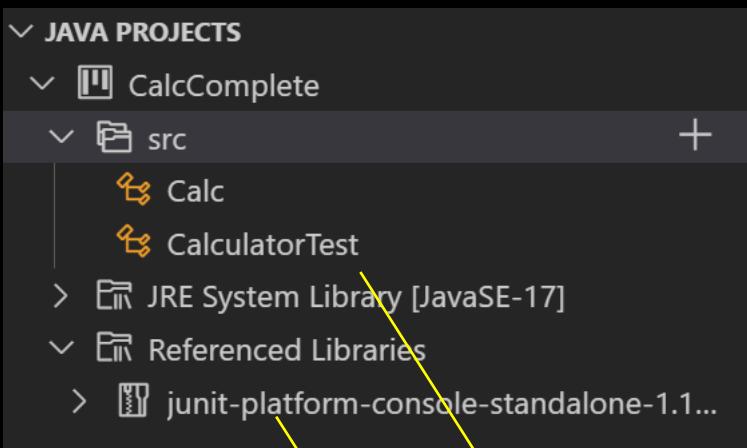
- Let's consider a **VS CODE** project with a class **Calc** (a simple calculator) to test
 - sum() and sub()
- The project is available on Aulaweb
 - Go to Aulaweb and download **Calc.zip**
 - Unzip archive file
 - In VS CODE
 - Open folder
 - Select archive file



```
public class Calc {  
  
    public int sum(int a, int b) {  
        return a + b;  
    }  
  
    public int sub(int a, int b) {  
        return a - b;  
    }  
}
```

Calculator Project

(we want to develop a testsuite for it)



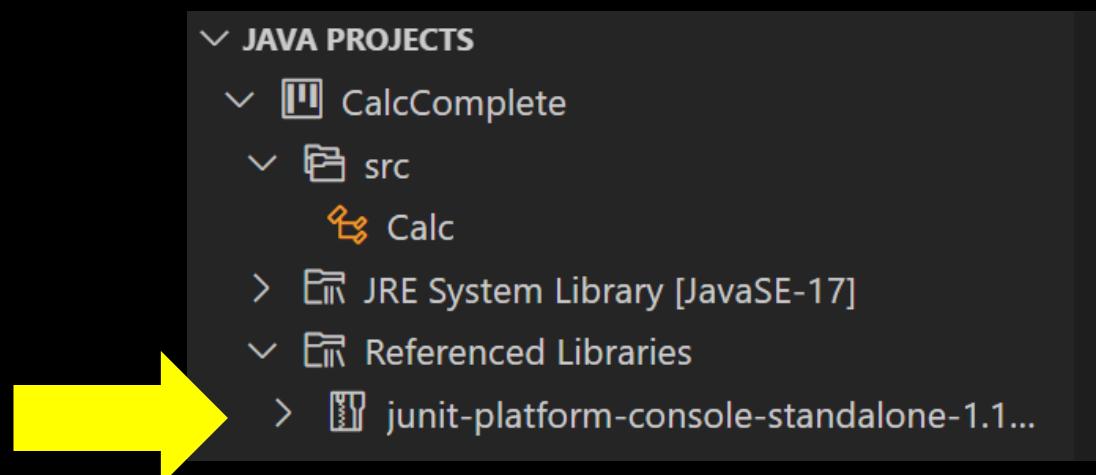
```
public class Calc {  
    public int sum(int a, int b) {  
        return a + b;  
    }  
  
    public int sub(int a, int b) {  
        return a - b;  
    }  
}
```

JUnit test class we want to create

Jar files of the JUnit 5 testing framework

JUnit test class

- Let's create a test class (**CalculatorTest**)
 - Check that the **Test Runner for Java** is installed
 - Go to the Market place () and search it
 - Click on Testing ()
 - Select 'Enable Java Tests'
 - Select 'JUnit Jupiter'
 - Open JAVA PROJECTS view

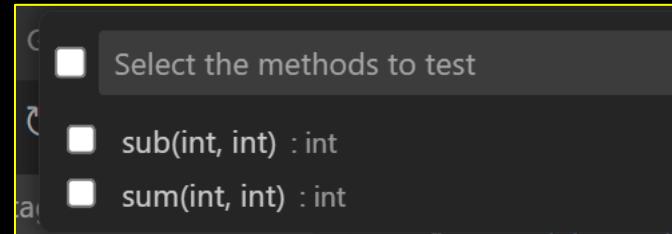


Test Methods selection

- On the ‘opened’ Calc class
 - Right-click on the Mouse and select ‘Source actions’
 - Select ‘Generate tests’
 - Select JUnit 5 (because library Junit Jupiter contains JUnit 4 and 5)
 - Insert the name of the Class ‘CalculatorTest’
 - Select the methods to test

- both

```
import org.junit.jupiter.api.Test;  
public class CalculatorTest {  
    @Test  
    void testSub() {  
    }  
  
    @Test  
    void testSum() {  
    }  
}
```



Wizard result

```
import org.junit.jupiter.api.Test;  
  
public class CalculatorTest {  
    @Test  
    public void testSum() {  
        }  
        @Test  
        public void testSub() {  
            }  
    }
```

JUnit framework import (classes and annotations)

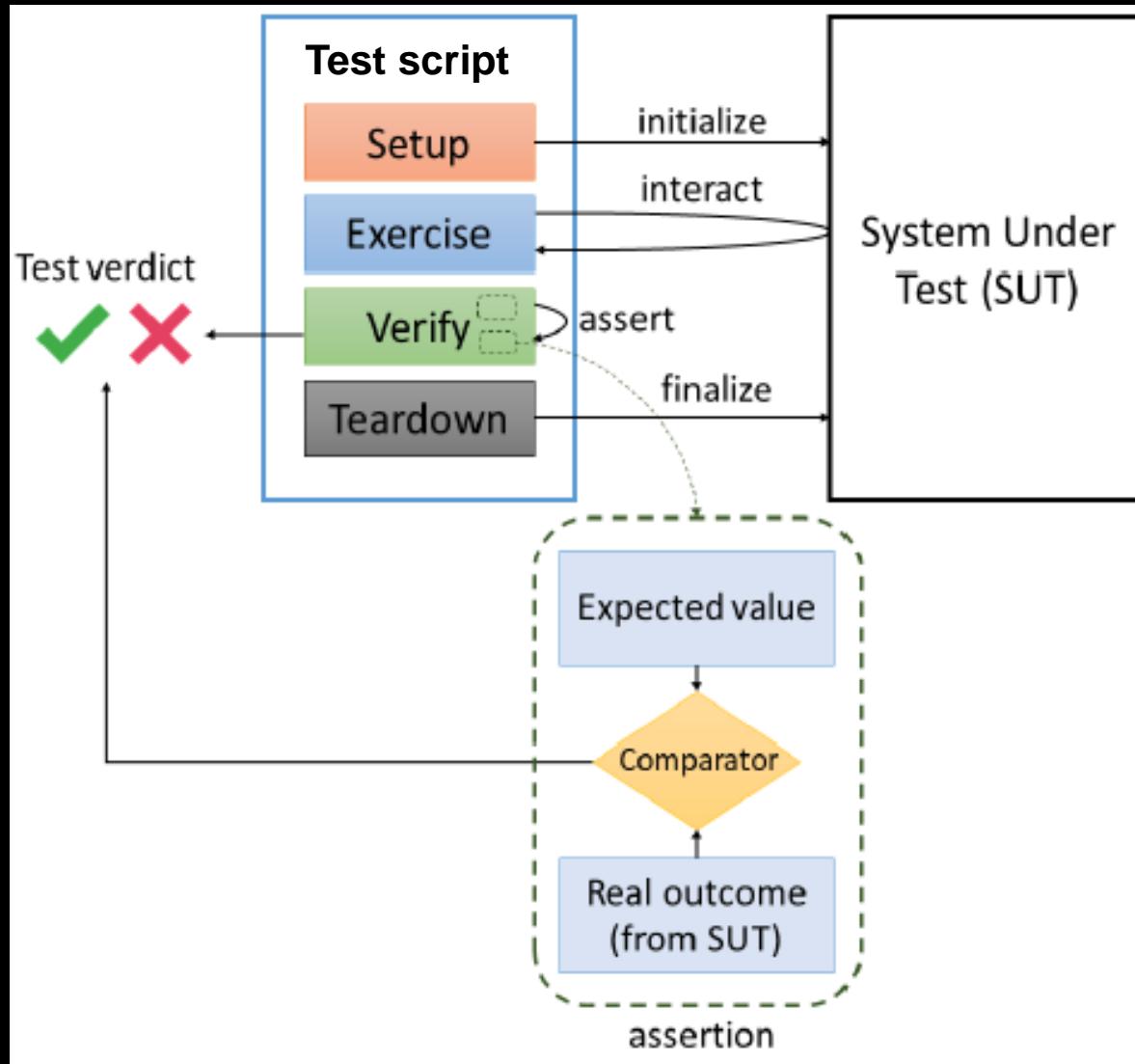
Test class name

It is an annotation. It indicates that the method is a test method

Test method name

Body of the method. Here it is possible to insert the test code

Test script structure



CalculatorTest (our first test class)

```
import static org.junit.jupiter.api.Assertions.*;  
import org.junit.jupiter.api.Test;
```

```
public class CalculatorTest {
```

```
    Calculator c;
```

```
@Test
```

```
public void testSum() {
```

```
    c = new Calcolator();
```

Setup

```
    System.out.println("testSum");
```

```
    assertEquals(3, c.sum(2, 1));
```

Verify  assert

```
}
```

Exercise

```
@Test
```

```
public void testSub() {
```

```
    c = new Calcolator();
```

Setup

```
    System.out.println("testSub");
```

```
    assertEquals(3, c.sub(6, 3));
```

Verify  assert

```
}
```

Exercise

```
}
```

JUnit test case execution

Test class execution

Test script execution



```
1 import static org.junit.Assert.assertEquals;
2 import org.junit.jupiter.api.Test;
3 public class CalculatorTest {
4     Calc c;
5
6     @Test
7     void testSum() {
8         c = new Calc();
9         System.out.println(x:"PrintSum");
10        assertEquals(3, c.sum(a:2,b:1));
11    }
12
13    @Test
14    void testSub() {
15        c = new Calc();
16        System.out.println(x:"PrintSum");
17        assertEquals(0, c.sub(a:6,b:3));
18    }
19 }
20
```

Test Runner result

Green passed
Red failed

```
1 import static org.junit.Assert.assertEquals;
2 import org.junit.jupiter.api.Test;
3 public class CalculatorTest {
4     Calc c;
5
6     @Test
7     void testSum() {
8         c = new Calc();
9         System.out.println("PrintSum");
10        assertEquals(3, c.sum(a:2,b:1));
11    }
12
13    @Test
14    void testSub() {
15        c = new Calc();
16        System.out.println("PrintSum");
17        assertEquals(0, c.sub(a:6,b:3)); Expected [0] but was [3]
```

Expected [0] but was [3] testSub()

| Expected | Actual |
|----------|--------|
| -0 | +3 |

↑ ↓ ⏪ | ⏴ ⏵ ×

✗ ✗ Test run at 3/3/202...

✗ ✗ testSub()

Expected [0] but ...
java.lang.Assertion...

Test Runner view

TESTING

Filter (e.g. text, !exclude, @tag)

✖ 1/2 896ms ⚡

- ✓ ✖ CalcComplete 28ms
 - ✓ ✖ <Default Package> 28ms
 - ✓ ✖ CalculatorTest 28ms
 - ✓ testSum() 2.0ms
 - ✖ testSub() 26ms

Failure vs Error in VS Code

Green passed
Red failed

Failure



```
3/4 tests passed (75.0%)
    ✓ (X) SumNumbers 8.0ms
        ✓ (X) {} test 8.0ms
            ✓ (X) SubTest 5.0ms
                ✘ (X) testSub() 5.0ms
                    > (✓) SumTest 3.0ms
```

```
6 import code.calculator;
7
8 public class SubTest {
9     @Test
10    public void testSub() {
11        assertEquals(expected: 1, Cal
java.lang.AssertionError: expected:[1] but was:[3] at test.Su
java.lang.AssertionError: expected:[1] but was:[3]
at test.SubTest.testSub(SubTest.java:11)
```

Error



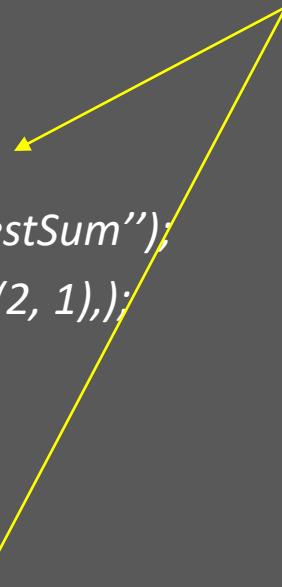
```
3/4 tests passed (75.0%)
    ✓ (O) SumNumbers 8.0ms
        ✓ (O) {} test 8.0ms
            ✓ (O) SubTest 6.0ms
                ✘ (O) testSub() 6.0ms
                    > (✓) SumTest 2.0ms
                        ✓ (✓) testSum() 2.0ms
                        ✓ (✓) testSumZero() 0.0ms
                        ✓ (✓) testSumNegative() 0.0ms
```

```
6 import code.calculator;
7
8 public class SubTest {
9     @Test
10    public void testSub() {
11        assertEquals(expected: 1, Cal
java.lang.ArithException: / by zero at code.Calculator
java.lang.ArithException: / by zero
at code.Calculator.sub(Calculator.java:15)
at test.SubTest.testSub(SubTest.java:11)
```

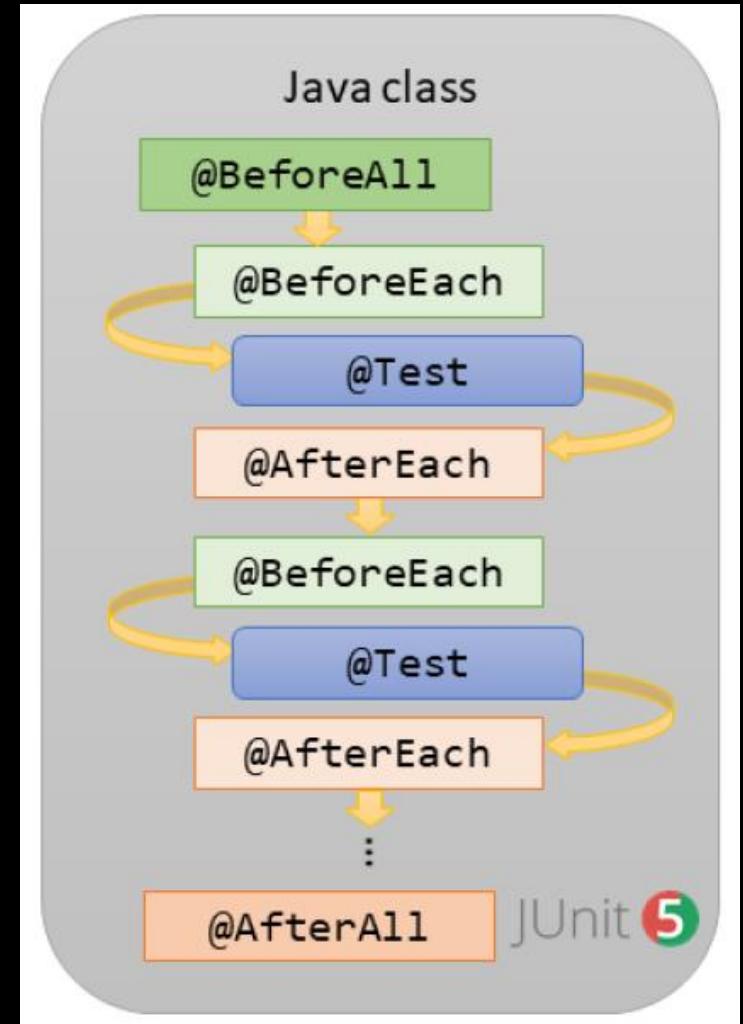
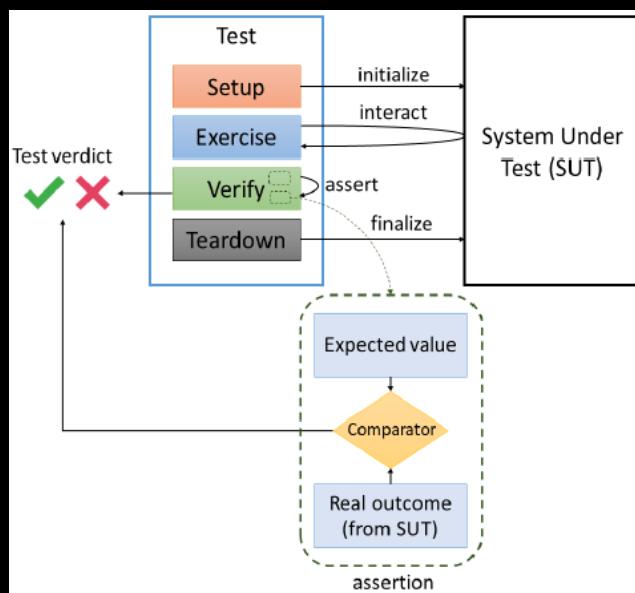
CalculatorTest

```
import static org.junit.jupiter.api.Assertions.*;  
import org.junit.jupiter.api.Test;  
  
public class CalculatorTest {  
    Calculator c;  
  
    @Test  
    public void testSum() {  
        c = new Calcolator();  
        System.out.println("testSum");  
        assertEquals(3, c.sum(2, 1));  
    }  
  
    @Test  
    public void testSub() {  
        c = new Calcolator();  
        System.out.println("testSub");  
        assertEquals(3, c.sub(6, 3));  
    }  
}
```

CODE DUPLICATION!



JUnit 5 ‘setUp’ and ‘Teardown’



BeforeEach / AfterEach

@BeforeEach is used to initialize the objects to test

- Method @BeforeEach called before each test method
- The state is the same for each Test Method

@AfterEach is used to release resources (objects)

- Method @AfterEach called after each Test Method

```
ShoppingCart cart;  
Book book1, book2;  
  
@BeforeEach  
protected void setUp() {  
    cart = new ShoppingCart();  
    book1 = new Book("JUnit", 29.95);  
    book2 = new Book("XP", 18.25);  
    cart.addItem(book1);  
    cart.addItem(book2);}
```

```
public class CalculatorTest {  
    @BeforeEach setUp() {...}  
    @AfterEach tearDown {...}  
  
    @Test testSum() {...}  
    @Test testSub() {...}  
    @Test testDivision(...) {...}  
}
```



setUp()
testSum()
tearDown()

setUp()
testSub()
tearDown()

setUp()
testDivision()
tearDown()

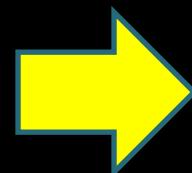
CalculatorTest

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class CalculatorTest {
    Calculator c;
    @BeforeEach
    Public void setUp(){
        c = new Calculator();
    }
    @Test
    public void testSum() {
        System.out.println("testSum");
        assertEquals(3, c.sum(2, 1));
    }
    @Test
    public void testSub() {
        System.out.println("testSub");
        assertEquals(3, c.sub(6, 3));
    }
}
```

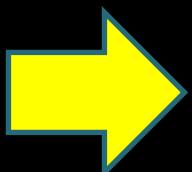
BeforeEach vs. BeforeAll

```
public class CalculatorTest {  
    @BeforeEach setUp() {...}  
    @AfterEach tearDown {...}  
    @Test testSum() {...}  
    @Test testSub() {...}  
    @Test testDivision(...) {...}  
}
```



```
setUp()  
    testSum()  
tearDown()  
setUp()  
    testSub()  
tearDown()  
setUp()  
    testDivision()  
tearDown()
```

```
public class CalculatorTest {  
    @BeforeAll setUp() {...}  
    @AfterAll tearDown {...}  
    @Test testSum() {...}  
    @Test testSub() {...}  
    @Test testDivision(...) {...}  
}
```



```
setUp()  
    testSum()  
    testSub()  
    testDivision()  
tearDown()
```

Assertions

Without AssertJ or Hamcrest

- Boolean condition
 - `assertTrue(boolean condition, "message for fail");`
 - `assertFalse(boolean condition);`
- For objects, int, long, boolean, strings, byte ...
 - `assertEquals(expected_value, actual, "message for fail");`
 - for objects is used the corresponding `equals method` (to define)
 - equals for array does not compare the content but the reference to arrays
- To compare arrays
 - `assertArrayEquals(expectet_value, actual);`
- For float e double values
 - `assertEquals (expected, actual, error);`
- Check if expected and actual refer to the same object
 - `assertSame(expected_value, actual);`
 - `assertNotSame(expected_value, actual);`
- Fail the test immediately
 - `fail ("message for fail");`

Tolerance

With more assertions

```
@Test  
public void testSumMoreAssert() {  
    Calculator c = new Calculator();  
    assertEquals(2, c.sum(1, 1));  
    assertEquals(0, c.sum(1, 0));  
    assertEquals(0, c.sum(0, 0));  
}
```



- If an **assertion is false**:
 1. test fails immediately (**red**)
 2. the rest of the method is skipped
 3. the test error message is printed (if any)
- If an **assertion is true**:
 1. execution continues normally
 2. if all the assertions are true the test method passes (**green**)

JUnit 5 provides also

Exception Testing

```
@Test  
public void testNullException() {  
    String aa = null;  
    assertThrows(NullPointerException.class, () -> {aa.length();});  
}
```

Timed Test methods

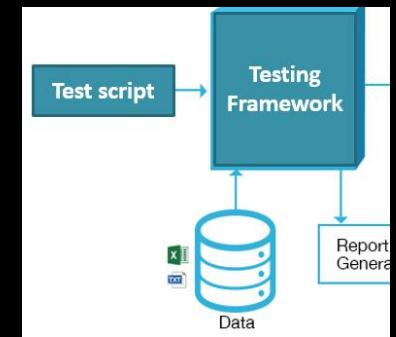
```
@Test  
public void doPaymentNotExceed15Seconds() {  
    OrderService orderService = new OrderService();  
    assertTimeout(ofSeconds(15), () -> {  
        orderService.doPayment();   
    });  
}
```

This method should run in 15 secs

Create a testsuite

```
@RunWith(JUnitPlatform.class)  
@SelectClasses({TestStack4.class, ParametrizedStackIntTest.class})  
public class TestAll {// nothing}
```

Parametric/Parameterized (data-driven) Test Methods



```
@ParameterizedTest  
@ValueSource(strings = { "Hello", "JUnit" })  
void testWithValueSource(String word) {  
    assertNotNull(word);  
}
```

| Test Results | | 26ms |
|--------------|-------------------------------|------|
| ▼ | ✓ HelloParams | 26ms |
| ▼ | ✓ testWithValueSource(String) | 26ms |
| ✓ [1] Hello | | 25ms |
| ✓ [2] JUnit | | 1ms |

Coverage

- The coverage of a test suite T measure **how much** of the program under test P is tested by the testsuite T
 - Important to evaluate a testsuite!
 - More is the coverage and more is ‘good’ T
- To measure the **Coverage** of T w.r.t P, we need to find:
 - An *unit of coverage* with the following two properties:
 - It is possible to count the **total number of units** in P
 - It is possible to identify **the units of P that are executed** during the execution of T
- That means that we can compute the **percentage** of executed units: **executed units / total units**



Code/Line coverage example

- Unit of coverage: **line of code**
 - It is possible to count the line of code of P
 - It is possible to count the lines executed
- TestSuite 1 = {(x=5)}
 - Coverage = $7/8 = 87,5\%$
- TestSuite 2 = {(x=5), (x=22)}
 - Coverage = $8/8 = 100\%$

```
1 scanf("%d", &x);
2 a = x + 1;
3 b = x - 1;
4 if (a < 10)
5   x++;
6 if (b > 20)
7   x--;
8 printf("%d\n", x);
```

```
1 scanf("%d", &x);
2 a = x + 1;
3 b = x - 1;
4 if (a < 10)
5   x++;
6 if (b > 20)
7   x--;
8 printf("%d\n", x);
```

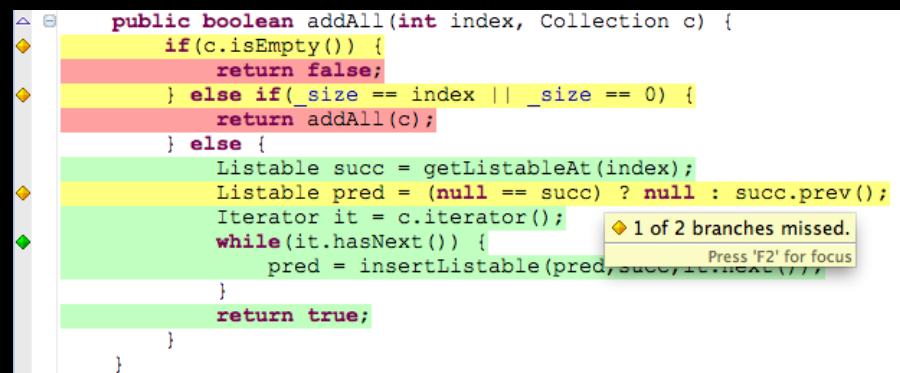
Executed

Testsuite 2 is better than Testsuite 1!

Java Code Coverage in Eclipse

- Launching 
 - Select the test class
 - Coverage as
 - JUnit test
- Coverage View

Source code Coloured



```
public boolean addAll(int index, Collection c) {  
    if(c.isEmpty()) {  
        return false;  
    } else if(_size == index || _size == 0) {  
        return addAll(c);  
    } else {  
        Listable succ = getListableAt(index);  
        Listable pred = (null == succ) ? null : succ.prev();  
        Iterator it = c.iterator();  
        while(it.hasNext()) {  
            pred = insertListable(pred, succ, it.next());  
        }  
        return true;  
    }  
}
```

| Element | Coverage | Covered Lines | Missed Lines | Total Lines |
|--|----------|---------------|--------------|-------------|
| commons-collections | 80.7 % | 11092 | 2646 | 13738 |
| src | 80.7 % | 11092 | 2646 | 13738 |
| org.apache.commons.collections | 77.1 % | 3991 | 1188 | 5179 |
| org.apache.commons.collections.bag | 66.9 % | 234 | 116 | 350 |
| org.apache.commons.collections.bidimap | 91.2 % | 964 | 93 | 1057 |
| AbstractBidiMapDecorator.java | 85.7 % | 6 | 1 | 7 |
| AbstractBidiMapDecorator | 85.7 % | 6 | 1 | 7 |
| AbstractBidiMapDecorator(BidiMap) | 100.0 % | 2 | 0 | 2 |
| getBidiMap() | 100.0 % | 1 | 0 | 1 |
| getKey(Object) | 100.0 % | 1 | 0 | 1 |
| inverseBidiMap() | 0.0 % | 0 | 1 | 1 |

Green executed
Yellow partially executed
Red not executed

Java Code Coverage in VS Code

- It seems there are two options:
 - Using VS Code ‘insiders’
 - Another version of VS CODE
 - Recent option
 - Using the VS Code plug-in ‘Coverage Gutters’
 - Not easy procedure
 - Problems with versions
 - Maven building tool

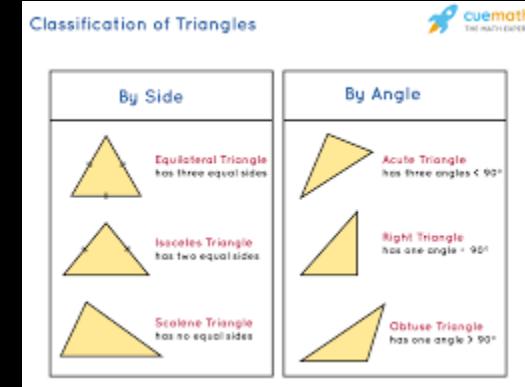


<https://devblogs.microsoft.com/java/java-on-visual-studio-code-update-december-2023/>

<https://medium.com/@karlrombauts/setting-up-unit-testing-for-java-in-vs-code-with-maven-3dc75579122f>

First Assignment (not mandatory)

- Designing and Implementing a **JUnit testsuite** for the **class Triangle** (you can find it on AW) trying to maximize the code coverage
- For the coverage there are three options:
 - Using VS Code ‘insiders’
 - Using plug-in ‘Coverage Gutters’
 - Computing manually the coverage
 - Easy task



JUnit 5

INTRODUCTION

- <https://www.youtube.com/watch?v=2E3WqYupx7c&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o>
- <https://www.youtube.com/watch?v=kp6ei9mEggc&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=2>
- <https://www.youtube.com/watch?v=deMlViVF-mE&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=3>
- <https://www.youtube.com/watch?v=1xRcRYKOTHs&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=5>
- https://www.youtube.com/watch?v=EROUlf2Ac_I&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=7
- <https://www.youtube.com/watch?v=e5uGwzz2I-s&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=8>
- <https://www.youtube.com/watch?v=jWIcKrxwaJU&list=PLqq-6Pq4lTTa4ad5JISViSb2FVG8Vwa4o&index=9>

USEFUL webinars

- <https://www.youtube.com/watch?v=1xRcRYKOTHs> - JUnit 5 architecture
- <https://www.youtube.com/watch?v=L5RA3KMLG5Y> - JUnit 5 assertions
- <https://www.youtube.com/watch?v=fpp6VgoEflk> - Assert methods
- <https://www.youtube.com/watch?v=Q29PFZhErUU> - Test exceptions
- <https://www.youtube.com/watch?v=RA619syDTps> - Life cycle and test antipatterns to avoid
- <https://www.youtube.com/watch?v=6ojQqqavyzY> - Using JUnit lifecycle hook annotations
- <https://www.youtube.com/watch?v=uzeFV-VzOmo> - @BeforeAll and @AfterAll

Functional and Security Testing Course

2023-2024

**An introduction to Generative AI
(ChatGPT and GitHub Copilot)
in Software Testing – lesson 4**

Filippo Ricca
DIBRIS, University of Genova

Generative AI

- **Generative artificial intelligence** (Generative AI, GenAI) is AI capable of generating text, images, code or other data in response to **prompts**
- Generative AI models **learn the patterns** and structure of their **input training data** and then **generate new data** that has similar characteristics
- Improvements in **deep neural networks** enabled an **AI boom** of generative AI systems
 - these include **large language model (LLM) chatbots** such as: **ChatGPT, Copilot, Gemini and LLaMA**

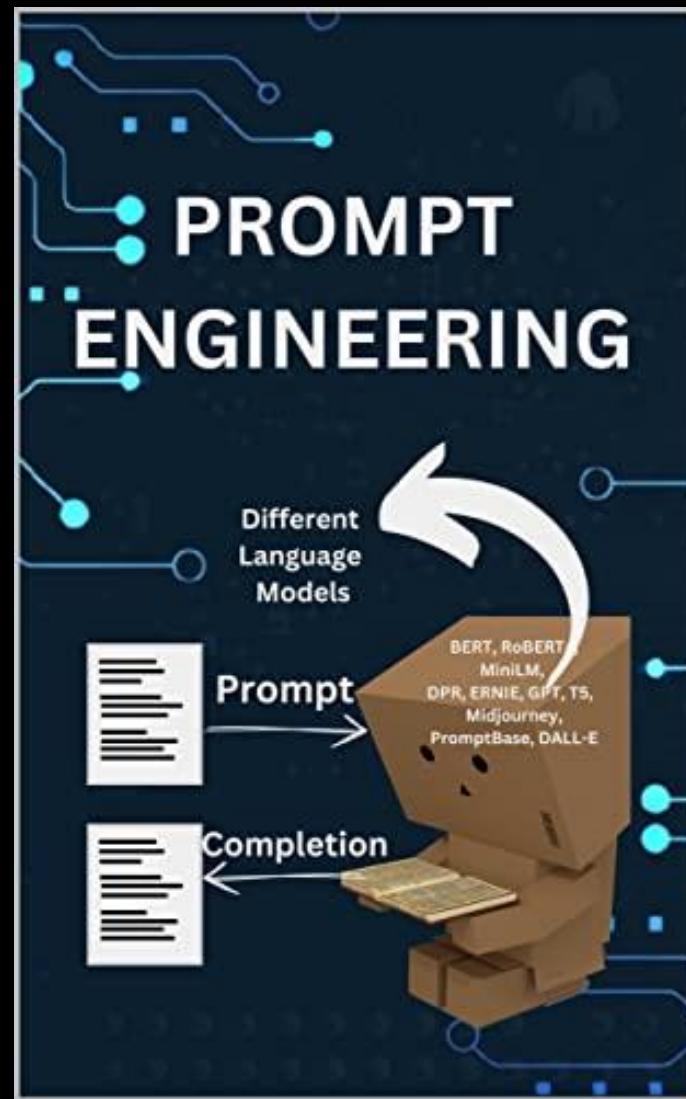


Prompt Engineering

- **Prompt engineering** is the process of structuring text that can be interpreted and understood by a **generative AI** model
- A **prompt** is **natural language text** describing the task that an AI should perform
- A prompt for a text-to-text language model can be:
 - a query such as "what is Fermat's little theorem?"
 - a command such as "write a poem about leaves falling"
 - or a longer statement including context, instructions, and conversation history

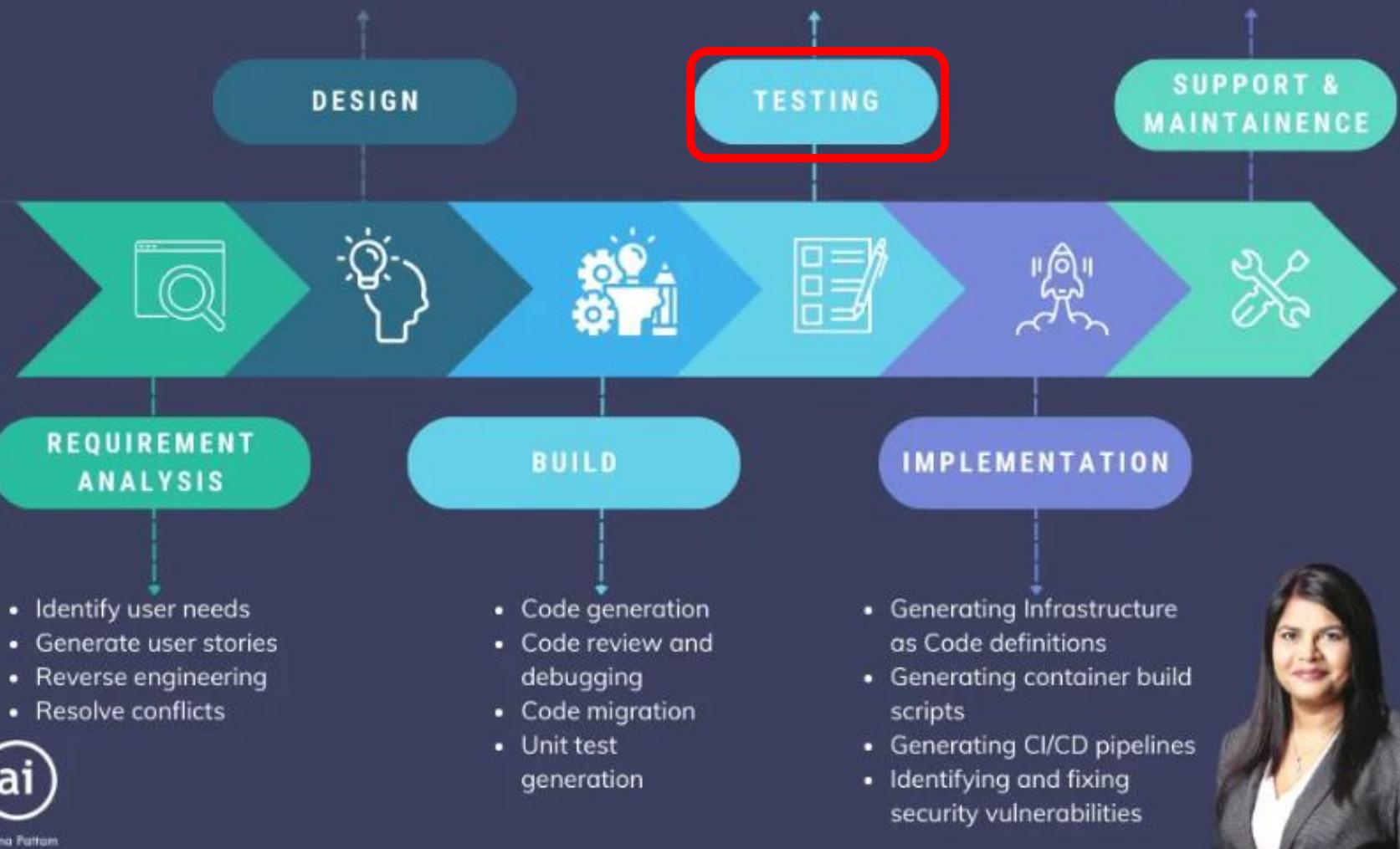


Give ‘good’ prompts: Not an easy task !



GENERATIVE AI IN SOFTWARE ENGINEERING

- Generate architecture diagrams
 - Select optimal technologies
 - Generate data models
 - Generate Wireframes
- Test case generation
 - Test data sets generation
 - Testing automation
 - Root cause analysis
- Anomaly Detection & Alerting
 - Feedback Analysis & Bug Prioritization
 - Incident Analysis & Problem Resolution
 - AI-Powered Support Tools



Aruna Patam



Generative AI in Software Testing

How can Generative AI help with Software Testing?



ChatGPT



ChatGPT

Generative AI enhances software testing by automating various tasks:

1. **Test Data Generation:** Creates diverse test data for thorough testing.
2. **Test Case Generation:** Generates test cases based on software specifications.
3. **Fuzz Testing:** Automates the generation of random inputs to uncover bugs.
4. **Regression Testing:** Automatically generates test cases to check for regressions.
5. **Automated Test Oracles:** Creates expected outputs for automated testing.
6. **Exploratory Testing Support:** Suggests test scenarios for exploratory testing.

Test cases
Test scripts

These capabilities streamline testing, increase coverage, and improve defect detection, but human expertise remains crucial for interpretation and validation.



ChatGPT (test script generation)

- **Prompt:** Could you write a JUnit test method for the following Java class?

```
public class BankAccount {  
    private int balance = 0;  
  
    public BankAccount(int startingBalance) {  
        this.balance = startingBalance;  
    }  
  
    public boolean withdraw(int amount) {  
        if(balance >= amount) {  
            balance -= amount;  
            return true;  
        }  
        return false;  
    }  
  
    public int deposit(int amount) {  
        balance += amount;  
        return balance;  
    }  
  
    public int getBalance() {  
        return balance;  
    }  
}
```

The suggestion



ChatGPT

Sure, here's a JUnit test method for the `BankAccount` class:

java

Copy code

```
import org.junit.Test;
import static org.junit.Assert.*;

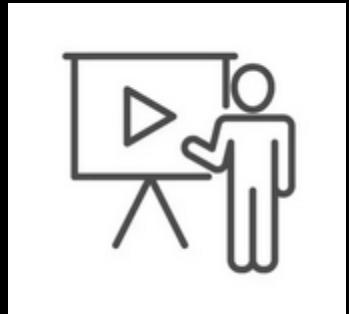
public class BankAccountTest {

    @Test
    public void testWithdrawSufficientFunds() {
        BankAccount account = new BankAccount(100); // Creating a bank account with :
        assertTrue(account.withdraw(50)); // Withdraw 50, should return true
        assertEquals(50, account.getBalance()); // Check if balance is updated correctly
    }

    @Test
    public void testWithdrawInsufficientFunds() {
        BankAccount account = new BankAccount(100); // Creating a bank account with :
```

Some refinements

- Please, refactor the code in **JUnit 5**, this is **JUnit 4**
- There is a lot of **code duplication**, Could you use `@BeforeEach` to improve the code?
- Assertions are really simple, could you convert them using '**AssertThat**'?
- Could you generate **parameterized test methods**?
- How much is the code coverage of the test suite you generated?
- Could you consider some **edge and corner cases** and generate for them some test methods?
- ...



Quick demo

ChatGPT: Test and Data Generation

ChatGPT (test script generation)

- **Prompt:** Can you write an **E2E Selenium Web Driver test method for the following Gherkin?**

```
Gherkin

Feature: Account Holder withdraws cash

Scenario: Account has sufficient funds
    Given The account balance is $100
        And the card is valid
        And the machine contains enough money
    When the Account Holder requests $20
        Then the ATM should dispense $20
        And the account balance should be $80
        And the card should be returned
```

We will introduce Selenium WebDriver and Gherkin in the next lessons

The suggestion



ChatGPT

Sure, here's an example of an end-to-end Selenium WebDriver test method written in Java for the provided Gherkin scenario:

java

Copy code

```
import org.junit.After;
import org.junit.Before;
import org.junit.Test;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;

public class ATMWithdrawalTest {
```

ChatGPT (test data generation)

- Could you create some **JUnit test** for the following Java **class User**?
- I need more data, could you **create some test data** for the User class?
- Put them in an array and create a **parametric/parameterized test**
- **More data** at least ten
- I want realist Italian phone numbers
- I need a complete code with the test data included in an array and passed as parameters to the test method

| | |
|---|----------------------|
| a | User |
| - | email : String |
| - | password : String |
| - | phoneNumber : String |
| - | address : Address |
| - | verified : boolean |
| - | roles : List<Role> |
| - | enabled : boolean |

Udemy Course

udemy Categorie Cerca... Udemy Business Insegna su Udemy Accedi Registrati

IT e software > Altri corsi IT e software > Controllo qualità

CHATGPT for Software Testing | Generative AI | 2024

Learn how to use Chat GPT for day to day work in Software Testing. CHAT GPT for Programming and Testing

Best seller 4,2 ★★★★☆ (75 valutazioni) 361 studenti
Creato da Dan Andrei Bucureanu
Ultimo aggiornamento in data 3/2024 Inglese Inglese [Auto]

Visualizza in anteprima questo corso

12,99 € 39,99 € 68% di sconto
A questo prezzo per ancora **2 giorni!**

Aggiungi al carrello

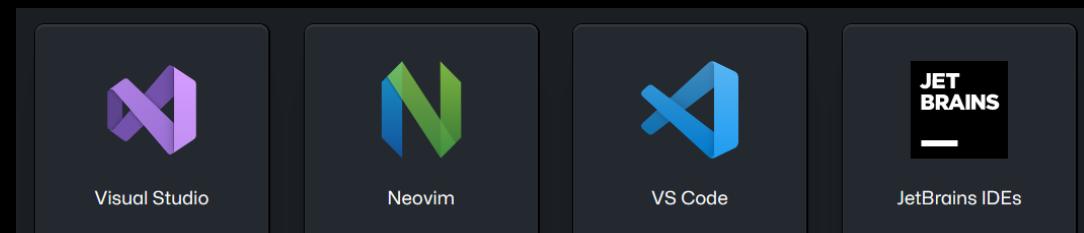
Acquista ora

QUICK INTRODUCTION TO GITHUB COPILOT

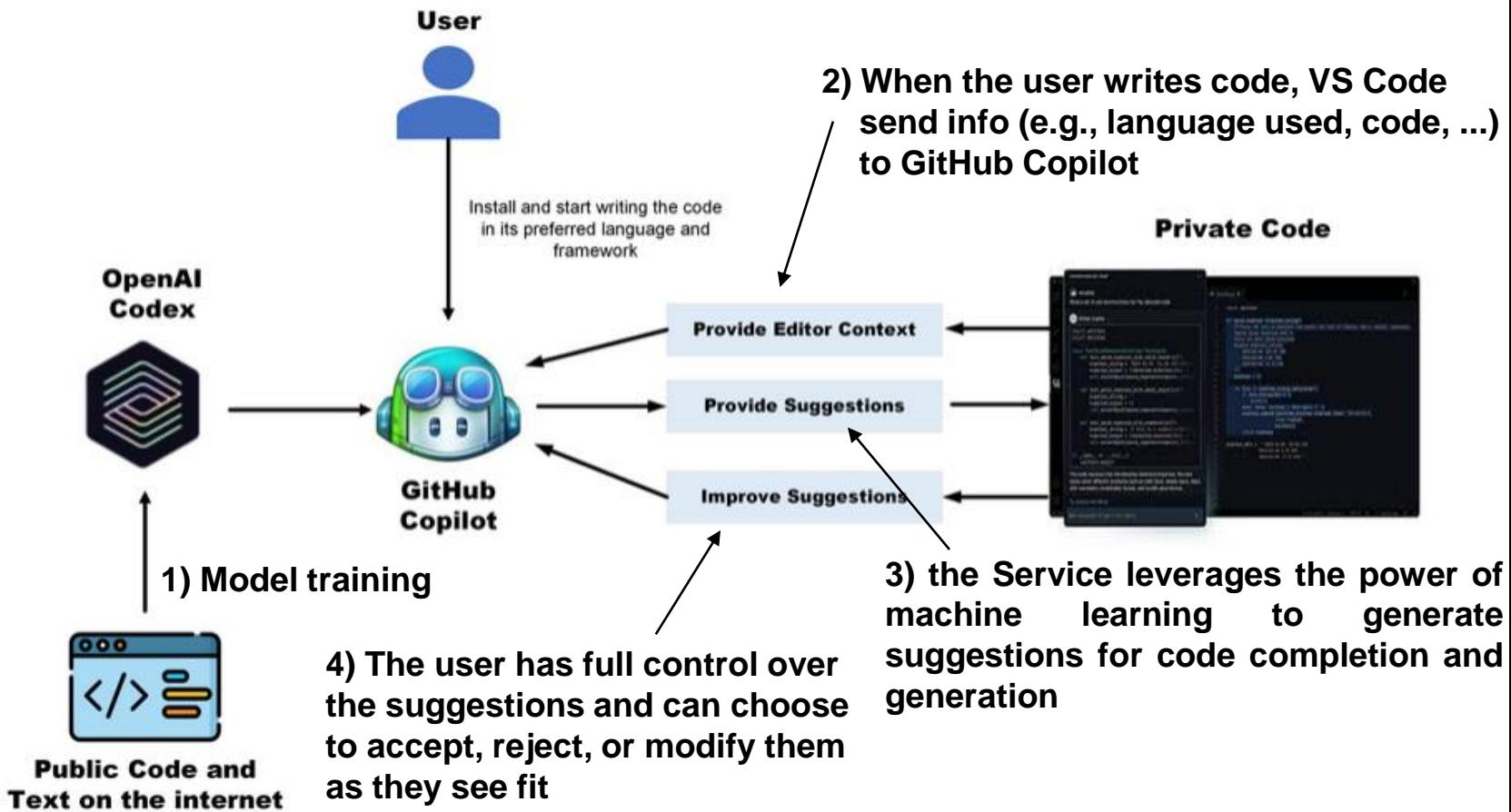
GitHub Copilot



- Defined as: '**AI pair programmer**'
- It provides a great help during the coding phase
- Based on **OpenAI Codex Model**
 - **AI model** capable of understanding natural language and producing code in response
 - Trained on billions of lines of public code (**all Github repos**)
- Equipped with a **chatbot** (in new versions)
- It supports various programming languages and various technologies/frameworks
- Integrated into several IDEs and Code Editors



How does it work?



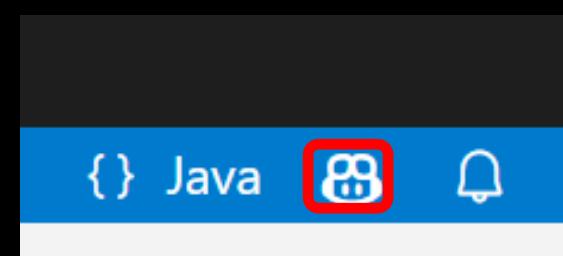
What can we do with Copilot?

- Get **inline code suggestions** while we are writing and iterating on code
- Start a **chat conversation** to generate or refactor source code, produce documentation comments, or generate test cases / test scripts
- Get help with **fixing errors** in our code, or resolve error while running commands in the terminal
- Ask questions to accelerate **learning** a **new programming language** or framework
- We will use it mainly for **Test Generation** ←



GitHub Copilot: SetUp

- It is necessary to have a **GitHub account**
- You must enable your GitHub account to use Copilot
 - Select ‘Settings’ after having clicked on you profile
 - Select ‘Copilot’
 - Enable GitHub Copilot
 - Copilot is a paid tool but as students you can use it for free by joining an appropriate student program
 - https://education.github.com/discount_requests/application
- Install in VS Code the necessary Copilot plug-in
 - also ‘Copilot chat’
 - Go to the marketplace, search GitHub Copilot and click on ‘Install’
- If the procedure was successful you should see the Github Copilot enabled symbol at the bottom right



GitHub Colipot: Keyboards shortcuts

- Accept inline code suggestion — **Tab**
- Dismiss inline code suggestion — **Esc**
- Show next suggestion — **Alt +] or Option (⌥) +]**
- Show previous suggestion — **Alt + [or Option (⌥) + [**
- Trigger suggestion — **Alt + \ or Option (⌥) + **
- Open 10 suggestions in a separate pane — **Ctrl + Enter**

<https://docs.github.com/en/copilot/configuring-github-copilot/configuring-github-copilot-in-your-environment>



How to use Copilot?

- **Inline suggestions**
 - As you start writing code Copilot presents suggestions
 - Use ‘Tab’ to accept
- **Use code comments to provide prompts to Copilot**
 - E.g., // create a Java class User with name, surname
- **Chat view**
 - Ask Copilot for help with any task or question in the Chat view
- **Inline Chat**
 - Press ‘**Ctrl + I**’ where you want to open it in the code

GITHUB COPILOT

GitHub Copilot

Welcome @bamurtaugh, I'm your Copilot and I'm here to help you get things done faster. I can identify issues, explain and even improve code.

You can ask generic questions, but what I'm really good at is helping you with your code. For example:

- ❖ Generate unit tests for my code
- ❖ Explain the selected code
- ❖ Propose a fix for the bugs in my code

If you want to learn more about my capabilities, [check out the Copilot documentation](#).

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and [share feedback](#) so that we can learn and improve.

Ask Copilot a question or type '?' for topics ➤

Code Generation starting from Comments

- It is able to convert ‘comments’ in code
 - It is sufficient to provide a prompt containing the logic
- A typical use case is for a user to type a comment
 - E.g., "**// write a binary search method**"
 - It is sufficient to give the method name
 - GitHub Copilot suggest a method able to satisfy our request
 - ‘in gray’
 - Tab for accepting it

```
// write a binary search algorithm
const binarySearch = (arr, target) => {
    let left = 0;
    let right = arr.length - 1;
    let middle = Math.floor((left + right) / 2);
    while (arr[middle] !== target && left <= right) {
        if (target < arr[middle]) {
            right = middle - 1;
        } else {
            left = middle + 1;
        }
        middle = Math.floor((left + right) / 2);
    }
    return arr[middle] === target ? middle : -1;
}
```

Code Documentation Generation

- It is able to infer the logic of a piece of code and produce a comment
 - For example the documentation of methods and classes
- It is also able to explain a portion of code in detail
 - ‘Program understanding’

The screenshot shows a code editor interface with a dark theme. A code review or documentation tool is overlaid on the editor. At the top, there is a status bar with the number '35'. Below it, a code block is shown:

```
35
36+    /**
37+     * Returns an ArrayList of prime numbers up to n using the Sieve of Eratosthenes algorithm.
38+     * @param n the upper limit of the range to find prime numbers
39+     * @return an ArrayList of prime numbers up to n
40+
41     public static ArrayList<Integer> eratosthenesSieve(int n) {
```

Below the code, there is a search bar containing '/doc' and a blue 'Accept' button. To the right of the search bar, there are 'Discard' and 'Undo' buttons. At the bottom right, there is a message 'Changed 5 lines' with a reply icon.

GitHub Copilot: Language translation

- Given a code written in one programming language, it is capable of converting it into another language

```
def eratosthenes_sieve(n):
    primes = []
    sieve = [True] * (n + 1)
    for p in range(2, n + 1):
        if sieve[p]:
            primes.append(p)
            for i in range(p * p, n + 1, p):
                sieve[i] = False
    return primes
```

Python



Java

```
public static ArrayList<Integer> eratosthenesSieve(int n) {
    ArrayList<Integer> primes = new ArrayList<Integer>();
    boolean[] sieve = new boolean[n + 1];
    for (int i = 2; i <= n; i++) {
        sieve[i] = true;
    }
    for (int p = 2; p <= n; p++) {
        if (sieve[p]) {
            primes.add(p);
            for (int i = p * p; i <= n; i += p) {
                sieve[i] = false;
            }
        }
    }
    return primes;
}
```

GitHub Copilot: Refactoring and Fix

- Given a portion of code (e.g. a class) it is capable of refactoring, changing part of the logic or finding errors and then eliminating them

```
def total_price(items):
    return sum(item['price'] * item['quantity'] for item in items)
```

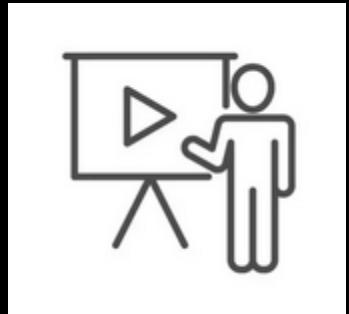


```
def total_price(items):
    total = 0
    for item in items:
        total += item['price'] * item['quantity']
    return total
```

GitHub Copilot: Test Code Generation

- GitHub Copilot is **able to generate JUnit test methods** starting from the name of the method or a textual description of the test case (in a comment)
 - e.g., for a BankAccount class
 - // Add a test method for the deposit method
 - Or for a Calculator class

```
// add unit test for the add method
no usages  new *
@Test
public void testAdd() {
    Calculator calculator = new Calculator();
    assertEquals( expected: 5, calculator.add( a: 2, b: 3));
}
```



Quick demo

GitHub Copilot: Test Code

Generation

GitHub Copilot: Chat

- It is a **coding chatbot** based on **OpenAI GPT-4**
- Provides real-time support and help on coding tasks

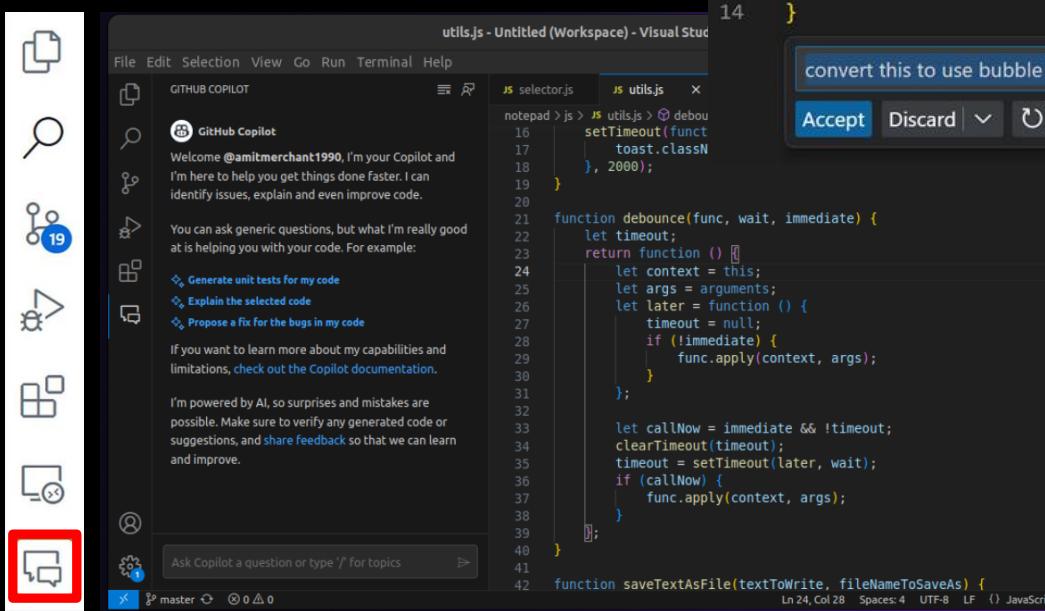
Inline Chat: 'Ctrl+I'

```
1  function sort(arr) {
2      const { length } = arr;
3      let temp;
4      for (let i = 1; i < length; i++) {
5          let j = i;
6          temp = arr[i];
7          while (j > 0 && (arr[j - 1] > temp)) {
8              arr[j] = arr[j - 1];
9              j--;
10         }
11         arr[j] = temp;
12     }
13     return arr;
14 }
```

convert this to use bubble sort

Accept Discard ⌂

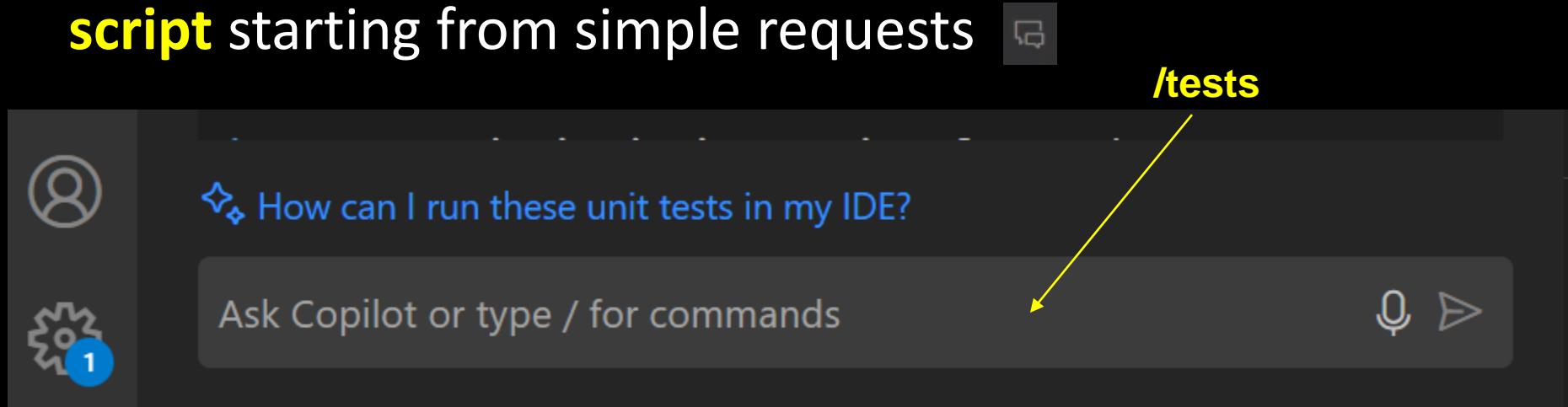
Changed 13 lines ↗ ↘



Chat View

GitHub Copilot Chat: Test Generation ⁽¹⁾

- The GitHub Copilot **Chat** is **able to generate JUnit test script** starting from simple requests



riccaF /tests Generate unit test

riccaF @workspace /tests generate parameterized test methods

GitHub Copilot Chat: Test Generation (2)

- Then it is possible to integrate the suggestion in the project using **copy&paste** or the button



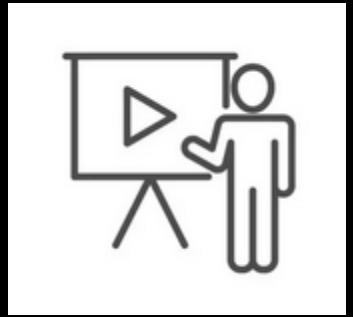
If you want to customize individual colors in your theme, you can use the `editor.tokenColorCustomizations` and `editor.semanticTokenColorCustomizations` settings in your `settings.json` file. These settings allow you to override editor syntax colors and font styles from the currently selected color theme.

Here's an example of how to use `editor.tokenColorCustomizations` to change the color of comments to green:

```
{  
  "editor.tokenColorCustomizations": {  
    "comments": "#00FF00"  
  }  
}
```



Copy



Quick demo

GitHub Copilot **Chat**: Test Code Generation

Second Assignment (not mandatory)

- Complete the tasks at:

<https://github.com/MorickClive/JunitVendorMachine>

- Test the Wallet class from: vendormachine.users.util.Wallet
 - Test the DrinkVendingMachine class from: vendormachine.vendors.DrinkVendingMachine
 - Add and complete tests for the 'vendormachine' project

- **Using GitHub Copilot!!!**

- Inline prompts or chat (or both)

- For computing coverage I suggest:

- Using plug-in ‘Coverage Gutters’ (see lesson 3)
 - VS Code Insiders
 - Computing manually the coverage



To Recap

- Generative AI
- Prompt Engineering
- **ChatGPT**
 - JUnit Test methods generation
 - Selenium Test methods generation
 - Test data generation
- **GitHub Copilot**
 - Helpful during coding
 - Software testing
 - Prompts: inline comments
 - Chat



Functional and Security Testing Course

2022-2023

**Manual Web app Testing and Gherkin
lesson 5**

Filippo Ricca
DIBRIS, University of Genova

WEB APPLICATION TESTING



Functional Web Testing

Goal: Improving the correctness of Web apps

- Should test all involved parts (client/server)
 - Everything can go wrong...
 - Heterogeneous systems
 - Pay attention to dynamism!
 - **highly dynamic and compelling UI!**
- Client side
 - HTML: Malformed HTML page?
 - JavaScript: Runtime Errors? Failure?
- Server side
 - PHP, Java...: Runtime Errors in Server scripts? Failure?
 - DBMS: Malformed SQL query string?



As a result: real bug in Modern apps

Express-Cart (1)



Duckworth Woolfill Jacket
\$188.00
Options
Size: S
Colour: Harvest
Quantity: -1 +
Add to cart

Inspired by the timeless, functional style of your grandfather's work coat, the Foraker features brass buttons and 4 patch pockets. Crafted in Bristol, Tennessee, our 10oz organic duck canvas is light enough for an early summer morning, but rugged enough to handle your days work.

- 100% Organic Duck Canvas.



Express-cart is a modern Web app: Node.js, Express, AngularJS, MongoDB, Bootstrap ...

As a result: real bug in Modern apps

Express-Cart (2)

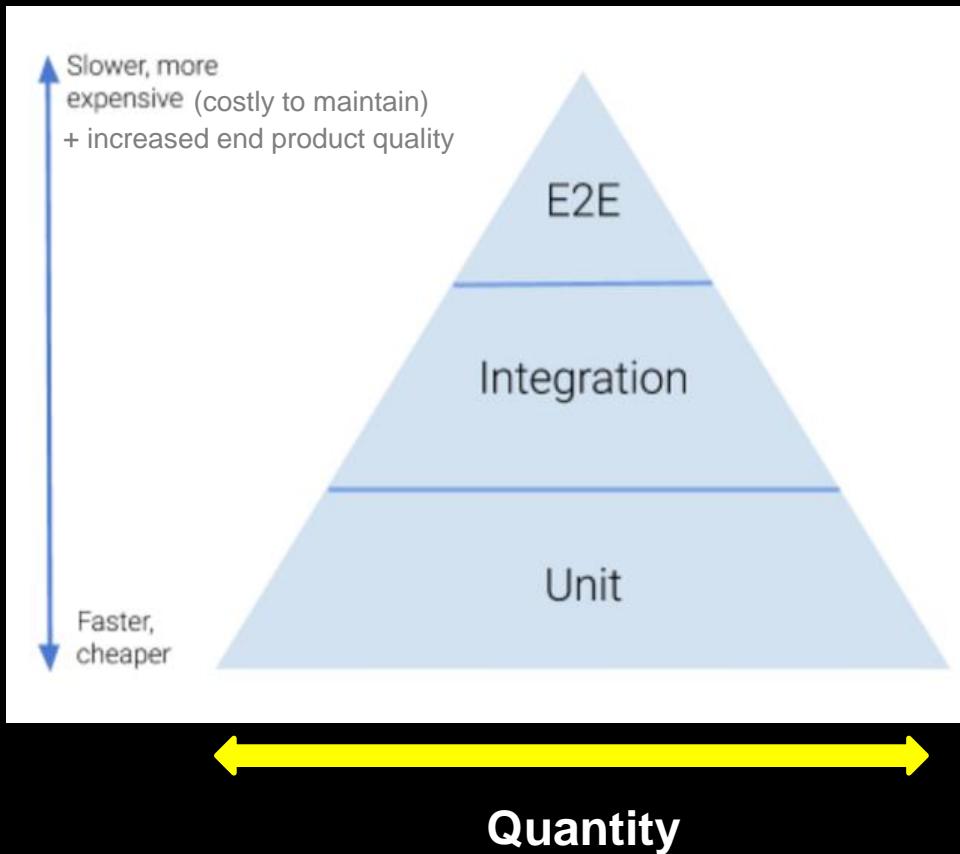
Cart contents

| | | |
|---|---|-------------------|
|  | Duckworth Woolfill Jacket | |
| | S / Harvest | \$-188.00 |
| | - -1 + | |
| | | Shipping: \$10.00 |
| | | Total: \$-178.00 |

Empty cart Checkout

If you remember ...

Test Pyramid Rule



**Testing the system
as a whole (GUI)**



**Individual units are
combined and tested
as a group**



**Testing of a single
function/class**



In theory, the distribution of test cases in a project should resemble a pyramid

Unit and Integration testing

(difficult to apply!)

- **Unit Testing**

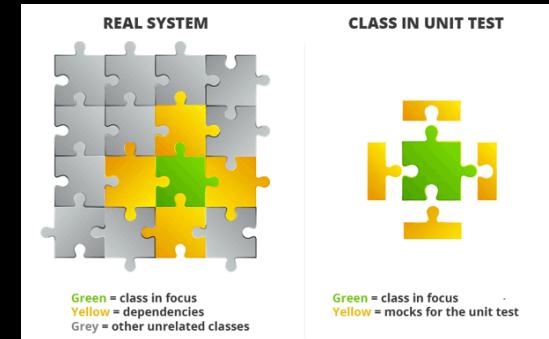
- **Challenge:** isolating Your Units

- Components of Web apps are interconnect; there are always **multiple dependencies**. You have **to mock many different classes/functions** and writing even a small test might be actually a big job ...

- **Integration Testing**

- **Challenge:** populating the units with data (setup)

- the time and resources necessary to create and maintain such test data becomes too complex and overwhelming



To understand better why unit testing is complex:

JS prettyDate() function

Blog

prettyDate(): the date is usually prettified (a 'day ago' instead of May 12, 2019)

Powered by SteemPress

↑ 795 20.16 \$

12 Comments [reply](#)

[Log in](#)
[Register](#)

Sort by ▾



redpill
[a day ago](#)

This is super exciting. I'm posting this comment directly from dappcentral.io!!!

↑ 1 0.28 \$ [reply](#)



adonisabril
[a day ago](#)

@howo and gang are really on to something here. Steempress is by far the most useful dapp in the steem universe.

↑ 1 0.28 \$ [reply](#)

Dependencies

```
<html>
<head>
<title>Mangled date examples</title>
<script>
function prettyDate(time) {
    ...
}

window.onload = function() {
    var links = document.getElementsByTagName("a");
    for (var i = 0; i < links.length; i++) {
        if (links[i].title) {
            var date = prettyDate(links[i].title);
            if (date) {
                links[i].innerHTML = date;
            }
        }
    }
}
</script>
</head>
<body>
<ul>
- blah blah blah...



Posted January 28th, 2008 by John Resig

<!-- more list items --&gt;
&lt;/ul&gt;
&lt;/body&gt;</pre>
```

How can we test
prettyDate()
It strongly depend on
the Web page ...

5 days ago

Html Blog

- blah blah blah...
- Posted [January 28th, 2008](#) by [John Resig](#)
- blah blah blah...
- Posted [January 29th, 2008](#) by [John Resig](#)



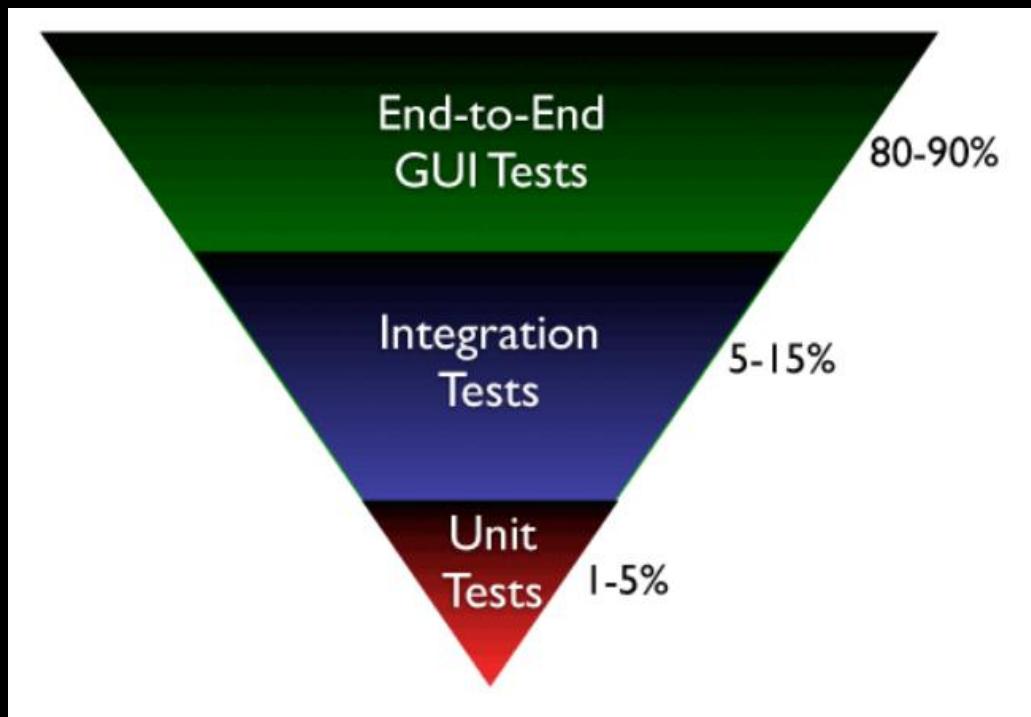
Code Refactoring

- **Steps to be conducted :**
 - Extract the function in a separate file
 - So to include it in a separate Test page
 - Select a Javascript Testing framework
 - E.g., QUnit
 - Modify prettyDate()
 - Add a parameter (current date, not necessary before because recovered by the browser)
 - Pass the current date to prettyDate() and some input dates

Test page

```
...
<script src="qunit.js"></script>
<script src="prettydate.js"></script>
<script>
test("prettydate basics", function() {
    var now = "2008/01/28 22:25:00";
    equal(prettyDate(now, "2008/01/28 22:24:30"), "just now");
    equal(prettyDate(now, "2008/01/28 22:23:30"), "1 minute ago");
    equal(prettyDate(now, "2008/01/28 21:23:30"), "1 hour ago");
    equal(prettyDate(now, "2008/01/27 22:23:30"), "Yesterday");
    ...
});
</script>
...
```

E2E is very often the first and only choice in many organizations!

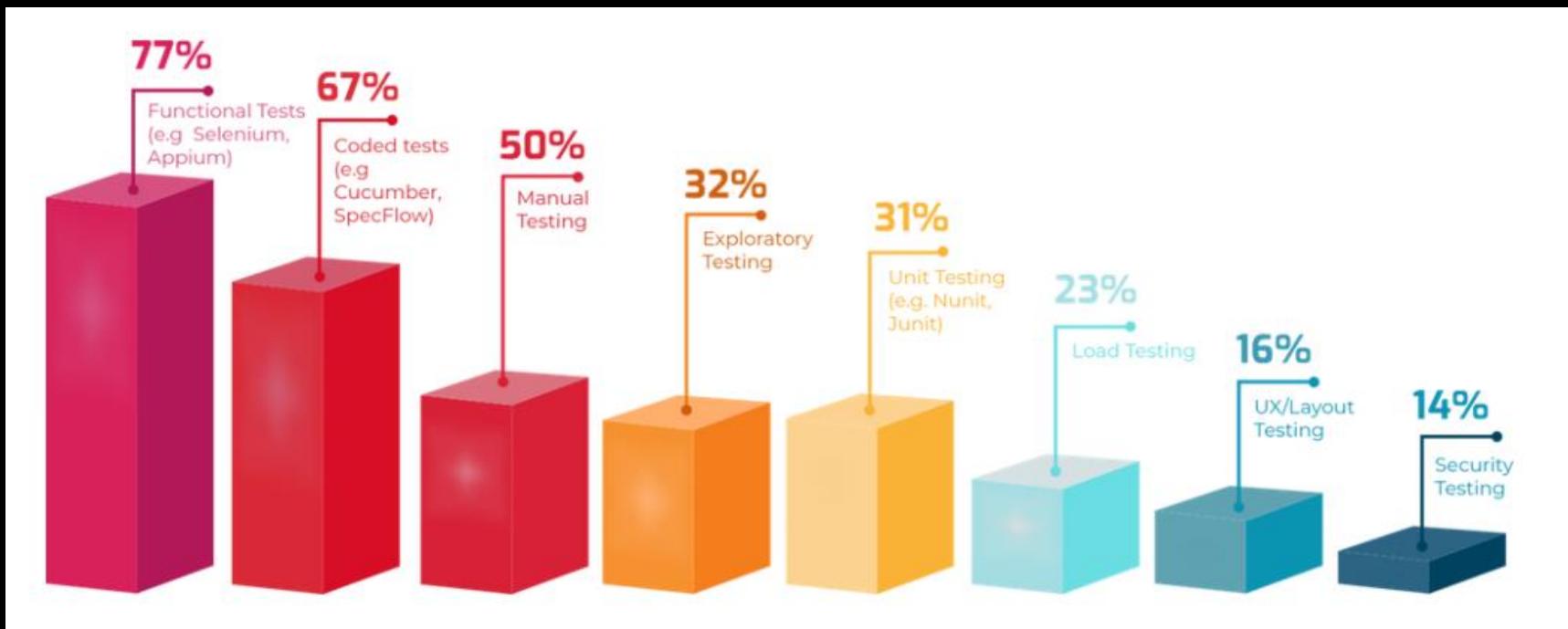


Exactly the inverse of what says the Test Pyramid rule!

... and this is not good but a fact

Survey

- An **extensive global survey** conducted by Tricentis provides several essential insights on the **testing trends**
- The testing tendencies show that teams prefer to use E2E functional testing (Selenium, Appium)



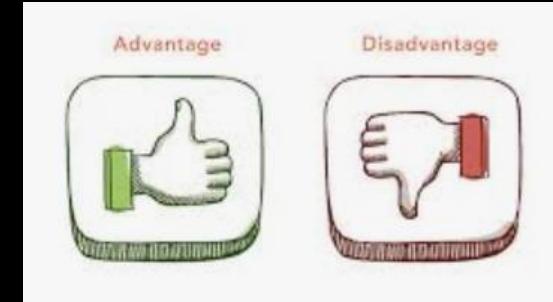
Manual testing for a Web app

- Process in which the human tester **create and execute E2E test cases** on the Application Under Test (AUT)
- Two steps:
 - Test cases creation
 - Test cases execution (manually)



Manual testing is necessary!

- Manual testing is **still an essential component of the software development process** in 2023!
- **Disadvantages:**
 - Time consuming and expensive
 - Resource-intensive and Repetitive
 - Not suitable for regression testing
- **Advantages:**
 - **Complex Scenarios:** Some scenarios may be too complex or difficult to automate effectively
 - Not everything can be easily automated
 - E.g., look and feel issues of a web application



E2E Test case

Web App under test

The screenshot shows a Google search results page. The search bar at the top contains the query "ERC". Below the search bar, there are three navigation links: "Web", "Images", and "More". The "Web" link is underlined, indicating it is the active category. The main search result is a link to the European Research Council's website, titled "ERC: European Research Council" and with the URL "erc.europa.eu/". A snippet of the website content describes it as a funding body for scientific excellence. At the bottom of the search results, there are additional links: "Funding and Grants", "About ERC", "Contact us", and "Projects and Results".

Test case (can be executed manually)

1. Open www.google.com
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: European Research Council"

A test case is a set of actions (**Test steps**) + inputs (**Test data**) done on a AUT to verify that it works as expected (**Expected result**)

Manual Testing (two problems)

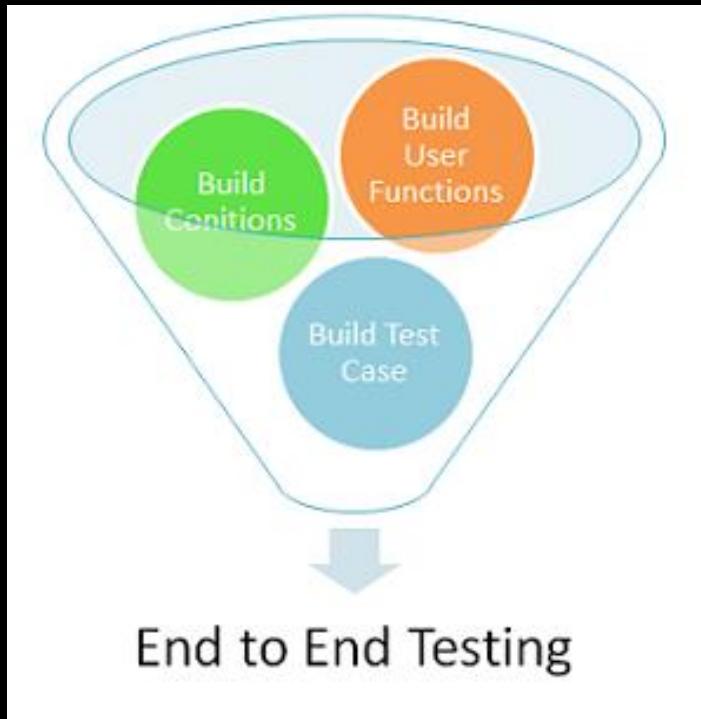
1. How to produce test cases for a target WAs?
 - We need **approaches/techniques** that can guide us
2. How to express, document and store test cases (to be executed manually)?
 - Useful ,e.g., for
 - Regression testing
 - Different options
 - **Tables** 
 - **Ad-hoc languages**
 - **Gherkin language**
 - ‘Ease’ the transition to **Automated Testing**

| Test Case ID | Test Case Description | Test Steps | Test Data | Expected Results | Actual Results | Pass/Fail |
|--------------|--|---|---------------------------------------|---|----------------|-----------|
| TU01 | Check Customer Login with valid Data | 1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit | Userid = guru99 Password = pass99 | User should Login into an application | As Expected | Pass |
| TU02 | Check Customer Login with invalid Data | 1. Go to site http://demo.guru99.com 2. Enter UserId 3. Enter Password 4. Click Submit | Userid = guru99 Password = glass99 | User should not Login into an application | As Expected | Pass |

CREATING E2E TEST CASES

How to create E2E Test Cases?

A systematic black box approach (for Web apps)



Select User Functions

- List down the **features** (user function) of the system that we want to test

Build Conditions based on User Function

- Building a set of **conditions** for each user function selected

Build Test Scenarios

- Building the **test scenarios** for the user functions defined (a set of steps or **pages**)

Build Multiple Test cases

- Build one or more **test cases** for each scenario defined. Test cases should include each condition

Input selection phase



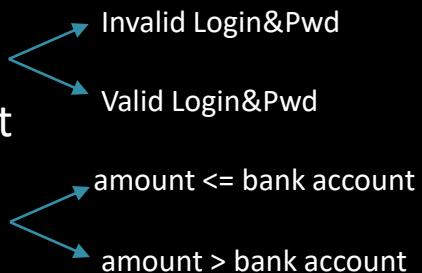
E2E Testcase Generation



*"Let's consider a **Bank web app** providing different user functions (feature) ...*

1. Login into the banking system
2. Check balance amount in the account
3. Transfer some amount from your account to some other bank account
4. Recharge mobile phone
5. Pay invoice
6. Logout

1° Select User Functions



2° Build Conditions based on User Functions

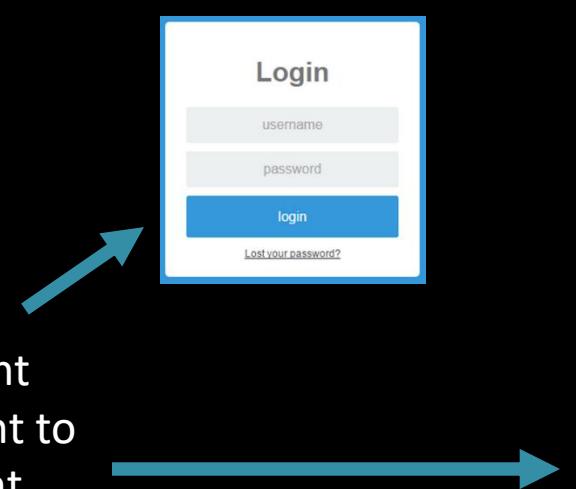
1. Login into the system (ok)
2. **Check balance amount**
3. Transfer some amount to another bank account
(amount \leq bank account)
4. **Check balance amount**
(assert balance=balance@pre-amount)

3° Build Test Scenarios (path in web app)

A test case is an instance of a test scenario where the inputs are selected and inserted

Input selection phase

1. Login into the system
2. Check balance amount
3. Transfer some amount to another bank account
4. Check balance amount



3º Build Test Scenario
(path in web app)

The screenshot shows a 'Transfer Funds' form with the following fields:
- Destination bank: A dropdown menu.
- Account Number: An input field.
- Account Type: A dropdown menu.
- Beneficiary Name: An input field.
- Amount: An input field.
- Click for more: A link.
- Continue: A red button at the bottom.

4º Build multiple Test cases

How many testcases? We have to select a reasonable number of good inputs!

How many input data?

The rule of thumb is to have at least one test case for each condition

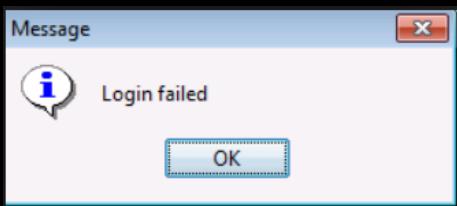
In our example:

Username user
Pwd ****



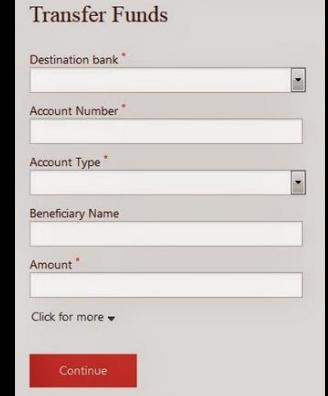
Login screen showing a successful login message: "Login Successful".

Valid Login&Pwd



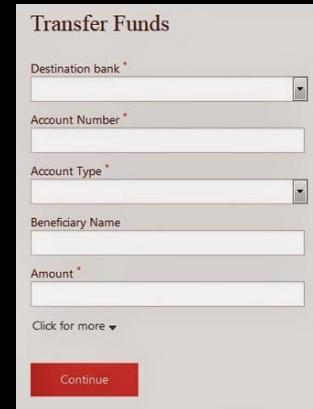
Invalid Login&Pwd

Amount ****



Transfer Funds form with fields: Destination bank, Account Number, Account Type, Beneficiary Name, Amount, and a "Continue" button.

amount <= bank account



Transfer Funds form with fields: Destination bank, Account Number, Account Type, Beneficiary Name, Amount, and a "Continue" button.

amount > bank account

One test case for each condition could not be sufficient!

Exhaustive Testing

- **Exhaustive testing** is a test approach in which **all possible data combinations** are used for testing

Web app Form

List Box - 0,1,2,3,4,5,6,7,8,9

Check Box - Checked or Unchecked

Radio Button - ON or OFF

Text Box - Any Value between 1 and 100

Exhaustive testing is not possible



```
List Box = 10  
Check Box = 2  
Radio Button = 2  
Text Box = 100
```

Total Number of Test Cases using Cartesian Method : $10 \times 2 \times 2 \times 100 = 4000$
Total Number of Test Cases including Negative Cases will be > 4000

4000 test cases!!!

Negative cases are invalid input (e.g., a string when an integer is required)

Reducing the combinations ...

List Box - 0,1,2,3,4,5,6,7,8,9

Check Box - Checked or Unchecked

Radio Button - ON or OFF

Text Box - Any Value between 1 and 100



For each variable only 2 values (max 3) are considered

List Box - 0, others

Check Box - Checked or Unchecked

Radio Button - ON or OFF

Text Box – valid, invalid, char

Conventional shortcut!

Exhaustive Combination results in > 4000 Test Cases.

Conventional Software Testing technique results in 24 Test Cases.

Pairwise (all-pairs) Testing

Pairwise testing technique reduces the number of test cases by testing **all pairs of values of variables** rather than attempting to test all of the combinations

Most faults are caused by interactions of at most two factors!

Pairwise-generated test suites cover all combinations of two values

| Text Box | List Box | Check Box | Radio Button |
|-----------------------|---------------|-----------|--------------|
| Valid Int | 0 | check | ON |
| Valid Int | <u>others</u> | uncheck | OFF |
| Invalid Int | 0 | uncheck | ON |
| Invalid Int | others | check | OFF |
| AlphaSpecialCharacter | 0 | check | OFF |
| AlphaSpecialCharacter | <u>others</u> | uncheck | ON |

6 test cases

Reduction

Exhaustive Combination results in > 4000 Test Cases.

Conventional Software Testing technique results in 24 Test Cases.

Pair Wise Software Testing technique results in just 6 Test Cases.

99.85%

Maintaining a good fault discovery rate!

Pairwise Testing Tools

- Difficult to compute the **matrix** manually!
- There are a lot of tools ...
 - <http://pairwisetesting.com/tools.html>
- **E.g., Pairwiser**



Pairwiser

Pairwise Testing and
Test Generation Tool

More Info

Sign up - its free!

How to select input data?

List Box - 0,1,2,3,4,5,6,7,8,9

Check Box - Checked or Unchecked

Radio Button - ON or OFF

Text Box - Any Value between 1 and 100



For each variable only 2 values (max 3) are considered

List Box - 0, others

Check Box - Checked or Unchecked

Radio Button - ON or OFF

Text Box – valid, invalid, char

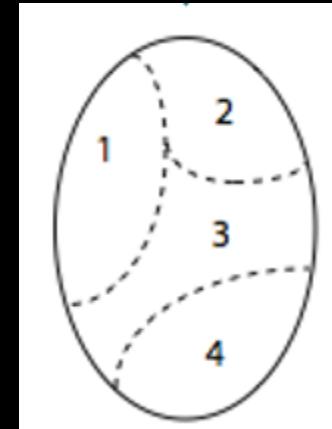


We have to select data input so as to find the largest number of errors

Boundary Testing: a good option!

- Boundary testing **is the process of testing extreme ends or boundaries** between partitions of the input values
- There are scientific studies showing that often errors are located on the boundaries of the input data partitions

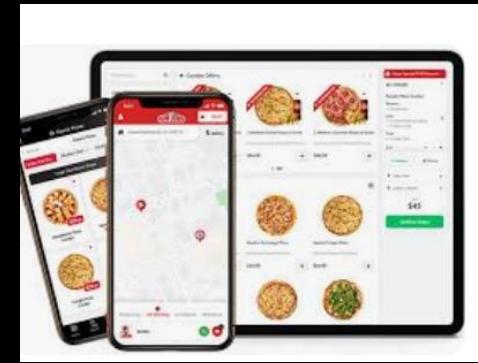
Input Domain



Input Domain is divided in partitions created assuming that the SUT will exhibit the same behavior on all elements of the partition

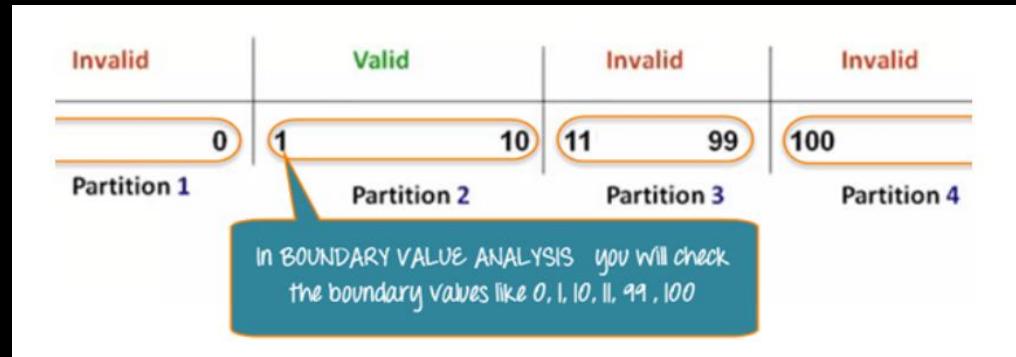
An Example of Boundary Testing

Order Pizza: Submit



Example:

- Let's consider the behavior of an hypothetical **Order Pizza Web app** above
- Pizza values 1 to 10 are considered valid. A success message is shown
- Values from 11 to 99 are considered invalid for order and an error message will appear, “**Only 10 Pizza can be ordered**”
- Values greater than 100 are invalid. Error message: “**number too big**”



Boundary value analysis

<https://www.youtube.com/watch?v=P1Hv2sUPKeM>

DOCUMENTING TEST CASES: GHERKIN LANGUAGE

Gherkin

- Gherkin is a ‘business readable language’ **used to define and describe test cases**
- Gherkin is a ‘line oriented language’
 - each line starts with a **Gherkin keyword**
 - each line is a step
 - the end of line is considered to be the end of an instruction or statement
- Comments are only permitted at the start of a new line
 - ‘#’ symbol



Gherkin keywords

- The purpose of the **Feature** keyword is to provide a high-level description of a software feature, and to group related scenarios
 - The feature we want to test
 - A simple text describe it
 - Feature = collection of scenario

Only one scenario (main scenario)

Feature: Login functionality of social networking site Facebook.

Given: I am a facebook user.

When: I enter username as username.

And I enter the password as the password

Then I should be redirected to the home page of facebook

Given

When

Then

What
software will
look like to
user

Things that
the user will
do

What the
user should
expect

Important Terms used in Gherkin

Given, When, Then

More Scenarios

Feature: Transfer money from a bank account to another

Scenario: Account balance is sufficient

Given bank account NL01INGB0123456789

And the account contains 364 euro

When the account holder wants to transfer 237 euro

Then the transfer proceeds because balance is sufficient

Scenario: Account balance is insufficient

Given bank account NL01INGB0123456789

And the account contains 237 euro

When the account holder wants to transfer 364 euro

Then the transfer does not proceed because balance is insufficient

One Feature but two scenarios (two test cases)

Login ‘without’: And, Or, But

Feature: Registration, Login and MyAccount

Scenario: Verify Login Functionality

Given I am on the homepage

Given I should see "Sign in" link

When I follow "Sign in" link

When I fill "email address" with "goswami.tarun77@gmail.com"

When I fill "password" with "tarun@123"

When I click "sign in" button

Then I should see "My Account" Heading

Then I should see "my account" link

Then I should not see "Login" Heading

Login ‘with’: And, Or, But

Feature: Registration, Login and MyAccount

Scenario: Verify Login Functionality

Given I am on the homepage

And I should see "Sign in" link

When I follow "Sign in" link

And I fill "email address" with "goswami.tarun77@gmail.com"

And I fill "password" with "tarun@123"

And I click "sign in" button

Then I should see "My Account" Heading

And I should see "my account" link

But I should not see "Login" Heading

Same as before but more readable!

Parameters

'Background' is used to define prerequisites valid for each scenario

Background:

Given I have logged into my account

@REQ-001 @Regression @EndToEnd

Scenario Outline: Withdraw money from account

Given my balance is "<balance>"

When I withdraw "<withdraw>"

Then I receive "<receive>"

And my new balance is "<remaining>"

And I am told "<message>"

Examples:

| balance | withdraw | receive | remaining | message |
|---------|----------|---------|-----------|-----------------------|
| 100 | 31 | 0 | 100 | "Please try again." |
| 100 | 30 | 30 | 70 | "Thank you." |
| 100 | 900 | 0 | 100 | "Insufficient funds." |

- 'Angle bracket' are used to express parameters in Scenarios**
- Data values are inserted in the Table below**

Gherkin business language

Simple to comprehend also for non – developers

We would like to encourage new users to buy in our shop.
Therefore we offer 10% discount for their first order.

```
public void CalculateDiscount(Order order)
{
    if (order.Customer.IsNew)
        order.FinalAmount =
            Math.Round(order.Total * 9/10);
}
```

Register as "bart_bookworm"
Go to "/catalog/search"
Enter "ISBN-0955683610"
Click "Search"
Click "Add to Cart"
Click "View Cart"
Verify "Subtotal" is "\$33.75"

← Before

After →

We would like to encourage new users to buy in our shop.
Therefore we offer 10% discount for their first order.

Given the user has not ordered yet
When the user adds a book with the price of EUR 37.5 into the shopping cart
Then the shopping cart sub-total is EUR 33.75.

Gherkin videos

- <https://www.youtube.com/watch?v=KP0vpVLatMc>
- <https://www.youtube.com/watch?v=i0Q5orC5jSQ>

To Recap



- Goal: Manual testing of Web apps

- **Test cases creation**

- Systematic black box approach
 - Input data selection
 - » Boundary value analysis (to select input data)
 - » Pairwise testing (to reduce the combinations)

- Gherkin language to express test cases

- **Test cases execution**

- Manual testers read ‘Gherkin test cases’ and execute them on the AUT

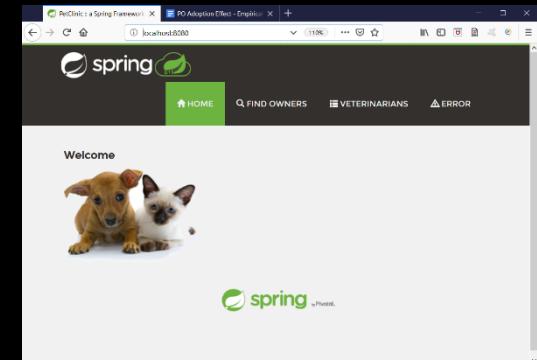


PETCLINIC WEB APP

(USED AS SAMPLE WEB APP)

Petclinic functionalities

- Petclinic is a **veterinary clinic** web application allowing veterinarians to manage data about **pets** and their **owners**
- Simple web app providing functionalities:
 - Add Owner, Find Owner, Edit Owner
 - Add Pet to an Owner, Edit Pet
 - Add Visit



A screen ...

localhost:8080/owners/2

The screenshot shows a web application interface for managing pet owners. At the top, there's a navigation bar with links for HOME, FIND OWNERS, VETERINARIANS, and an ERROR page. The main content area has two sections: "Owner Information" and "Pets and Visits". The "Owner Information" section contains fields for Name (Betty Davis), Address (638 Cardinal Ave.), City (Sun Prairie), and Telephone (6085551749). Below these fields are two buttons: "Edit Owner" and "Add New Pet". The "Pets and Visits" section displays a list of pets for the owner. One pet, named Basil, is shown with details: Birth Date (2012-08-06) and Type (cat). To the right of the pet list are buttons for "Edit Pet" and "Add Visit". At the bottom of the page is the Spring logo.

Owner Information

| | |
|-----------|-------------------|
| Name | Betty Davis |
| Address | 638 Cardinal Ave. |
| City | Sun Prairie |
| Telephone | 6085551749 |

Edit Owner Add New Pet

Pets and Visits

| | | |
|-------------|------------------------|-----------|
| Name: Basil | Birth Date: 2012-08-06 | Type: cat |
|-------------|------------------------|-----------|

Visit Date Description

Edit Pet Add Visit

spring by Pivotal.

Information of an owner

PetClinic

- Petclinic is a Spring Boot application built using Maven
- Spring Boot is a framework used to build Web applications
 - <https://spring.io/projects/spring-boot>
- Maven is a build automation tool used primarily for Java projects
 - <https://maven.apache.org/>

Our task

1. Install PetClinic
2. Create a collection of test cases to test the main features of PetClinic
 - Applying the Systematic approach as much as possible
 - Using Gherkin language to express them
3. Test PetClinic Manually

PetClinic Installation

- Go to <https://github.com/spring-projects/spring-petclinic>
- Petclinic is a [Spring Boot](#) application built using [Maven](#). You can build a jar file and run it from the command line:

```
git clone https://github.com/spring-projects/spring-petclinic.git  
cd spring-petclinic  
.mvnw package  
java -jar target/*.jar
```

- You can then access Petclinic here: <http://localhost:8080/>

Gherkin test case

- Feature: **SearchOwner**

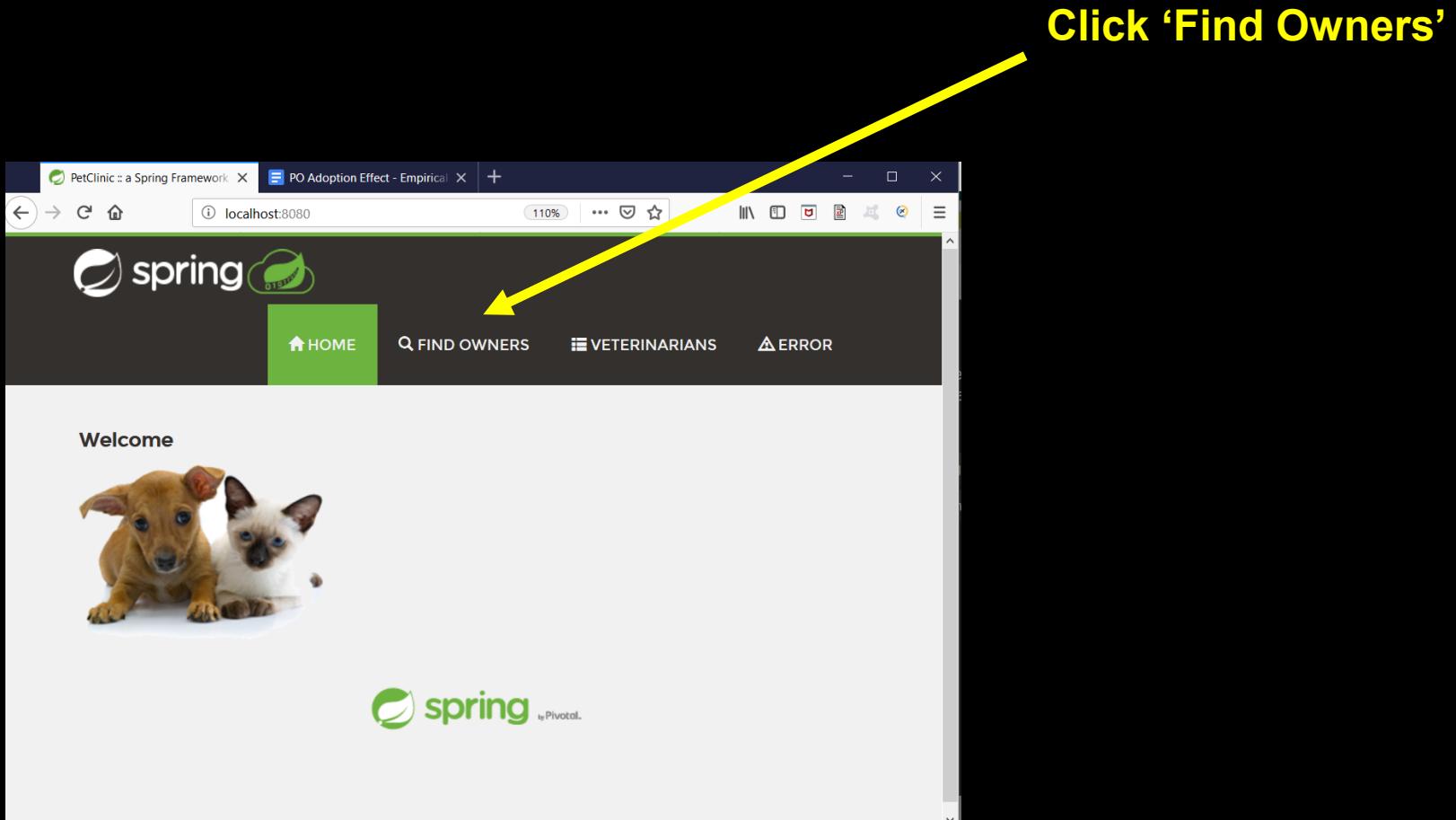
As a veterinarian I want to Search an Owner

Scenario: A veterinarian searches an existing Owner

- **Given** I am on the Home page of PetClinic
- **When** I Click “Find Owners”
- **And** I Insert the Owner’s last name ‘**Black**’
- **And** I Click the button “Find Owner”
- **Then** I should see the Black Owner’s info displayed

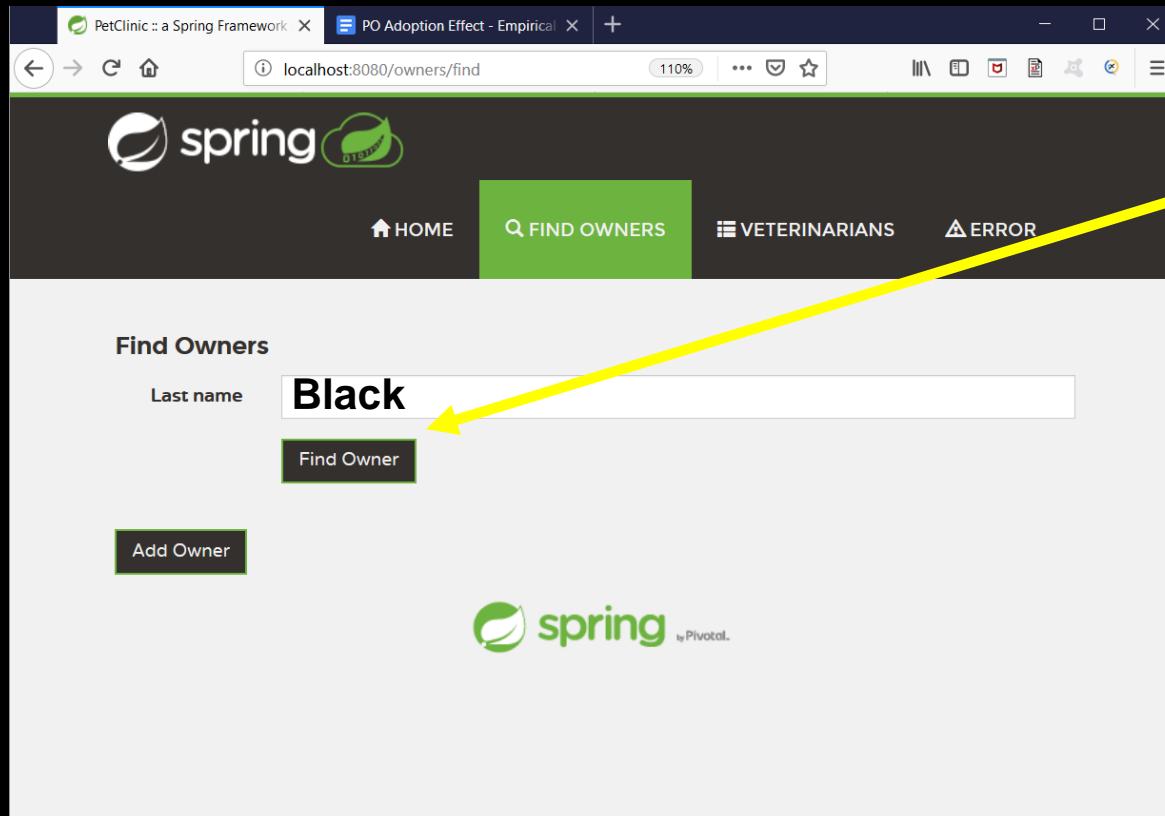
Test case execution

- Given I am on the Home page of PetClinic
- When I Click “Find Owners”



Test Case execution

- And I Insert the Owner's last name ‘Black’
- And I Click the button “Find Owner”



Click 'Find Owner' button

Test Case execution

- Then I should see the Black Owner's info displayed

PetClinic :: a Spring Framework X PO Adoption Effect - Empirical X +

localhost:8080/owners/7 110% ⚡

spring

HOME FIND OWNERS VETERINARIANS ERROR

Owner Information

| | |
|-----------|----------------|
| Name | Bob Black |
| Address | 1450 Oak Blvd. |
| City | Monona |
| Telephone | 1234567890 |

Edit Owner Add New Pet

Pets and Visits

| Name | Birth Date | Type | Visit Date | Description |
|--------|------------|------|------------|----------------------------|
| aRNOLD | 2015-01-10 | bird | 2019-03-26 | ssssssssssssssssssssssssss |
| | | | 2019-03-26 | dsssd |

Edit Pet Add Visit

Functional and Security Testing Course

2023-2024

**Automated E2E Testing with Selenium
lesson 6**

Filippo Ricca
DIBRIS, University of Genova

To Recap



- Goal: Manual testing of Web apps

- **Test cases creation**

- Systematic black box approach
 - Input data selection
 - » Equivalence partitioning
 - » Boundary value analysis
 - » Pairwise testing (to reduce the combinations)

- Gherkin language to express test cases

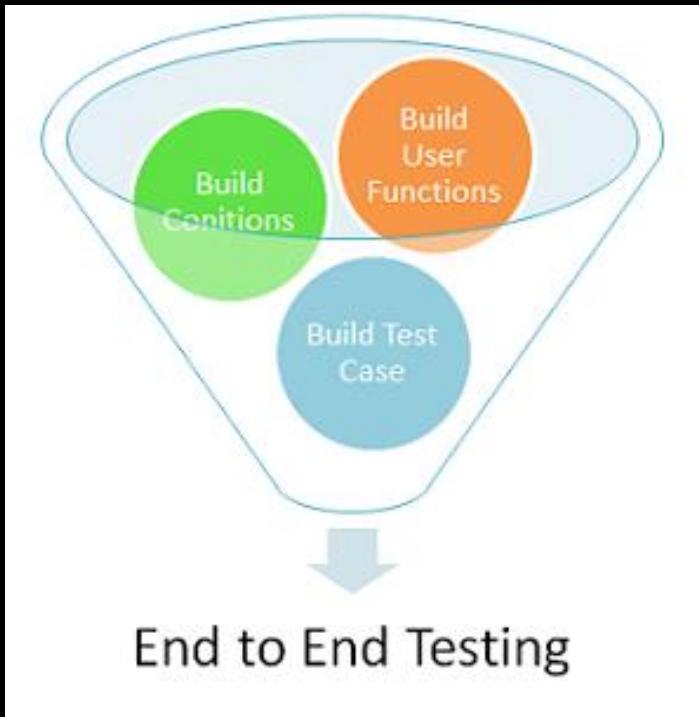
- **Test cases execution**

- Manual testers read Gherkin test cases and execute them on the AUT



Last lecture

A systematic black box approach (for Web apps)



Build User Functions

- List down the features of the system that we want to test

Build Conditions based on User Function

- Building a set of **conditions** for each user function defined

Build Test Scenarios

- Building the **test scenarios** for the user functions defined (steps or **pages**)

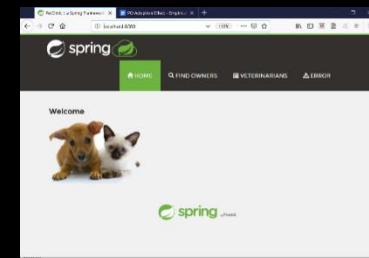
Build **Multiple** Test cases

- Build one or more test cases for each scenario defined. Test cases should include each condition

Inputs selection phase →

How to create E2E Test Cases?

Gherkin test case example



- **Feature: SearchOwner**

As an user I want to Search an Owner

Scenario: An user searches an existing Owner

- **Given** I am on the Home page of PetClinic
- **When** I Click “Find Owners”
- **And** I Insert the Owner’s last name ‘**Black**’
- **And** I Click the button “Find Owner”
- **Then** I should see the Black Owner’s info displayed

Three problems (to realize Automated software testing)

- How to produce test cases for a target WAs?
 - We need an **approach/strategy** that can guide us
- How to express, document and store test cases
 - We need a specification language
- How to transform test cases in test scripts?
 - We need a **Web testing framework/tool**



Automated End-to-End Web testing

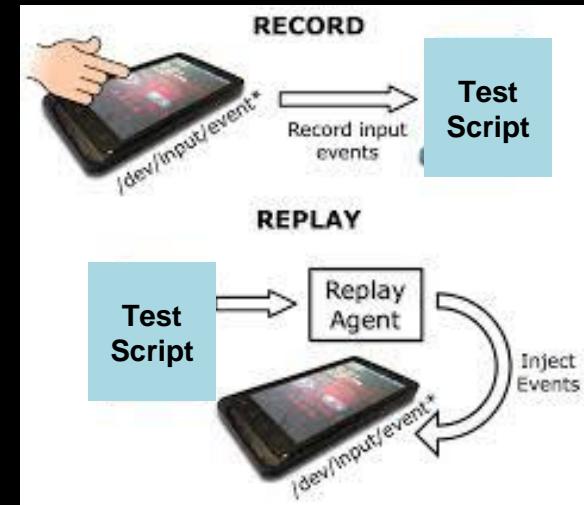


Testing Frameworks/Tools

Two categories:

- **Capture/Replay**

- **Recording** the actions performed by the tester
 - using a specific tool/framework
 - e.g., Selenium IDE, Katalon IDE, ...
 - Re-executing them automatically

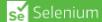


- **Programmable (or Script-based)**

- Test scripts are **software artefacts** created:
 - Using standard programming languages and IDE
 - e.g., Java + VS Code
 - Resorting to specific Web testing frameworks
 - Selenium WebDriver(+JUnit), TestIM, Katalon Studio, ...



Selenium Ecosystem

 Selenium

<https://www.selenium.dev/>

About Downloads Documentation Projects Support Blog English  Search this site...

Selenium automates browsers. That's it!

What you do with that power is entirely up to you.

Primarily it is for automating web applications for testing purposes, but is certainly not limited to just that. Boring web-based administration tasks can (and should) also be automated as well.

Log4j Vulnerability 

Note: Selenium does **not** use Log4j, thus is unaffected by [CVE-2021-45105](#).



Selenium WebDriver

If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use Selenium WebDriver, a collection of language specific bindings to drive a browser - the way it is meant to be driven.

[READ MORE ▶](#)



Getting Started

Selenium IDE

If you want to create quick bug reproduction scripts, create scripts to aid in automation-aided exploratory testing, then you want to use Selenium IDE; a Chrome, Firefox and Edge add-on that will do simple record-and-playback of interactions with the browser.

[READ MORE ▶](#)



Selenium Grid

If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS, then you want to use Selenium Grid.

[READ MORE ▶](#)

- Software ecosystems are **collections of elements** interacting with a **shared market** underpinned by a common technological background
- In the case of Selenium, its ecosystem involves the official **core projects** and other related projects, libraries, and other actors
- Language bindings, driver managers, locator tools, frameworks, browser infrastructure (cloud providers), and communities



Selenium IDE (Capture/replay)

- **Firefox/Chrome/Edge extension!**

- Add-on simple to install

- During recording Selenium saves all information in a ‘three element’ Table format (JSON file, format ‘.side’)

- Actions done by the user

- Each line/record consists of three elements:

- **Command** – tells Selenium what to do
 - e.g. “open”, “type”, “click”
 - **Target** – tells Selenium which Web element a command refers to
 - e.g. textbox, header, table
 - **Value** – used for any command that might need a value of some kind
 - e.g. type something into a textbox

Selenium IDE - UI

| Command | Target | Value |
|---------|------------------------------|--|
| 1 | open | /contest/holiday-feast-win-free-ticket-s |
| 2 | set window size | 786x818 |
| 3 | select frame | index=0 |
| 4 | type | id=ember634 keyword |
| 5 | type | id=ember641 [REDACTED]@gmail.com |
| 6 | click | css=.ssButton |
| 7 | wait for element not present | id=ember634 10000 |

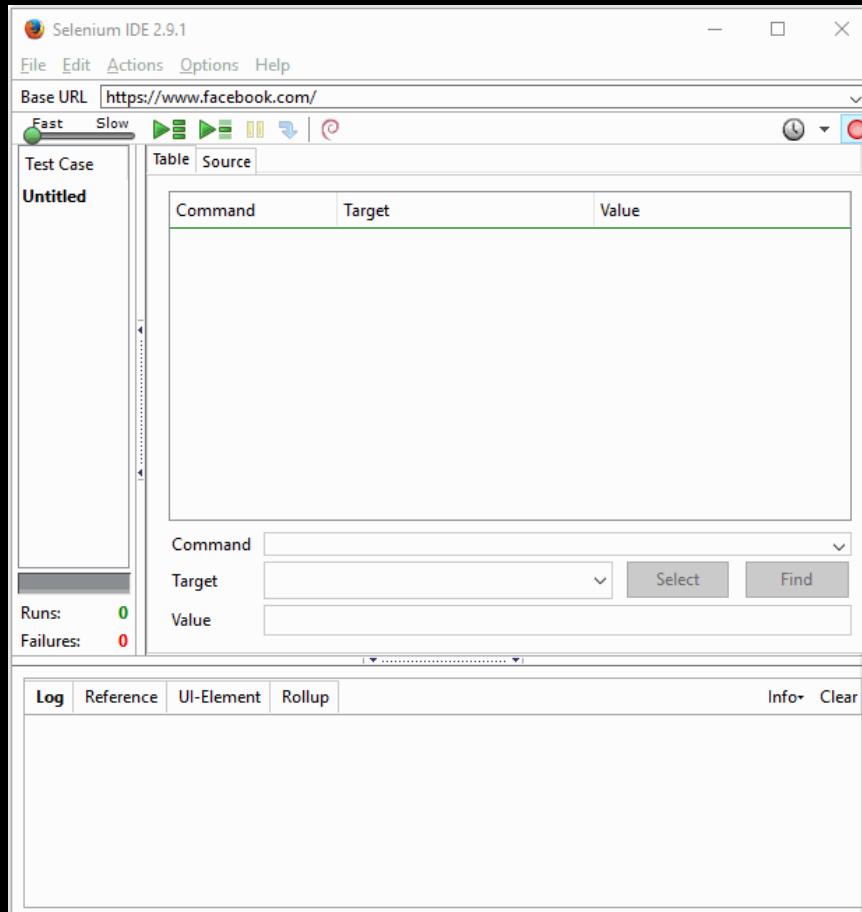
Selenese language

Type id=UH Selenium



UH Selenium

Capture and Replay approach



Selenium IDE 2.9.1

File Edit Actions Options Help

Base URL <https://www.facebook.com/>

Test Case Untitled

Runs: 0 Failures: 0

Log Reference UI-Element Rollup Info Clear

Facebook - Log In or Sign ... X +

Search Forgotten your password?

Sign Up

It's free and always will be.

First name Surname

Mobile number or email address

Re-enter mobile number or email address

New password

Birthday

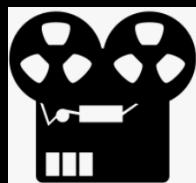
Day Month Year Why do I need to provide my date of birth?

Female Male

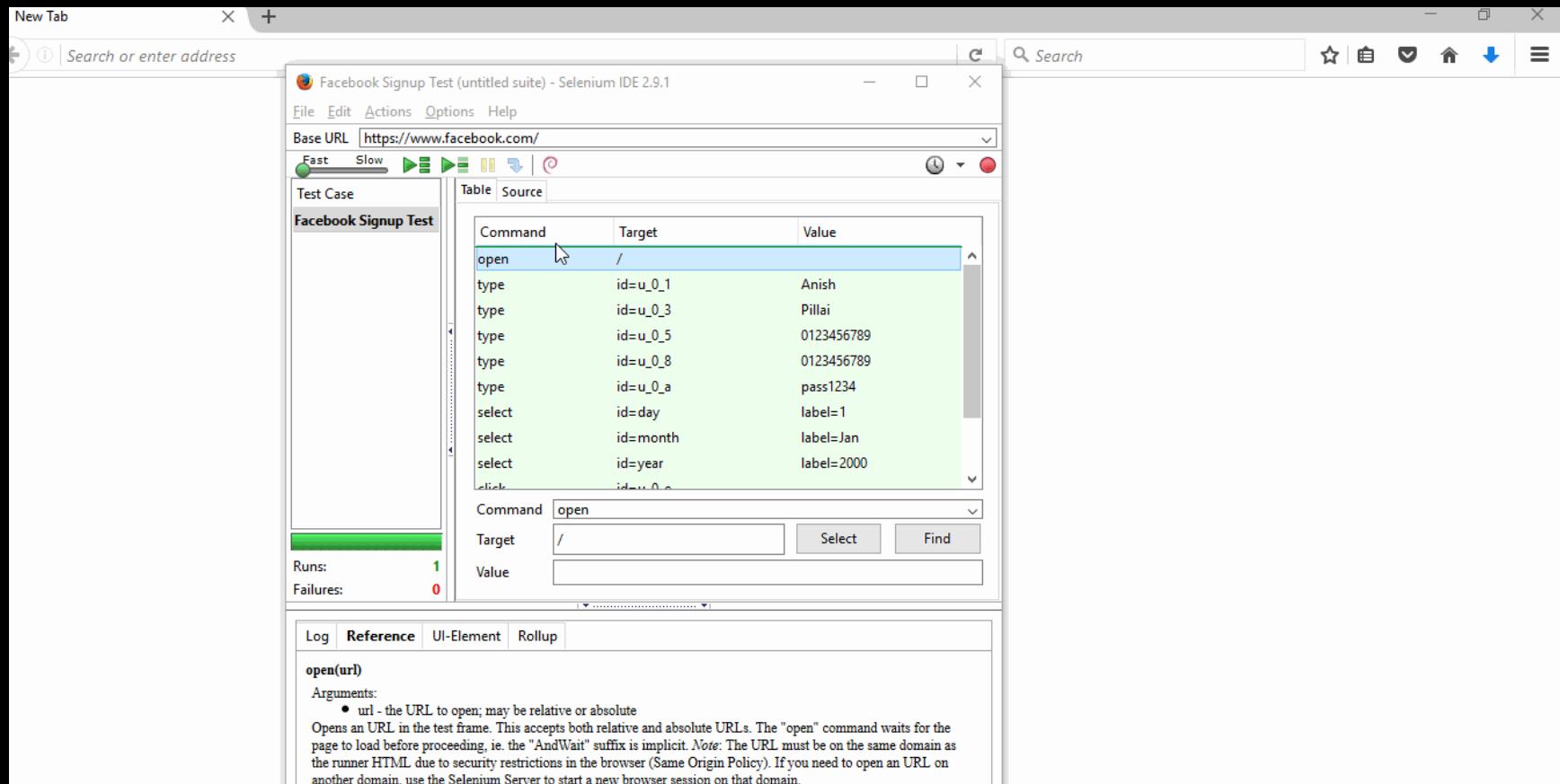
By clicking Sign Up, you agree to our [Terms](#) and that you have read our [Data Policy](#), including our [Cookie Use](#).

Sign Up

The screenshot shows the Selenium IDE interface on the left, connected to a Facebook sign-up page on the right. The IDE window displays a test case titled 'Untitled' with a table for recording commands. The Facebook page shows the sign-up form with fields for first name, surname, mobile number, password, birthday, gender selection, and a terms and conditions agreement. A large green 'Sign Up' button is at the bottom.



Capture and **Replay** approach



Selenium IDE example

Web App under test

The screenshot shows a Google search results page for the query "ERC". The search bar at the top contains "ERC". Below the search bar, there are navigation links for "Web", "Images", and "More". The first search result is a link to "ERC: European Research Council" with the URL "erc.europa.eu/". The snippet below the link describes the European Research Council as a funding body set up to stimulate scientific excellence. At the bottom of the snippet, there are links for "Funding and Grants", "About ERC", "Contact us", and "Projects and Results".

Test case

1. Open www.google.com
2. Insert "ERC" in input field
3. Click submit button
4. Check first result = "ERC: European Research Council"

Executable test case (Test script)



Selenium IDE

Test Script 1

| | | |
|--------------|---|--------------------------------|
| open | http://www.google.com | |
| type | name=q | ERC |
| clickAndWait | id=id_fZI | Target/Locator Assertion |
| assertText | path=//*[@@id='res1']/h3/a | ERC: European Research Council |

Assertions must be added manually using Selenium IDE



Selenium WebDriver



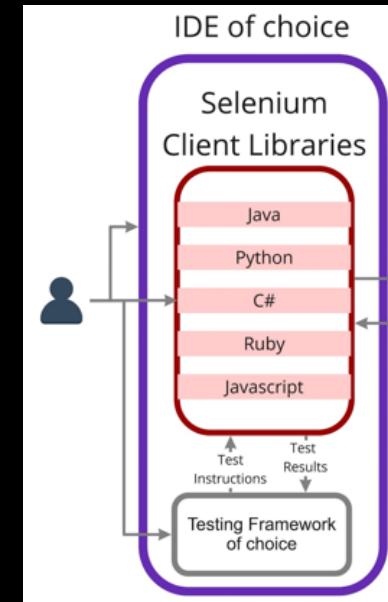
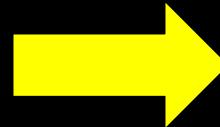
- **Selenium-WebDriver**
 - Control the browser **by programming**
 - Primarily it is for automating web applications for **testing purposes**, but is certainly not limited to just that
 - boring web-based administration tasks can be automated as well ...
 - Different programming languages supported
 - Java, JavaScript, Python, Ruby, C# ...
- **Selenium-WebDriver supports multiple browsers**



Programmable (Script-based) category!

Selenium WebDriver ‘testing’

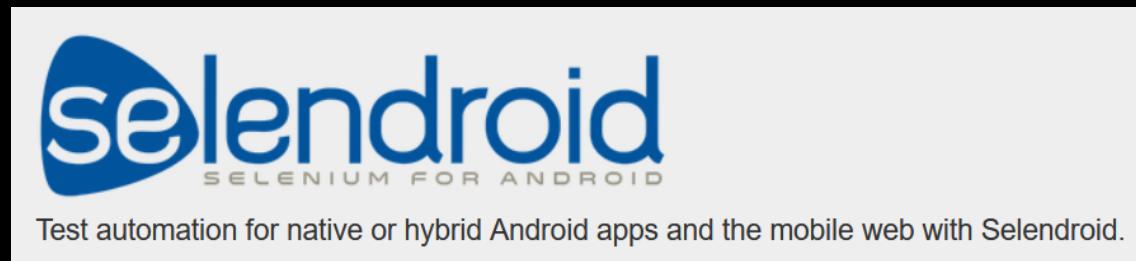
- For **E2E** testing purposes of Web apps
 - In combination with **JUnit** or **TestNG**
 - Setup/Test/TearDown
 - Assertions
 - Acceptance/Regression testing of Web apps
- For **browser compatibility** testing
- For **security** testing purposes
 - Brute force attacks
 - SQL injection, Password security ...
 - to show/manifest the presence of a vulnerability
 - E.g., Integration with OWASP ZAP (<https://www.zaproxy.org/>)
- For **performance** testing puropses
 - E.g., Integration with Apache Jmeter (<https://jmeter.apache.org/>)



For different types of testing!

Can Selenium be used for Mobile Testing?

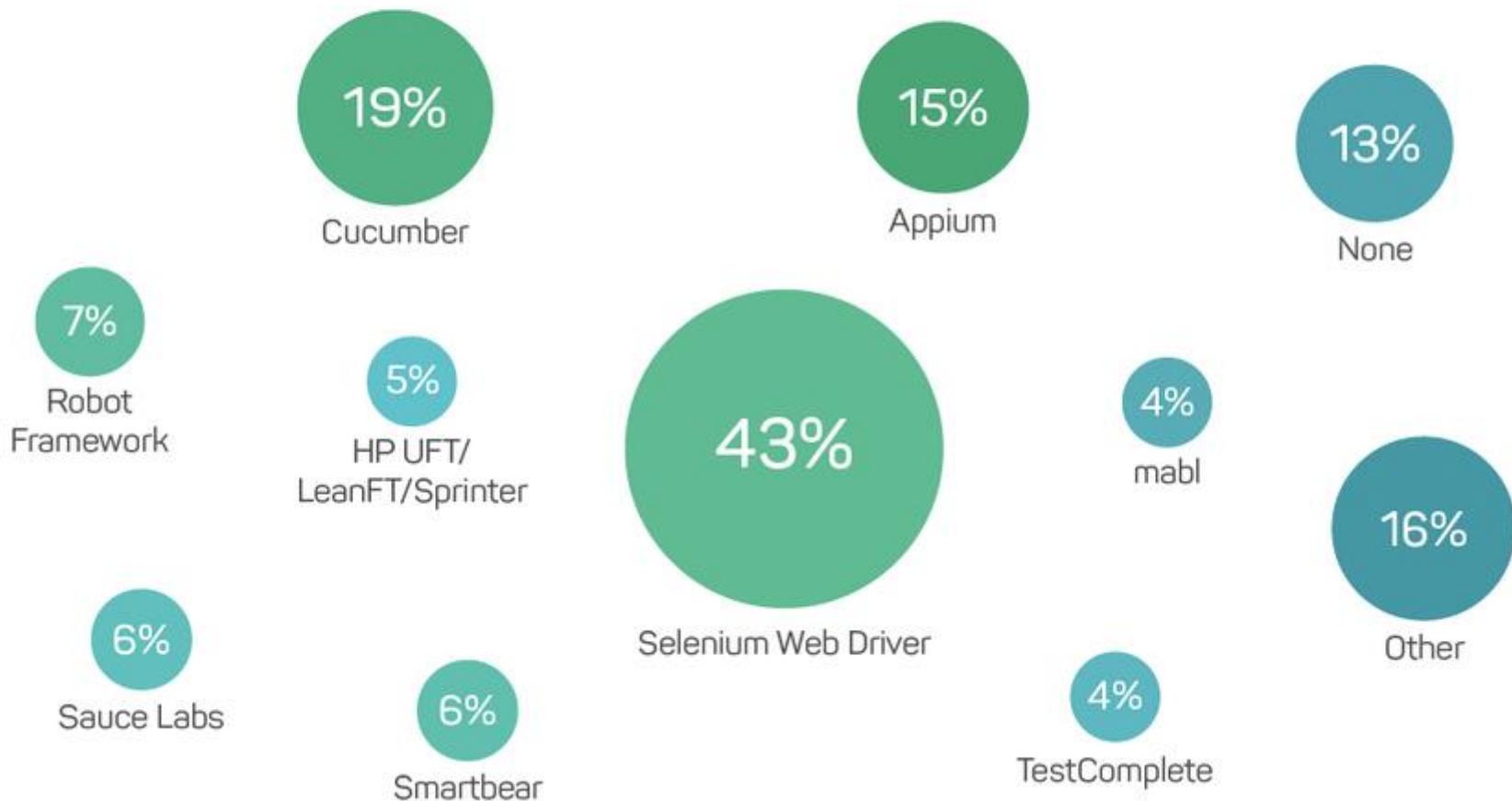
- Not really ...
- You cannot use Selenium for mobile app testing, but there are **frameworks based on Selenium** specific for **Mobile automated testing**
- These frameworks are:
 - Selendroid
 - Appium



Most Popular Testing framework

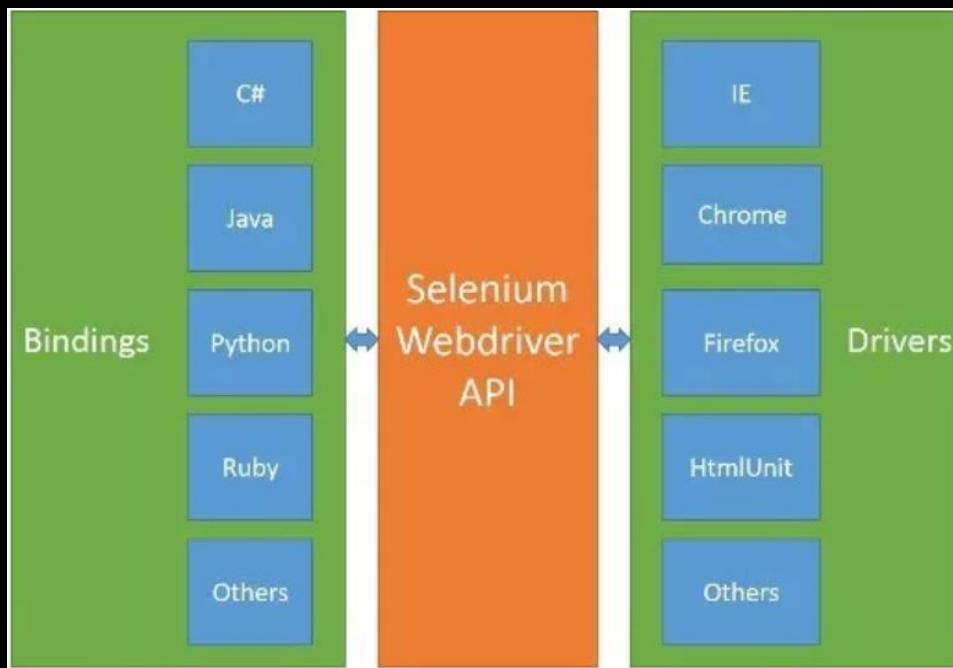
"What testing tools do you use?

Select all that apply."



Selenium WebDriver

- WebDriver is designed to provide a simple and uniformed programming interface (**API**)
 - *A well designed Object Oriented API that provides improved support for Web-app testing*



**'API = set of classes, methods, ...
usable in test scripts!'**

**.get()
.click()
.sendKeys()**

Selenium WebDriver example

Web App under test

The screenshot shows a Google search results page for the query "ERC". The search bar at the top contains "ERC". Below the search bar, there are three tabs: "Web", "Images", and "More". The "Web" tab is selected. The first search result is a link to "ERC: European Research Council" with the URL "erc.europa.eu/". The snippet below the link describes the European Research Council as a funding body for scientific excellence. At the bottom of the snippet, there are links for "Funding and Grants", "About ERC", "Contact us", and "Projects and Results".

Test case

1. Open www.google.com
2. Insert “ERC” in input field
3. Click submit button
4. Check first result = “ERC: European Research Council”



Selenium WebDriver

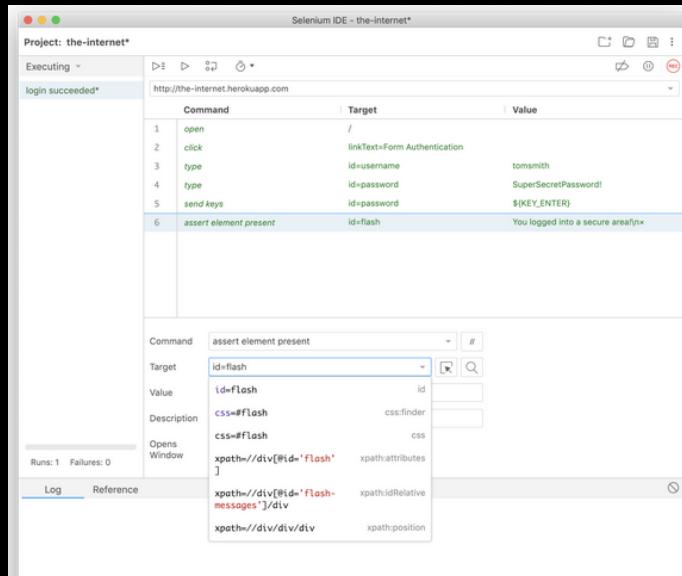
Executable test case (Test script)

```
public void searchTest() {
    WebDriver driver = new FirefoxDriver();
    driver.get("http://www.google.com");
    WebElement element = driver.findElement(By.name("q"));
    element.sendKeys("ERC");
    element.submit();
    WebElement res = driver.findElement(By.xpath("//*[@id='res1']/h3/a"));
    assertEquals(res.getText(), "ERC: European Research Council");
    driver.quit();
}
```

Locator: A red box highlights the line `driver.findElement(By.name("q"));`. An arrow points from the word "Locator" to this box.

JUnit Assertion: A green box highlights the line `assertEquals(res.getText(), "ERC: European Research Council");`. The words "JUnit Assertion" are written below it.

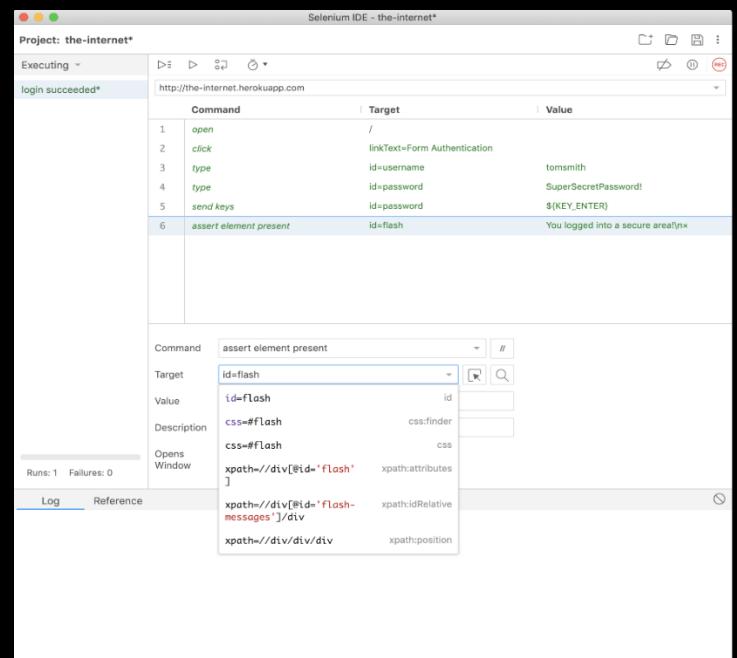
Selenium IDE



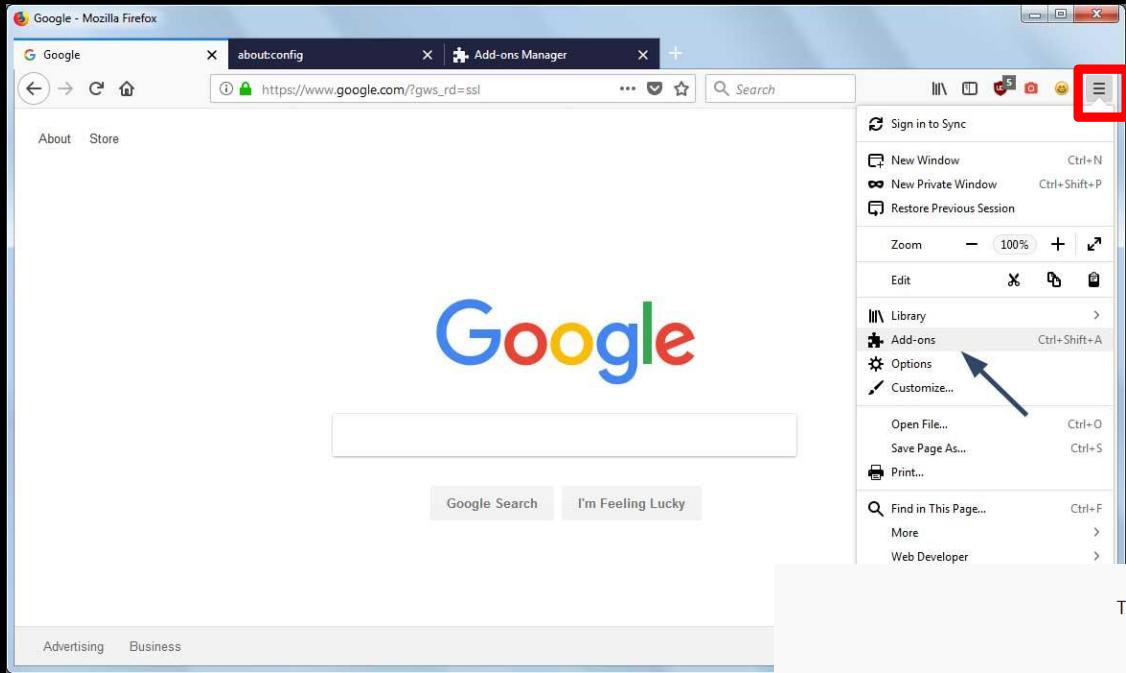
Selenium IDE Recorder

- Firefox/Chrome/Edge extension!
 - Add-on simple to install
- Offered functionalities:
 - Easy '**record and replay**'
 - Test scripts editing (**IDE**)
 - **Debug**: set breakpoints, execute step by step, pause
 - Save (import/export)
 - Test scripts and test suites
 - format '.side'
 - Export in Webdriver test scripts

Selenium IDE Recorder - UI



Add-on simple to install



Depending on the version the Firefox menu could change

A screenshot of the Firefox Add-ons Manager. At the top, there's a search bar with "Trova altre estensioni" and a search term "Selenium IDE". Below it, there are two sections: "Gestione estensioni" and "Attive". Under "Attive", three extensions are listed: "Avast Online Security", "ChroPath", and "Cisco Webex Extension". Each extension has a toggle switch and a more options button. An arrow points from the text "Depending on the version the Firefox menu could change" towards the "Attive" section of the add-ons manager.

Selenium test script

- During recording Selenium saves all information in a Table

Selenese language

| Command | Target | Value |
|---------|------------------------------|-------|
| open | http://softeng.disi.unige.it | |
| click | link=Projects | |
| click | xpath=(//a[contains(text(), |)]) |

- Each line/record consists of:

- Command** – tells Selenium what to do
 - e.g. “open”, “type”, “click”
- Target** – tells Selenium which HTML element a command refers to
 - e.g. textbox, header, table
- Value** – used for any command that might need a value of some kind
 - e.g. type something into a textbox

Locator
(e.g., id=search_input)

Type id=UH Selenium → UH Selenium

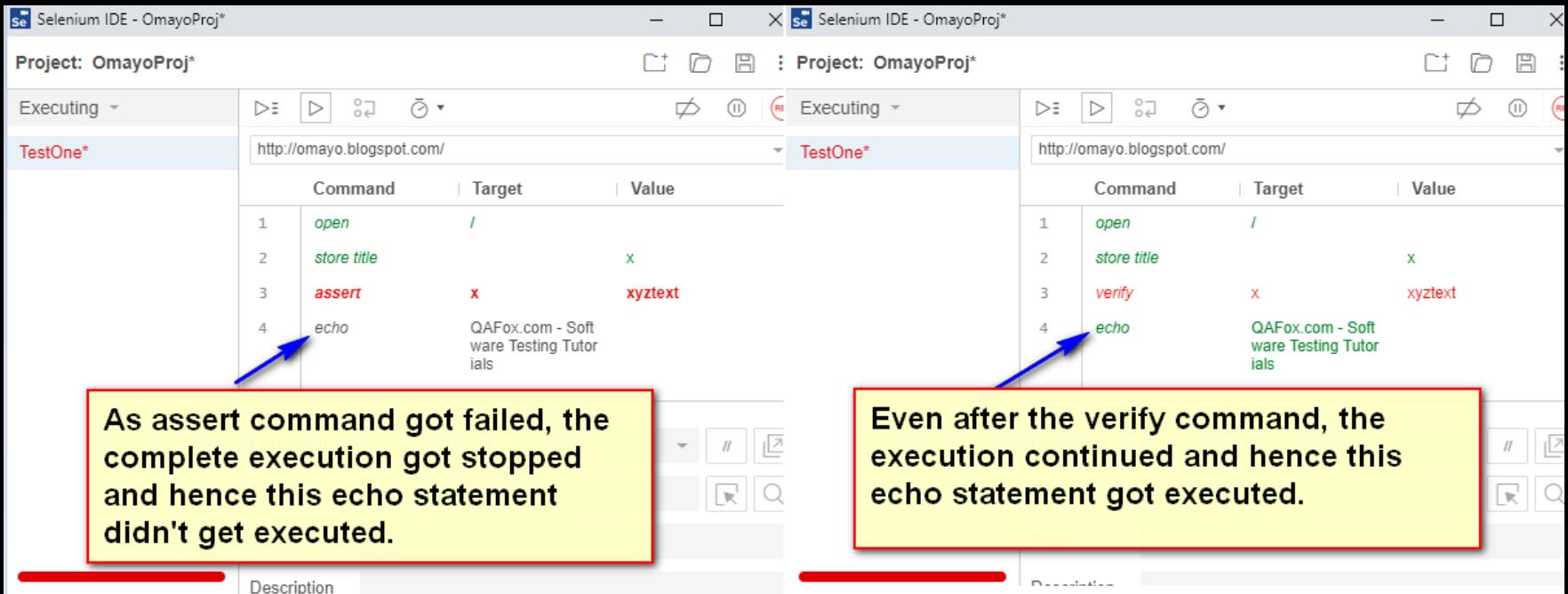
Assertions in Selenium IDE

- Selenese allows multiple ways of checking for UI elements using **assertions**
- Selenium IDE offers some predefined-assertions
- Assertions are used to check if, e.g.,:
 - a web element is present somewhere on the page?
 - **assert element present** id=idgv
 - a web element with id ‘equal to’ idgv
 - specific text is at a specific location on the page?
 - **assert text** id=firstHeading Genova
 - the text ‘Genova’ is in the page in a web element with id ‘equals to’ firstHeading
 - Is the title of the page ‘XYZ’?
 - **assert title** XYZ

Much more on this ...

Not only Assertions ...

- **Assertions** (hard) and **verifications** (soft assertions) are not the same!
- If an assertion fails, the script will be aborted but if a verification fails the script will continue



| | Command | Target | Value |
|---|-------------|--|---------|
| 1 | open | / | |
| 2 | store title | | x |
| 3 | assert | x | xyztext |
| 4 | echo | QAFox.com - Soft ware Testing Tutor ials | |

↑

As assert command got failed, the complete execution got stopped and hence this echo statement didn't get executed.

| | Command | Target | Value |
|---|-------------|--|---------|
| 1 | open | / | |
| 2 | store title | | x |
| 3 | verify | x | xyztext |
| 4 | echo | QAFox.com - Soft ware Testing Tutor ials | |

↑

Even after the verify command, the execution continued and hence this echo statement got executed.

How to add Assertions/Verifications

assert text 'locator' 'text'

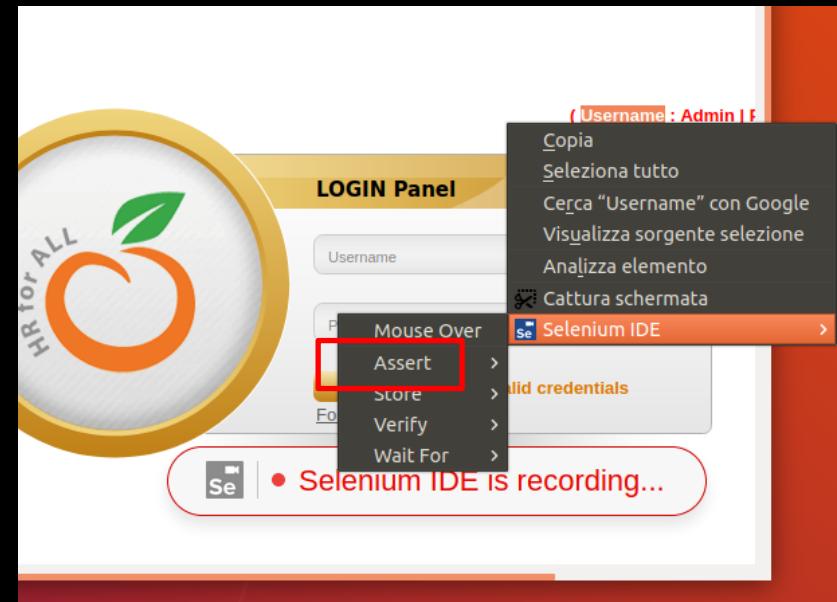
➤ This command is used to check if a particular text is present in a specific location or not

➤ To add this command, **right click on the text item** that you want to verify

➤ **During recording**

➤ Once right clicked you can find an option “**Selenium IDE**”. On clicking it you will find different options

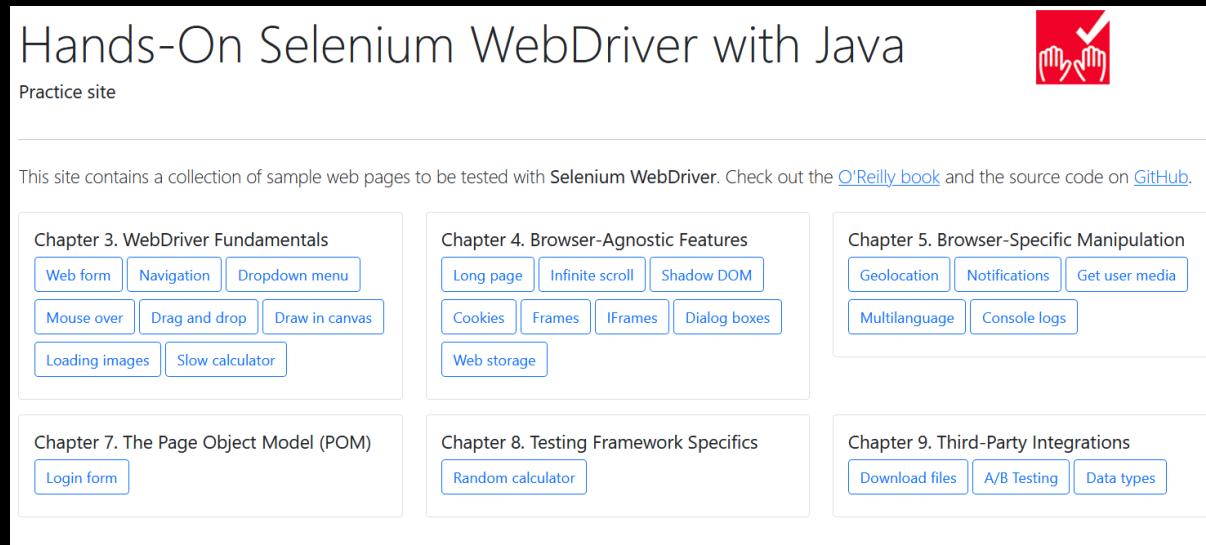
➤ Assert -> text, select it



Our first E2E test in Selenium IDE

- Open the page

<https://bonigarcia.dev/selenium-webdriver-java/>



- Click on Web form
- Check that the title is ‘Hands-On Selenium WebDriver with Java’
- Close the browser

Creating a Testsuite

Selenium IDE - OmayoProj*

Project: OmayoProj*

Tests ▾ +

Tests Ctrl+1
Test suites Ctrl+2
Executing Ctrl+3

http://omayo.blogspot.com

| | Command | Target | Value |
|---|-----------------|-------------------------------------|----------|
| 1 | open | / | |
| 2 | set window size | 1366x728 | |
| 3 | click | name=q | |
| 4 | type | name=q | QAFox.co |
| 5 | click | css=.gsc-search-button:nth-child(1) | |

TestX*
TestY*
TestZ*

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites ▾ +

Search tests...

http://omayo.blogspot.com

Default Suite

| | Command | Target |
|---|-----------------|----------|
| 1 | open | / |
| 2 | set window size | 1366x728 |

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites ▾ +

Search tests...

Default Suite

Add new suite

SUITE NAME

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites ▾ +

Search tests...

http://omayo.blogspot.com

Default Suite

TestSuiteOne

| | Command | Target |
|---|-----------------|-------------------------------------|
| 1 | open | / |
| 2 | set window size | 1366x728 |
| 3 | click | name=q |
| 4 | type | name=q |
| 5 | click | css=.gsc-search-button:nth-child(1) |

Add tests to a Testsuite

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites + ▶ Default Suite ▶ TestSuiteOne

Search tests... http://omayo.blogspot.com

| | Command | Target | Value |
|---|-----------------|-------------------------------------|-----------|
| 1 | open | / | |
| 2 | set window size | 1366x728 | |
| 3 | click | name=q | |
| 4 | type | name=q | QAFox.com |
| 5 | click | css=.gsc-search-button:nth-child(1) | |

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites + ▶ Default Suite ▶ TestSuiteOne

Search tests... http://omayo.blogspot.com

Command Target Value

| | | | |
|---|-----------------|-------------------------------------|-----------|
| 1 | open | / | |
| 2 | set window size | 1366x728 | |
| 3 | click | name=q | |
| 4 | type | name=q | QAFox.com |
| 5 | click | css=.gsc-search-button:nth-child(1) | |

Add tests ▶ Rename Delete Settings Export

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites + ▶ Default Suite ▶ TestSuiteOne

Search tests... http://omayo.blogspot.com

Select tests

TestOne
TestThree
TestTwo
TestX
TestY
TestZ

SEARCH TESTS

SELECT CANCEL

Selenium IDE - OmayoProj*

Project: OmayoProj*

Test suites + ▶ Default Suite ▶ TestSuiteOne*

TestOne*
TestThree*
TestTwo*

Search tests... http://omayo.blogspot.com

Command Target Value

| | | | |
|---|-----------------|-------------------------------------|-----------|
| 1 | open | / | |
| 2 | set window size | 1366x728 | |
| 3 | click | name=q | |
| 4 | type | name=q | QAFox.com |
| 5 | click | css=.gsc-search-button:nth-child(1) | |

Organising Test in Testsuite

<https://www.youtube.com/watch?v=aWyKKIlepxQ>

Test: Successfully Login example

<https://bonigarcia.dev/selenium-webdriver-java/login-form.html>

Hands-On Selenium WebDriver with Java

Practice site

Login form

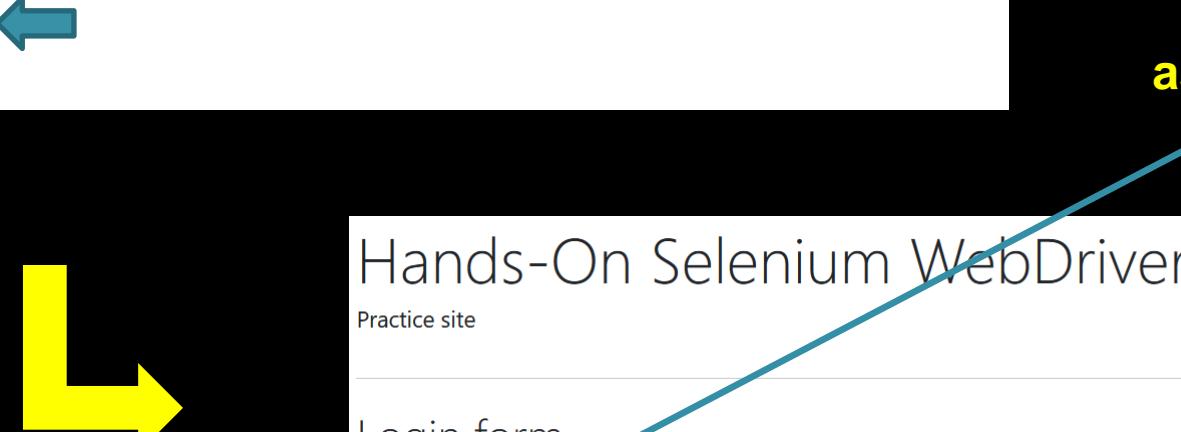
Login

user

Password

user

Submit



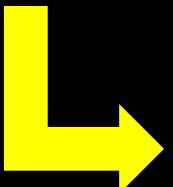
assertion

Hands-On Selenium WebDriver with Java

Practice site

Login form

Login successful



Test: Unsuccessfully Login example

<https://bonigarcia.dev/selenium-webdriver-java/login-form.html>

Hands-On Selenium WebDriver with Java

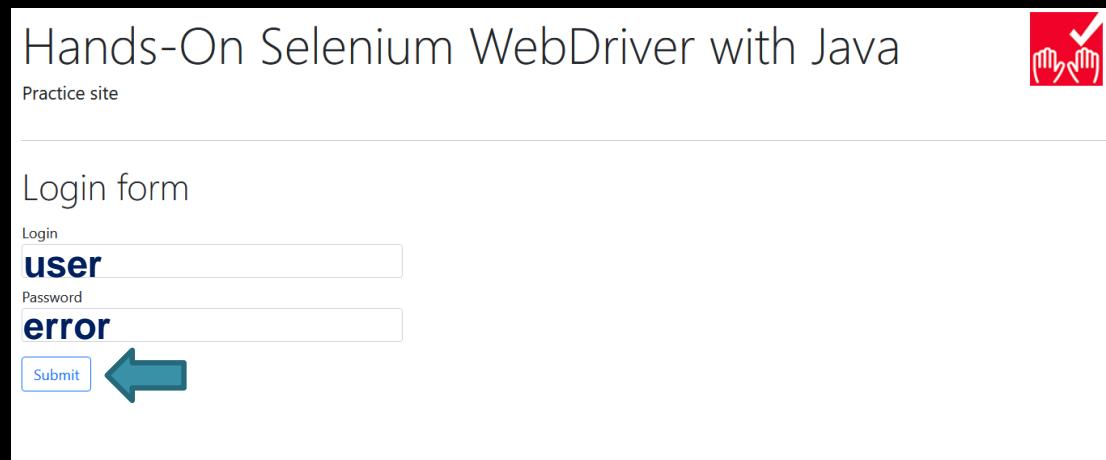
Practice site

Login form

Login
user

Password
error

Submit



A screenshot of a web browser displaying a login form. The 'Login' field contains the text 'user'. The 'Password' field contains the text 'error'. Below the form is a red rectangular box containing the text 'Invalid credentials'. A blue arrow points from the word 'error' in the password field towards this red box.

assertion

Hands-On Selenium WebDriver with Java

Practice site

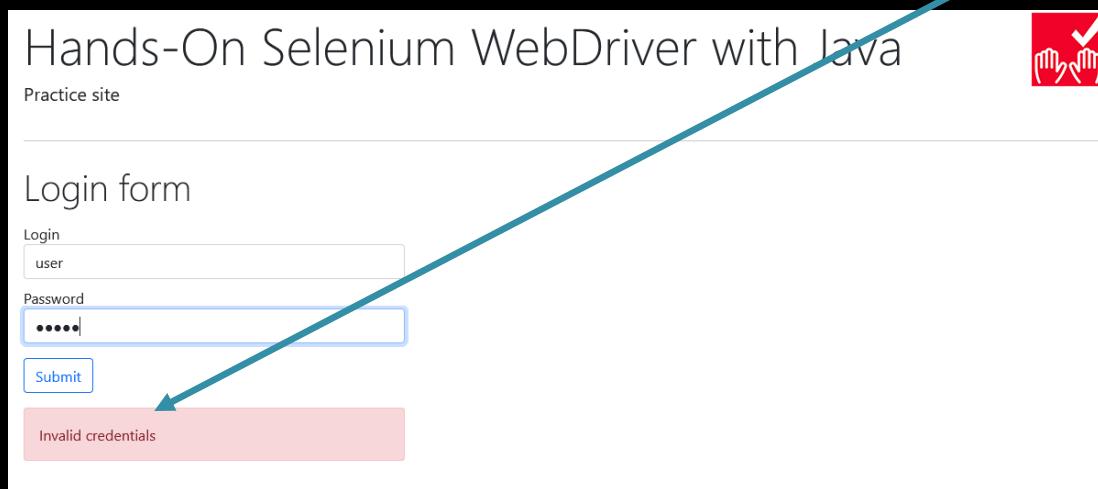
Login form

Login
user

Password
.....

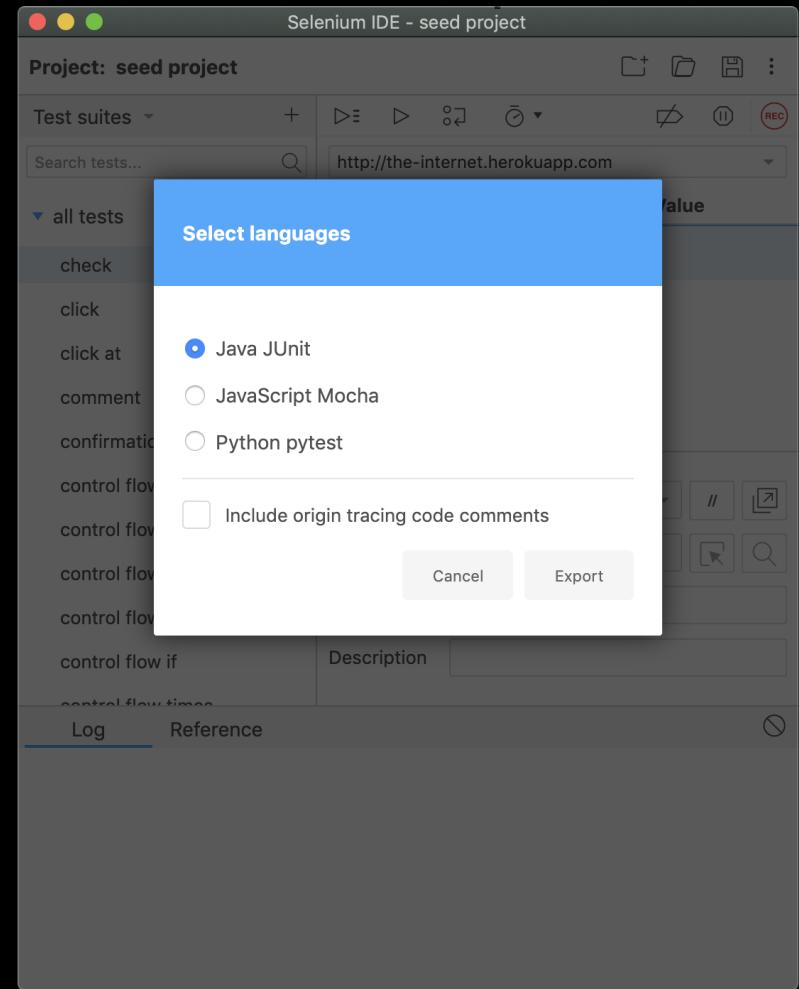
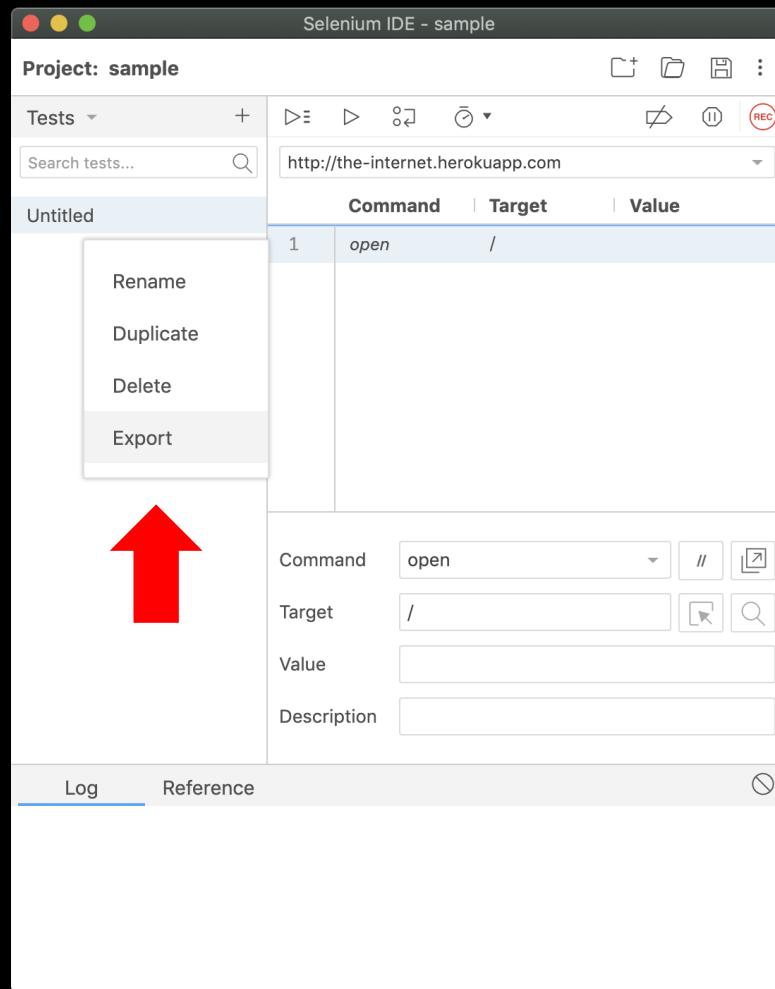
Submit

Invalid credentials



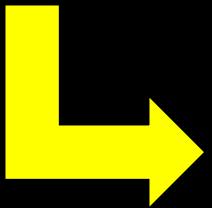
A screenshot of a web browser displaying a login form. The 'Login' field contains the text 'user'. The 'Password' field contains the text '.....'. Below the form is a red rectangular box containing the text 'Invalid credentials'. A large yellow L-shaped arrow points from the top-left towards this red box.

Export



Export example

| Command | Target | Value |
|-------------------|--|-------|
| 1 open | /selenium-webdriver-java/ | |
| 2 set window size | 1198x672 | |
| 3 click | linkText=Web form | |
| 4 assert title | Hands-On Selenium Web Driver with Java | |



```
public class FirsttestTest {
    private WebDriver driver;
    private Map<String, Object> vars;
    JavascriptExecutor js;
    @Before
    public void setUp() {
        driver = new FirefoxDriver();
        js = (JavascriptExecutor) driver;
        vars = new HashMap<String, Object>();
    }
    @After
    public void tearDown() {
        driver.quit();
    }
    @Test
    public void firsttest() {
        driver.get("https://bonigarcia.dev/selenium webdriver java/");
        driver.manage().window().setSize(new Dimension(1198, 672));
        driver.findElement(By.linkText("Web form")).click();
        assertThat(driver.getTitle(), is("Hands-On Selenium WebDriver with Java"));
    }
}
```

Time issue

- There are **delays** between submitting a request and receiving the response



- **Manual testing:** a Human tester can wait until the response page is loaded
 - A Testing framework/tool doesn't know!
- Sometimes a test script doesn't work if
 - Submit a request
 - Verify the response immediately
- **Solution:**
 - **Adding an explicit wait!**
 - It stops the testing framework until some HTML object appears

Test: slow calculator

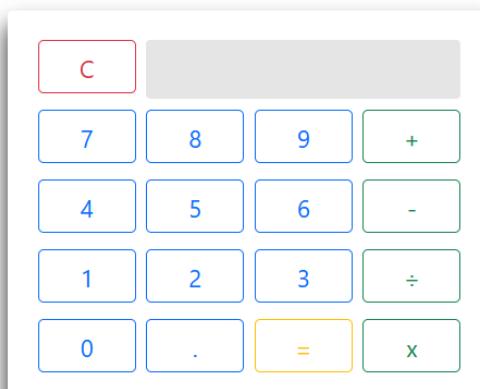
- <https://bonigarcia.dev/selenium-webdriver-java/slow-calculator.html>

Hands-On Selenium WebDriver with Java 

Practice site

Slow calculator

This calculator waits seconds to get the result of basic arithmetic operations.



The calculator interface consists of a grid of buttons. The top row contains a red 'C' button and a grey clear display area. Below are four rows of four buttons each: the first three rows contain digits 7, 8, 9; 4, 5, 6; and 1, 2, 3 respectively, with the fourth button in each row being an operator (+, -, ÷, x). The bottom row contains a blue '0' button, a blue '.' button, an orange '=' button, and a green 'x' button.

Copyright © 2021-2022 [Boni García](#)

Selenium IDE Waits

- **Built in wait strategies** in the Selenium IDE aren't always sufficient
 - Selenium IDE has a '**default waiting mechanism**'
- When that happens, you can use one of the available '**wait**' commands
 - Set execution speed
 - set a delay which will follow each Selenium command
 - **Pause** (Stop execution for a given fixed time)
 - **wait for element present**
 - **wait for text**
 - **wait for element visible**
 -

| Command | Target | Value |
|---------|--------------------------------|-------|
| open | http://omayo.blogspot.com/ | |
| type | name=q | QAFox |
| pause | 10000 | |
| click | xpath=//input[@value='Search'] | |

Test: Images

- <https://bonigarcia.dev/selenium-webdriver-java/loading-images.html>

Hands-On Selenium WebDriver with Java



Practice site

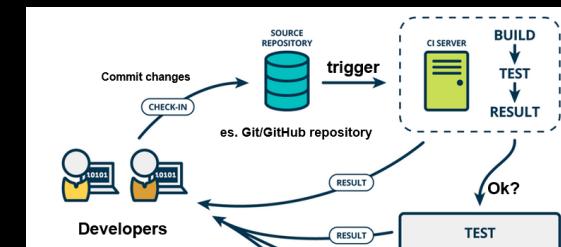
Loading images

Done!



Limitation of Selenium IDE

- Selenium IDE = **good solution for small test suites and quick sessions of testing!**
- Not suitable for creating '**real big**' industrial test suites
- It is difficult:
 - Creating complex test scripts and data-driven test scripts
 - Conditions, loops
 - Executing Browser-Compatibility Testing
 - Maintaining test scripts
 - Using the produced test scripts in a modern software development process
 - ...



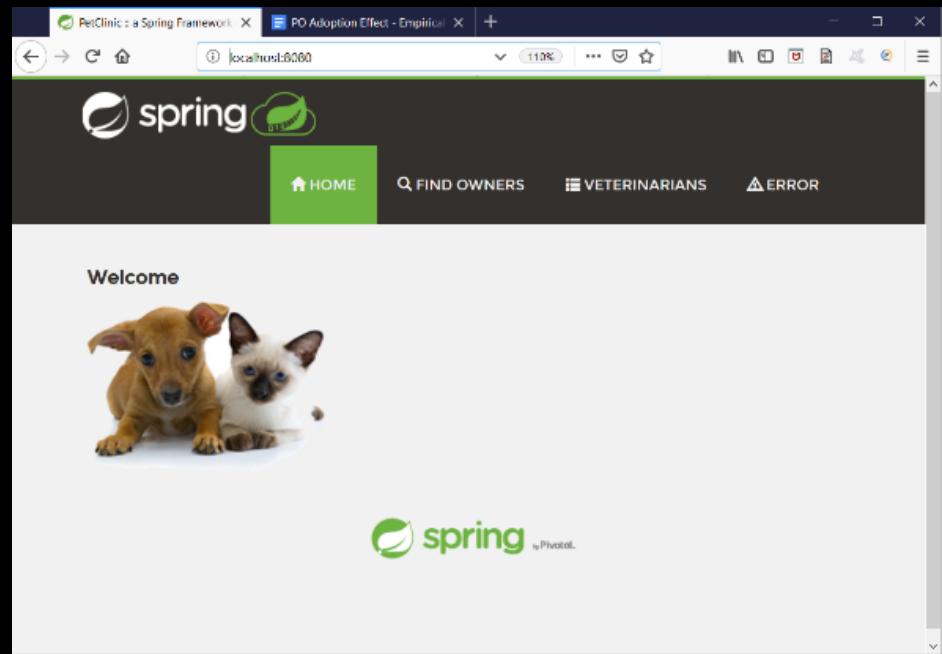
Solution? --> using the programmable approach!

Selenium IDE video tutorial

https://www.youtube.com/watch?v=4I7xay_NV8A

Assignment (not evaluated)

- Building a Selenium IDE testsuite for Petclinic
- You can start from the Gherkin test cases you built



Functional and Security Testing Course

2023-2024

Selenium WebDriver: Architecture and SetUp – lesson 7

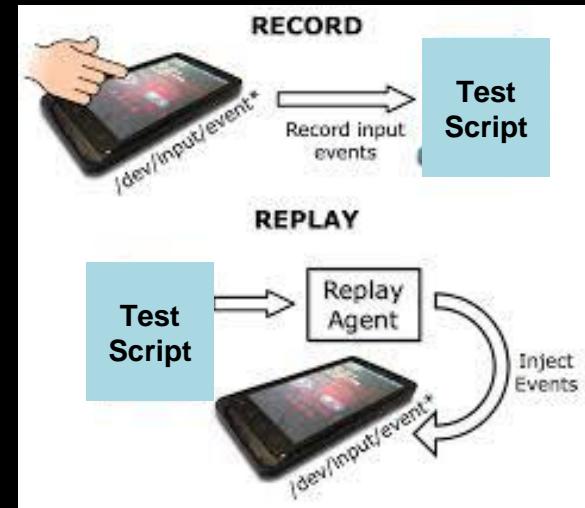
Filippo Ricca
DIBRIS, University of Genova

How Test scripts can be produced

Two ways:

- Capture/Replay

- Recording the actions performed by the tester
 - using a specific tool
 - e.g., Selenium IDE or Katalon recorder
 - Manual Fix
- Re-executing them automatically



- Programmable

- Test scripts are **software artefacts** created:
 - Using standard programming languages and IDE
 - e.g., Java + Eclipse
 - Resorting to specific testing frameworks
 - e.g., Selenium WebDriver

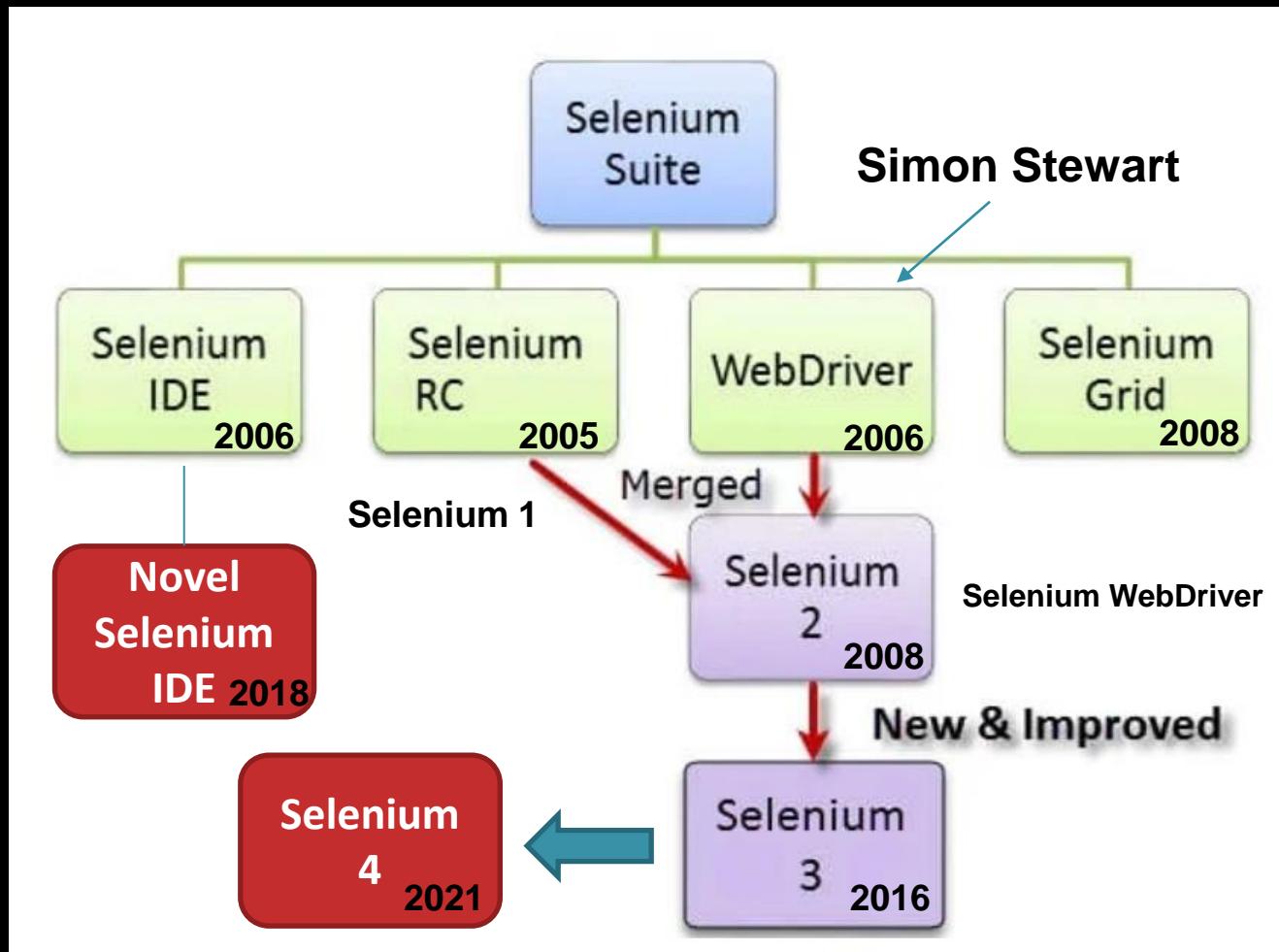


Why the name Selenium?

- The name **Selenium** came from a **joke**
- During Selenium's development, another automated testing framework was popular made by the company called **Mercury Interactive**
 - **QTP** (now UFT) before it was acquired by HP
- Since Selenium is a well-known **antidote for Mercury poisoning** they decided to call the testing framework in this way



Selenium Ecosystem History





Selenium WebDriver 4



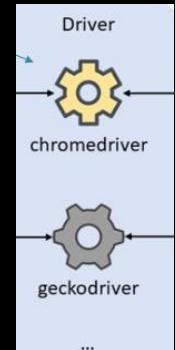
- **Selenium-WebDriver**
 - Control the browser **by programming**
 - Primarily it is for automating web applications for **testing purposes**, but is certainly not limited to just that
 - boring web-based administration tasks can be automated as well ...
 - Different programming languages supported
 - Java, JavaScript, Python, Ruby, C# ...
- **Selenium-WebDriver supports multiple browsers**



Programmable (Script-based) category!

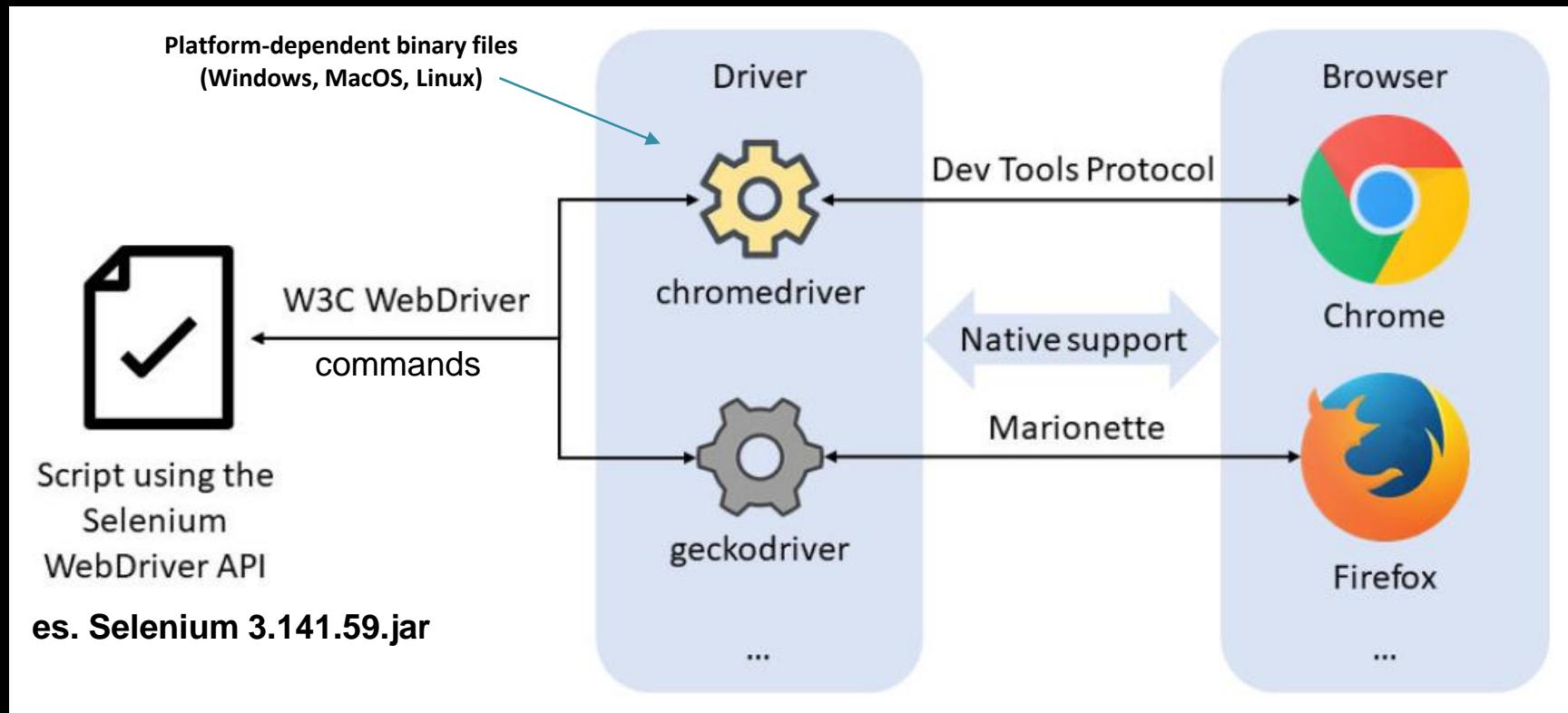
Selenium WebDriver 4

- Selenium WebDriver provides **a cross-platform API** in different languages
 - e.g. Interface WebDriver, methods click(), get(), ...
- Selenium WebDriver 4 uses **the native support implemented by each browser** to carry out the automation process
- For this reason a component called **Driver** between the script using the Selenium Web-Driver API and the browser is needed!



The driver receives commands from the script and transform them in actions

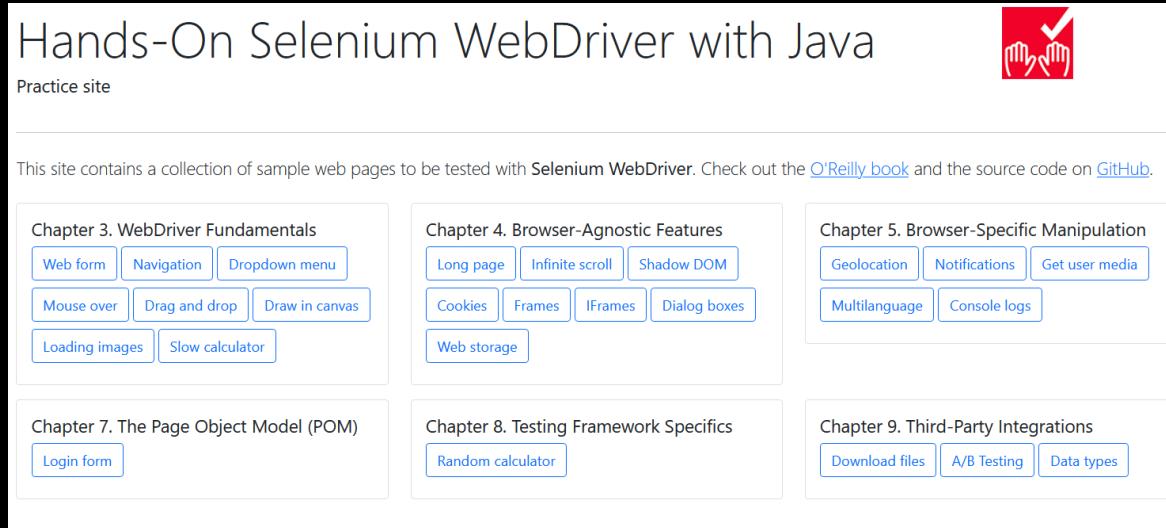
Architecture of Selenium WebDriver 4



In the first releases of Selenium WebDriver (i.e., in Selenium 2), these commands (also known as the *Selenium protocol*) were JSON messages over HTTP (the so-called *JSON Wire Protocol*). Nowadays, this communication follows a standard specification named *W3C WebDriver*.

Our first end-to-end test

- Open the following page with Chrome
- <https://bonigarcia.dev/selenium-webdriver-java/>



- Get the title (*Hands-On Selenium WebDriver with Java*)
- Check that the title is ‘Hands-On Selenium WebDriver with Java’

A first look to the code

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // not complete ...  
        WebDriver driver = new ChromeDriver();  
  
        String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/";  
        String expectedTitle = "Hands-On Selenium WebDriver with Java";  
        String actualTitle = "";  
  
        // launch Chrome and direct it to the Base URL  
        driver.get(baseUrl);  
  
        // get the actual value of the title  
        actualTitle = driver.getTitle();  
  
        // compare the actual title of the page with the expected one"  
        if (actualTitle.contentEquals(expectedTitle)){  
            System.out.println("Test Passed!");  
        } else {  
            System.out.println("Test Failed");  
        }  
  
        // close Chrome  
        driver.quit();  
    }  
}
```

Test script using Selenium WebDriver API without JUnit

**VS Code Project (no build tools)
with Selenium WebDriver
3.141.59 (manual driver
management) and Chrome**

Necessary “ingredients”

- **Visual Studio Code**, recent version
 - E.g., version 1.87.2
- **Selenium WebDriver API**
 - Java language binding
 - Es. Selenium 3.141.59.jar
- Couple: ‘**Browser**’ – ‘**Driver**’



Manually build

| Browser | Driver |
|-----------------|--------------|
| Chrome/Chromium | chromedriver |
| Edge | msedgedriver |
| Firefox | geckodriver |

Selenium Webdriver API (official)



C#

Stable: [4.8.1 \(February 17, 2023\)](#)

[Changelog](#)

[API Docs](#)



Ruby

Stable: [4.8.1 \(February 17, 2023\)](#)

[Changelog](#)

[API Docs](#)



Java

Stable: [4.8.1 \(February 17, 2023\)](#)

[Changelog](#)

[API Docs](#)



Python

Stable: [4.8.2 \(February 18, 2023\)](#)

[Changelog](#)

[API Docs](#)

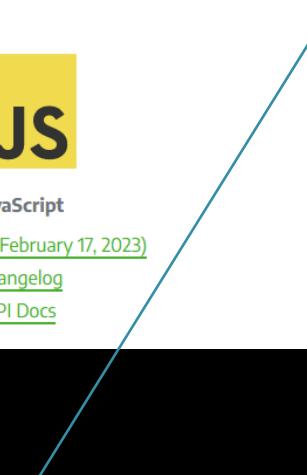


JavaScript

Stable: [4.8.1 \(February 17, 2023\)](#)

[Changelog](#)

[API Docs](#)



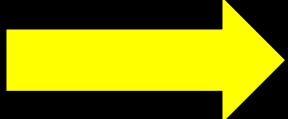
The libraries adding ‘Selenium commands’ to the ‘language’

‘Unofficial’ Selenium Webdriver API

| Name | Language | License | Maintainer | Website |
|----------------------------|------------|--------------|------------------------|---|
| hs-webdriver | Haskell | BSD-3-Clause | Adam Curtis | https://github.com/kallisti-dev/hs-webdriver |
| php-webdriver | PHP | MIT | Facebook, community | https://github.com/php-webdriver/php-webdriver |
| RSelenium | R | AGPLv3 | rOpenSci | https://github.com/ropensci/RSelenium |
| Selenium | Go | MIT | Miki Tebeka | https://github.com/tebeka/selenium |
| Selenium-Remote- Driver | Perl | Apache 2.0 | George S. Baugh | https://github.com/teodesian/Selenium-Remote-Driver |
| webdriver.dart | Dart | Apache 2.0 | Google | https://github.com/google/webdriver.dart |
| wd | JavaScript | Apache 2.0 | Adam Christian | https://github.com/admc/wd |

Task: download ‘Selenium API for Java’

- Open <https://www.selenium.dev/downloads/>
- Go to the link Previous Releases ‘here’
 - At the end of the page
- Select **Selenium 3.141.59** and download jar file
 - [selenium-server-standalone-3.141.59.jar](#)
 - Latest stable version **4.19.1**
- Copy the file in a folder



Previous Releases

| Name | Size |
|---|---------|
| IEDriverServer_Win32_3.141.59.zip | 1.0 MB |
| IEDriverServer_x64_3.141.59.zip | 1.2 MB |
| selenium-html-runner-3.141.59.jar | 12.9 MB |
| selenium-java-3.141.59.zip | 7.2 MB |
| selenium-server-3.141.59.zip | 10.0 MB |
| selenium-server-standalone-3.141.59.jar | 10.2 MB |

[CLOSE ▶](#)

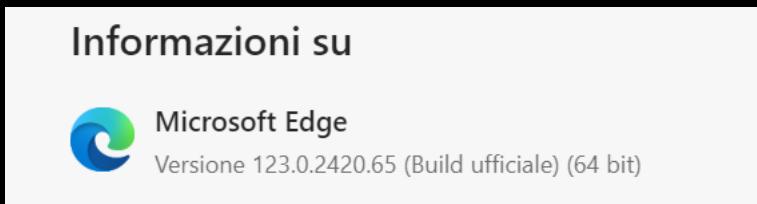
Browsers and drivers supported by Selenium WebDriver

| Browser | Driver | Operating System | Maintainer | Download |
|----------------------|----------------|-----------------------|-------------------|---|
| Chrome/ Chromium | chromedriver | Windows/ Mac/Linux | Google | https://chromedriver.chromium.org/ |
| Edge | msedgedriver | Windows/ Mac/Linux | Microsoft | https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/ |
| Firefox | geckodriver | Windows/ Mac/Linux | Mozilla | https://github.com/mozilla/geckodriver/ |
| Opera | operadriver | Windows/ Mac/Linux | Opera Software AS | https://github.com/operasoftware/operachromiumdriver |
| Internet Explorer | IEDriverServer | Windows | Selenium project | https://www.selenium.dev/downloads/ |
| Safari | safaridriver | Mac | Apple | Built-in |

https://www.selenium.dev/documentation/webdriver/getting_started/install_drivers/

Downloading ‘the proper driver’

- Chrome and Edge (although not Firefox, unfortunately) follow the same versioning schema for drivers and browsers
- So, for instance, if you use **Edge 123.0.2420.65**, you also need to use **msedgedriver 123.0.2420.65**



Release 123

Version 123.0.2420.81

x86 x64 Mac M1 Mac
Linux ARM64

Version 123.0.2420.65

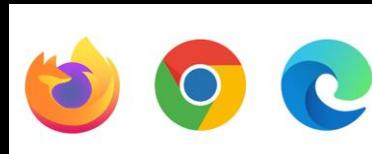
x86 x64 Mac M1 Mac
Linux ARM64

Version 123.0.2420.53

x86 x64 Mac M1 Mac
Linux ARM64

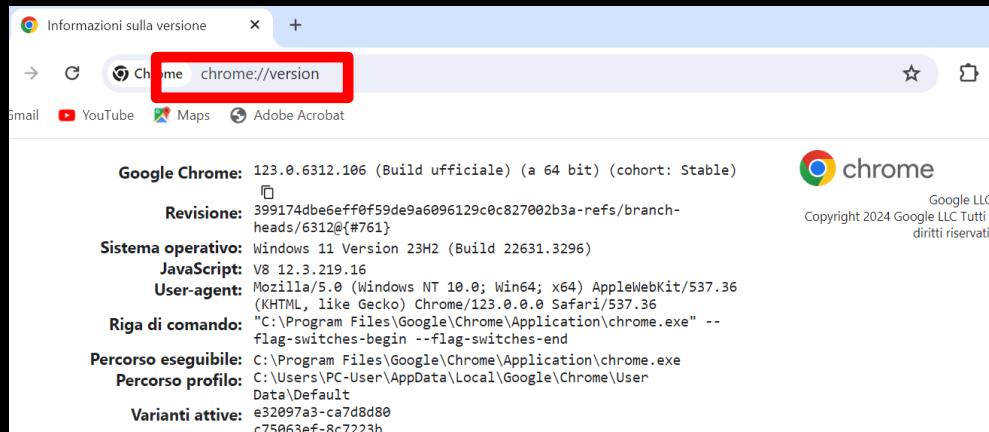
Version 123.0.2420.51

x86 x64 ARM64



Task: Download ChromeDriver

- Check **Google Chrome version**



- Open
 - https://www.selenium.dev/documentation/webdriver/getting_started/install_drivers/
- Select 'Downloads' in the line Chrome
- Select a compatible Driver with your Browser
 - to use Chrome 123, you need to download ChromeDriver 123.0.6312.106



Compatibility problem

- Because of the **automatic upgrade of Browsers**, the compatibility Browser-Driver is not guaranteed
- As a result, a Selenium WebDriver script based on an old driver probably stops working
 - Error: '*this version of chromedriver only supports chrome version N*'
- **Manual driver management** is costly in terms of effort and problematic in terms of maintenance



API 'complete' and 'complex'

Selenium Web Driver API

Changing frames & windows,
managing alerts, and cookies,
selecting drivers for browsers

Domain

Manage the automation speed, wait
for the web application events

Synchronisation

Selenium
WebDriver
Commands

Open a browser (`get()`), move to a page from
another one, back, forward, etc ([WebDriver
interface](#))

Navigation

Get information about the website and its
elements (`FindElement()`). ([WebDriver interface](#))

Interrogation

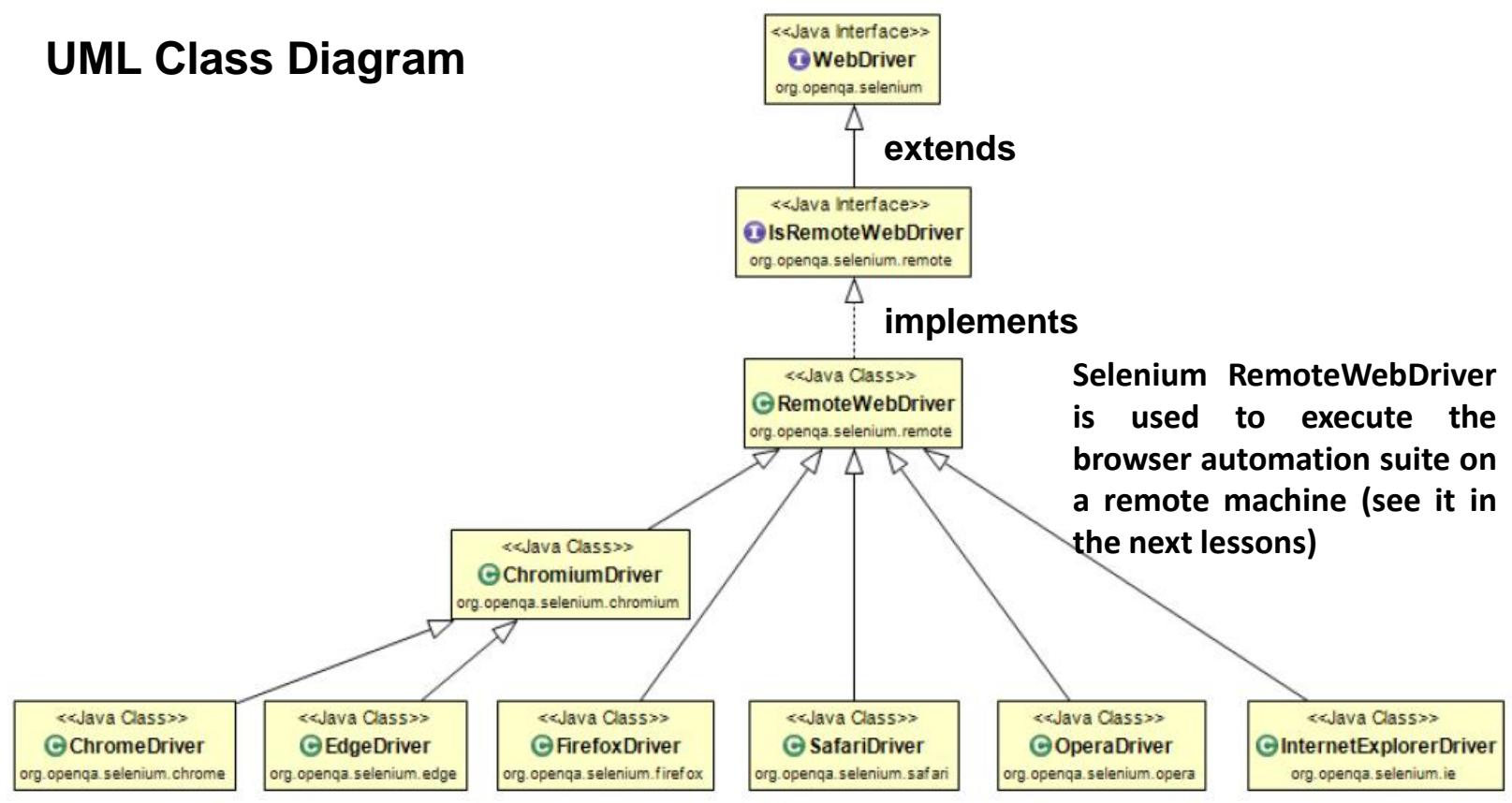
Clicking on links and buttons, filling
forms, cleaning texts, pressing keys, drag
& dropping ([WebElement class](#))

Manipulation

Commands belong to five categories

WebDriver interface

UML Class Diagram



```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

WebDriver driver = new ChromeDriver();
```

Manual Driver Management

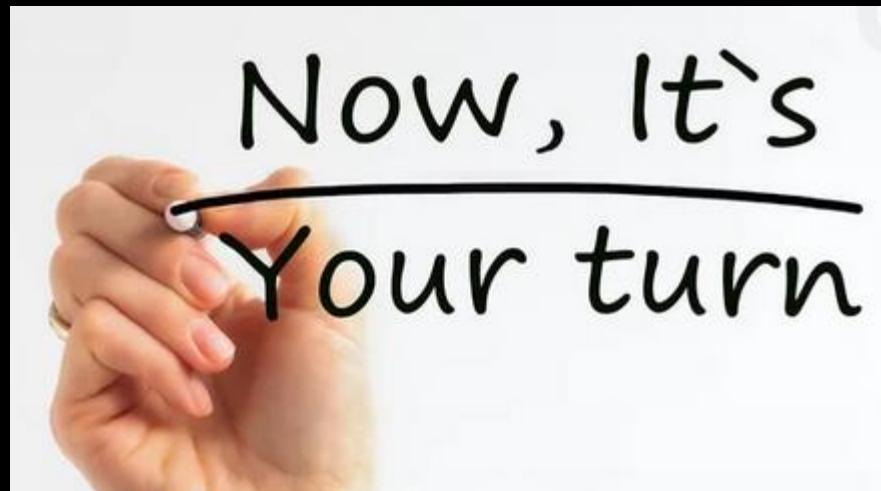
- Before instantiating an object of the WebDriver hierarchy, it is mandatory to resolve its corresponding driver
 - The test script needs to know where the driver is located in the file system
- For example, to control **Chrome** with **ChromeDriver**, first, we need to manage **chromedriver**
- If we manually make this management, our Java code will look like the following:

```
System.setProperty("webdriver.chrome.driver", "/path/to/drivers/chromedriver");
WebDriver driver = new ChromeDriver();
```

Example: first test in VS Code

```
System.setProperty("webdriver.chrome.driver", "/path/to/drivers/chromedriver");  
  
public class HelloWorld {  
    public static void main(String[] args) {  
        // not complete ... ←  
        WebDriver driver = new ChromeDriver();  
  
        String baseUrl = "https://bonigarcia.dev/selenium webdriver java/";  
        String expectedTitle = "Hands-On Selenium WebDriver with Java";  
        String actualTitle = "";  
  
        // launch Chrome and direct it to the Base URL  
        driver.get(baseUrl);  
  
        // get the actual value of the title  
        actualTitle = driver.getTitle();  
  
        // compare the actual title of the page with the expected one"  
        if (actualTitle.contentEquals(expectedTitle)){  
            System.out.println("Test Passed!");  
        } else {  
            System.out.println("Test Failed");  
        }  
  
        // close Chrome  
        driver.quit();  
    }  
}
```

- Creating a VS code project with Selenium 3.14 and Chrome

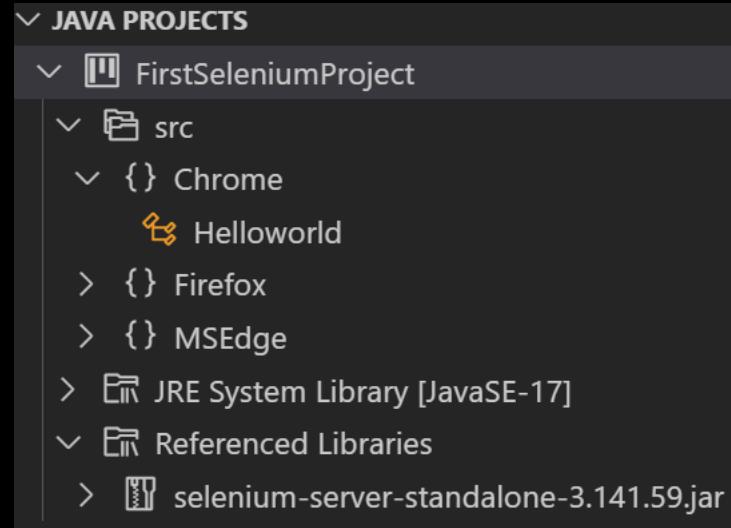


Task: creating a VS Code project

Steps:

- Create a VS code project
 - Create Java project
 - No build tools
- Create a package named Chrome
 - Move App.java in it and rename in Helloworld.java
 - Copy the code
- Add the selenium-server-standalone-3.141.59.jar to Referenced Libraries
- Set the property webdriver.chrome.driver
- Run: 

```
public class HelloWorld {  
    public static void main(String[] args) {  
        // not complete ...  
        WebDriver driver = new ChromeDriver();  
  
        String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/";  
        String expectedTitle = "Hands-On Selenium WebDriver with Java";  
        String actualTitle = "";  
  
        // launch Chrome and direct it to the Base URL  
        driver.get(baseUrl);  
  
        // get the actual value of the title  
        actualTitle = driver.getTitle();  
  
        // compare the actual title of the page with the expected one"  
        if (actualTitle.contentEquals(expectedTitle)){  
            System.out.println("Test Passed!");  
        } else {  
            System.out.println("Test Failed");  
        }  
  
        // close Chrome  
        driver.quit();  
    }  
}
```



**Maven Project with Selenium
WebDriver 4.19.1 (automated
driver management) and
Chrome**

Building tools

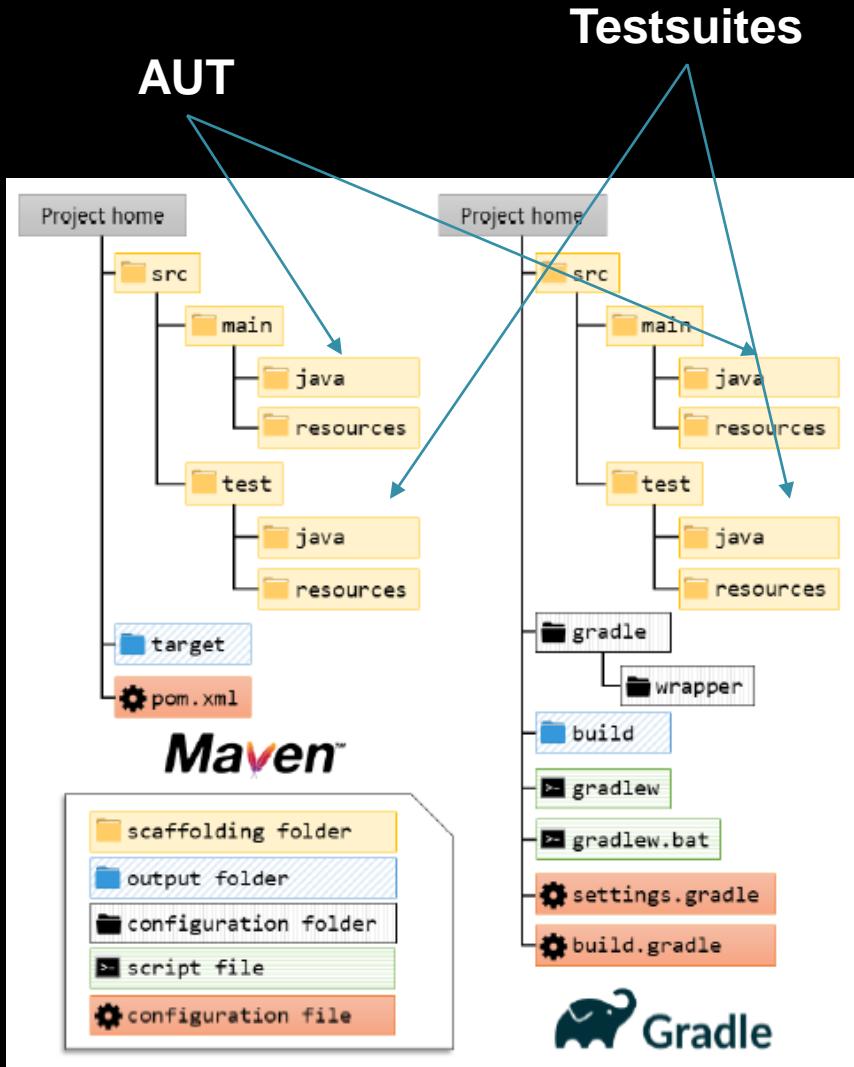
- Build tools are software utilities used to automate the **creation of executable applications from source code**
 - JAR and WAR files
- These tools ease the project management **in terms of dependency management**, testing, or release
- Overall, build tools are a convenient way to automate the development of software projects, both in build servers (e.g., Jenkins, **GitHub actions**) and developer machines



<https://maven.apache.org/>

Project Layout

- The *project layout* is the directory structure used to store the different assets of a software project (e.g., source code, binary files, static resources, and so on)
- Both **Maven** and **Gradle** use a similar layout for Java projects



The target (build) directory is used to house all output of the build

Configuration files

- A big difference between the two build tools is the configuration file
- On the one hand, this file is unique and called **pom.xml** (Project Object Model) in Maven
- On the other hand, there are two files in Gradle for configuration, called **settings.gradle** and **build.gradle**

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" >
  <modelVersion>4.0.0</modelVersion>
  <groupId>WebDriverTest</groupId>
  <artifactId>WebDriverTest</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>4.1.1</version>
      <scope>test</scope>
    </dependency>

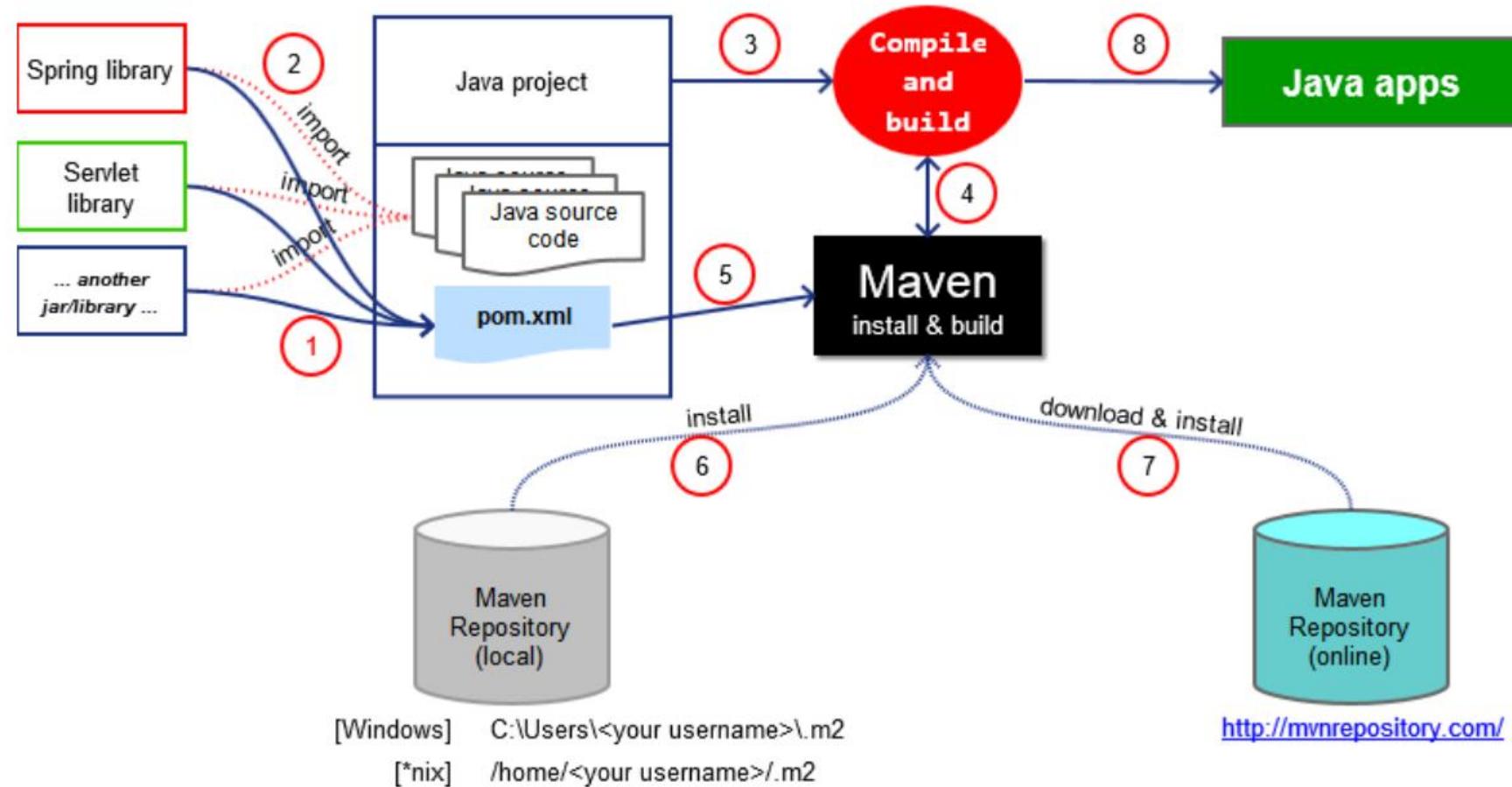
    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>7.4.0</version>
      <scope>test</scope>
    </dependency>
  
```



Maven

- Based on **Project Object Model (POM)**
 - **Configuration file of the project**
 - **XML file** which describes (among other things) the **dependencies** between the project and the necessary libraries
- Maven automatically downloads Java libraries from specific remote repositories by downloading them locally
 - Simplification and greater **portability**
 - Thus it is possible to move the project from one environment to another with the **certainty of always using the same versions of the libraries!!!**

How Maven works



The libraries are first searched in the local repo otherwise in the central repo and downloaded

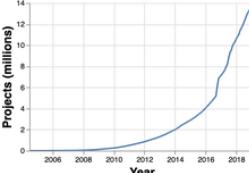
It is not the only MVN repository but the most used

MVN Repository

- <https://mvnrepository.com/>

MVN REPOSITORY

Indexed Artifacts (30.7M)



Search for groups, artifacts, categories

Search

What's New in Maven

WSO2 Carbon Utilities
org.wso2.carbon » org.wso2.carbon.utils » 4.8.0-m2
OSGi Bundle for Carbon Utils
Last Release on Oct 24, 2022

WSO2 Carbon UI
org.wso2.carbon » org.wso2.carbon.ui » 4.8.0-m2
org.wso2.carbon.ui
Last Release on Oct 24, 2022

WSO2 Carbon Default Authenticator Stub
org.wso2.carbon » org.wso2.carbon.authenticator.stub » 4.8.0-m2
WSO2 Carbon Default Authenticator Stub
Last Release on Oct 24, 2022

Popular Categories

- Testing Frameworks
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- Core Utilities
- JVM Languages
- Mocking
- Language Runtime

Central Repository provided and maintained by the 'Maven community'

POM File

When you want to add a dependency you need to go to the MVN repository, search for the library of interest and copy the relevant XML lines into the POM of our project



```
<project xmlns="http://maven.apache.org/POM/4.0.0" >
  <modelVersion>4.0.0</modelVersion>
  <groupId>WebDriverTest</groupId>
  <artifactId>WebDriverTest</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
    <dependency>
      <groupId>org.seleniumhq.selenium</groupId>
      <artifactId>selenium-java</artifactId>
      <version>4.1.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
      <groupId>org.testng</groupId>
      <artifactId>testng</artifactId>
      <version>7.4.0</version>
      <scope>test</scope>
    </dependency>
```

How to install Maven

Installazione

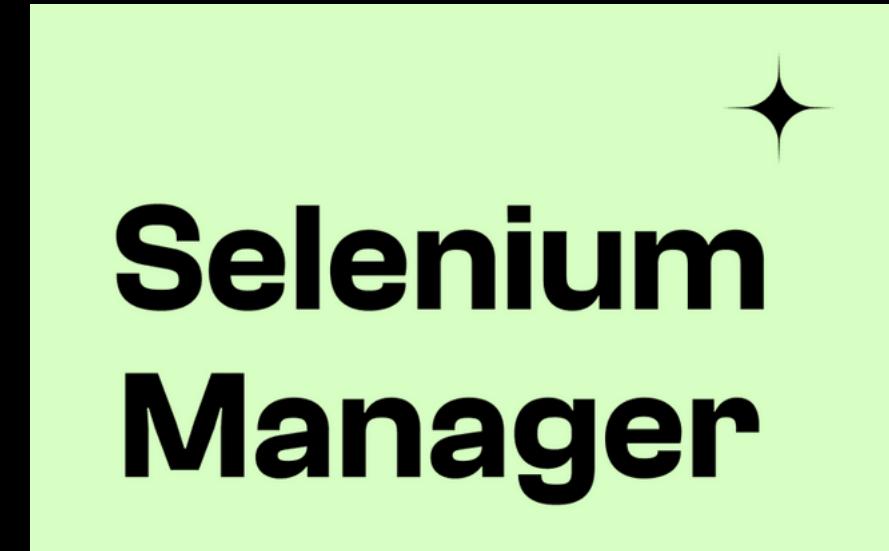
- Andare a <http://maven.apache.org/download.html>
- Necessario avere già installato Java Development Kit (JDK)
- Scaricare Maven e decomprimerlo in una cartella
`/usr/local/apache-maven`
- Creare (export) le variabili d'ambiente
 - `M2_HOME=/usr/local/apache-maven/apache-maven-X.Y.Z`
 - `M2=$M2_HOME/bin`
- Aggiungere (export) al path
 - `PATH=$M2:$PATH`
- Assicurarsi di avere nel path la variabile d'ambiente `JAVA_HOME` che punta alla JDK
- Verificare la corretta installazione
 - `mvn --version`

<https://maven.apache.org/install.html>

<https://phoenixnap.com/kb/install-maven-windows>

Automated Driver Management

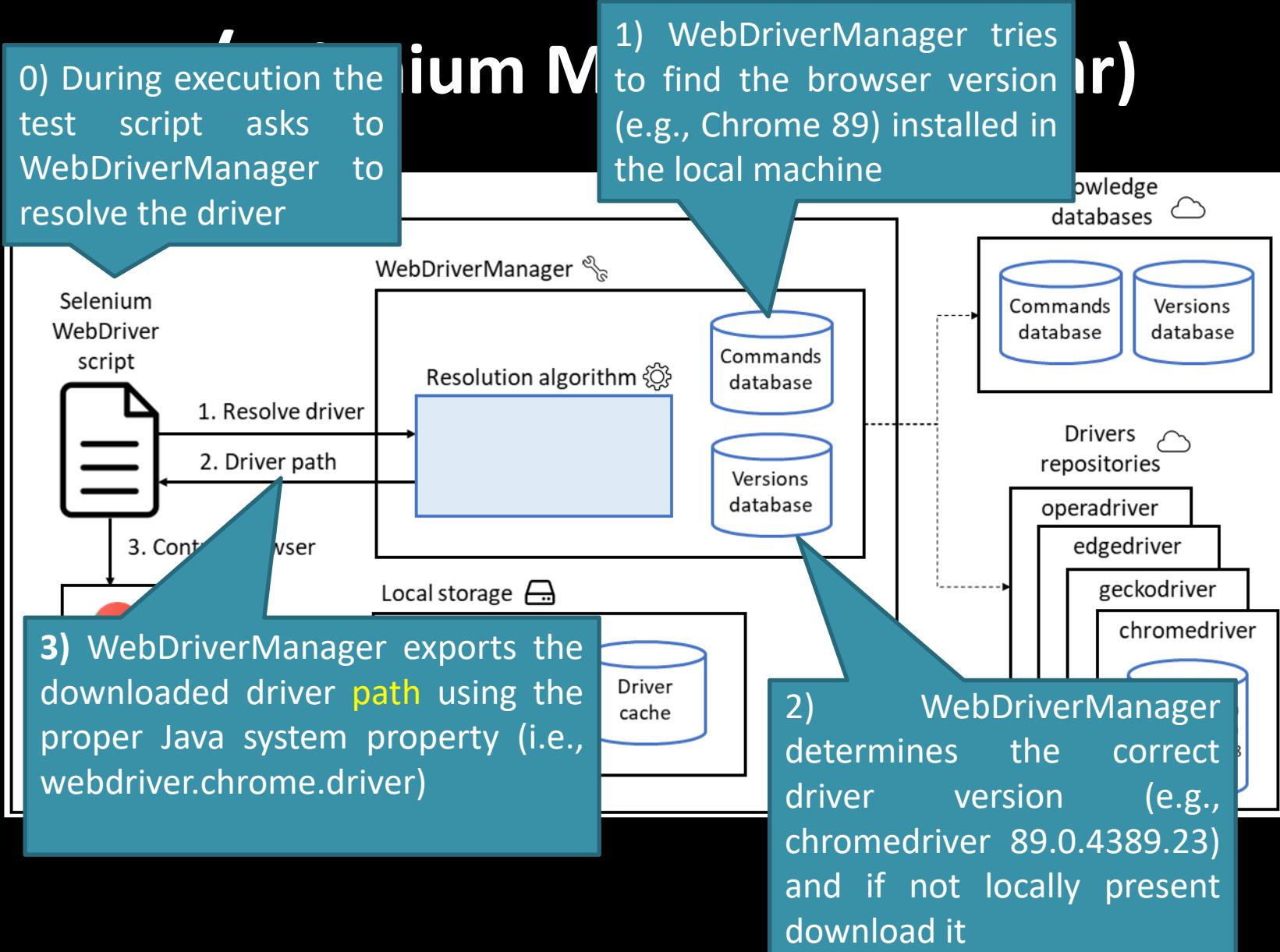
- **Manual Driver management** is costly in terms of effort and problematic in terms of maintenance
- For this reason, recently some specific libraries able to manage automatically drivers have been devised:
WebDriverManager and **Selenium Manager**



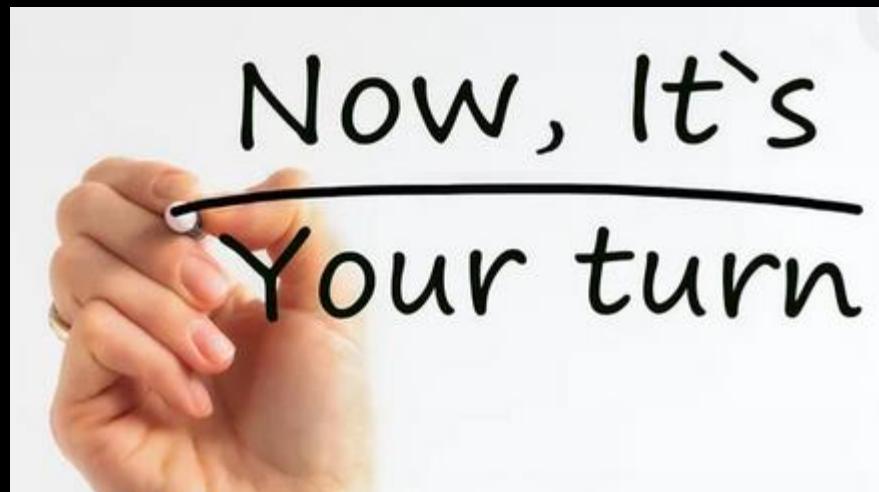
What are WebDriverManager and Selenium Manager?

- They are open-source Java library **able to carry out automatically the management**
 - download, setup, and maintenance
- of the drivers required by Selenium WebDriver (e.g., chromedriver, geckodriver, msedgedriver, etc.) in a **fully automated manner**
- WebDriverManager (Third-party library):
 - <https://bonigarcia.dev/webdrivermanager/>
- Selenium Manager (beta) (belong to the Selenium ecosystem):
 - shipped out of the box with each Selenium release as of version 4.6
 - https://www.selenium.dev/documentation/selenium_manager/

WebDriverManager architecture



- Creating a Maven VS code project with Selenium 4.19.1 and Chrome

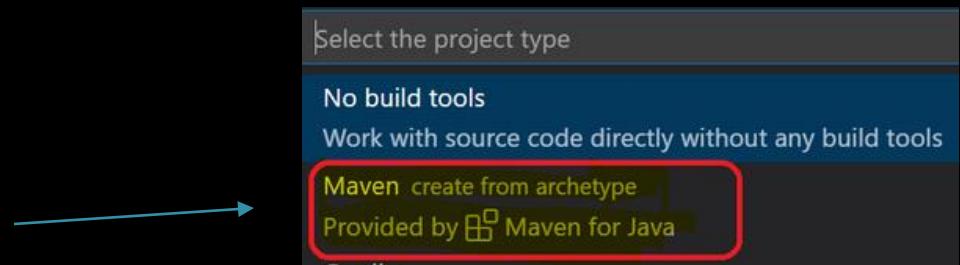


Task: creating a Maven project

Steps:

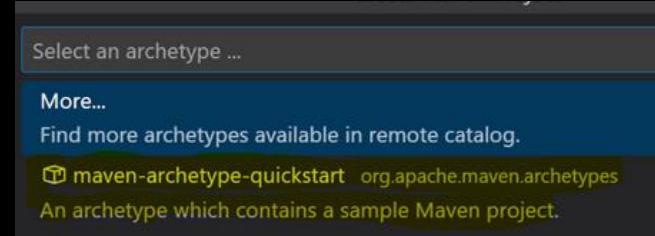
1. Create a Maven project

- Create Java project
- Maven create from archetype
- Select maven-archetype-quickstart
 - 1.4, com.example, first-maven-project, 1.0.0, Y



2. Rename App.java in Helloworld.java

- Copy the Helloworld code in it
- 'without the set property'



3. Open pom.xml and add selenium dependency

- Go to the MVN repository and search for Selenium Java 4.19.1
- Copy XML dependency and paste it into the dependencies section in pom.xml

4. Run:

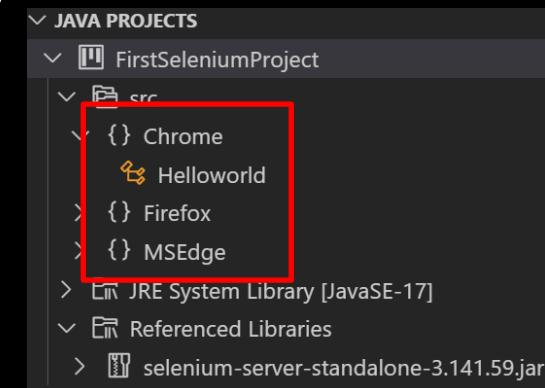
WebDriver methods

| Method | Return | Description |
|---------------------|------------------|---|
| get(String url) | void | Load a web page in the current browser |
| getCurrentUrl() | String | Get the URL currently loaded in the browser |
| getTitle() | String | Get the title (<title> HTML tag) of the current web page |
| findElement(By by) | WebElement | Find the first WebElement using a given locator in the current web page (see " Locating WebElements " on page 95 for further details) |
| findElements(By by) | List<WebElement> | Find all the WebElement using a given locator in the current web page (see also " Locating WebElements " on page 95) |
| getPageSource() | String | Get the HTML source code of the current web page |
| navigate() | Navigation | Access the browser history and navigate to a given URL (see Chapter 4) |
| getWindowHandle() | String | Get the <i>window handle</i> , i.e., a unique identifier for the open window in the current browser (see Chapter 4 for further details) |
| getWindowHandles() | Set<String> | Get the set of window handles currently opened in the current browser (see also Chapter 4) |
| switchTo() | TargetLocator | Select a frame or window in the current browser (see Chapter 4) |
| manage() | Options | Generic utility for managing different aspects of the browser (e.g., browser size and position, cookies, timeouts, logs, and the Input Method Engine (IME) API) |
| close() | void | Close the current window, quitting the browser if there is no more windows opened |
| quit() | void | Close all windows and quit the browser |

Assignment (not evaluated) multi browser project (manual driver management)

Add to the previous project with only Chrome the other browsers

1. Download **geckodriver** (for Firefox)
2. Create a package named **Firefox**
3. Copy Helloworld in it
4. Set the correct property (webdriver.gecko.driver)
5. Create Webdriver driver = new FirefoxDriver()
6. Download **edgeDriver** (for Edge)
7. Create a package named **Edge**
8. Copy Helloworld in it
9. Set the correct property (webdriver.edge.driver)
10. Create Webdriver driver = new EdgeDriver()



Functional and Security Testing Course

2023-2024

Selenium WebDriver API – lesson 8

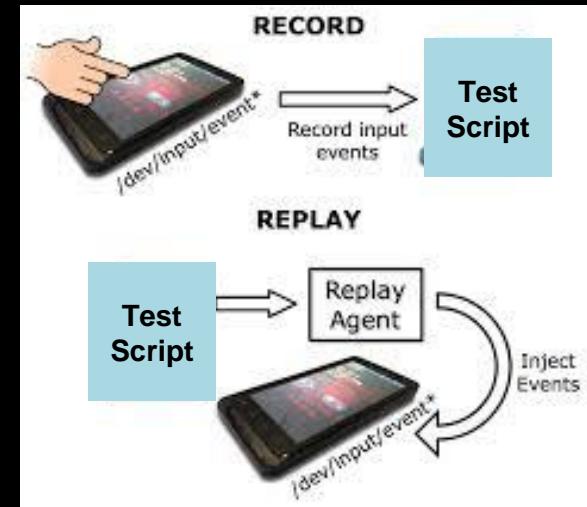
Filippo Ricca
DIBRIS, University of Genova

How Test scripts can be produced

Two ways:

- Capture/Replay

- Recording the actions performed by the tester
 - using a specific tool
 - e.g., Selenium IDE or Katalon recorder
 - Manual Fix
- Re-executing them automatically



- Programmable (script based)

- Test scripts are **software artefacts** created:
 - Using standard programming languages and IDE
 - e.g., Java + Eclipse
 - Resorting to specific testing frameworks
 - e.g., Selenium WebDriver



API complete and complex

Selenium Web Driver API

changing frames & windows,
managing alerts, and cookies,
selecting drivers for browsers

Domain

manage the automation speed, wait
for the web application events

Synchronisation

Selenium
WebDriver
Commands

open a browser, move to a page from another
one, back, forward, etc ([WebDriver interface](#))

Navigation

get information about the website and its
elements (`FindElement()`). ([WebDriver interface](#))

Interrogation

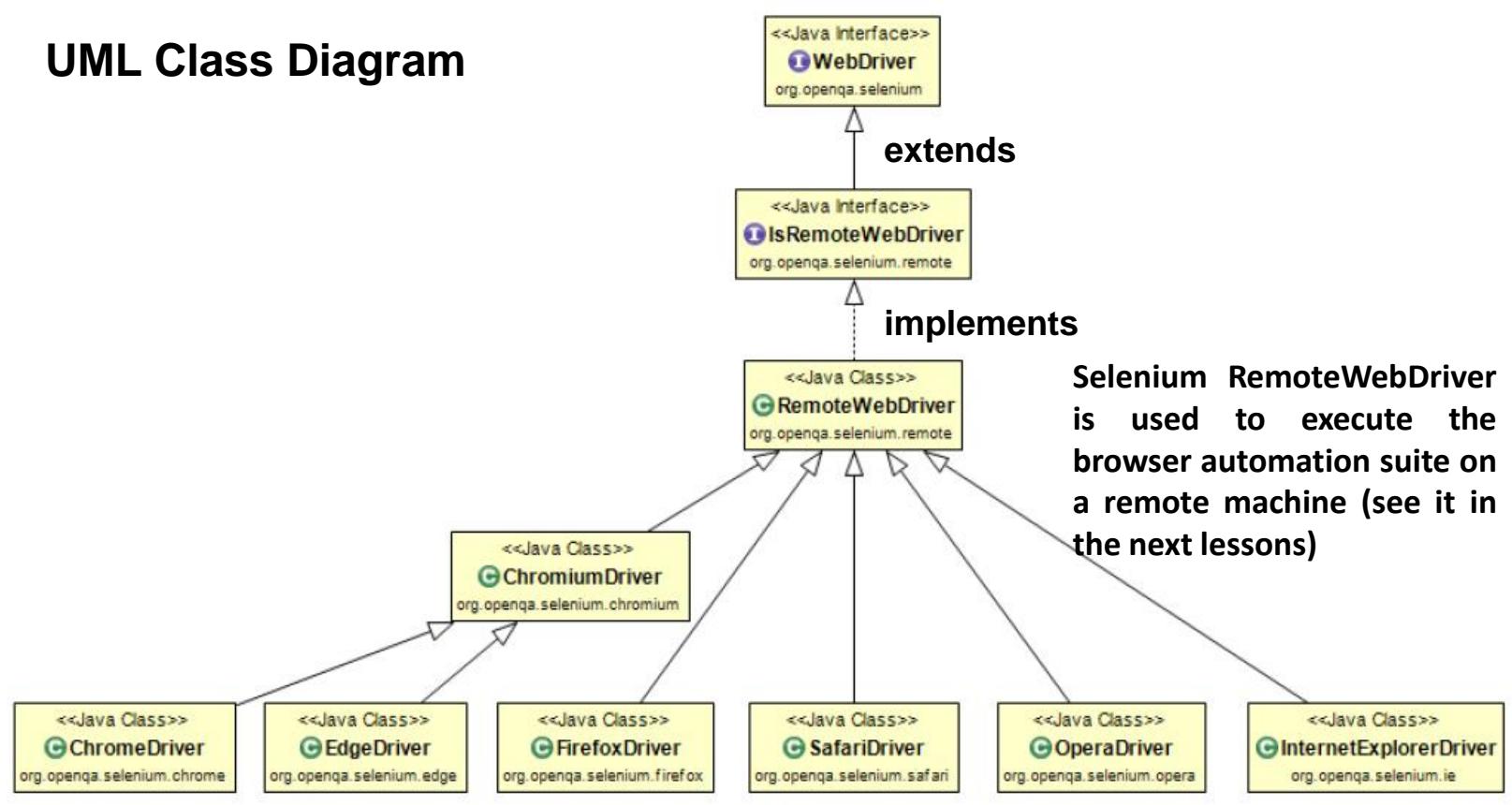
clicking on links and buttons, filling forms,
cleaning texts, pressing keys, drag &
dropping ([WebElement class](#))

Manipulation

Commands belong to five categories

WebDriver interface

UML Class Diagram



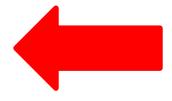
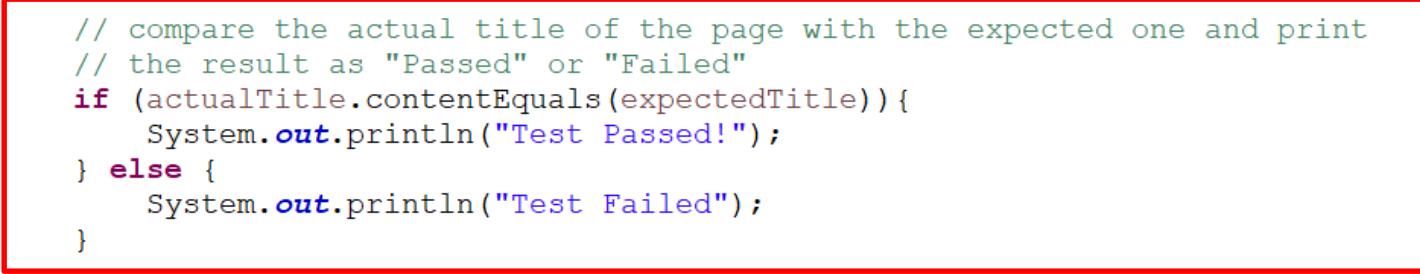
```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

WebDriver driver = new ChromeDriver();
```

Our HelloWorld example

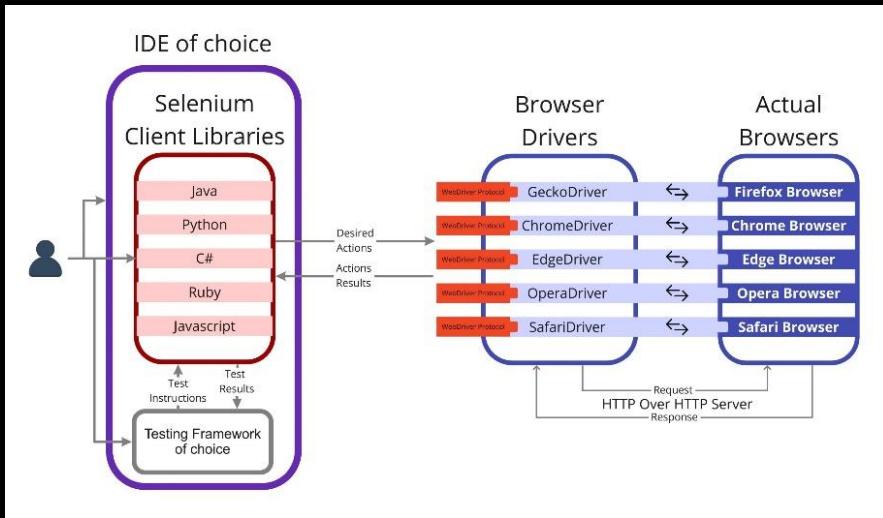
```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver; Testing framework import

public class Helloworld {
    public static void main(String[] args) throws InterruptedException {

        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\PC-User\\\\Desktop\\\\FSTT 2023\\\\chromedriver.exe"); To be substituted by an assertion
```

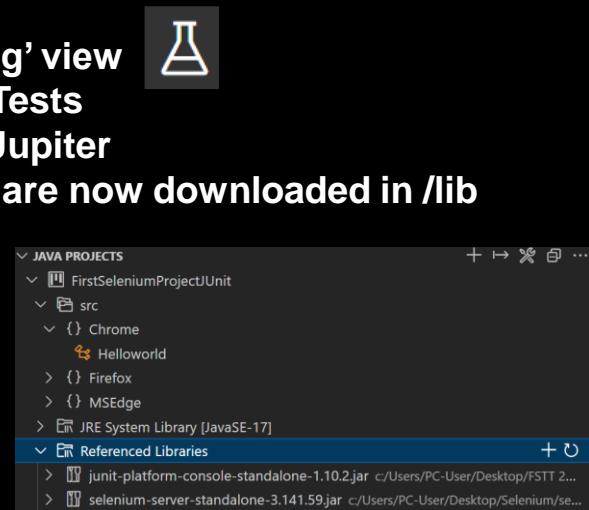
Unit testing frameworks

- To create **real** E2E Webdriver tests we have to embedd the Selenium WebDriver calls within tests created with a particular **Unit testing framework and add assertions**
- The alternative are: JUnit 4, JUnit 5 and TestNG



This means to add another dependency (es. JUnit 5) to our project

- Select 'Testing' view
- Enable Java Tests
- Select JUnit Jupiter
- Test libraries are now downloaded in /lib



Refactored HelloWorld (with JUnit 5)

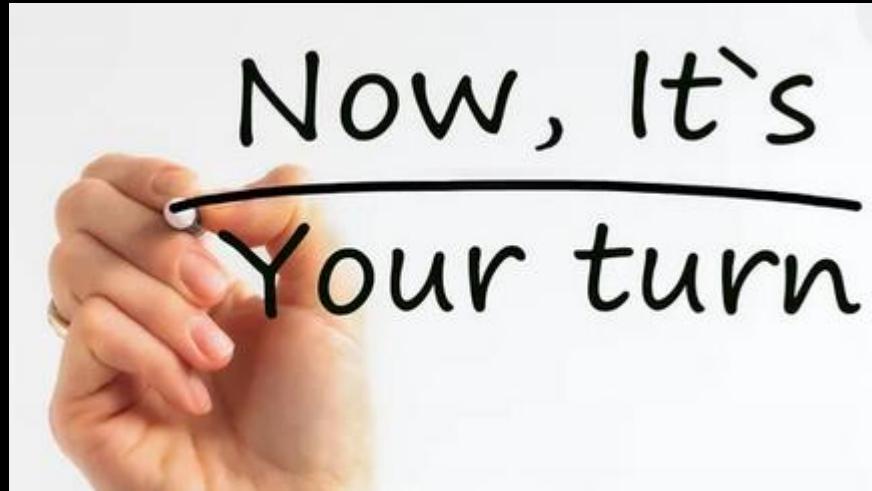
```
public class Helloworld {  
  
    private WebDriver driver;  
  
    @BeforeAll  
    static void setUpClass() {  
        System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\PC-User\\\\Des  
    }  
    @BeforeEach  
    void setup() {  
        driver = new ChromeDriver();  
    }  
    @AfterEach  
    void teardown() {  
        if (driver != null) {  
            driver.quit();  
        }  
    }  
    @Test  
    void test1() {  
  
        String baseUrl = "https://bonigarcia.dev/selenium webdriver-java/";  
        String expectedTitle = "Hands-On Selenium WebDriver with Java";  
        String actualTitle = "";  
  
        // launch Chrome and direct it to the Base URL  
        driver.get(baseUrl);  
    }  
}
```

JUnit lifecycle

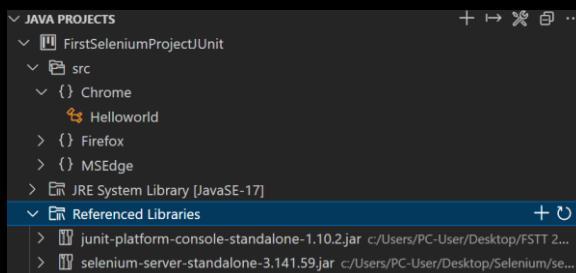
Not necessary if we are using Selenium WebDriver > 4.6

Assertions ...

- Open the ‘old’ manual created (No Maven) VS Code Helloworld project
 - the one with Selenium Jar file vers. 3.14.59
- Add the library JUnit 5
- Refactor the code using the JUnit lifecycle and Assertions (assertEquals())
- Execute your first test script



- Select ‘Testing’ view 
- Enable Java Tests
- Select JUnit Jupiter
- Test libraries are now downloaded in /lib



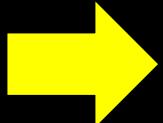
Pay attention to insert the correct import!

```
import org.junit.jupiter.api.Test;
```

```
import static  
org.junit.jupiter.api.Assertions.*;
```

WebDriver methods

```
WebElement button = driver.findElement(By.id("pippo"))
button.click();
```



| Method | Return | Description |
|---------------------|------------------|---|
| get(String url) | void | Load a web page in the current browser |
| getCurrentUrl() | String | Get the URL currently loaded in the browser |
| getTitle() | String | Get the title (<title> HTML tag) of the current web page |
| findElement(By by) | WebElement | Find the first WebElement using a given locator in the current web page (see " Locating WebElements " on page 95 for further details) |
| findElements(By by) | List<WebElement> | Find all the WebElement using a given locator in the current web page (see also " Locating WebElements " on page 95) |
| getPageSource() | String | Get the HTML source code of the current web page |
| navigate() | Navigation | Access the browser history and navigate to a given URL (see Chapter 4) |
| getWindowHandle() | String | Get the <i>window handle</i> , i.e., a unique identifier for the open window in the current browser (see Chapter 4 for further details) |
| getWindowHandles() | Set<String> | Get the set of window handles currently opened in the current browser (see also Chapter 4) |
| switchTo() | TargetLocator | Select a frame or window in the current browser (see Chapter 4) |
| manage() | Options | Generic utility for managing different aspects of the browser (e.g., browser size and position, cookies, timeouts, logs, and the Input Method Engine (IME) API) |
| close() | void | Close the current window, quitting the browser if there is no more windows opened |
| quit() | void | Close all windows and quit the browser |

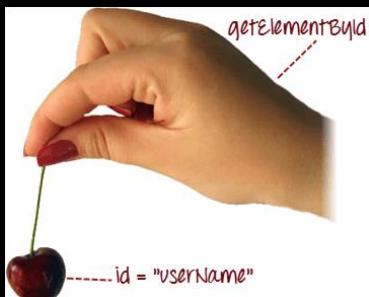
WebElement methods

```
WebElement button = driver.findElement(By.id("pippo"))
button.click();
```



| Method | Return | Description |
|--------------------------------------|------------------|---|
| click() | void | Perform a mouse click (i.e., a left-click) in the current element |
| submit() | void | Send a web form (when the element is a form) |
| sendKeys(CharSequence... keysToSend) | void | Simulate typing with the keyboard (e.g., in input text elements) |
| clear() | void | Reset the value of an input text element |
| getTagName() | String | Get the tag name of the element |
| getDomProperty(String name) | String | Get value of a DOM property |
| getDomAttribute(String name) | String | Get value of the element attribute as declared in its HTML markup |
| getAttribute(String name) | String | Get the value of the DOM property, if it exists. If not, the value of the attribute value is returned |
| getAriaRole() | String | Get element role as defined on the W3C WAI-ARIA specification |
| getAccessibleName() | String | Get element accessible name as defined on the W3C WAI-ARIA specification |
| isSelected() | boolean | Determine if a checkbox, option in a select, or radio button, is selected |
| isEnabled() | boolean | Determine if an element is enabled or not (e.g., a form field) |
| isDisplayed() | boolean | Determine if an element is visible or not |
| getText() | String | Get visible text of the element including its sub-elements (if any) |
| getSize() | Dimension | Get the width and height of the rendered element |
| getRect() | Rectangle | Get the location and size of the rendered element |
| getCssValue(String propertyName) | String | Get the value of a CSS property of the element |
| findElements(By by) | List<WebElement> | Find all sub-elements that matches the locator within the current element |
| findElement(By by) | WebElement | Find the first sub-elements that matches the locator within the current element |

**Depending on the Web element type
we can do different actions**



LOCATORS

How to locate an element

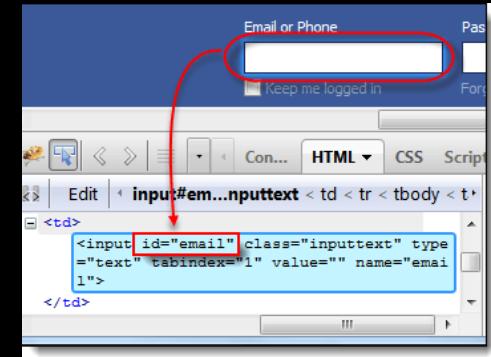
- **By id**
 - HTML: <input id="email" ... />
 - WebDriver:

```
driver.findElement( By.id("email") );
```
- **By name**
 - HTML: <input name="cheese" type="text"/>
 - WebDriver:

```
driver.findElement( By.name("cheese") );
```
- **By Xpath**
 - HTML

```
<html>
  <input type="text" name="example" />
  <input type="text" name="other" />
</html>
```
 - WebDriver:

```
driver.findElement( By.xpath("/html/input[1]") );
```

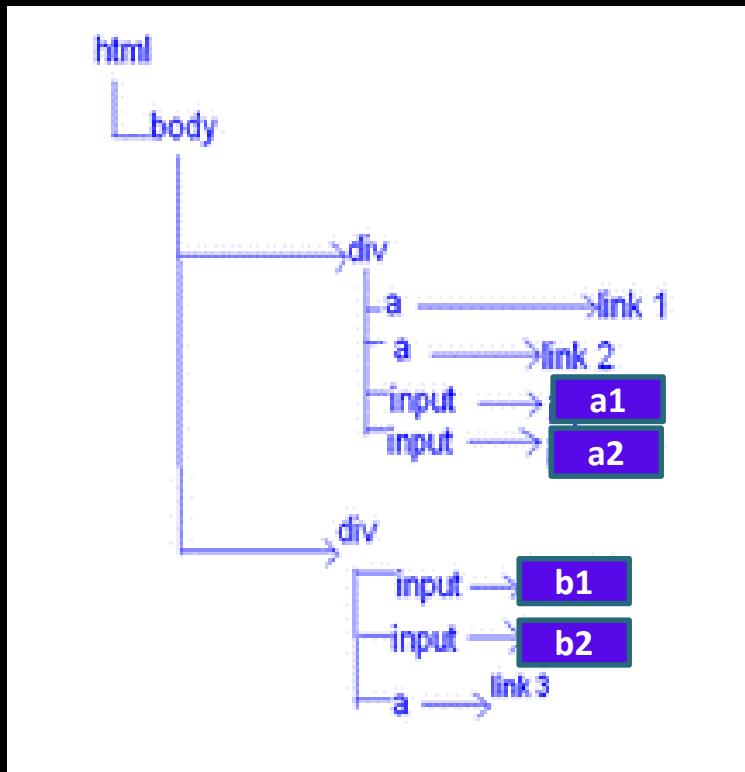


| Selenium Supported Locators | |
|-----------------------------|-----------------|
| XPath | Name |
| CSSSelector | LinkText |
| ClassName | PartialLinkText |
| ID | TagName |

Different types of locators!

Absolute vs. relative XPath

XPath (XML Path Language) is a **query language** for selecting nodes from an XML document



DOM tree representation

Absolute XPath. It begins with single slash “/” which means start the search from the **root** node

- `/html/body/div/input` => a1, a2, b1, b2
- `/html/body/div[1]/input[2]` => a2

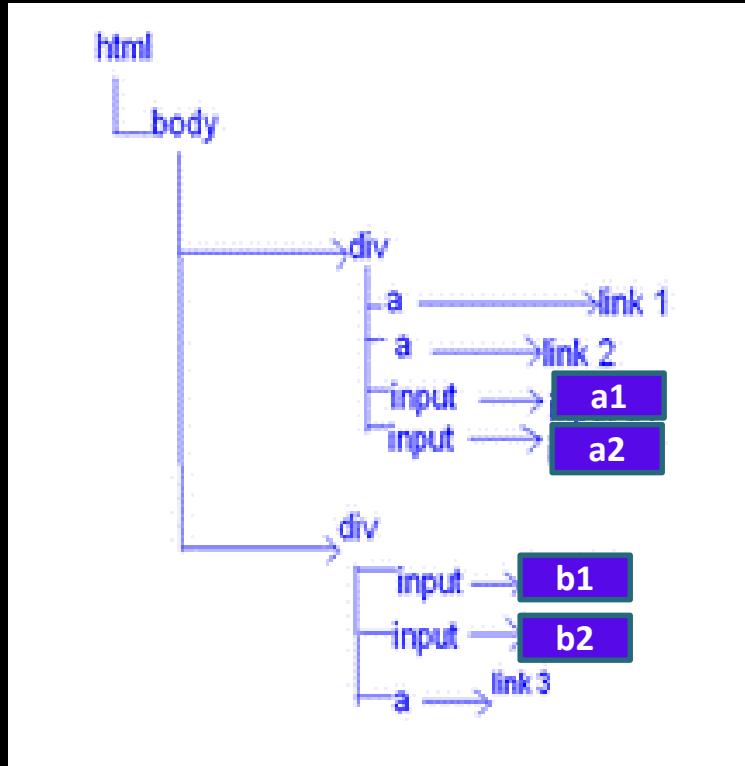
Relative XPath. It begins with double slash “//” which represents **search in the entire web page**

- `//input` => a1, a2, b1, b2
- `//div/input[1]` => a1, b1
- `//div[2]/input[2]` => b2

Level of a XPath it is the number of ‘/’ or ‘//’ symbols

Wild cards in XPath

- A star ('*') is a wild card, it means:
 - **'all the elements at this level'**



- `/html/body/div[1]/*` => link1, link2, a1, a2
- `//div/*` => every child of every div node
- `//*` => all the elements of the entire DOM

XPath

https://www.youtube.com/watch?v=r_AP1I3T9yM

Facts about Locators: no perfect locator

1. Maintaining locators must be calculated as part of the **cost of test suite maintenance**
2. There is no perfect, unbreakable, locator!

WAUT - Release N

Name:

John

Surname:

Doe

Mobile:

1234

Target Element

'XPath Expr'
//*[id="userInfo"]/tr[3]/td[2]

Test

OK

FAILURE

Locator selects
Male and not 1234!

WAUT - Release N+1

Name:

John

Surname:

Doe

Gender:

Male

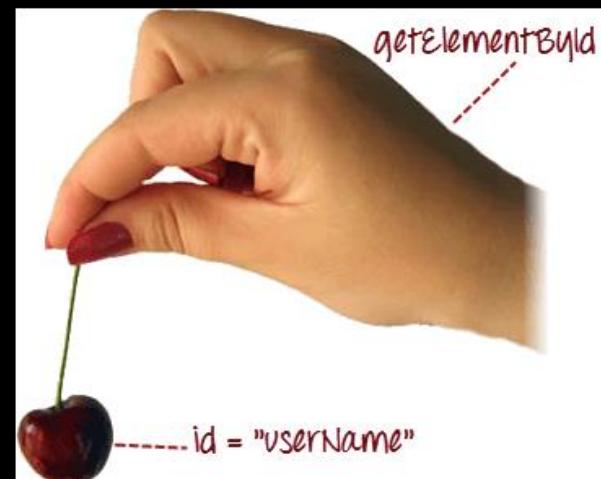
Mobile:

1234

Target Element

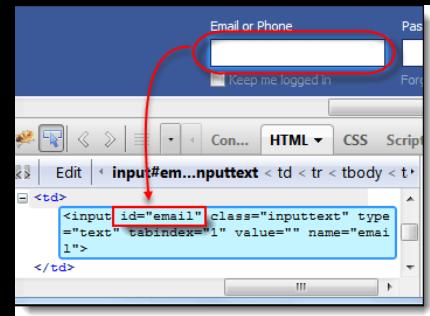
Facts about Locators: Ids the best

- “Meaningful” Ids locators are the preferred way to locate an element
 - **Experimental study** [M. Leotta et alt., WCRE 2013]
 - Ids turned out to be the most robust
 - less than 2% of the 459 Id locators were broken, while 60% of the 791 XPath required to be fixed



Facts about Locators:

Ids don't always exist



- **Ids are the best choice**, however...
 - Ids don't always exist
 - adding **meaningful** Ids everywhere is impractical or not viable
 - Their uniqueness is not enforced
 - In some cases, they are ‘auto-generated’ and change at run-time, so they are unreliable
 - Some development frameworks generate them

```
driver.findElement(By.id("id_fZI_1")).click();
```

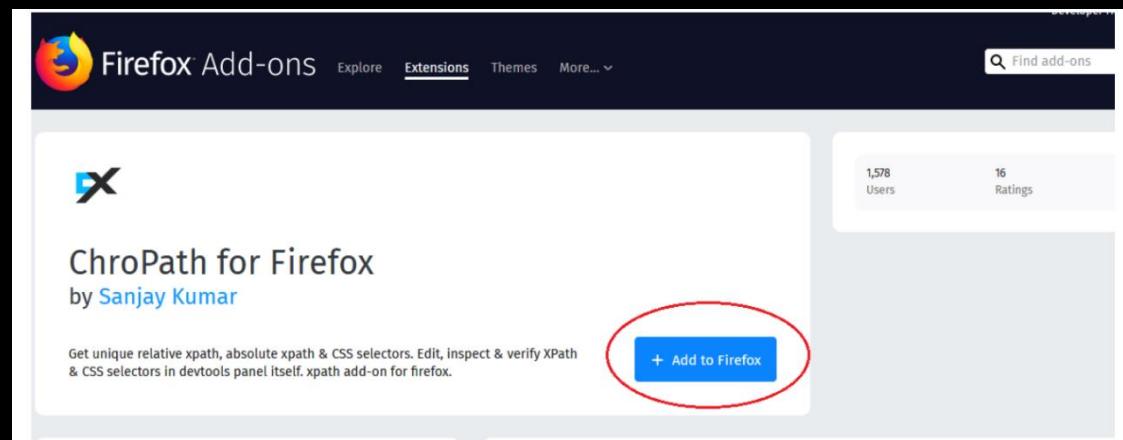
- So often it is not possible to use IDs locators
 - An alternative is using XPath expressions



How to compute XPath

- There are plug-ins for Firefox/Chrome to automatically display the XPath of a Web element
 - e.g., **ChroPath** (Firefox and Chrome) and **xPath Finder**
 - **ChroPath** available for Firefox and Chrome
- **Inspect** the element or click on DOM node to generate unique locators

For installing ChroPath
go to
the ‘Firefox store’ and
search for it ...



How to use ChroPath



1) Open Web Console

Hands-On Selenium WebDriver with Java

Practice site

2) Select the Element

Web form

Text input

Dropdown (select)

Dropdown (datalist)

Color picker

Date picker

Analisi pagina Console Debugger Rete Editor stili Prestazioni Memoria Archiviazione Accessibilità Applicazione

Cerca in HTML

Text input
<input id="my-text-id" class="form-control" type="text" name="my-text" myprop="myvalue" xpath="1">
</label>
 <label class="form-label w-100">
 Password
 <input class="form-control" type="password" name="my-password" autocomplete="off">
 </label>

:hover .cls + ☀️ ☀️

Pseudo-elementi

Questo elemento

elemento :: { inline } contentScript.css:9 [xpath="1"], [css="1"] :: { outline: 2px dashed #07bb46 !important; }

Layout Calcolate Modifiche Compatibilità Caratteri Animazioni ChroPath

Selectors type selector and press enter 5.0 1 element matching.

Rel XPath //input[@id='my-text-id']
Abs XPath /html[1]/body[1]/main[1]/div[1]/form[1]/div[1]/label[1]/input[1]
CSS sel... #my-text-id

Issue in FF ? | 6.0 live for Chrome | Download for other browsers | Add Review | Join ChroPath Community | Follow Us

3) Search ChroPath Tab

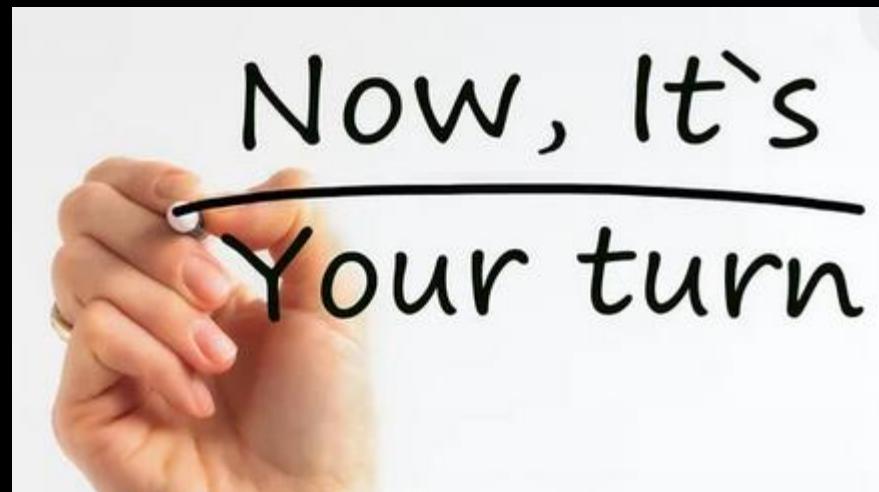
4) Read Rel XPath

A screenshot of the ChroPath extension interface integrated into the Firefox browser. A red arrow points from the 'Text input' field in the main browser to the 'Text input' field in the ChroPath tab. Another red arrow points from the 'ChroPath' tab in the browser's toolbar to the 'ChroPath' tab in the extension's interface. The ChroPath tab displays the selected element's Rel XPath as //input[@id='my-text-id'].

How to install and use ChroPath

<https://www.youtube.com/watch?v=hpCZnulz1k4>

- Install ChroPath
- Try to use ChroPath



LOGIN EXAMPLE

Task: Successfully Login example

<https://bonigarcia.dev/selenium-webdriver-java/login-form.html>

Hands-On Selenium WebDriver with Java

Practice site

Login form

Login

user

Password

user

Submit



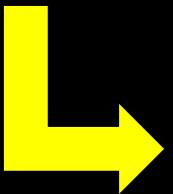
assertion

Hands-On Selenium WebDriver with Java

Practice site

Login form

Login successful



Task: Unsuccessfully Login example

<https://bonigarcia.dev/selenium-webdriver-java/login-form.html>

Hands-On Selenium WebDriver with Java

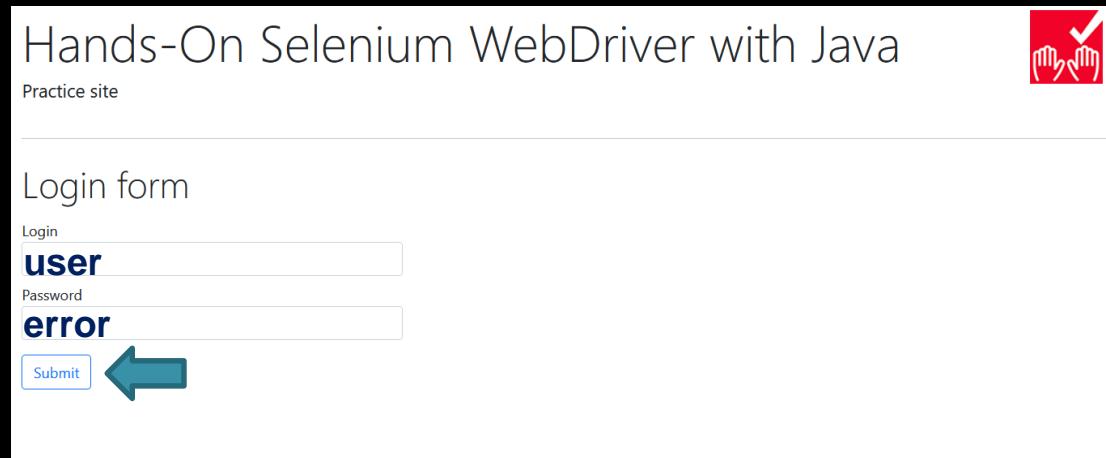
Practice site

Login form

Login
user

Password
error

Submit



A screenshot of a web browser displaying a login form. The page title is "Hands-On Selenium WebDriver with Java" and the subtext is "Practice site". The form has two input fields: "Login" containing "user" and "Password" containing "error". A blue "Submit" button is at the bottom. To the right of the "Submit" button is a red square icon with a white checkmark and a hand icon. A blue arrow points from the "Submit" button towards this icon.

assertion

Hands-On Selenium WebDriver with Java

Practice site

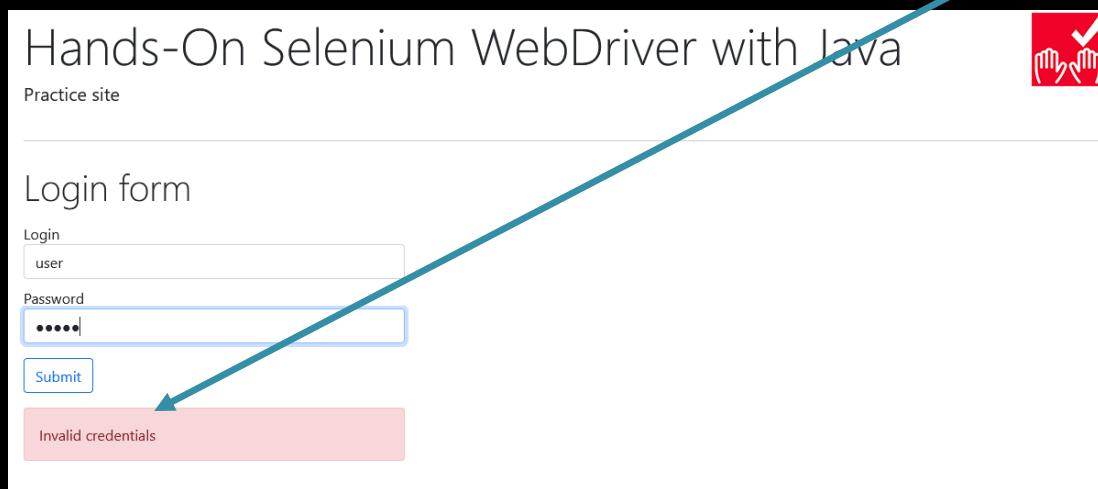
Login form

Login
user

Password

Submit

Invalid credentials



A screenshot of a web browser displaying a login form. The page title is "Hands-On Selenium WebDriver with Java" and the subtext is "Practice site". The form has two input fields: "Login" containing "user" and "Password" containing "*****". A blue "Submit" button is at the bottom. Below the "Submit" button is a pink rectangular box containing the text "Invalid credentials". To the left of the "Submit" button is a large yellow L-shaped arrow pointing towards the "Invalid credentials" box. To the right of the "Submit" button is a red square icon with a white checkmark and a hand icon.

Successfully Login example

```
@Test
public void testLoginOK() {
    System.out.println("testLoginOK");
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/login-form.html";
    driver.get(baseUrl);
    System.out.println(driver.getCurrentUrl());

    driver.findElement(By.id("username")).sendKeys("user");
    driver.findElement(By.id("password")).sendKeys("user");
    driver.findElement(By.className("mt-2")).click();
    //driver.findElement(By.xpath("//button[@type='submit']")).click();
    //driver.findElement(By.cssSelector("button")).click();
    System.out.println(driver.getCurrentUrl());
    // si passa nella pagina login-success.html
    assertTrue(driver.findElement(By.id("success")).isDisplayed());
    assertEquals(driver.findElement(By.id("success")).getText(), "Login successful");
}
```

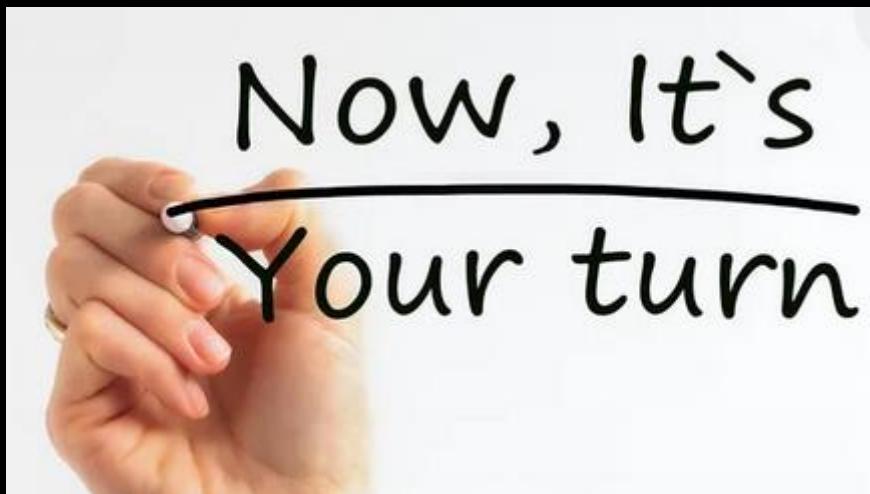
Experiment with different locators!

Unsuccessfully Login example

```
@Test
public void testLoginNoOK() {
    System.out.println("testLoginNoOK");
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/login-form.html";
    driver.get(baseUrl);
    System.out.println(driver.getCurrentUrl());

    driver.findElement(By.id("username")).sendKeys("user");
    driver.findElement(By.id("password")).sendKeys("errore");
    driver.findElement(By.className("mt-2")).click();
    System.out.println(driver.getCurrentUrl());
    // si rimane nella stessa pagina
    // web element 'invalid' compare solo se credenziali sbagliate, implementato con Java
    assertTrue(driver.findElement(By.id("invalid")).isDisplayed());
}
```

- Open the ‘old’ created Maven VS Code Helloworld project
 - the one with Selenium 4.19.1 dependency
- Add to the POM file the JUnit Jupiter dependency
 - JUnit Jupiter (Aggregator) dependency in MVN repository
- Create a LoginTest class
 - With two Test methods: testLoginOk and testLoginNotOk
- Execute the LoginTest class
- If you want deliver it in the Login Form Selenium WebDriver assignment



Functional and Security Testing Course

2023-2024

Page Object Design Pattern – lesson 9

Filippo Ricca
DIBRIS, University of Genova

AGENDA

- Simple Test Script Login example
- Test Script quality
- **Page Object Design Pattern**
- Test Script Refactoring
 - Towards POs
 - Different Refactorings



Login example

<https://bonigarcia.dev/selenium-webdriver-java/login-form.html>

Hands-On Selenium WebDriver with Java

Practice site

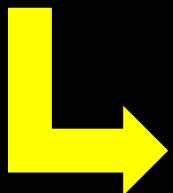
Login form

Login

user

Password

error

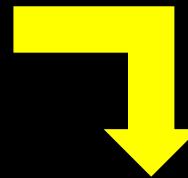


Hands-On Selenium WebDriver with Java

Practice site

Login form

Login successful



stay on login page

Hands-On Selenium WebDriver with Java

Practice site

Login form

Login

user

Password

•••••

Invalid credentials

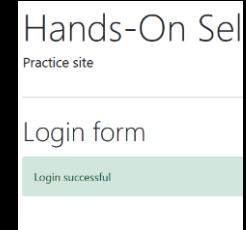
goto login-success page

Plain Test Script (NO Page objects)

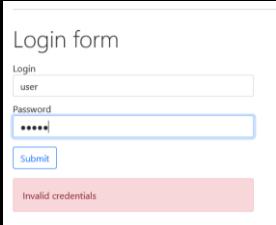
```
@Test
public void testLoginOK() {
    System.out.println("testLoginOK");
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/login-form.html";
    driver.get(baseUrl);
    System.out.println(driver.getCurrentUrl());

    driver.findElement(By.id("username")).sendKeys("user");
    driver.findElement(By.id("password")).sendKeys("user");
    driver.findElement(By.className("mt-2")).click();
    //driver.findElement(By.xpath("//button[@type='submit']")).click();
    //driver.findElement(By.cssSelector("button")).click();
    System.out.println(driver.getCurrentUrl());
    // si passa nella pagina login-success.html
    assertTrue(driver.findElement(By.id("success")).isDisplayed());
    assertEquals(driver.findElement(By.id("success")).getText(), "Login successful");
}
```

Test Ok Login
(goto login-success page)



Test Not Ok Login
(remain loginForm)



```
@Test
public void testLoginNoOK() {
    System.out.println("testLoginNoOK");
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/login-form.html";
    driver.get(baseUrl);
    System.out.println(driver.getCurrentUrl());

    driver.findElement(By.id("username")).sendKeys("user");
    driver.findElement(By.id("password")).sendKeys("errore");
    driver.findElement(By.className("mt-2")).click();
    System.out.println(driver.getCurrentUrl());
    // si rimane nella stessa pagina
    // web element 'invalid' compare solo se credenziali sbagliate, implementato con Java
    assertTrue(driver.findElement(By.id("invalid")).isDisplayed());
}
```

Test Script quality



- Test scripts are difficult to read
 - A lot of implementation details
- Duplication of locators and code across test scripts
 - **no reuse !**
- Often changes in the Web app breaks multiple tests
 - Fragile test scripts



IDEA

Adopting the Page Object Pattern!

Page Object Pattern

- A **level of abstraction** between the **test scripts** and the **web pages**
 - with the aim of **reducing the coupling** among them



- **Idea:** creating a **page class** for **each web page**
- Each **method** in the page class **encapsulates** a page's functionality
 - e.g., Login

Page Object Pattern: the Idea

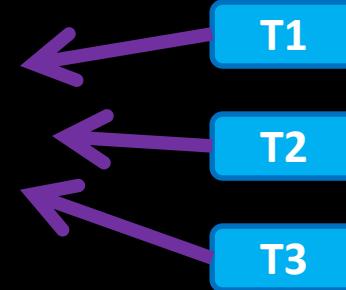
Username:

Password:

Login

```
public class LoginPage {  
    ...  
    public HomePage login( ... ) {  
        ...  
        ...  
    }  
    ... Other methods ...  
}
```

Test scripts use login() method offered by LoginPage



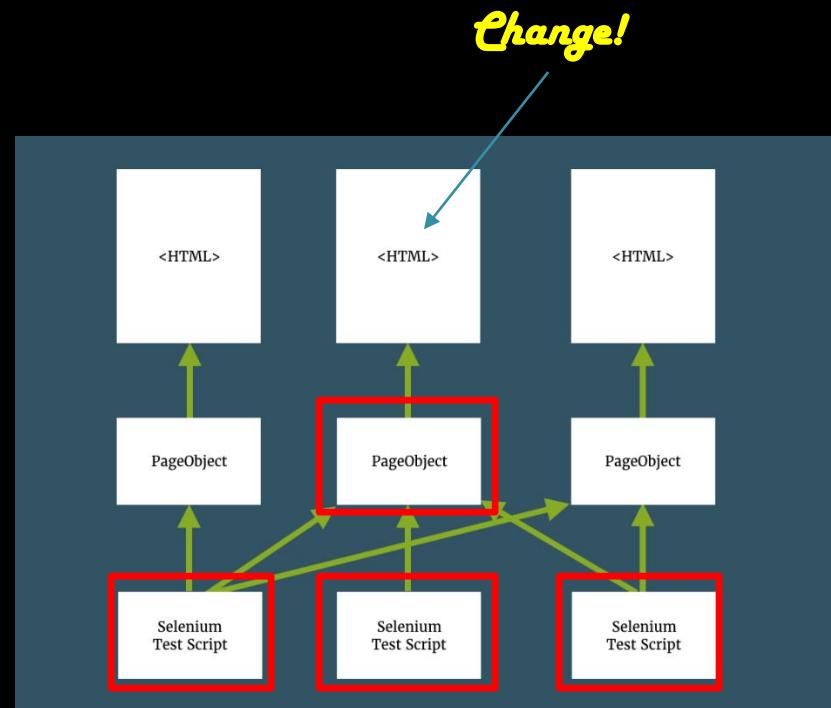
Implementation details are here!

Only Test Logic Here!



Advantages

- **Test scripts are simpler and more readable**
 - Implementation details are in the POs
- **Reuse**
 - The same method is called by several Test scripts
 - e.g., login()
- **Maintenace effort reduction**
 - A change in a Web page can affect only one PO
 - not a bunch of Test scripts!

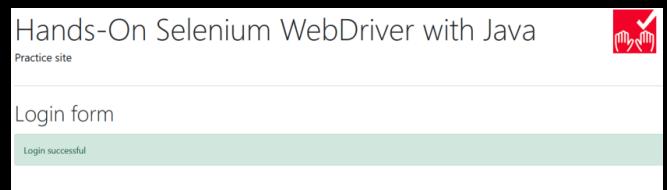
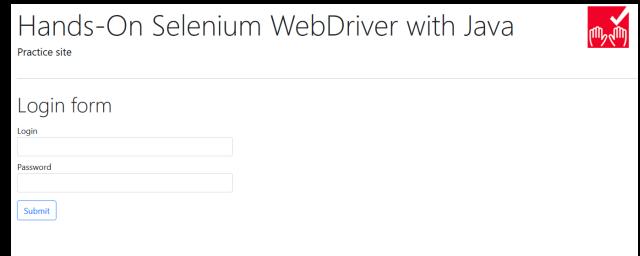


Login Test Scripts Refactoring

(Page Objects creation)

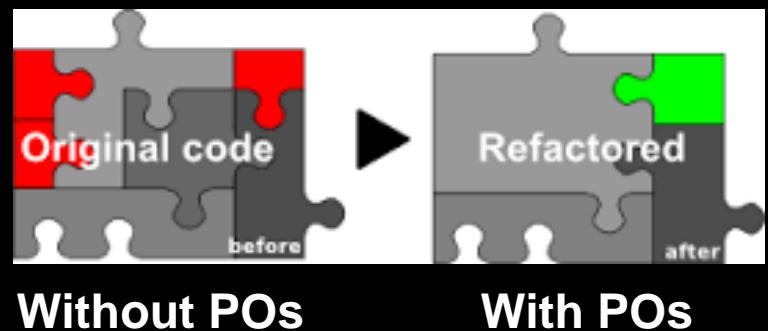
1. Page Objects creation

- Two POs (one for page)
 - LoginFormPO
 - LoginSuccessPO

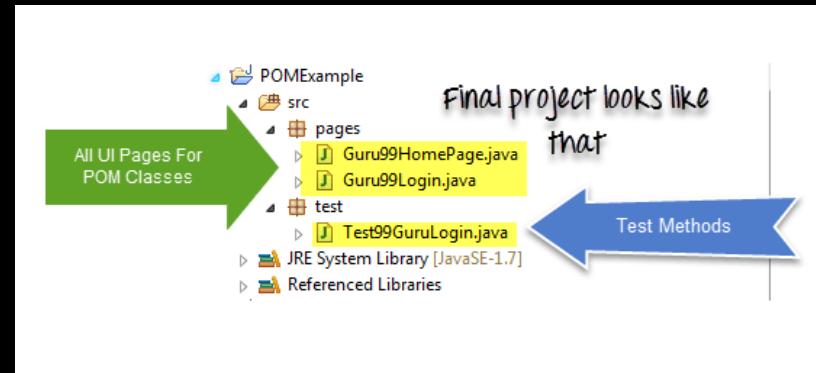
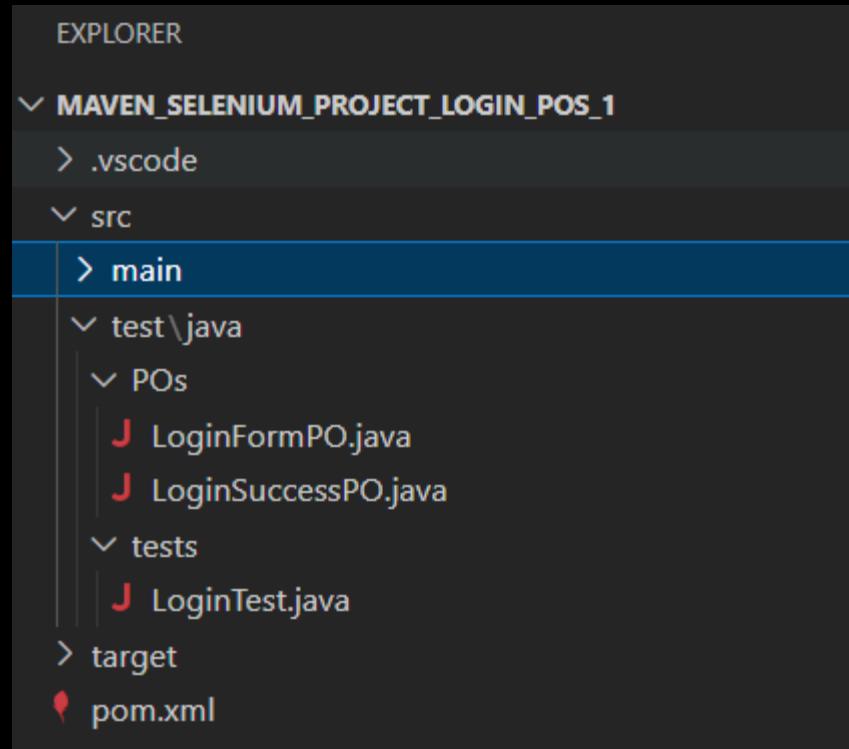


2. Test scripts building

- Using the POs



Project structure



Keep separated POs from test scripts!

Page Object Creation: LoginForm

```
public class LoginFormPO {    ←----- Page class in object repository
    private WebDriver driver;

    public LoginFormPO(WebDriver driver) {
        this.driver = driver;
        driver.get(
            "https://bonigarcia.dev/selenium-webdriver-java/login-form.html");
    }

    public void with(String username, String pwd) {
        driver.findElement(By.id("username")).sendKeys(username);
        driver.findElement(By.id("password")).sendKeys(pwd);
        driver.findElement(By.className("mt-2")).click();
    }

    public boolean invalidBoxIsPresent() {
        return driver.findElement(By.id("invalid")).isDisplayed();
    }
}
```

Getter Method

Assertions must be
inserted in Test Scripts



Page Object Creation: LoginSuccess

```
public class LoginSuccessPO {  Page class in object repository  
    private WebDriver driver;  
  
    public LoginSuccessPO(WebDriver driver) {  
        this.driver = driver;  
    }  
  
    public boolean successBoxIsPresent() {  
        return driver.findElement(By.id("success")).isDisplayed();  
    }  
}
```

Getter Method

Assertions must be
inserted in Test Scripts



(Refactored) Test script

```
public class BasicLoginTest {  
  
    private WebDriver driver;  
    private LoginFormPO login;  
    private LoginSuccessPO loginSuccess;  
  
    @BeforeAll  
    public static void beforeAll() { System.setProperty("webdriver.chrome.driver", "C:\\\\Users\\\\PC-User\\\\De  
  
    @BeforeEach  
    public void beforeEach() {driver = new ChromeDriver();}  
  
    @Test  
    public void testLoginOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "user");  
        // goto login-success page  
        loginSuccess = new LoginSuccessPO(driver);  
        assertTrue(loginSuccess.successBoxIsPresent());  
    }  
  
    @Test  
    public void testLoginNoOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "error");  
        // remain in the login-form page  
        assertTrue(login.invalidBoxIsPresent());  
    }  
}
```

POs definition (fields)

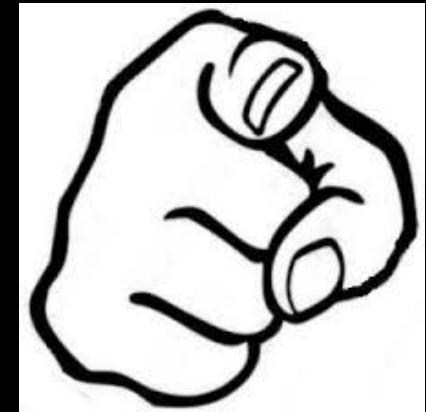
Not necessary if we are using Selenium WebDriver > 4.6

PO methods are called

Getter methods are used to create assertions

Now it's your turn!

- Open the 'old' created VS Code Login project (if you want available on AW)
- Refactor the LoginTest class with two Test methods using POs
 - testLoginOk and testLoginNotOk
- Execute the LoginTest class



First Project improvement

- POs
 - Locators as fields (this simplifies maintenance!)

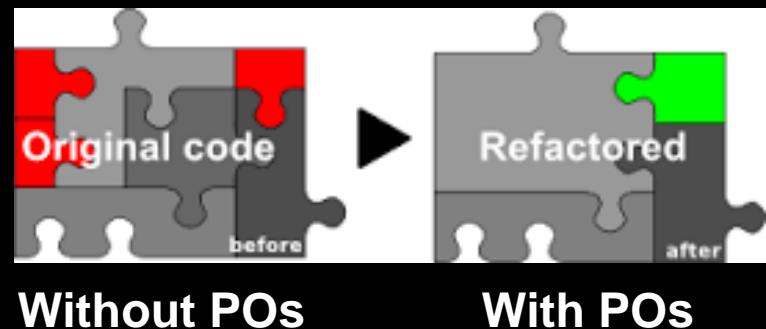
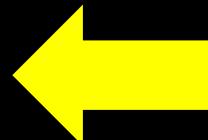
```
By usernameInput = By.id("username");
By passwordInput = By.id("password");
By submitButton = By.cssSelector("button");
By successBox = By.id("success");
```

- Test scripts
 - Isolating **Selenium WebDriver setup** in a specific class
 - @BeforeAll, @BeforeEach, @AfterEach

Test Script Refactoring

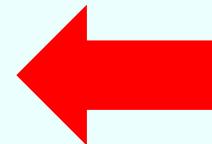
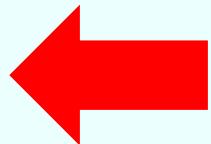
(Locators as fields)

- Page Objects creation
 - Two POs (one for page)
 - **LoginFormPO**
 - **LoginSuccessPO**
- Test scripts building
 - Using the POs



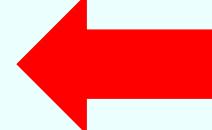
LoginForm PO

```
public class LoginFormPO {  
  
    private WebDriver driver;  
  
    private By usernameInput = By.id("username");  
    private By passwordInput = By.id("password");  
    private By submitButton = By.cssSelector("button");  
    private By invalidBox = By.id("invalid");  
  
    public LoginFormPO(WebDriver driver) {  
        this.driver = driver;  
        driver.get(  
            "https://bonigarcia.dev/selenium-webdriver-java/login-form.html");  
    }  
  
    public void with(String username, String pwd) {  
        driver.findElement(usernameInput).sendKeys(username);  
        driver.findElement(passwordInput).sendKeys(pwd);  
        driver.findElement(submitButton).click();  
    }  
  
    public boolean invalidBoxIsPresent() {  
        return driver.findElement(invalidBox).isDisplayed();  
    }  
}
```



LoginSuccess PO

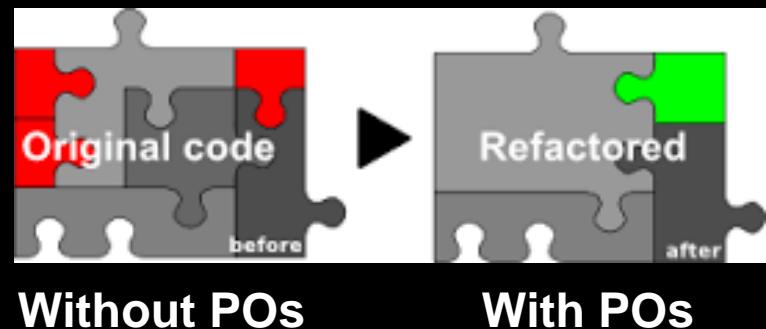
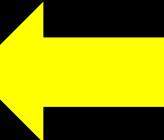
```
public class LoginSuccessPO {  
  
    private WebDriver driver;  
    private By successBox = By.id("success");  
  
    public LoginSuccessPO(WebDriver driver) {  
        this.driver = driver;  
    }  
  
    public boolean successBoxIsPresent() {  
        return driver.findElement(successBox).isDisplayed();  
    }  
}
```



Test Script Refactoring

(Selenium WebDriver setup)

- **Create setup test class**
- Page Objects creation
 - Two POs (one for page)
 - LoginFormPO
 - LoginSuccessPO
- Test scripts building
 - **extend setup test class**
 - Using the POs



Selenium WebDriver setup

```
public class DriverLifeCycleSetting {  
    protected WebDriver driver;          Make it accessible to subclasses  
    @BeforeAll                          Not necessary if we are using Selenium WebDriver > 4.6  
    public static void beforeAll() {  
        System.setProperty("webdriver.chrome.driver","C:\\\\Users\\\\PC-User\\\\Desl  
    }  
  
    @BeforeEach  
    public void beforeEach() {  
        driver = new ChromeDriver();  
        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5));  
    }  
  
    @AfterEach  
    public void afterEach() {  
        if (driver != null) {  
            driver.quit();  
        }  
    }  
}
```

The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception"

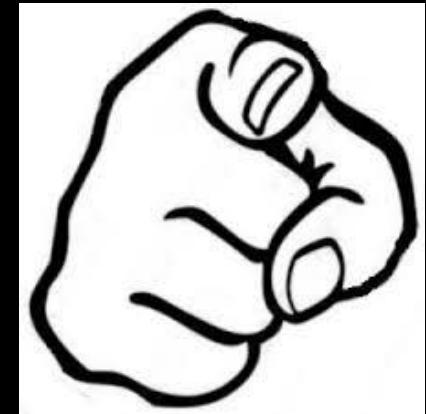
First step: creating a class to hold all the driver lifecycle management code

Test script (with setUp class)

```
public class LoginTest extends DriverLifeCycleSetting {  
  
    private LoginFormPO login;  
    private LoginSuccessPO loginSuccess;  
  
    @Test  
    public void testLoginOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "user");  
        // goto login-success page  
        loginSuccess = new LoginSuccessPO(driver);  
        assertTrue(loginSuccess.successBoxIsPresent());  
    }  
  
    @Test  
    public void testLoginNoOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "error");  
        // remain in the login-form page  
        assertTrue(login.invalidBoxIsPresent());  
    }  
}
```

Now it's your turn!

- Open the VS Code Login project and refactor it
- In PO classes transform Locators as fields
- Create setup test class
- Extend test scripts using setup test class
- Execute the LoginTest class

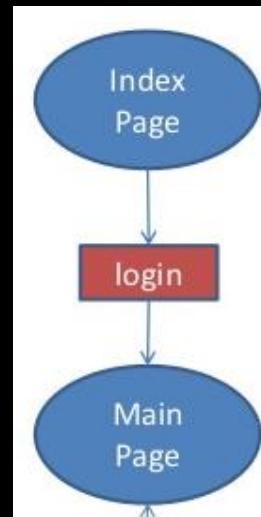


Second Project improvement

- POs
 - Create **BasePagePO**
 - To hide here **Selenium WebDriver calls**

```
public void type(By element, String text) {  
    find(element).sendKeys(text);  
}
```

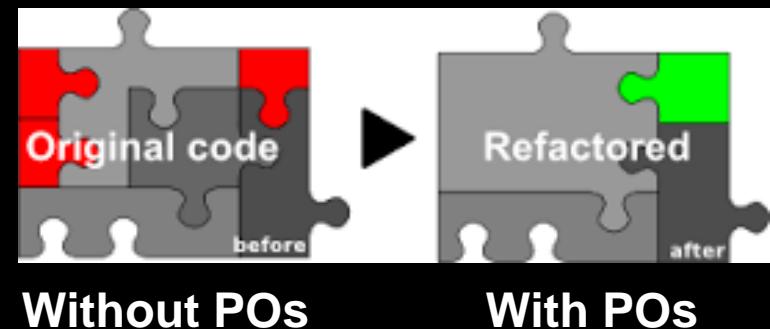
- Action methods should return POs (if any)
 - When you ‘navigate’ you should return a PO for the next page
 - Login example



Test Script Refactoring

(Create BasePagePO)

- **Create BasePagePO** ←
- Page Objects creation
 - Two POs (one for page)
 - LoginFormPO (extends BasePagePO)
 - Refactoring
 - LoginSuccessPO (extends BasePagePO)
 - Refactoring
 - Test scripts building
 - Using the POs



BasePagePO

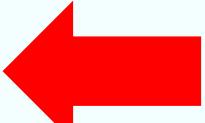
All the Selenium WebDriver calls should be in this class!

A sort of Domain Specific Language (DSL)

```
public class BasePagePO {  
    protected WebDriver driver;  
  
    public BasePagePO(WebDriver driver) {  
        this.driver = driver;  
    }  
    public void visit(String url) {  
        driver.get(url);  
    }  
    public WebElement find(By element) {  
        return driver.findElement(element);  
    }  
    public void click(By element) {  
        find(element).click();  
    }  
    public void type(By element, String text) {  
        find(element).sendKeys(text);  
    }  
    public boolean isIn(By element) {  
        return find(element).isDisplayed();  
    }  
    public String getUrl() {  
        return driver.getCurrentUrl();  
    }  
}
```

LoginFormPO

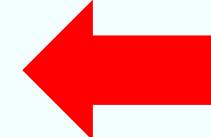
```
public class LoginFormPO extends BasePagePO {  
  
    private By usernameInput = By.id("username");  
    private By passwordInput = By.id("password");  
    private By submitButton = By.cssSelector("button");  
    private By invalidBox = By.id("invalid");  
  
    public LoginFormPO(WebDriver driver) {  
        super(driver);  
        visit("https://bonigarcia.dev/selenium-webdriver-java/login-form.html");  
    }  
  
    public BasePagePO with(String username, String pwd) {  
        type(usernameInput, username);  
        type(passwordInput, pwd);  
        click(submitButton);  
        if (getUrl().equals("https://bonigarcia.dev/selenium-webdriver-java/login-form.html"))  
            return this;  
        else return new LoginSuccessPO(driver);  
    }  
  
    public boolean invalidBoxIsPresent() {  
        return isIn(invalidBox);  
    }  
}
```



We can use the methods we created!

LoginSuccessPO

```
public class LoginSuccessPO extends BasePagePO {  
  
    private By successBox = By.id("success");  
  
    public LoginSuccessPO(WebDriver driver) {  
        super(driver);  
    }  
  
    public boolean successBoxIsPresent() {  
        return isIn(successBox);  
    }  
}
```

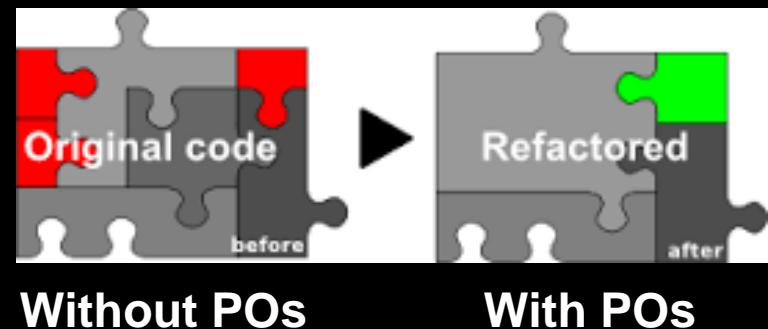


Test Script Refactoring

(Action methods returns POs)

- Page Objects creation
 - Two POs (one for page)
 - LoginFormPO
 - Refactoring
 - LoginSuccessPO
 - Refactoring
- Test scripts building (**Refactoring**)
 - Using the POs

← Actions POs methods returning POs

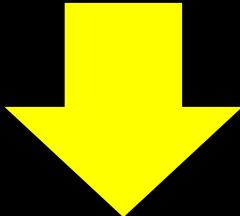


Action methods returning POs

When the button is clicked
a novel page is loaded

LoginFormPO (before)

```
public void with(String username, String pwd) {  
    driver.findElement(By.id("username")).sendKeys(username);  
    driver.findElement(By.id("password")).sendKeys(pwd);  
    driver.findElement(By.className("mt-2")).click();  
}
```



```
login.with("user", "user");  
// goto login-success page
```

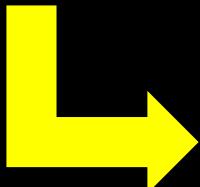
LoginFormPO (after)

```
public BasePagePO with(String username, String pwd) {  
    type(usernameInput, username);  
    type(passwordInput, pwd);  
    click(submitButton);  
    if (getUrl().equals("https://bonigarcia.dev/selenium-webdriver-java/login-form.html"))  
        return this;  
    else return new LoginSuccessPO(driver);  
}
```

Test script (final)

```
public class LoginTest extends DriverLifeCycleSetting {  
  
    private LoginFormPO login;  
    private LoginSuccessPO loginSuccess;  
  
    @Test  
    public void testLoginOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "user");  
        // goto login-success page  
        loginSuccess = new LoginSuccessPO(driver);  
        assertTrue(loginSuccess.successBoxIsPresent());  
    }  
  
    @Test  
    public void testLoginNoOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "error");  
        // remain in the login-form page  
        assertTrue(login.invalidBoxIsPresent());  
    }  
}
```

Before

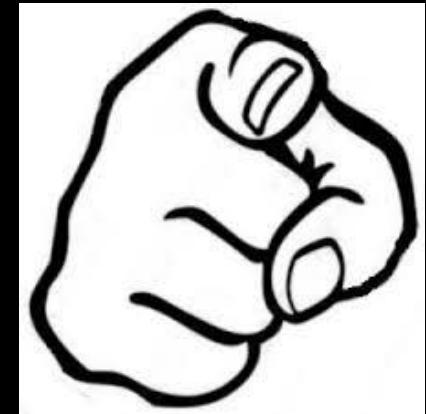


```
public class LoginTest extends DriverLifeCycleSetting {  
  
    private LoginFormPO login;  
    private LoginSuccessPO loginSuccess;  
  
    @Test  
    public void testLoginOK() {  
        login = new LoginFormPO(driver);  
        loginSuccess = (LoginSuccessPO) login.with("user", "user");  
        assertTrue(loginSuccess.successBoxIsPresent());  
    }  
  
    @Test  
    public void testLoginNoOK() {  
        login = new LoginFormPO(driver);  
        login.with("user", "error");  
        assertTrue(login.invalidBoxIsPresent());  
    }  
}
```

After

Now it's your turn!

- Open the VS Code Login project and refactor it
- Create BasePagePO
- Refactor test scripts so that action methods return POs
- Execute the LoginTest class



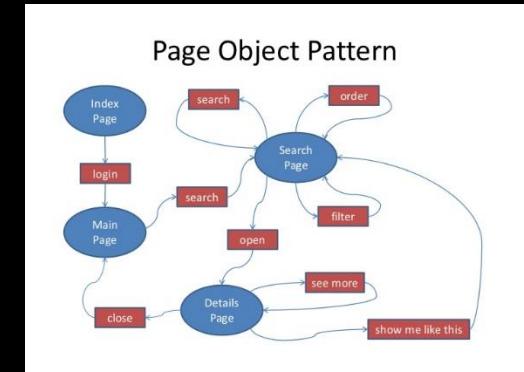
Page Objects Best practices



1. A page object should not have any assertion
 - but only **getter methods**
2. A page object should represent meaningful elements of a page and not necessarily a complete page
3. When you navigate you should return a page object for the next page
4. Building the **Page Object (navigation) Model**
 - POs + methods to move from a PO to another
5. **Factorize common components of a page**
 - Ex. 'menu' present in several pages = PO component
the pages containing the Menu = PO containers

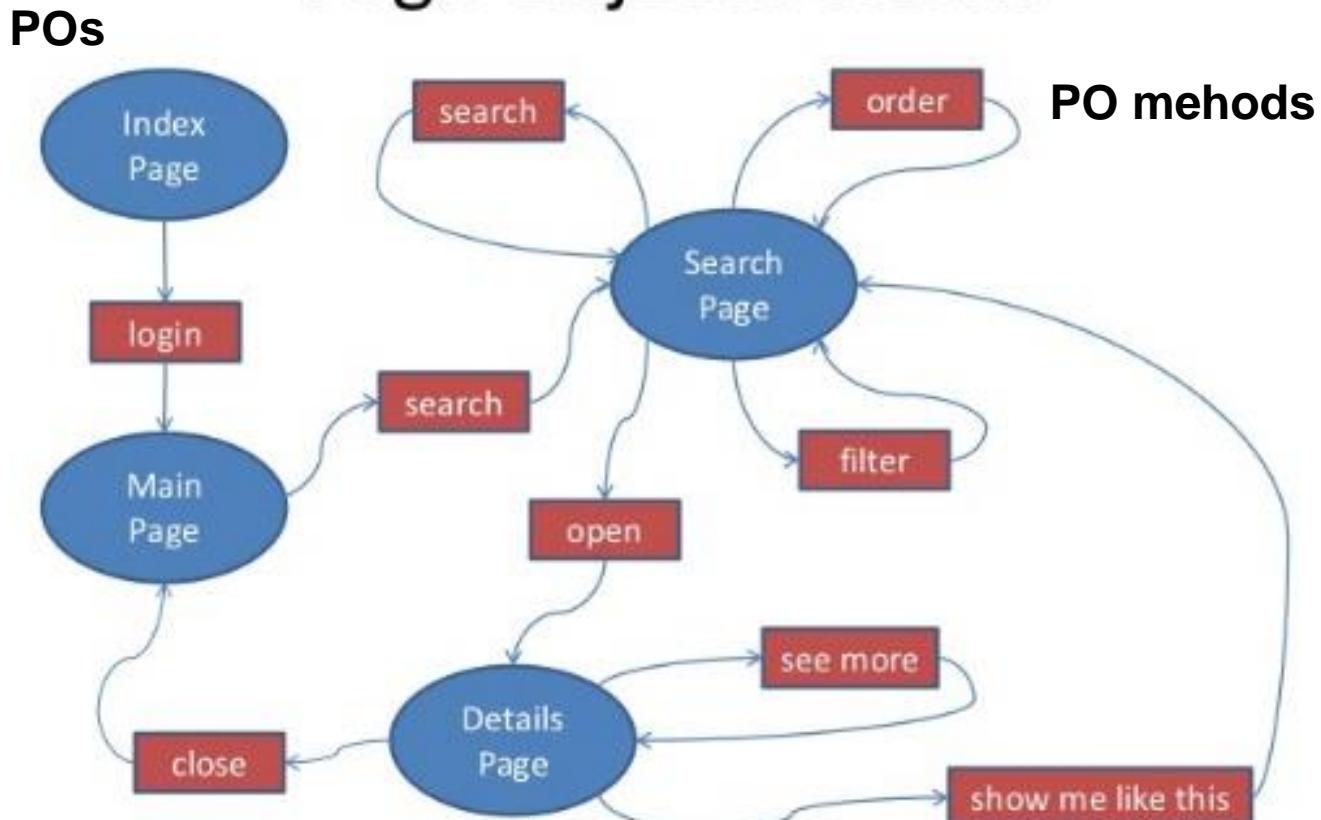
Page Object Model

- **Perform a breakdown of the application under test**
 - Enumerate various **Web pages** to interact with
 - they become the **POs**
 - List down **actions** to do with those pages, like entering form data or clicking buttons
 - they become the **action methods of POs**
- **Create a graph where from a page it is possible to navigate in another (if it exists a PO method)**



POM Example

Page Object Pattern



PO Containers and Components (1)

Home

The screenshot shows the homepage of the Software Engineering & Programming Languages Research Group. At the top, there's a navigation bar with links for Home, People, Publications, Projects, Research, Meetings, Theses & Stages, and Contacts. The 'Home' link is highlighted with a red box. Below the navigation, there's a section titled 'Welcome to the Software Engineering & Programming Languages Research Group web site'. It contains text about their research focus on reliable software systems and a list of current research interests. On the right, there's a graphic of a black robot holding a lightbulb, with text above it reading 'Tirocini Prove Finali Tesi' and 'Leggi QUI!'. Logos for DIBRIS and UNIGE are at the bottom left, and news items are at the bottom right.

People

The screenshot shows the 'People' page of the same website. The navigation bar has 'People' highlighted with a red box. The page lists several faculty members with their profiles, including Egidio Astesiano, Gianna Reggio, Mauro Cerioli, Elena Zucca, Filippo Ricca, and Davide Ancona. Each profile includes a photo, title, contact information, and a 'Personal Web Page' link. A sidebar on the left shows the 'People' link in the navigation bar.

Some HTML portions are in common (ex. Menu)

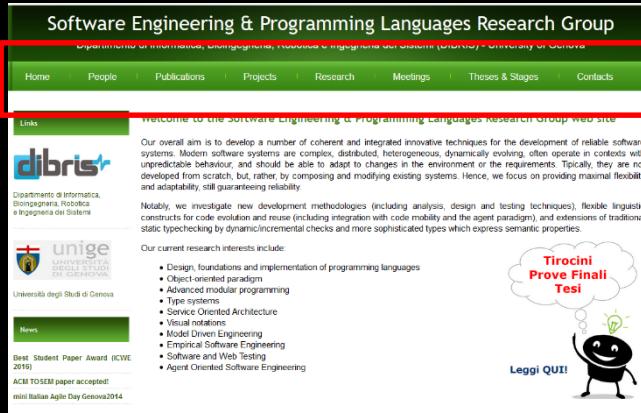
```
public class HomePO extends PageObject {  
    @FindBy(how=How.XPATH, xpath="...")  
    private WebElement home;  
    @FindBy(how=How.XPATH, xpath = "...")  
    private WebElement people;  
  
    public HomePO goToHome() {  
        home.click();  
        return new HomePO(driver);  
    }  
  
    public PeopleContainerPO goToPeople() {  
        people.click();  
        return new PeoplePO(driver);  
    }  
    ...  
}
```

```
public class PeoplePO extends PageObject {  
    @FindBy(how=How.XPATH, xpath="...")  
    private WebElement home;  
    @FindBy(how=How.XPATH, xpath = "...")  
    private WebElement people;  
  
    public HomePO goToHome() {  
        home.click();  
        return new HomePO(driver);  
    }  
  
    public PeopleContainerPO goToPeople() {  
        people.click();  
        return new PeoplePO(driver);  
    }  
    ...  
}
```

Bad choice!! A lot of duplicated code (all menu methods)!

PO Containers and Components (2)

Home



MenuComponentPO

```
public class MenuComponentPO extends PageObject {  
    @FindBy(how=How.XPATH, xpath = "...")  
    private WebElement home;  
    @FindBy(how=How.XPATH, xpath = "...")  
    private WebElement people;  
  
    public HomeContainerPO goToHome() {  
        home.click();  
        return new HomeContainerPO(driver);  
    }  
  
    public PeopleContainerPO goToPeople() {  
        people.click();  
        return new PeopleContainerPO(driver);  
    }  
  
    ...  
}
```

MenuComponentPO

extend

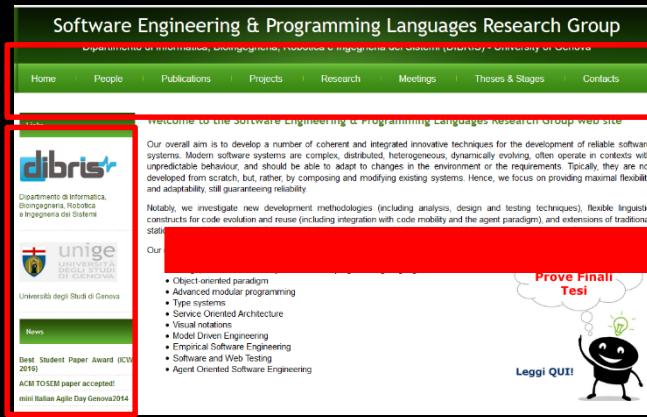
HomeContainerPO

PeopleContainerPO

PubContainerPO

...

PO Containers and Components (3)



```
public class MenuComponentPO extends PageObject {  
    ...  
}
```

```
public class LinksComponentPO extends PageObject {  
    ...  
}
```

MenuComponentPO

LinksComponentPO

Multiple Inheritance
Not easy in Java!

HomeContainerPO

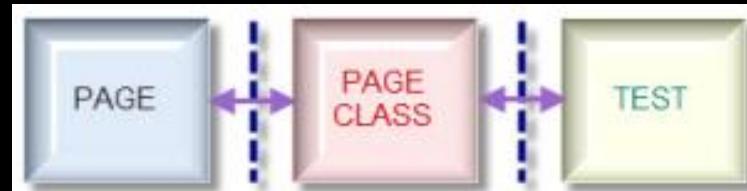
PeopleContainerPO

PubContainerPO

...

To Summarize ...

- **Page Objects are a design pattern**
 - we used it in Selenium WebDriver but this pattern is more general
- **POs are useful**
 - improve the quality of Test scripts
 - reduce the number of LOCs (to write / to maintain)
 - reduce the test suite maintenance time



Page Object pattern Videos

<https://www.youtube.com/watch?v=-OF-YBAQdGE>

<https://www.youtube.com/watch?v=tnWig6KfQ9w>

Functional and Security Testing Course

2023-2024

Selenium WebDriver API Web Form and Waiting – lesson 10

Filippo Ricca
DIBRIS, University of Genova

AGENDA



- **Web Form**
 - Text Input, Dropdown lists, Check and Radio buttons, Calendar ...
- **Waiting mechanisms**
 - Thread.sleep(), Implicit waits, Explicit waits, ...
- **User Gestures**
 - Double-click, Right-click, Mouse-over, Drag and drop, Copy and paste, ...

WEB FORM

Web Form

<https://bonigarcia.dev/selenium.webdriver-java/web-form.html>

Web form

Text input

Password

Textarea

Disabled input

Readonly input

Dropdown (select)

Dropdown (datalist)

File input

 Nessun file selezionato.

Checked checkbox

Default checkbox

Checked radio

Default radio

Color picker

Date picker

Example range

[Return to index](#)

Web form

Text input

Password

Textarea

Disabled input

Readonly input

[Return to index](#)

Text Input

```
Text input
<input id="my-text-id" class="form-control" type="text" name="my-text" myprop="myvalue">
```

To recover the value inserted in a input form

```
@Test
public void test_text_input() throws InterruptedException {
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";
    // launch Chrome and direct it to the Base URL
    driver.get(baseUrl);
    WebElement input = driver.findElement(By.id("my-text-id"));
    Thread.sleep(3000);
    input.sendKeys("ciao");
    assertEquals("ciao", input.getAttribute("value"));
    // the following does not work
    //assertEquals("ciao", input.getText());
    Thread.sleep(3000);
    input.clear();
    Thread.sleep(3000);
}
```

Web form

Text input

Password

Textarea

Disabled input

Readonly input

[Return to index](#)

Disabled Input

```
Disabled input
<input class="form-control" type="text" name="my-disabled" placeholder="Disabled input" disabled="">
```

```
@Test
```

```
public void test_disablad_input() throws InterruptedException {
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";
    // launch Chrome and direct it to the Base URL
    driver.get(baseUrl);
    // it will fail because the web element is not interactable
    WebElement input = driver.findElement(By.name("my-disabled"));
    input.sendKeys("ciao");
    assertEquals("ciao", input.getAttribute("value"));
    Thread.sleep(3000);
}
```

Web form

Text input

Password

Textarea

Disabled input

Disabled input

Readonly input

[Return to index](#)

ReadOnly Input

```
 Readonly input
<input class="form-control" type="text" name="my_READONLY" value=" Readonly input" readonly="">
```

```
@Test
public void test_READONLY_input() throws InterruptedException {
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";
    // launch Chrome and direct it to the Base URL
    driver.get(baseUrl);
    WebElement input = driver.findElement(By.name("my_READONLY"));
    input.sendKeys("ciao");
    // it will fail because the value is Readonly input
    assertEquals("ciao", input.getAttribute("value"));
    Thread.sleep(3000);
}
```

- Create a Test script for each element in this part of the Form
- Also for
 - Password
 - TextArea

Web form

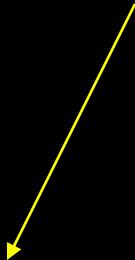
Text input

Password

Textarea

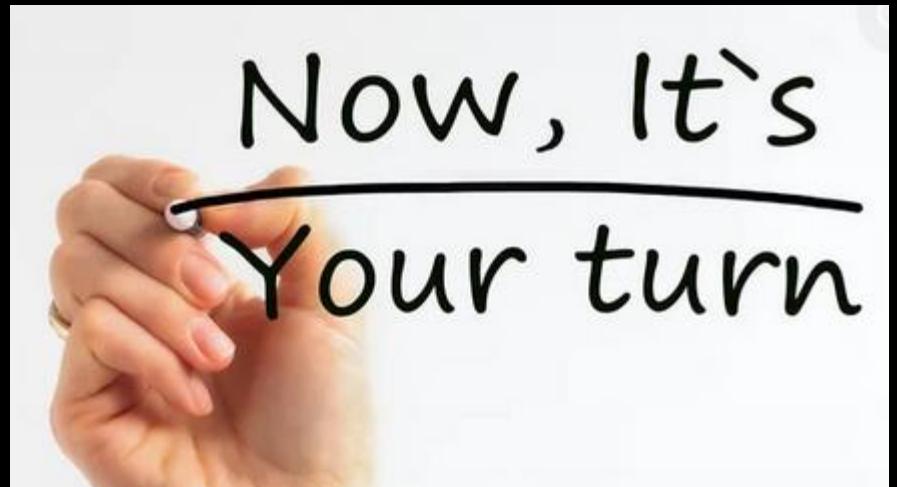
Disabled input

Readonly input



Suggestion:

- Start from a working VS code project
 - e.g., HelloWorld
- Modify it

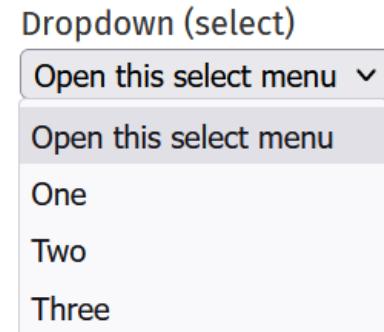


Managing Dropdown lists

HTML Code

```
1 <label class="form-label w-100">Dropdown (select)
2   <select class="form-select" name="my-select">
3     <option selected>Open this select menu</option>
4     <option value="1">One</option>
5     <option value="2">Two</option>
6     <option value="3">Three</option>
7   </select>
8 </label>
9
```

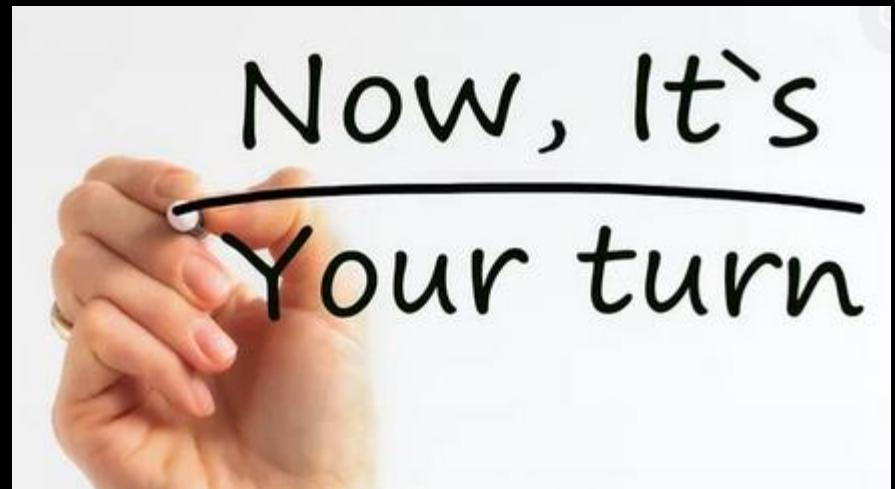
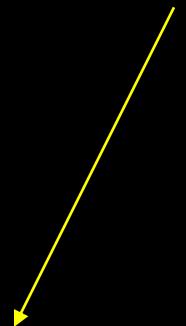
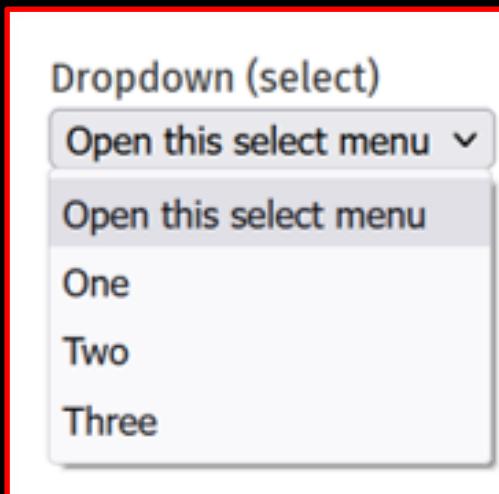
Browser



```
@Test
public void test_dropdownlist_vISIBLEtext() throws InterruptedException {
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";
    // launch Chrome and direct it to the Base URL
    driver.get(baseUrl);
    Select dropdown = new Select(driver.findElement(By.name("my-select")));
    Thread.sleep(3000);
    dropdown.selectByVisibleText("Three");
    Thread.sleep(3000);
    assertEquals("Three", dropdown.getFirstSelectedOption().getText());
}
```

selectByIndex - It is used to select an option based on its index, beginning with 0
selectByValue - It is used to select an option based on its 'value' attribute

- Create some Test scripts to test Dropdown List
- Use:
 - **SelectByVisibleText**
 - **SelectByIndex**
 - **SelectByValue**

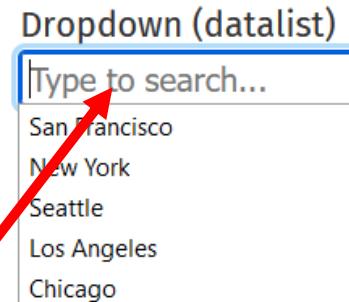


Managing Data Lists

HTML Code

```
1 <label class="form-label w-100">Dropdown (datalist)
2   <input class="form-control" list="my-options"
3     name="my-datalist" placeholder="Type to search...">
4
5     <datalist id="my-options">
6       <option value="San Francisco">
7       <option value="New York">
8       <option value="Seattle">
9       <option value="Los Angeles">
10      <option value="Chicago">
11    </datalist>
12  </label>
```

Browser

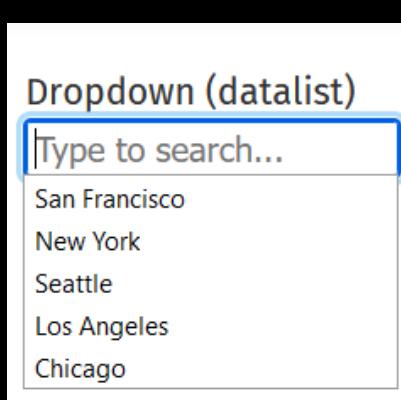


Here we can write a text or selecting an option

Different from Dropdown lists: Click, Select, Read and Write

Managing Data Lists

```
@Test
public void test_datalist() throws InterruptedException {
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";
    // launch Chrome and direct it to the Base URL
    driver.get(baseUrl);
    Thread.sleep(3000);
    WebElement datalist = driver.findElement(By.name("my-datalist"));
    datalist.click(); Click, Open Data List
    Thread.sleep(3000);
    WebElement option = driver.findElement(By.xpath("//datalist/option[2]")); Select
    String optionValue = option.getAttribute("value"); Read
    datalist.sendKeys(optionValue); Write
    Thread.sleep(3000);
    assertEquals("New York", datalist.getAttribute("value"));
}
```



Numeration starts from one

Managing Check and Radio buttons

HTML Code

```
<br>
<input id="vfb-7-1" type="radio" name="webform" value="Option 1"> overflow
Option1
<br> overflow
<input id="vfb-7-2" type="radio" name="webform" value="Option 2"> overflow
Option2
<br> overflow
<input id="vfb-7-3" type="radio" name="webform" value="Option 3"> overflow
Option3
<br> overflow
<br> overflow
<strong>Checkbox</strong> overflow
<br> overflow
<input id="vfb-6-0" type="checkbox" name="webform" value="checkbox1"> overflow
checkbox1
<br> overflow
<input id="vfb-6-1" type="checkbox" name="webform" value="checkbox2"> overflow
checkbox2
<br> overflow
<input id="vfb-6-2" type="checkbox" name="webform" value="checkbox3"> overflow
checkbox3
```

Browser

| |
|------------------------------------|
| Radio |
| <input type="radio"/> Option1 |
| <input type="radio"/> Option2 |
| <input type="radio"/> Option3 |
| Checkbox |
| <input type="checkbox"/> Checkbox1 |
| <input type="checkbox"/> Checkbox2 |
| <input type="checkbox"/> Checkbox3 |

Radio: used to choose one option

Checkbox: used to choose many options

```
@Test
public void tryCheckbox(){

    WebElement option1 = driver.findElement(By.id("vfb-6-0")); 1

    //This will Toggle On the Check box
    option1.click(); 2

    //Check whether the Check box is toggled on
    if(option1.isSelected()){ 3
        System.out.println("Checkbox is Toggled On");
    }
    else{
        System.out.println("Checkbox is Toggled Off");
    }

    // This should Toggle Off the Check box
    option1.click(); 4

    // Lets see whether its Toggled Off
    if(!option1.isSelected()){
        System.out.println("Checkbox is now Toggled Off !!");
    }

}
```

1. Locate the checkbox element by its ID

2. click() toggles on the checkbox

3. isSelected() gives the Toggle status of the checkbox

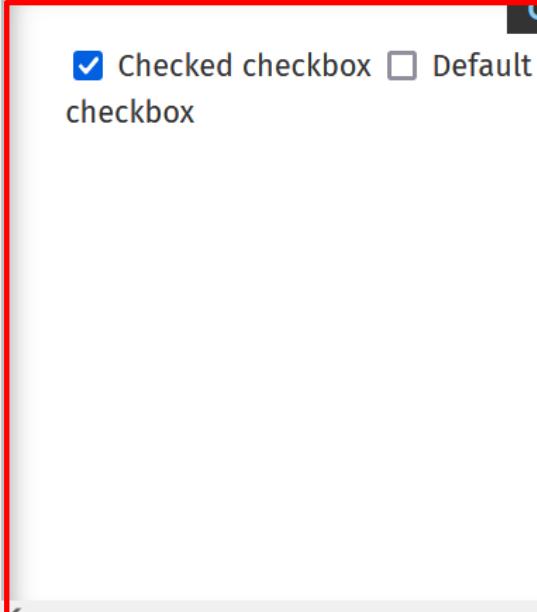
4. click() again on the checkbox turns the Toggle off

CheckBox Buttons

HTML Code

```
1 <div class="form-check">
2     <label class="form-check-label w-100">
3         <input class="form-check-input" type="checkbox"
name="my-check" id="my-check-1" checked> Checked
checkbox
4     </label>
5
6     <label class="form-check-label w-100">
7         <input class="form-check-input"
type="checkbox" name="my-check" id="my-check-2"> Default
checkbox
8     </label>
9 </div>
```

Browser



✓ Checked checkbox □ Default
checkbox

```
@Test
public void test_check_and_radio() throws InterruptedException {
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";
    // launch Chrome and direct it to the Base URL
    driver.get(baseUrl);
    Thread.sleep(3000);
    WebElement check = driver.findElement(By.id("my-check-2"));
    check.click();
    assertTrue(check.isSelected());
    Thread.sleep(3000);
}
```

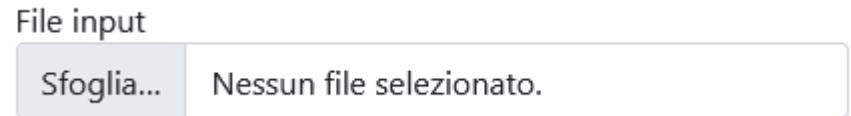
Uploading a File

HTML Code

```
<label class="form-label w-100">File input  
  <input class="form-control" type="file" name="my-file">  
</label>
```

Browser

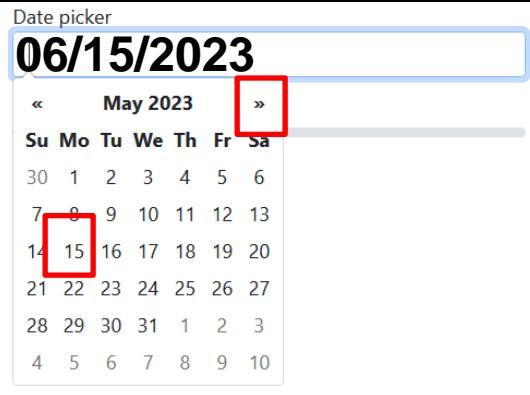
It creates a temporary file tempfileXXX.tmp



```
@Test  
public void test_uploadFile() throws InterruptedException, IOException {  
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";  
    // launch Chrome and direct it to the Base URL  
    driver.get(baseUrl);  
    Thread.sleep(3000);  
    WebElement inputFile = driver.findElement(By.name("my-file"));  
    Path tempFile = Files.createTempFile("tempfile", ".tmp");  
    String filename = tempFile.toAbsolutePath().toString();  
    System.out.println(filename);  
    Thread.sleep(3000);  
    inputFile.sendKeys(filename);  
    Thread.sleep(5000);  
    driver.findElement(By.tagName("form")).submit();  
    Thread.sleep(3000);  
    assertEquals(driver.getCurrentUrl(), "https://bonigarcia.dev/selenium-webdriver-jav  
}
```

filename is a string formed by absolute path + file name

Managing a Calendar



Test Script Example:

- 1) Open Date picker
- 2) Click right arrow (we switch to June)
- 3) Click on day 15
- 4) Assert date shown is 06/15/2023 (dd/mm/aaaa)

```
@Test  
public void test_datePicker() throws InterruptedException, IOException {  
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/web-form.html";  
    // launch Chrome and direct it to the Base URL  
    driver.get(baseUrl);  
    Thread.sleep(3000);  
    // Click on the date picker to open the calendar  
    WebElement datePicker = driver.findElement(By.name("my-date"));  
    datePicker.click();  
    Thread.sleep(3000); Click, Open Data picker  
    // Click on the right arrow using xpath  
    WebElement arrowRight = driver.findElement(By.xpath("//html[1]/body[1]/div[1]/div[1]/table[1]/tbody[1]/tr[1]/td[2]/a[1]"));  
    arrowRight.click();  
    Thread.sleep(3000); Click Arrow  
    // Click on the day 15  
    WebElement day = driver.findElement(By.xpath("//td[contains(text(), '15')]"));  
    day.click();  
    Thread.sleep(3000); Select and click day 15  
    String optionValue = datePicker.getAttribute("value");  
    assertEquals(optionValue, "06/15/2023");  
}
```

WAITING

Time issue

- There are **delays** between submitting a request and receiving the response



- A Human tester (during manual testing) can wait until the response page is loaded
 - Testing frameworks doesn't know!
- Sometimes a test script doesn't work if
 - Submit a request
 - Verify the response immediately
- **Solution:**
 - **Add a waiting command to delay 'a little' the execution!**
 - **In the previous examples, we used Thread.sleep()**

Loading Images

<https://bonigarcia.dev/selenium-webdriver-java/loading-images.html>

Hands-On Selenium WebDriver with Java

Practice site

Done!

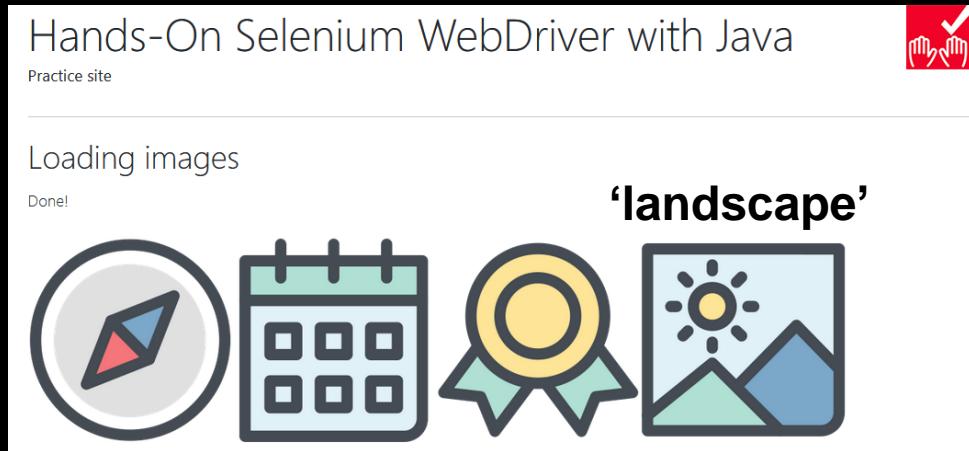
Loading images



The screenshot shows a web page titled "Hands-On Selenium WebDriver with Java" with a subtitle "Practice site". Below the title, there is a "Done!" button and the text "Loading images". At the bottom, there are four large, colorful icons: a compass rose, a calendar, a ribbon badge, and a landscape scene.

about 10 seconds to finish ...

Possible Test script (not working)

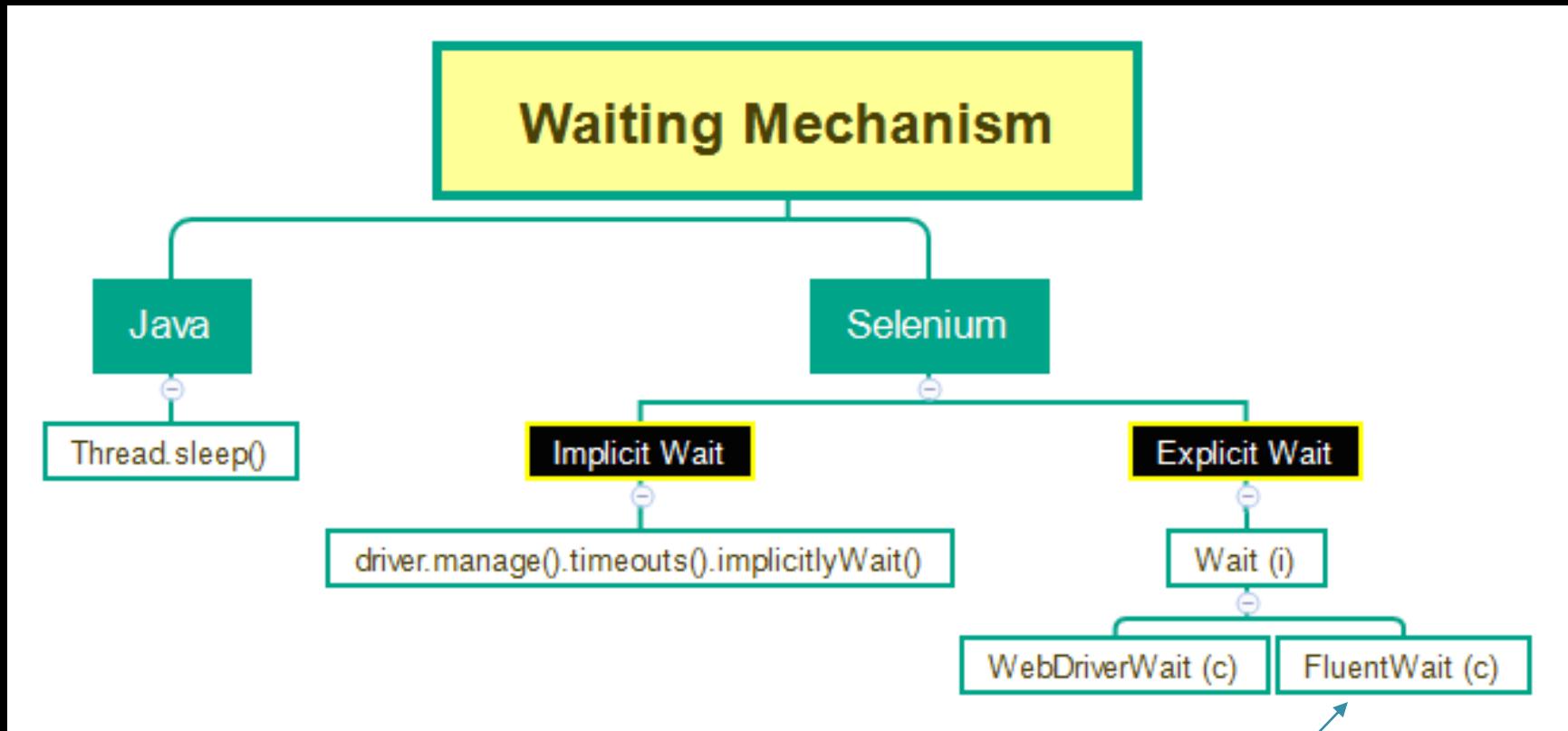


about 10 seconds to finish ...

Failure, web element not found!

```
@Test
public void testA_WithoutWait() throws InterruptedException {
    driver.get(
        "https://bonigarcia.dev/selenium-webdriver-java/loading-images.html");
    WebElement landscape = driver.findElement(By.id("landscape"));
    assertTrue(landscape.getAttribute("src").contains("landscape"));
}
```

Waiting Mechanisms



It is not part of the FSTT program course

Thread.sleep()

- **Thread.sleep(millis)**
 - Not a good choice (to avoid!)
 - It can produce flakiness
 - A test script can pass or fail in non-deterministic way
 - It can delay the test suite execution
 - Execution is stopped for a fixed time (e.g., 10 seconds)

Add before ‘WebElement line’: Thread.sleep(10000)

```
@Test
public void testA_WithoutWait() throws InterruptedException {
    driver.get(
        "https://bonigarcia.dev/selenium webdriver java/loading-images.html");
    WebElement landscape = driver.findElement(By.id("landscape"));
    assertTrue(landscape.getAttribute("src").contains("landscape"));
}
```

Implicit waits

- It implicitly delays the exception raised by Selenium of ‘N’ seconds in case the web element is not found (driver.findElement())
 - Ex. *driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(5))*
 - *For each findElement() exception is raised only after 5 seconds*
- Better than Thread.sleep(), Polling mechanism, no delay!
 - If the web element is found the execution starts before ‘N’ seconds

```
@Test
public void testD_ImplicitWait() throws InterruptedException {
    driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(seconds:10));
    driver.get(
        "https://bonigarcia.dev/selenium-webdriver-java/loading-images.html");
    WebElement landscape = driver.findElement(By.id("landscape"));
    assertTrue(landscape.getAttribute("src").contains(s:"landscape"));
}
```

Explicit waits

- Explicit wait stops the execution of the test script **based on a certain condition** (for a specified amount of time)
 - E.g., elementToBeClickable()

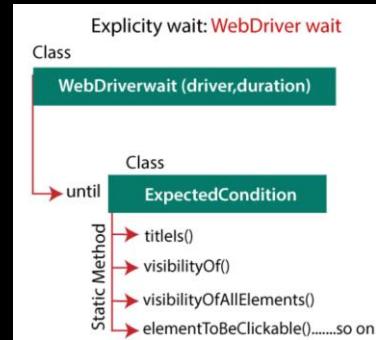
Timeout in seconds, after that time if the condition remain false, the exception is raised

```
WebDriverWait wait = new WebDriverWait(WebDriverReference, TimeOut);
```

```
WebElement element =  
wait.until(ExpectedConditions.elementToBeClickable(By.id("gh-btn"))));
```

Element with id 'gh-btn'

Types of expected conditions



presenceOfElementLocated

```
public static ExpectedCondition<WebElement> presenceOfElementLocated(By locator)
```

An expectation for checking that an element is present on the DOM of a page. This does not necessarily mean that the element is visible.

Parameters:

locator - used to find the element

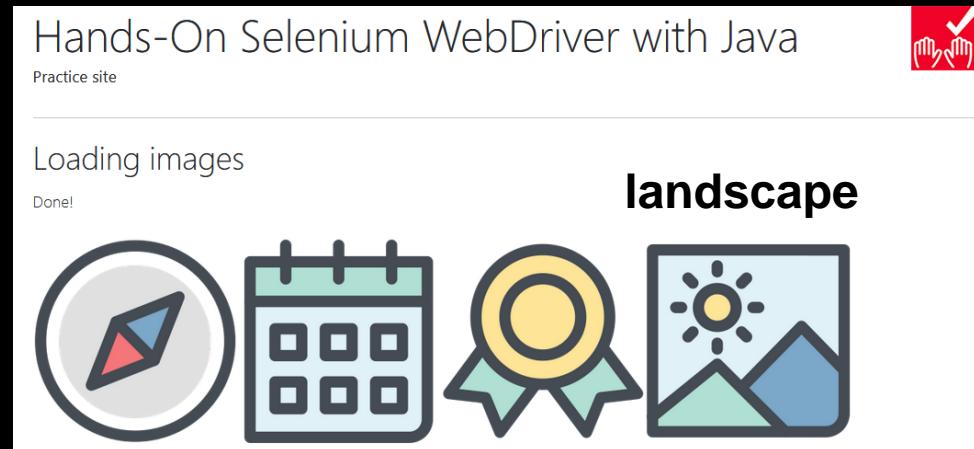
Returns:

the WebElement once it is located

There are many expected conditions available, which you can refer through the Selenium official page (it is in general complex to decide which to use!)

<https://www.selenium.dev/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html>

Test script with explicit wait



about 10 seconds to finish ...

```
@Test
public void testE_ExplicitWait() throws InterruptedException {
    driver.get(
        "https://bonigarcia.dev/selenium-webdriver-java/loading-images.html");
    WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(seconds:10));
    WebElement landscape = wait.until(ExpectedConditions.elementToBeClickable(By.id("landscape")));
    assertTrue(landscape.getAttribute("src").contains(s:"landscape")));
}
```

Explicit wait Selenium Video

<https://www.youtube.com/watch?v=x6ei6WOfQBo>

Slow Calculator

<https://bonigarcia.dev/selenium-webdriver-java/slow-calculator.html>

Slow calculator

This calculator waits seconds to get the result of basic arithmetic operations.



Slow Calculator Test Script

Slow calculator

This calculator waits seconds to get the result of basic arithmetic operations.



Failure!!!

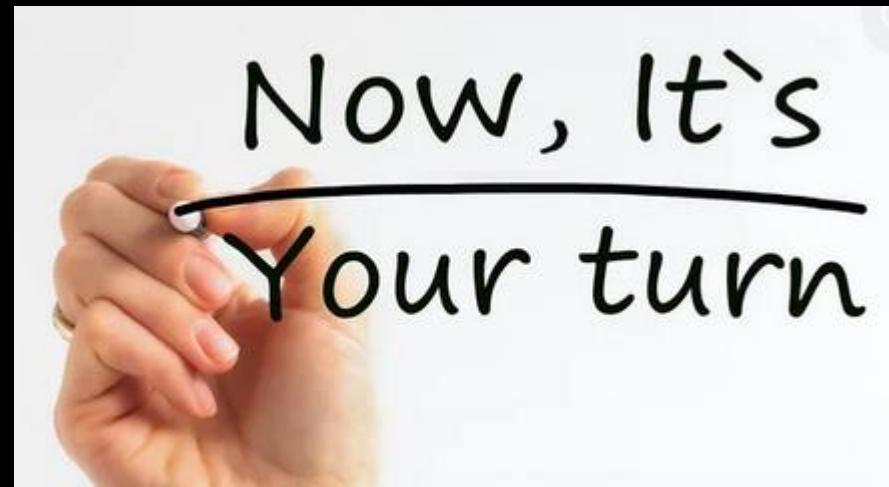
```
@Test  
public void test_Calculator_noWait() throws InterruptedException {  
    String baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/slow-calculator.html";  
    // launch Chrome and direct it to the Base URL  
    driver.get(baseUrl);  
    Thread.sleep(3000);  
    // 1 + 3  
    driver.findElement(By.xpath("//span[text()='1']")).click(); Thread.sleep(1000);  
    driver.findElement(By.xpath("//span[text()='+' ]")).click(); Thread.sleep(1000);  
    driver.findElement(By.xpath("//span[text()='3']")).click(); Thread.sleep(1000);  
    driver.findElement(By.xpath("//span[text()='=' ]")).click(); Thread.sleep(1000);  
  
    // ... should be 4, wait for it  
    assertEquals("4", driver.findElement(By.className("screen")).getText());  
}
```

← you have to insert a wait!

- Create some Test scripts to test Slow calculator
- Use:
 - Thread.sleep
 - Implicit Waits (is it possible?)
 - Explicit Waits
- If you want you can deliver it (assignment not mandatory)

Slow calculator

This calculator waits seconds to get the result of basic arithmetic operations.



USER GESTURES

User Gestures

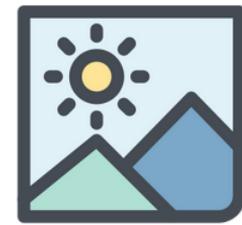
Dropdown menu

Use left-click here ▾

Use right-click here ▾

Use double-click here ▾

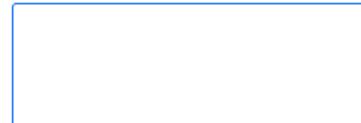
Mouse over



Drag and drop

Draggable panel

Drag me



Actions class

- Selenium WebDriver provides the class **Actions**, a powerful asset to **automate different user actions**, both for keyboard and mouse
- This class follows the ***builder pattern***
 - This way, you can chain several methods (i.e., different actions) and perform all of them at the end by calling to build().perform()
 - Ex. `WebElement dropdown3 = driver.findElement(By.id("my-dropdown-3"));
actions.doubleClick(dropdown3).build().perform();`

API Actions Class

Table 3-11. Actions methods

| Method | Description |
|--|--|
| <code>clickAndHold()</code> <code>clickAndHold(WebElement target)</code> | Clicks without releasing in the current position (or the middle of a given element) |
| <code>release()</code> <code>release(WebElement target)</code> | Release the left-click mouse button previously pressed with <code>clickAndHold()</code> |
| <code>click()</code> <code>click(WebElement target)</code> | Click on the current position (or a given element) |
|  <code>doubleClick()</code> <code>doubleClick(WebElement target)</code> | Double-click on the current position (or element) |
|  <code>contextClick()</code> <code>contextClick(WebElement target)</code> | Right-click on the current position (or element) |
| <code>moveToElement(WebElement target)</code> <code>moveToElement(WebElement target, int xOffset, int yOffset)</code> | Move mouse cursor to the middle (or shifted to a given offset) of a given element |
| <code>moveByOffset(int xOffset, int yOffset)</code> | Moves the mouse from its current position (0,0 by default) by the given offset |
|  <code>dragAndDrop(WebElement source, WebElement target)</code> <code>dragAndDropBy(WebElement</code> | This action consists of three steps: 1) Click-and-hold at the middle (or shifted by a given offset) of the source element location. 2) Move to mouse to the target element location. 3) Release the mouse click. |

Click, double-click, right click

<https://bonigarcia.dev/selenium-webdriver-java/dropdown-menu.html>

Dropdown menu

Use left-click here ▾

Use right-click here ▾

Use double-click here ▾

```
@Test
public void CtestDoubleClick() throws InterruptedException {
    driver.get("https://bonigarcia.dev/selenium-webdriver-java/dropdown-menu.html")
    Actions actions = new Actions(driver);
    WebElement dropdown3 = driver.findElement(By.id("my-dropdown-3"));
    actions.doubleClick(dropdown3).build().perform();
    Thread.sleep(3000);
    WebElement contextMenu2 = driver.findElement(By.id("context-menu-3"));
    assertTrue(contextMenu2.isDisplayed());
}
```

Mouse Over

<https://bonigarcia.dev/selenium-webdriver-java/mouse-over.html>

Mouse over



Compass



```
moveToElement(WebElement tar  
get)
```

Move mouse cursor to the middle (or shifted to a given offset) of a given element

```
@Test  
public void AtestMouseOverSimple() throws InterruptedException {  
    driver.get("https://bonigarcia.dev/selenium-webdriver-java/mouse-over.html");  
    Actions actions = new Actions(driver);  
    Thread.sleep(3000);  
    WebElement image = driver.findElement(By.xpath("//img[@src='img/compass.png']"));  
    actions.moveToElement(image).build().perform();  
    WebElement caption = driver.findElement(By.xpath("//p[contains(text(),'Compass')]"));  
    assertEquals(caption.getText(), "Compass");  
}
```

Drag and Drop

<https://bonigarcia.dev/selenium-webdriver-java/drag-and-drop.html>

Drag and drop



```
@Test
public void AtestDragAndDropSimple() throws InterruptedException {
    driver.get(
        "https://bonigarcia.dev/selenium-webdriver-java/drag-and-drop.html");
    Thread.sleep(3000);
    Actions actions = new Actions(driver);
    WebElement draggable = driver.findElement(By.id("draggable"));
    // pixel
    int offset = 100;
    // dragAndDropBy(WebElement source, int xOffset, int yOffset)
    actions.dragAndDropBy(druggable, offset, 0).build().perform();
    Thread.sleep(3000);
}
```



Copy And Paste

<https://bonigarcia.dev/selenium-webdriver-java/web-form.html>

Text input
hello World

Password

Textarea



```
@Test
public void testCopyPaste() throws InterruptedException {
    driver.get(
        "https://bonigarcia.dev/selenium-webdriver-java/web-form.html");
    Actions actions = new Actions(driver);
    WebElement inputText = driver.findElement(By.name("my-text"));
    WebElement textarea = driver.findElement(By.name("my-textarea"));
    Thread.sleep(3000);
    // per MAC sostituire con Keys.COMMAND
    Keys modifier = Keys.CONTROL;
    // inserisco hello world nell'input text field
    actions.sendKeys(inputText, "hello world");
    // attivo il key down del CONTROL (equivale a tenere premuto Ctrl)
    actions.keyDown(modifier);
    // mi posiziono sull'input text field
    actions.sendKeys(inputText, "a");
    // eseguo ctrl+c che copia il valore contenuto
    actions.sendKeys(inputText, "c").build().perform();
    Thread.sleep(3000);
    // eseguo ctrl+v che fa il paste nella textarea
    actions.sendKeys(textarea, "v").build().perform();
    Thread.sleep(3000);
    assertEquals(inputText.getAttribute("value"), textarea.getAttribute("value"));
}
```

Functional and Security Testing Course

2023-2024

Cross/Compatibility Browser Testing – lesson 12

Filippo Ricca
DIBRIS, University of Genova

AGENDA

- **Browser Compatibility Testing**
 - Selenium WebDriver
 - Selenium Grid
 - Cloud based solutions
 - LambdaTest
- Implementing Browser Compatibility testing with JUnit 5 and Selenium WebDriver



Browser Compatibility Testing

Browsers Compatibility Testing

(Cross Browser Testing)

- Browsers compatibility testing is a type of test used to check that **a web application works as expected in different browsers**



Example (cross browser issues)

The image shows a side-by-side comparison of a "SIGN UP" form as it appears in two different web browsers: Chrome and Firefox. Both versions of the form are set against a blue background and feature three input fields: "Your name", "Email Id", and "Password", followed by a green "Submit" button.

Chrome Browser: A callout bubble highlights the symmetrical layout of the form elements and the proper spacing between them. The "Submit" button is positioned at the bottom right of the form area.

Firefox Browser: A callout bubble highlights the lack of rounded corners on the input fields, which affects the overall visual presentation. The "Submit" button is also located at the bottom right of the form area.

Cross Browser Issue

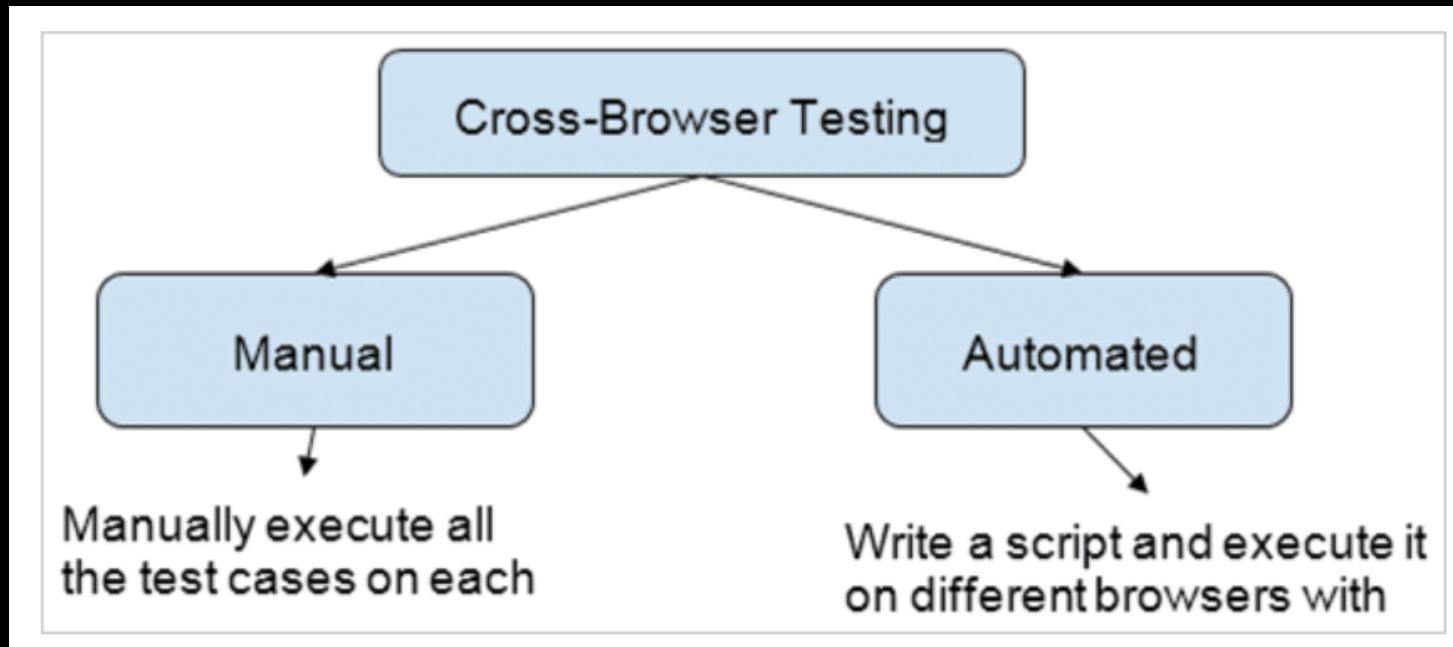
Text fields are positioned differently, and the corners of text boxes are different

The reasons for Cross Browser Issues

- **Different implementation:** each browser implements W3C standards, but not in the same way. Each browser treats HTML, CSS, and JavaScript differently
- **Vendor specific features:** each browser handles enhancements or deviations from W3C standards in their own way
- **Operating system:** the operating system on which the browser is executed can handle graphics and visual elements differently
- **Different screen resolutions:** a big number of variables that can impact not just the look of your website on the browser but how it behaves

Browsers are implemented by different vendors and browsers are executed in different platforms!

Manual vs. automated approaches



Two options to realize cross-browser testing



Huge number of combinations!

Cross browser testing is simple in principle but ...

How to do Cross Browser Testing?

HOWEVER THERE ARE LOT OF BROWSERS

- There are 5 major browsers - Chrome, Firefox, Safari, Edge, Internet Explorer
 - Each Browser has at least 10 versions
-
- There are 4 major operating systems - Windows, Mac, Android, iOS
 - Each operating system has at least 5 active versions

TOTAL = 2000+ Different Browser Environment



Different approaches for compatibility testing using Selenium

1. Using **Selenium WebDriver** (or Selenium IDE)
 - Possible only for few configurations (e.g., modern Chrome and Firefox)
2. Using **Selenium Grid**
 - Selenium Grid is a smart proxy server that makes possible **to run tests in parallel on multiple machines**
 - Not easy setup
3. Using specific **Cloud-based platforms**
 - E.g., **LambdaTest** or **SmartBear CrossBrowser Testing**
 - Selenium Grid is used ‘under the hood’ and the setup is easier
 - Paid services

Using Selenium WebDriver

- Cross-browser testing can be implemented using **JUnit 5 Parameterized tests**
- **Parameterized tests make it possible to run the same test script with different values multiple times**
- They are similar to traditional JUnit tests (@Test) but the annotation **@ParameterizedTest** must be used
- In addition, you have to add ‘a source’ that will provide the values for each execution
 - In our scenario, the **browser type** to be used



Parameterized Tests in JUnit5: an example

```
public class Numbers {  
    public static boolean isOdd(int number) {  
        return number % 2 != 0;  
    }  
}
```

‘Method that should return true if a number is odd’

% is the ‘remainder’ operator
i.e. Remainder of integer division

$4 \% 4 = 0$, 4 is even

$9 \% 4 = 1$, 9 is odd

```
class TestNumbers {  
    @ParameterizedTest  
    @ValueSource(ints = {1, 3, 5, -3, 15, Integer.MAX_VALUE}) // six numbers  
    void isOdd_ShouldReturnTrueForOddNumbers(int number) {  
        assertTrue(Numbers.isOdd(number));  
    }  
}
```

The method **isOdd_ShouldReturnTrue** is executed for each value in the ‘source’

Parameterized Tests in JUnit 5

https://www.youtube.com/watch?v=6Fqq_Lg5v_4

JUnit 5 compatibility testing

```
public enum Browser {  
    CHROME, FIREFOX, EDGE  
}
```

```
public class HelloWorldCrossBrowser {  
  
    private WebDriver driver;  
    private String baseUrl;  
  
    @ParameterizedTest  
    @EnumSource(Browser.class) // passing all 3 browsers  
    public void testA(Browser browser) throws InterruptedException {  
  
        driver = DriverFactory.getNewDriverInstance(browser);  
    }  
}
```

Returns a driver of type: Chromedriver, Firefoxdriver or Edgedriver

DriverFactory

```
public class DriverFactory {  
    public static WebDriver getNewDriverInstance(Browser browser) {  
        switch (browser) {  
            case CHROME:  
                return new ChromeDriver();  
            case FIREFOX:  
                return new FirefoxDriver();  
            case EDGE:  
                return new EdgeDriver();  
            default:  
                throw new IllegalArgumentException();  
        }  
    }  
}
```

```
public enum Browser {  
    CHROME, FIREFOX, EDGE  
}
```

The test script

```
package multibrowser;

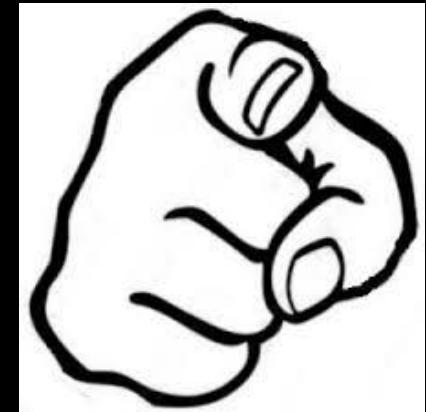
import static org.junit.jupiter.api.Assertions.assertEquals;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.EnumSource;
import org.openqa.selenium.WebDriver;

public class HelloWorldCrossBrowser {
    private WebDriver driver;
    private String baseUrl; Executed three times!

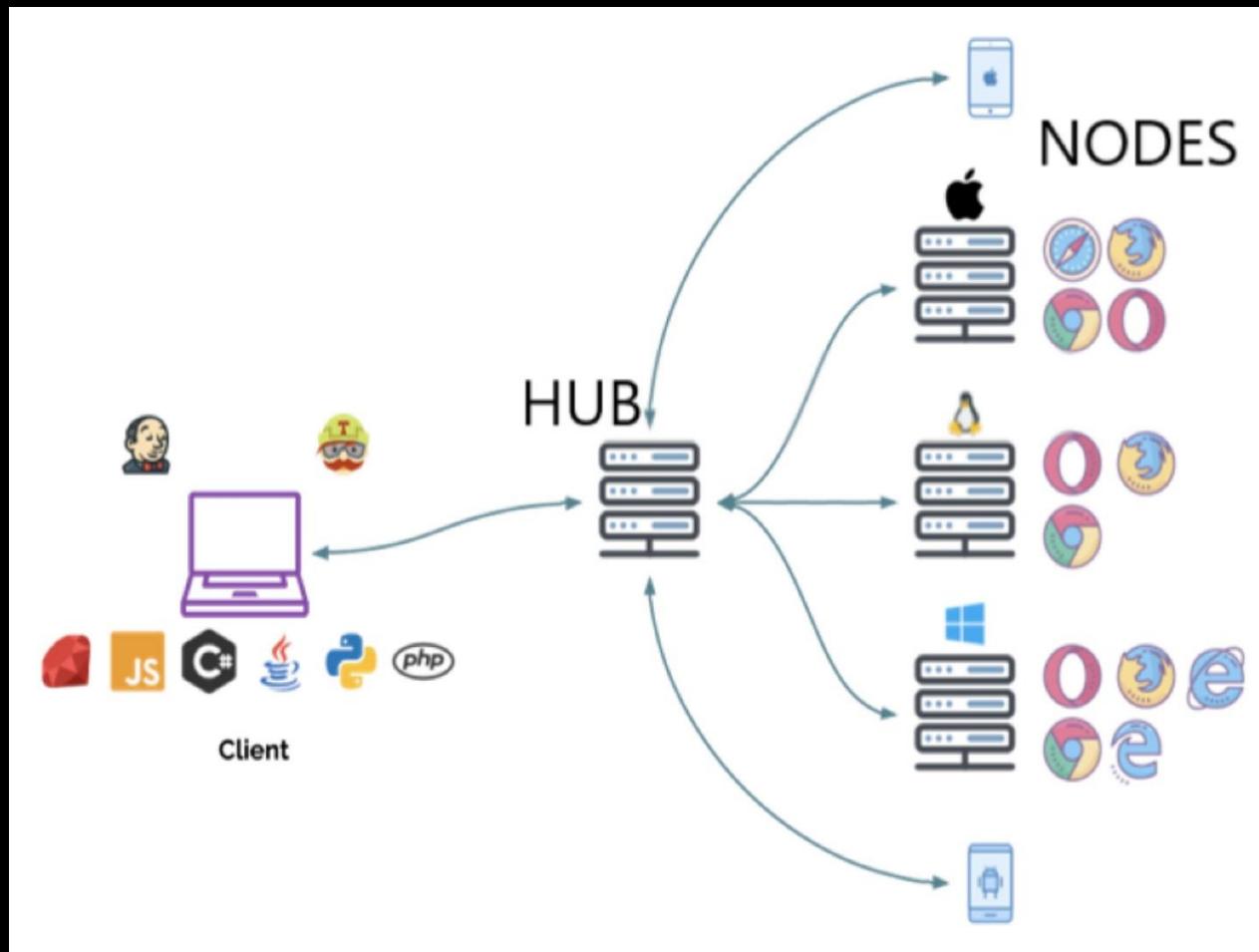
    @ParameterizedTest
    @EnumSource(Browser.class) // passing all 3 browsers
    public void testA(Browser browser) throws InterruptedException {
        driver = DriverFactory.getNewDriverInstance(browser);
        baseUrl = "https://bonigarcia.dev/selenium-webdriver-java/";
        // launch Browser and direct it to the Base URL
        driver.get(baseUrl);
        String expectedTitle = "Hands-On Selenium WebDriver with Java";
        assertEquals(expectedTitle, driver.getTitle());
    }
}
```

Now it's your turn!

Starting from the project about the Login Page with Page Objects (created previously), refactor it to test the Login page with different browsers (Firefox, Chrome and Edge)

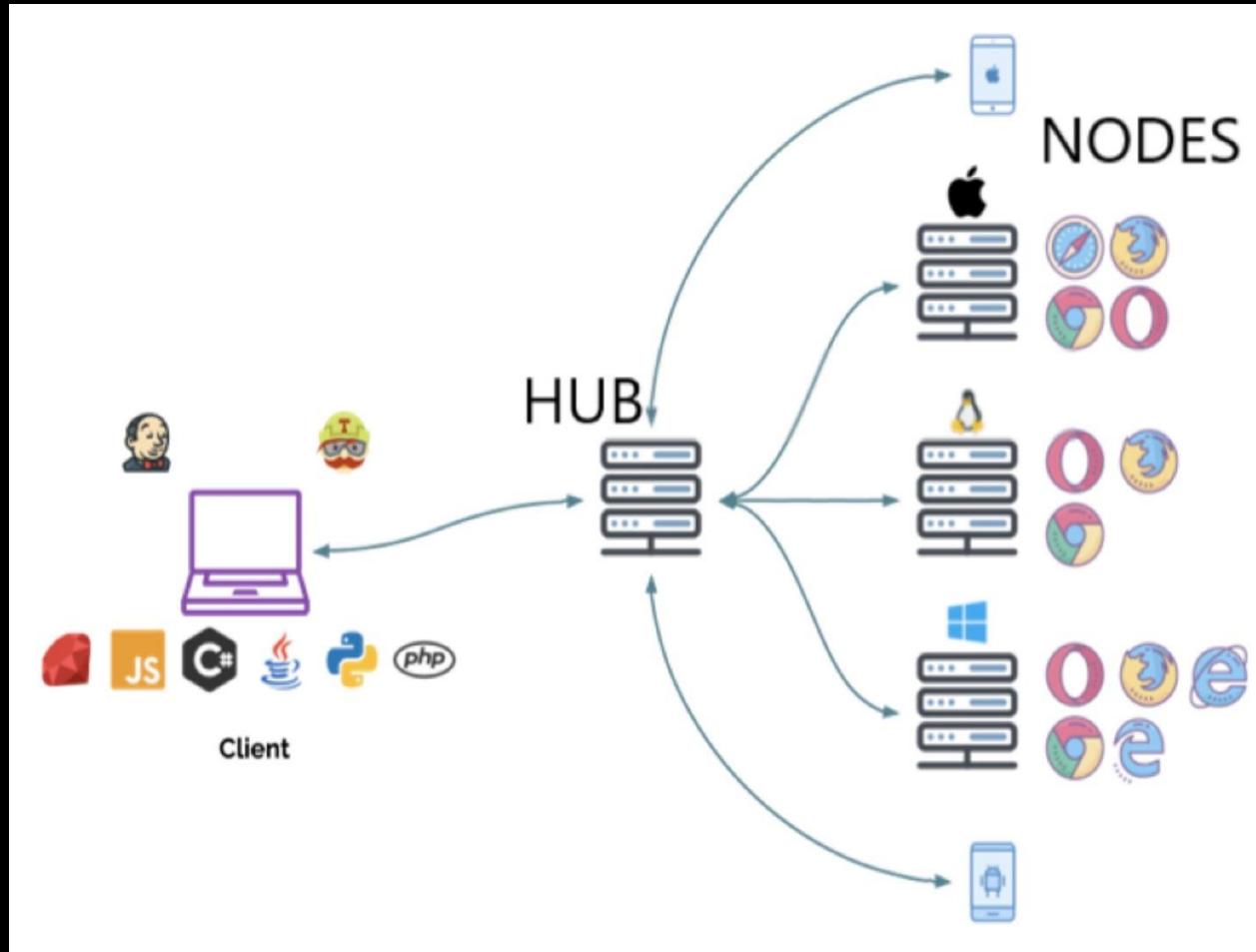


Using Selenium Grid (1)



Tests can be run in parallel on multiple machines called nodes. On each node it is possible to install different SO and browsers

Using Selenium Grid (2)



Setting up the platform to run tests on multiple browsers is quite complex and is outside the scope of this course!!!

Using Cloud-based testing platforms

- For example, LambdaTest



- **Cloud-based cross-browser testing platform**
 - It is possible to choose a wide range of Windows and Mac operating systems, along with all legacy and latest browsers
 - Web and Mobile platforms
 - Both manual and automated testing (Selenium)

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

LambdaTest

A screenshot of a web browser displaying the LambdaTest homepage. The URL 'lambdatest.com' is highlighted with a red box in the address bar. Below the address bar, a navigation bar includes links for Apps, Dell, CFT, Wiki, TOOLS, Related (3), Imported, Demo, and Topics to Teach, along with a 'Register Now >' button. The main content features the LambdaTest logo, a navigation menu with Live, Automation, Pricing, Resources, Support, and Log in options, and a prominent 'Start Free Testing' button. The central heading reads 'Cross Browser Testing Cloud'. A sub-headline states 'Perform Live Interactive and Automated Cross Browser Testing on 2000+ Real Browsers and Operating Systems Online'. A red box highlights the text '2000+ Real Browsers and Operating Systems Online'. At the bottom, there is a form for entering an email address with a 'Start Free Testing' button, and a note indicating '✓ No Credit Card Required', '✓ Free Sign Up', and 'Cancel Anytime'. A red arrow points from the bottom of the 'Free Sign Up' note towards the 'Start Free Testing' button. In the bottom right corner, there is a small video player interface with the text 'Sgancia questo video'.

Real time testing (platform selection)

The screenshot shows the LambdaTest platform selection interface. On the left, a sidebar contains icons for browser selection, proxy tunnel, and settings. The main area has a header with a URL input field containing "www.google.com" and a "START" button with a rocket icon. Below this are three columns: "VERSION" (listing Firefox 63 Beta, 62, 61, 60, 59), "OS" (listing Windows 10, 8.1, 8, 7, macOS Mojave, macOS High Sierra), and "RESOLUTION" (listing 1024 X 768, 1366 X 768, 1440 X 900, 1600 X 1200, 1920 X 1080, 2048 X 1536). The "1366 X 768" resolution is highlighted with a blue border.

Place your URL
www.google.com

Optional
Proxy Tunnel

START

VERSION

OS

RESOLUTION

1024 X 768

1366 X 768

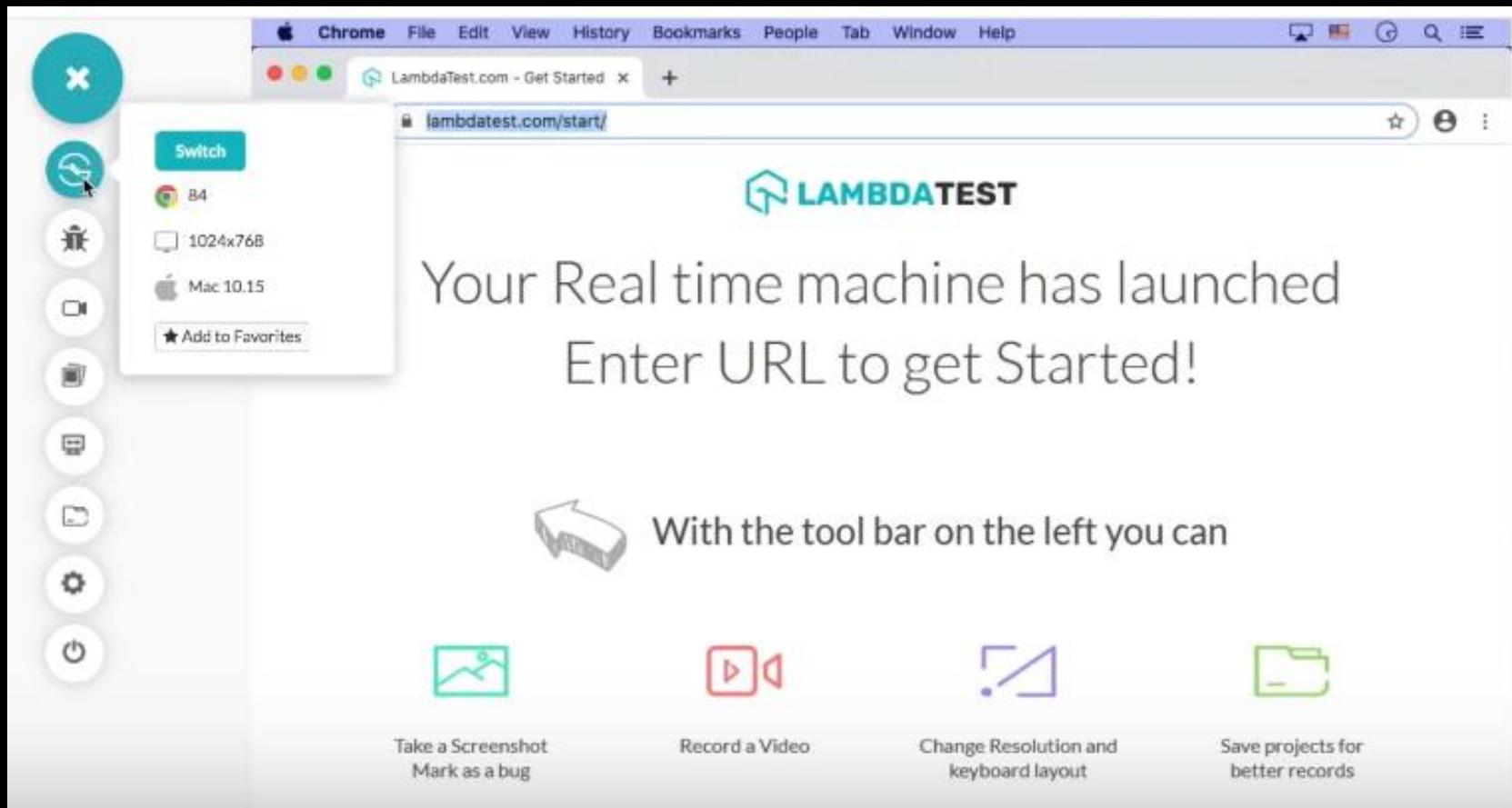
1440 X 900

1600 X 1200

1920 X 1080

2048 X 1536

Real time testing (execution)



Manual Testing can start with the chosen virtual machine!

LambdaTest Real time testing

<https://www.youtube.com/watch?v=hJ-eP8TcGuY>

Browser Screenshot Testing

The screenshot shows the LambdaTest dashboard interface. On the left, a sidebar menu is visible with various options like Dashboard, Real Time Testing, LT Browser, Automation, Visual UI Testing (which is expanded and highlighted with a red box), Test Logs, Issue Tracker, Integrations, and Projects. The main area is titled "Capture screenshots for multiple browsers & devices" and contains a URL input field with "https://lambda-test.github.io/sample-todo-app/" and a "CAPTURE" button. Below this, there's a summary section with browser icons and counts: 0/25 (Google Chrome), 0 (Mozilla Firefox), and 0 (Microsoft Edge). The main content area displays a grid of browser performance data for three operating systems: Windows 8.1, Windows 8, and Windows 7. Each row shows scores for Google Chrome, Mozilla Firefox, and Microsoft Edge across three metrics: CPU, Memory, and Disk.

| OS | Browser | CPU | Memory | Disk |
|-------------|-----------------|-----|--------|------|-----|--------|------|-----|--------|------|-----|--------|------|
| Windows 8.1 | Google Chrome | 88 | 87 | 86 | 85 | 84 | 83 | 74 | 73 | 72 | 88 | 87 | 86 |
| | Mozilla Firefox | 85 | 84 | 83 | 82 | 81 | 80 | 71 | 70 | 69 | 85 | 84 | 83 |
| | Microsoft Edge | 81 | 80 | 79 | 79 | 78 | 77 | 68 | 67 | 66 | 81 | 80 | 79 |
| Windows 8 | Google Chrome | 88 | 87 | 86 | 85 | 84 | 83 | 74 | 73 | 72 | 88 | 87 | 86 |
| | Mozilla Firefox | 85 | 84 | 83 | 82 | 81 | 80 | 71 | 70 | 69 | 85 | 84 | 83 |
| | Microsoft Edge | 81 | 80 | 79 | 79 | 78 | 77 | 68 | 67 | 66 | 81 | 80 | 79 |
| Windows 7 | Google Chrome | 88 | 87 | 86 | 85 | 84 | 83 | 74 | 73 | 72 | 88 | 87 | 86 |
| | Mozilla Firefox | 85 | 84 | 83 | 82 | 81 | 80 | 71 | 70 | 69 | 85 | 84 | 83 |
| | Microsoft Edge | 81 | 80 | 79 | 79 | 78 | 77 | 68 | 67 | 66 | 81 | 80 | 79 |

Screenshot comparison of the same webpage in different browsers!

Browsers selection

Capture screenshots for multiple browsers & devices

Place your URL
https://www.lambdatest.com/

Optional Select Tunnel

CAPTURE

25/25

List1 (0/25)

Basic Authentication

Login

Windows 10

| Browser | Resolution | OS | Device | Score |
|-------------|------------|-------------|---------|------------|
| Chrome | 1920x1080 | Windows 10 | Desktop | 97 |
| Chrome | 1920x1080 | Windows 10 | Desktop | 96 |
| Chrome | 1920x1080 | Windows 10 | Desktop | 95 |
| Firefox | 1920x1080 | Windows 10 | Desktop | 96 |
| Firefox | 1920x1080 | Windows 10 | Desktop | 95 |
| Firefox | 1920x1080 | Windows 10 | Desktop | 94 |
| IE | 1920x1080 | Windows 10 | Desktop | 83 |
| IE | 1920x1080 | Windows 10 | Desktop | 82 |
| IE | 1920x1080 | Windows 10 | Desktop | 81 |
| Edge | 1920x1080 | Windows 10 | Desktop | 97 |
| Edge | 1920x1080 | Windows 10 | Desktop | 96 |
| Edge | 1920x1080 | Windows 10 | Desktop | 95 |
| Opera | 1920x1080 | Windows 10 | Desktop | 94 |
| Opera | 1920x1080 | Windows 10 | Desktop | 93 |
| Opera | 1920x1080 | Windows 10 | Desktop | 92 |
| Safari | 1920x1080 | Windows 10 | Desktop | 91 |
| Safari | 1920x1080 | Windows 10 | Desktop | 90 |
| Safari | 1920x1080 | Windows 10 | Desktop | 89 |
| Android | 1920x1080 | Windows 10 | Mobile | 90 |
| Android | 1920x1080 | Windows 10 | Mobile | 89 |
| Android | 1920x1080 | Windows 10 | Mobile | 88 |
| iPhone | 1920x1080 | Windows 10 | Mobile | 90 |
| iPhone | 1920x1080 | Windows 10 | Mobile | 89 |
| iPhone | 1920x1080 | Windows 10 | Mobile | 88 |
| Tablet | 1920x1080 | Windows 10 | Mobile | 77 |
| Tablet | 1920x1080 | Windows 10 | Mobile | 76 |
| Tablet | 1920x1080 | Windows 10 | Mobile | 75 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 97 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 96 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 95 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 94 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 93 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 92 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 91 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 90 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Desktop | 89 |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Mobile | 61 more... |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Mobile | 72 more... |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Mobile | 52 more... |
| Windows 8.1 | 1920x1080 | Windows 8.1 | Mobile | 9 more... |

Windows 8.1

| Browser | Resolution | OS | Device | Score |
|---------|------------|-------------|---------|-------|
| Chrome | 1920x1080 | Windows 8.1 | Desktop | 97 |
| Chrome | 1920x1080 | Windows 8.1 | Desktop | 96 |
| Chrome | 1920x1080 | Windows 8.1 | Desktop | 95 |
| Firefox | 1920x1080 | Windows 8.1 | Desktop | 96 |
| Firefox | 1920x1080 | Windows 8.1 | Desktop | 95 |
| Firefox | 1920x1080 | Windows 8.1 | Desktop | 94 |
| IE | 1920x1080 | Windows 8.1 | Desktop | 83 |
| IE | 1920x1080 | Windows 8.1 | Desktop | 82 |
| IE | 1920x1080 | Windows 8.1 | Desktop | 81 |
| Edge | 1920x1080 | Windows 8.1 | Desktop | 97 |
| Edge | 1920x1080 | Windows 8.1 | Desktop | 96 |
| Edge | 1920x1080 | Windows 8.1 | Desktop | 95 |
| Opera | 1920x1080 | Windows 8.1 | Desktop | 94 |
| Opera | 1920x1080 | Windows 8.1 | Desktop | 93 |
| Opera | 1920x1080 | Windows 8.1 | Desktop | 92 |
| Safari | 1920x1080 | Windows 8.1 | Desktop | 91 |
| Safari | 1920x1080 | Windows 8.1 | Desktop | 90 |
| Safari | 1920x1080 | Windows 8.1 | Desktop | 89 |
| Android | 1920x1080 | Windows 8.1 | Mobile | 90 |
| Android | 1920x1080 | Windows 8.1 | Mobile | 89 |
| Android | 1920x1080 | Windows 8.1 | Mobile | 88 |
| iPhone | 1920x1080 | Windows 8.1 | Mobile | 90 |
| iPhone | 1920x1080 | Windows 8.1 | Mobile | 89 |
| iPhone | 1920x1080 | Windows 8.1 | Mobile | 88 |
| Tablet | 1920x1080 | Windows 8.1 | Mobile | 77 |
| Tablet | 1920x1080 | Windows 8.1 | Mobile | 76 |
| Tablet | 1920x1080 | Windows 8.1 | Mobile | 75 |

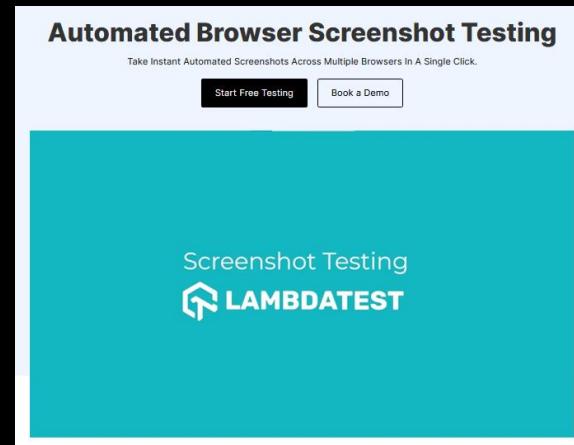
Download, Share and Analyze Screenshots

The screenshot displays a user interface for managing and sharing screenshots from cross-browser testing. At the top, there's a header with a logo, a 'Configure Tunnel' button, an 'Upgrade' button, and a user profile icon. Below the header, the 'Testing URL' is set to <https://www.lambdatest.com/>. The 'Project' is listed as 'Untitled' and the 'Version' is '515'. A progress bar shows '25/25 Screenshots Generated' with '25 Desktop 0 Mobile/Tablet'. Below this, there are two rows of five screenshots each, representing different browser configurations. Each screenshot preview includes the browser icon, the test environment (e.g., 'Cross Browser Testing Cloud'), and the operating system/browser combination (e.g., 'Windows 10 Firefox 96'). A red box highlights the 'Share Screenshots' button in the toolbar above the screenshots. On the far right, there's a blue circular button with a white speech bubble icon.

| Screenshot Preview | Environment | OS / Browser |
|--------------------|-----------------------------|---------------------------|
| | Cross Browser Testing Cloud | Windows 10 Firefox 96 |
| | Cross Browser Testing Cloud | Windows 10 Firefox 95 |
| | Cross Browser Testing Cloud | Windows 10 Firefox 92 |
| | Cross Browser Testing Cloud | Windows 8.1 Firefox 95 |
| | Cross Browser Testing Cloud | Windows 8.1 Firefox 96 |
| | Cross Browser Testing Cloud | Windows 10 Firefox 96 |
| | Cross Browser Testing Cloud | Windows 10 Firefox 95 |
| | Cross Browser Testing Cloud | Windows 10 Firefox 92 |
| | Cross Browser Testing Cloud | Windows 8.1 Firefox 95 |
| | Cross Browser Testing Cloud | Windows 8.1 Firefox 96 |

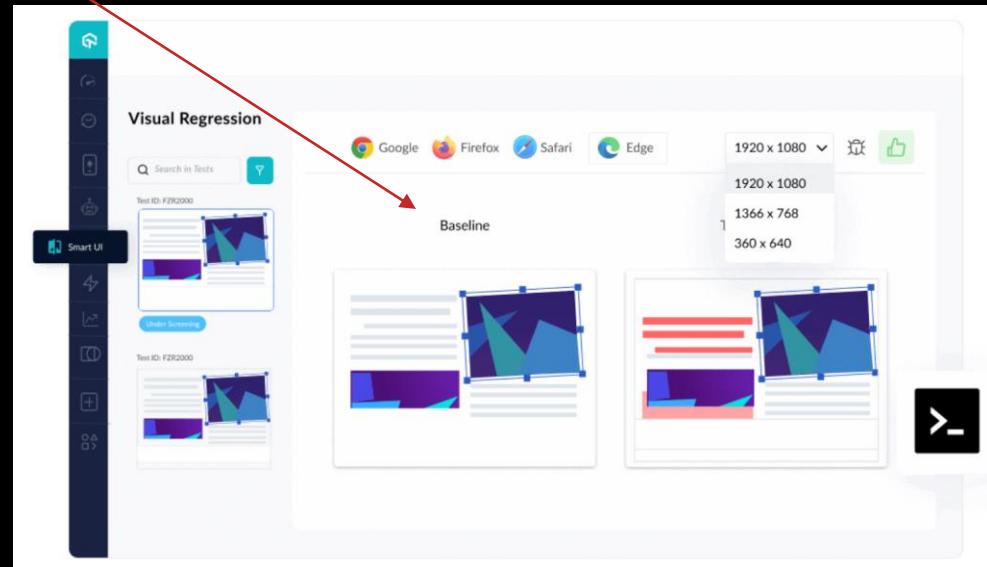
LambdaTest Screenshot testing

<https://www.lambdatest.com/automated-screenshot>



‘Smart’ Visual UI Testing

- A baseline (**gold standard**) is created for a web page
- Subsequent versions of the web page are then visually compared against this baseline
 - Differences are highlighted
- This approach is particularly useful for:
 - Regression Testing
 - Browser Compatibility Testing



Login Form example

Test: Login Form PAge

VIEW HIGHLIGHT DIFFS ANNOTATIONS AUTO MAINTENANCE ACCESSIBILITY Scope: Default

Baseline | 1/1 Strict | Linux | Chrome 90.0 | 1000x660 | Desktop

The screenshot shows a login form with a red highlight difference on the left side of the page, indicating a change between the baseline and checkpoint versions.

Checkpoint | 1/1 Strict | Linux | Chrome 90.0 | 1000x660 | Desktop

The screenshot shows a login form with a green highlight difference on the right side of the page, indicating a change between the baseline and checkpoint versions.

Log In Form

Username Enter your username

Password Enter your password

Remember Me

[Twitter](#) [Facebook](#) [LinkedIn](#)

Log In Form

Username Enter your username

Password Enter your password

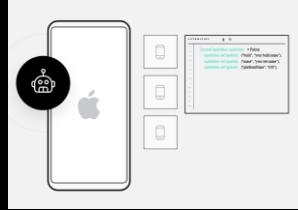
Remember Me

[Twitter](#) [Facebook](#) [LinkedIn](#)

LambdaTest Smart Visual UI testing

<https://www.youtube.com/watch?v=sOaQ-ttEBzs>

Selenium Test Automation



- It is possible to **run Selenium test automation scripts** across 2000+ online desktop and mobile environments for browser compatibility testing

The screenshot displays the LambdaTest platform interface. On the left, a sidebar menu includes 'Automation', 'Web Automation' (which is selected), and 'App Automation'. The main area shows a test run titled 'JUnitSampleTest' under 'JUnitSampleTest'. The status bar indicates 'Passed' with a green checkmark, '124.0' devices, '10' parallel tests, '1920x1080' resolution, '6s' duration, and 'OFF' for notifications. Below this, a 'All Commands' tab is active, showing a list of 14 commands with their execution times and details:

- Element (id: f.280998F503A3D2CAD64AC25220..5CECE2BBA5F41DED541C7FB5.e.3)
- Element (css selector: #addbutton)
- Clicks on (id: f.280998F503A3D2CAD64AC25220..5CECE2BBA5F41DED541C7FB5.e.12)
- Element (xpath: /html/body/div/div/div/ul/li[6]/span)
- Element (css selector: .list-item)

A search bar at the top of the command list allows filtering by command name. To the right, a large preview window shows a browser screenshot of a sample application with a list of items and a progress bar indicating the test's execution status.

- Follow LambdaTest tutorial
 - ‘Run Your First test on LambdaTest’
 - <https://www.lambdatest.com/support/docs/junit-with-selenium-running-junit-automation-scripts-on-lambdatest-selenium-grid/>
- Modify the project to test the slow Calculator implemented by Boni Garcia
 - <https://bonigarcia.dev/selenium-webdriver-java/slow-calculator.html>
- If you want, you can deliver your project on AW (assignment not mandatory)



LambdaTest Test Automation

<https://www.youtube.com/watch?v=dfnVAAw6PV4>



Last lesson: final balance

- **Basics of Web applications**
- **Functional testing of Web apps**
 - Manual vs automated testing
 - Software testing fundamentals (approaches and techniques)
 - Unit testing tools (e.g., **JUnit/TestNG**)
 - Automated testing approaches and tools for Web apps
 - Ex. **Selenium WebDriver** (used also in the Security testing part)
- **Continuous Integration** (a little) *Only theory. I will provide you a tutorial*
 - Tool: **Jenkins** and/or **GitHub actions**
- **API Testing** (if there is time)
 - Tool: **Postman**
- **Mobile Testing** (if there is time)
 - Testing framework: **Appium**