









LEAN TEST FRAMEWORK FOR WEB TESTING

ELIAS NOGUEIRA

@eliasnogueira

THE STACK

LANGUAGE	TESTING	DATA GENERATION	LOG & REPORTS	INFRA
	 	 javafaker	 Allure 	 docker  ZALENIUM

BASIC ITEMS FOR A TEST ARCHITECTURE

with focus on web automation



BASE ARCHITECTURE

- Clean Architecture
- Design Pattern
- Testing Patterns



DATA GENERATION

- Fake
- Static creation
- Dynamic creation



PAGE OBJECT MODEL

- Page Objects
- Page Factory
- Abstraction
- Waiting Strategy



PARALLEL EXECUTION

- Infrastructure
- Containers



LOGS AND REPORTS

- Exception logs
- General reports
- Evidence

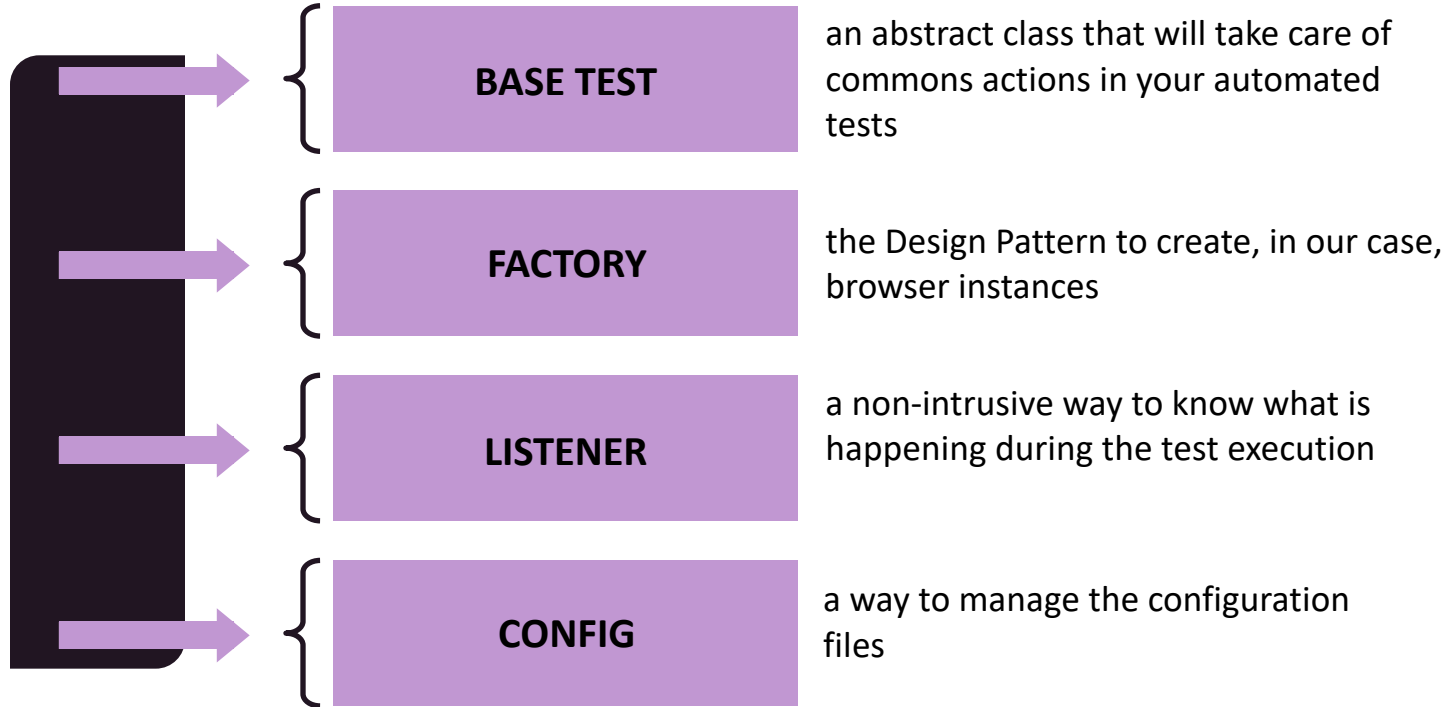


PIPELINE

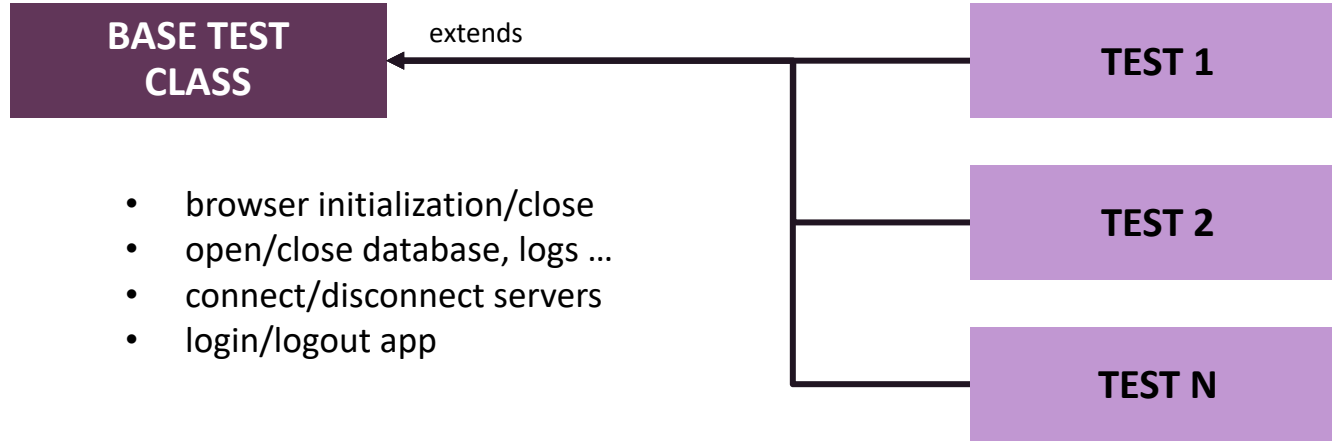
- Execution strategy

BASE ARCHITECTURE

to apply DRY and KISS



BASE TEST CLASS



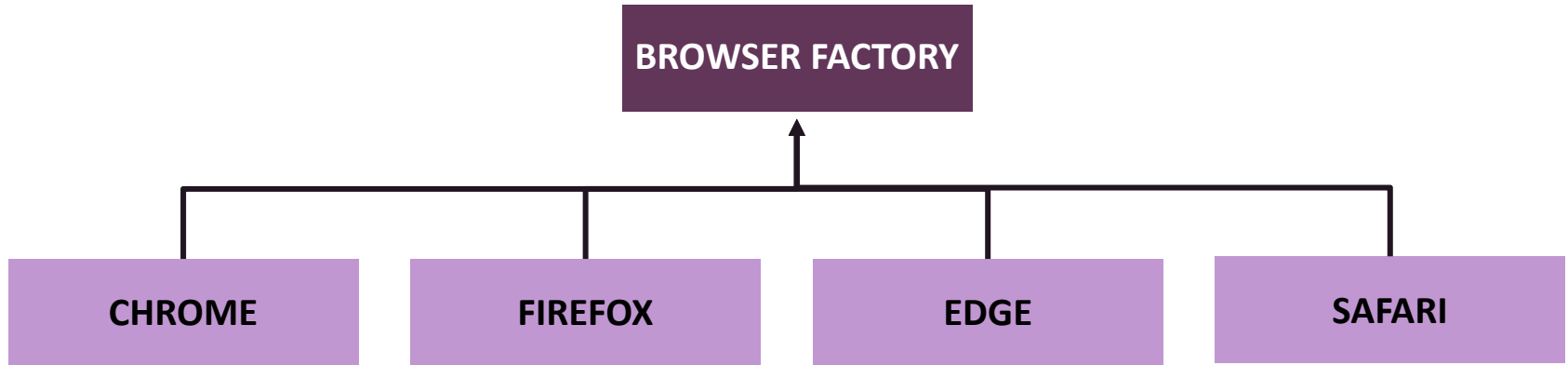
Smart use of inheritance

- test inherits common test actions

One test case per class

- provide an easy way to add more tests
- ease division of tests in suites

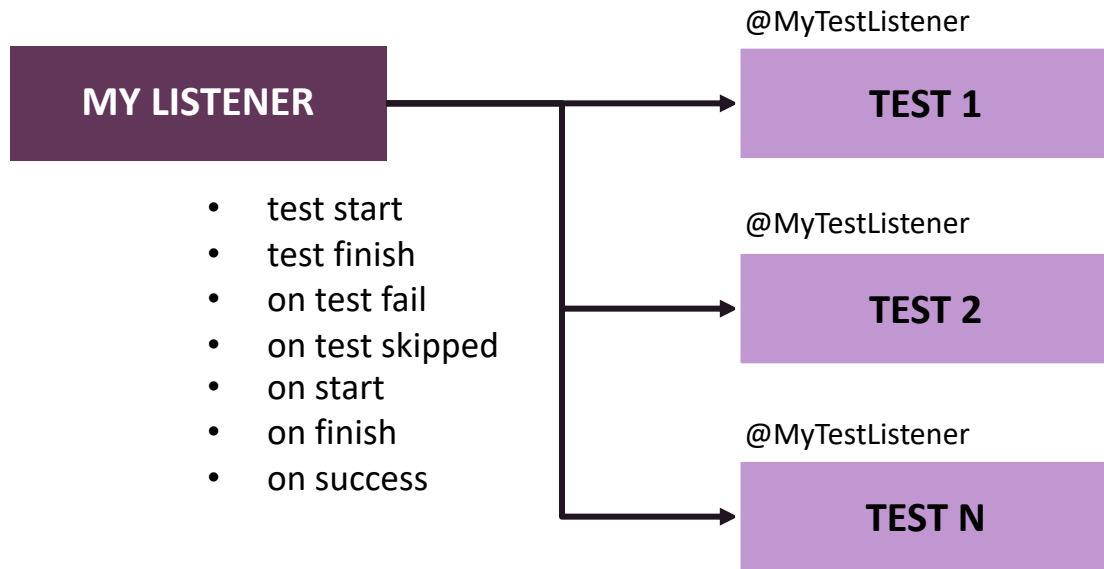
FACTORY CLASS



Apply Factory Design Pattern will help us to create a browser instance and make easy the parallel execution in many environments.

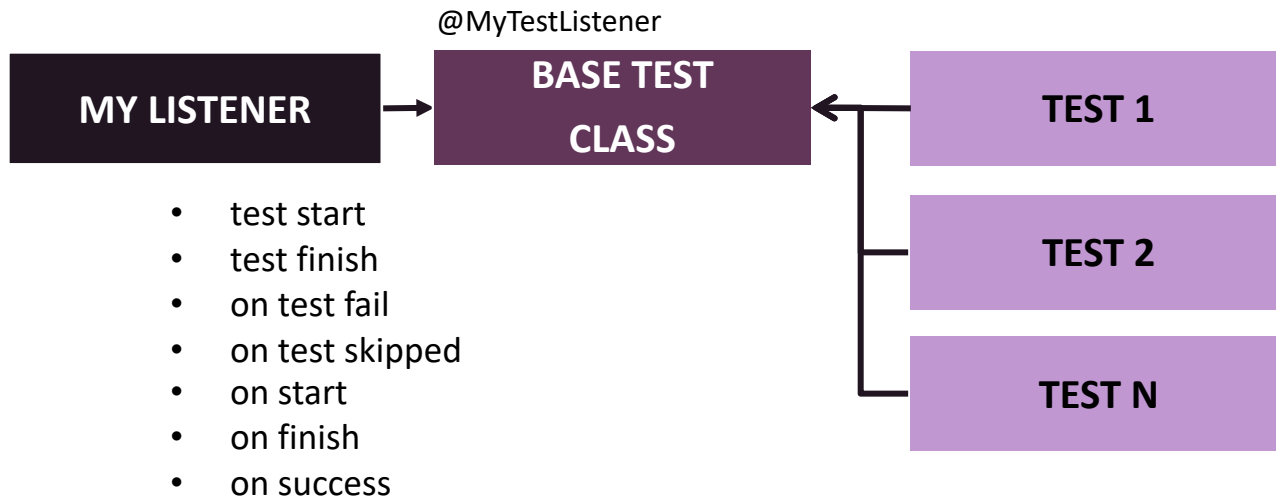
LISTENERS

Using TestNG we can use some listeners that allow modifying (or just watch) the test behaviors. Helpful o watch the test lifecycle and do something.



LISTENERS

But the best approach is to add the listener to the Base Test class, so all the tests that extend this class will have the listener.



CONFIGURATION MANAGEMENT

There are many ways to create a configuration management with different configuration file types.

We must try to avoid:

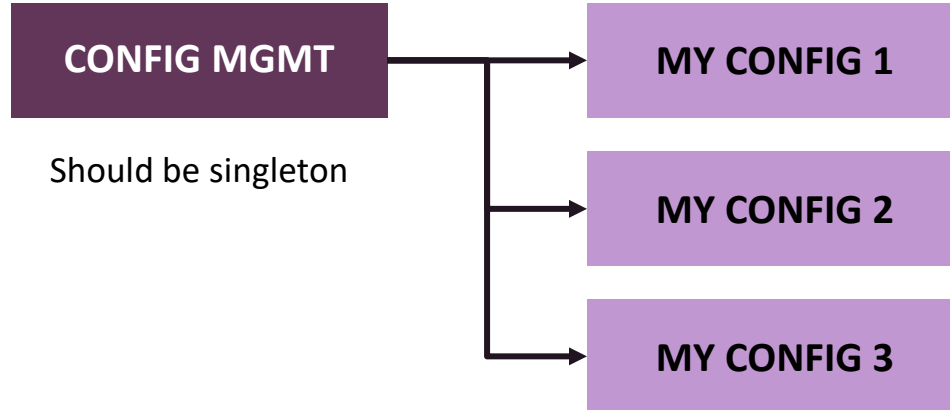
- Call multiple times the file (should be a singleton)
- Provide a way to load multiples files based on your requirements

CONFIGURATION MANAGEMENT

There are many ways to create a configuration management with different configuration file types.

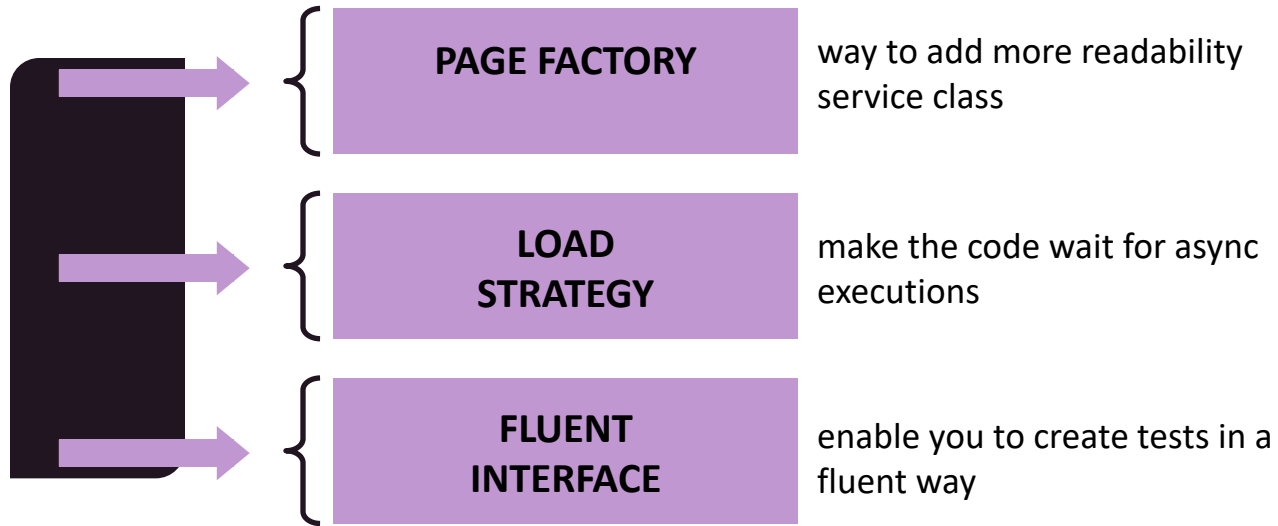
We must try to avoid:

- Call multiple times the file (should be a singleton)
- Provide a way to load multiples files based on your requirements



PAGE OBJECTS MODEL

more maintainability and readability

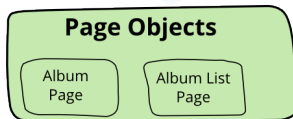


PAGE OBJECTS

Page Object is a class that serves as an interface to a page of your web page. The class provides methods to do page actions. Tests will use these methods.

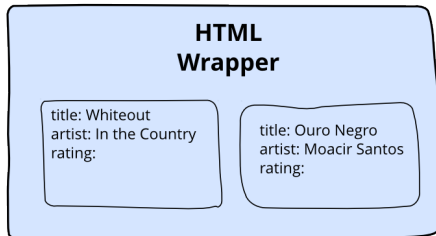
this API is about the application

`selectAlbumWithTitle()
getArtist()
updateRating(5)`



this API is about HTML

`findElementsWithClass('album')
findElementsWithClass('title-field')
getText()
click()
findElementsWithClass('ratings-field')
setText(5)`



PAGE OBJECTS

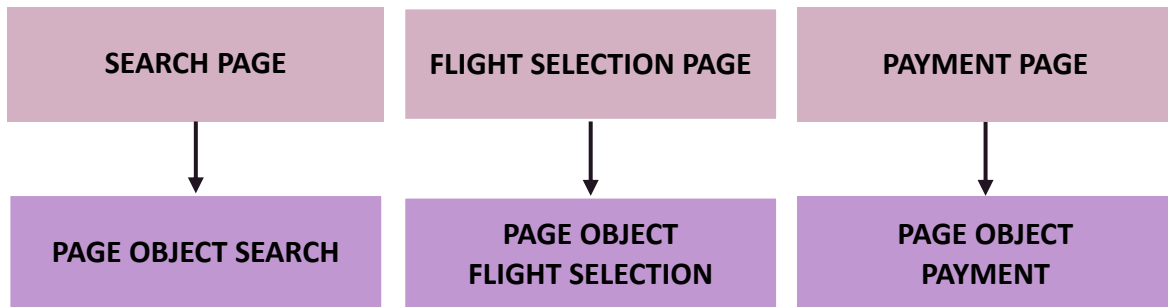
SEARCH PAGE

FLIGHT SELECTION PAGE

PAYMENT PAGE

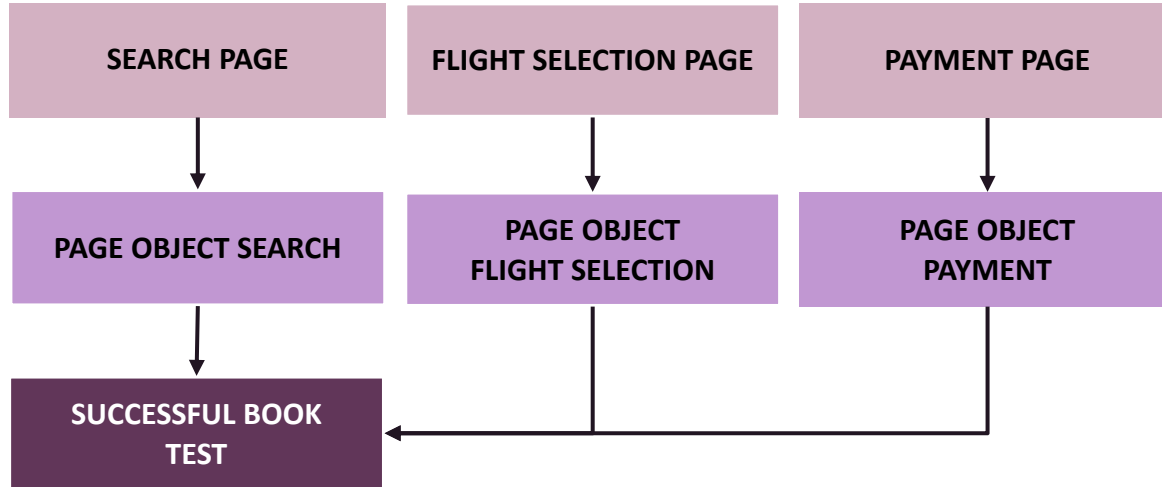
Let's say we have to test an application that sells flight tickets.
We would have those 3 pages.

PAGE OBJECTS



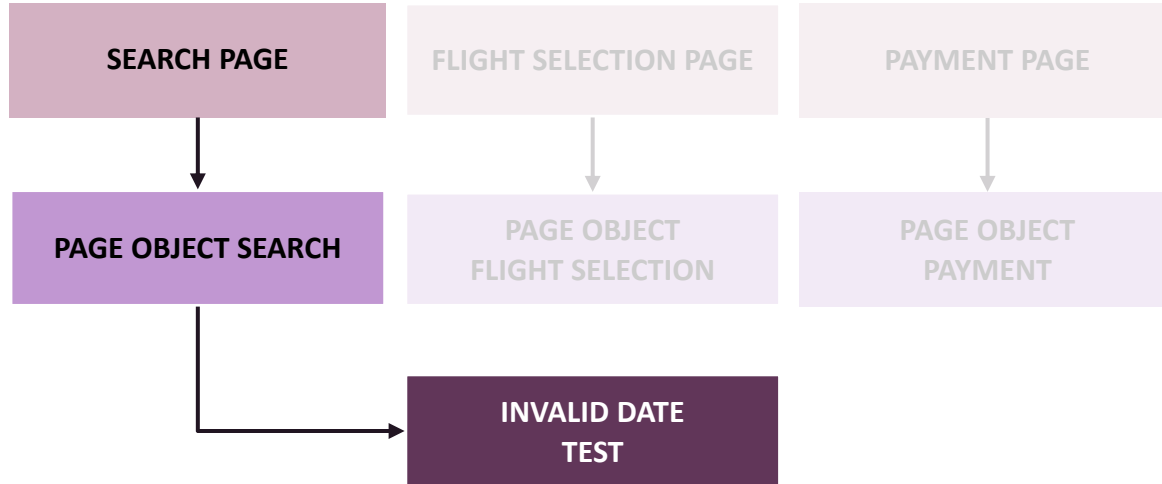
We will create a class for each page and add all the interactions we want to do with it. It's called a Page Object.

PAGE OBJECTS



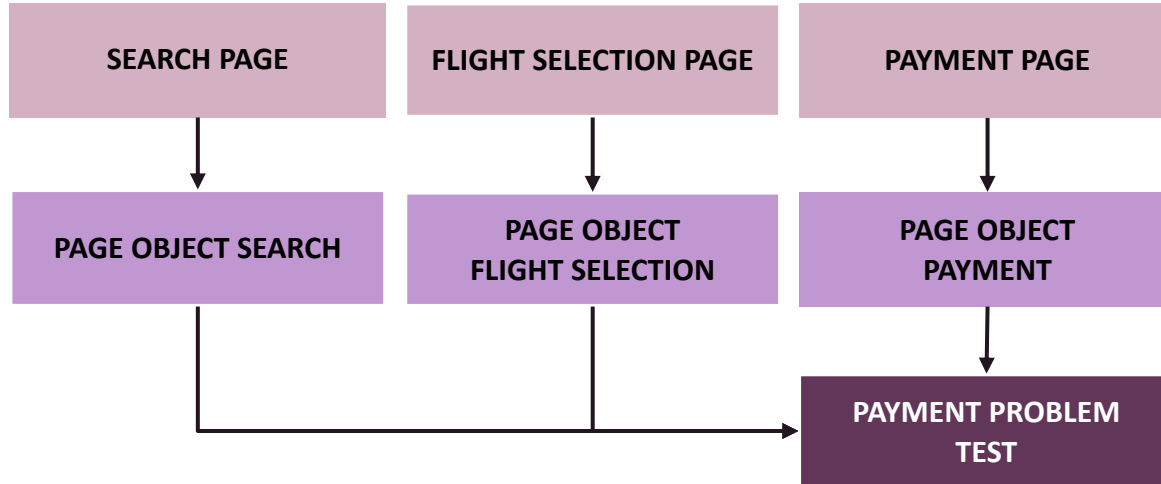
For a successful test, we would consume those 3 Page Objects.

PAGE OBJECTS



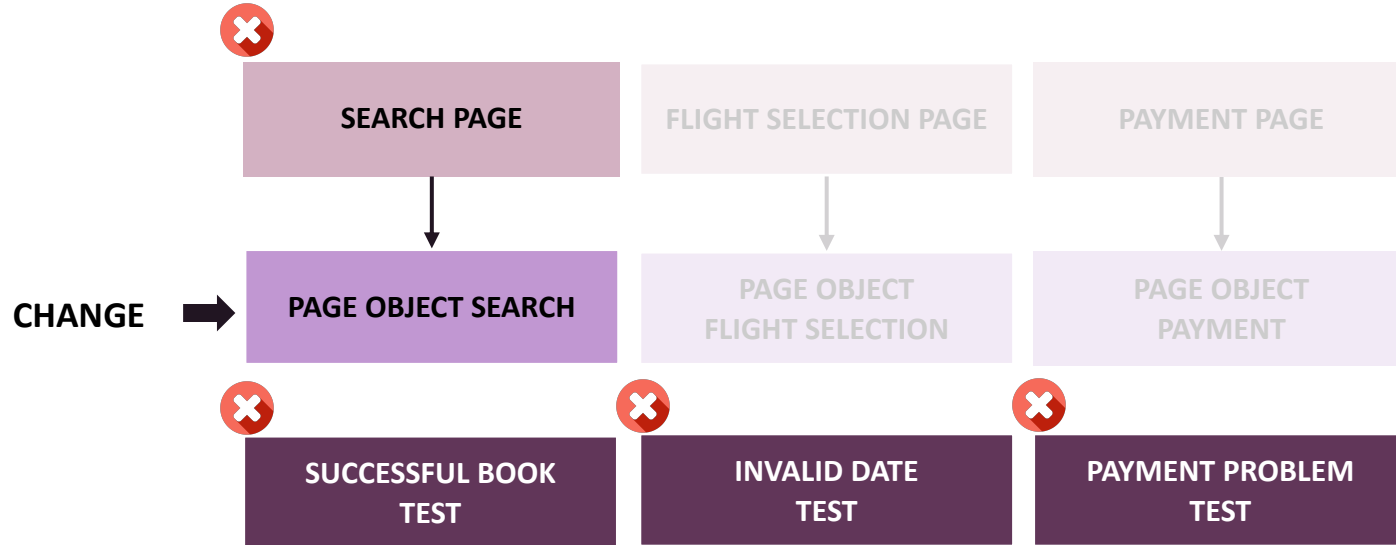
For an invalid date test, we would consume only one Page Object.

PAGE OBJECTS



For a payment problem simulation test,
we would use those 3 Page Objects.

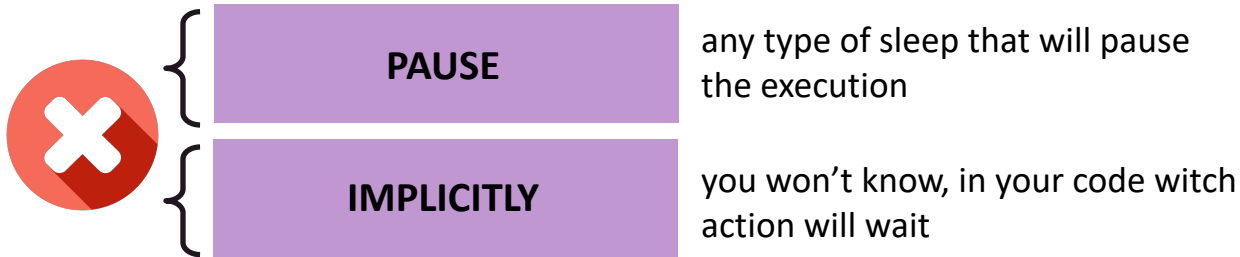
PAGE OBJECTS



If the Frontend developer do some changes in the Search Page that affects the tests, we just need to change the Page Object for that page, not the tests.

LOAD STRATEGY

A Load Strategy is responsible for wait for a certain time by any event on the web page, most of the time related to async requests (Ajax).



with this strategy, you can see in the code which element will take time

the best choice to use with Page Factory strategy



FLUENT INTERFACE

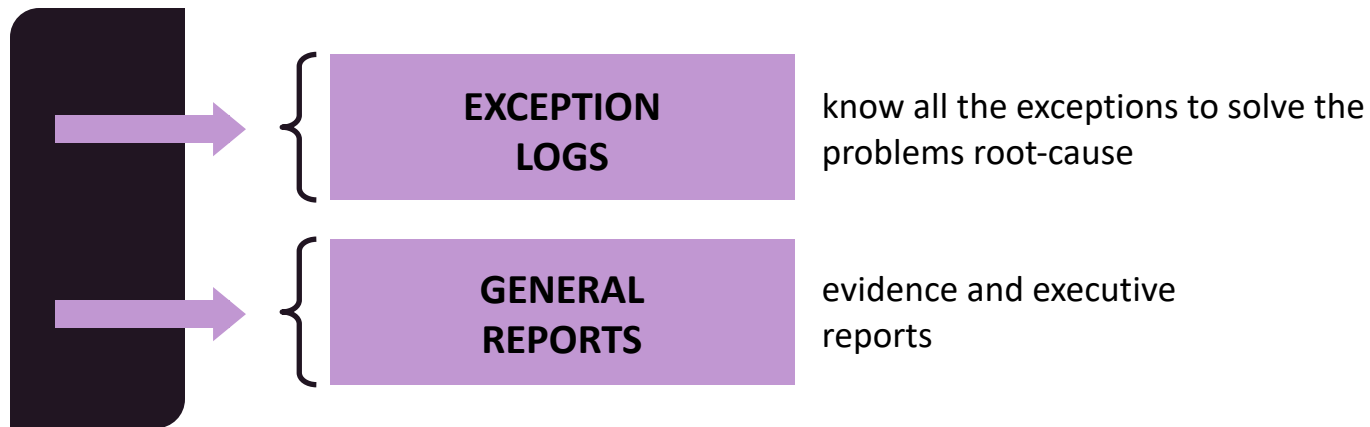
Creates a method chaining to perform a series of actions to make the code more readable and easier to use.

```
@Test
public void testWithoutFluentInterface() {
    GeneralMenuPage menu = new GeneralMenuPage();
    menu.clickinExperience();
    menu.clickInOurFleet();
    menu.clickInSeatingCharts();
}

@Test
public void testWithFluentInterface() {
    GeneralMenuPage menu = new GeneralMenuPage();
    menu.clickinExperience().clickInOurFleet().clickInSeatingCharts();
}
```

LOGS AND REPORTS

because we need to know about any error



EXCEPTION LOGS

By using any log strategy, saving a log file, we can understand the common errors that occurred during the test execution.

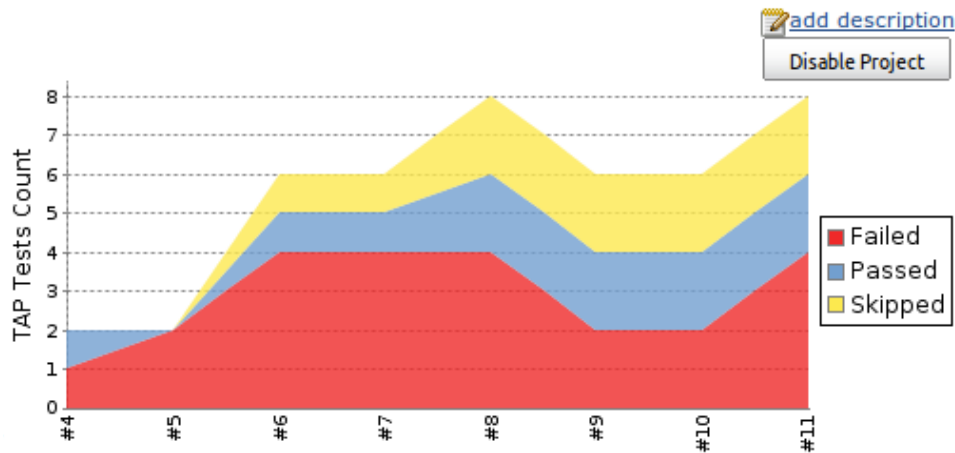
These errors can be of:

- assertion errors
- timeout exceptions
- locator exception
- an exception on your architecture

If you want to analyze test errors across teams a good way is using *Elasticsearch* with *Grafana/Kibana* or using *Report Portal*.

GENERAL REPORTS

Generate xUnit reports to attach on your CI/CD and, rapidly, see the test status.

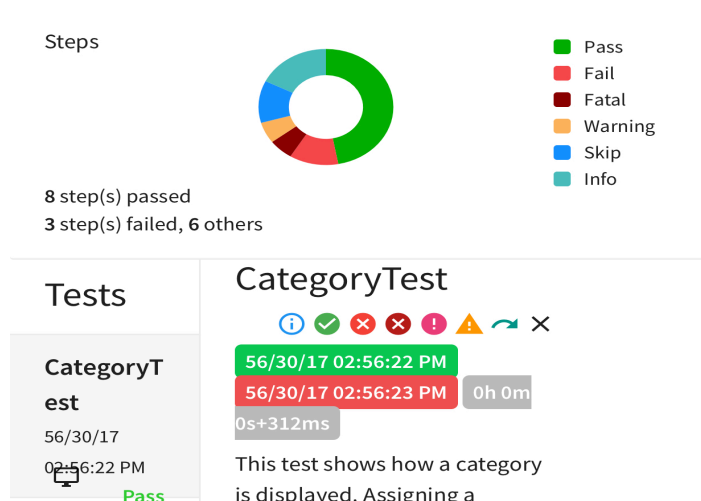


[2 hr ago](#)

GENERAL REPORTS

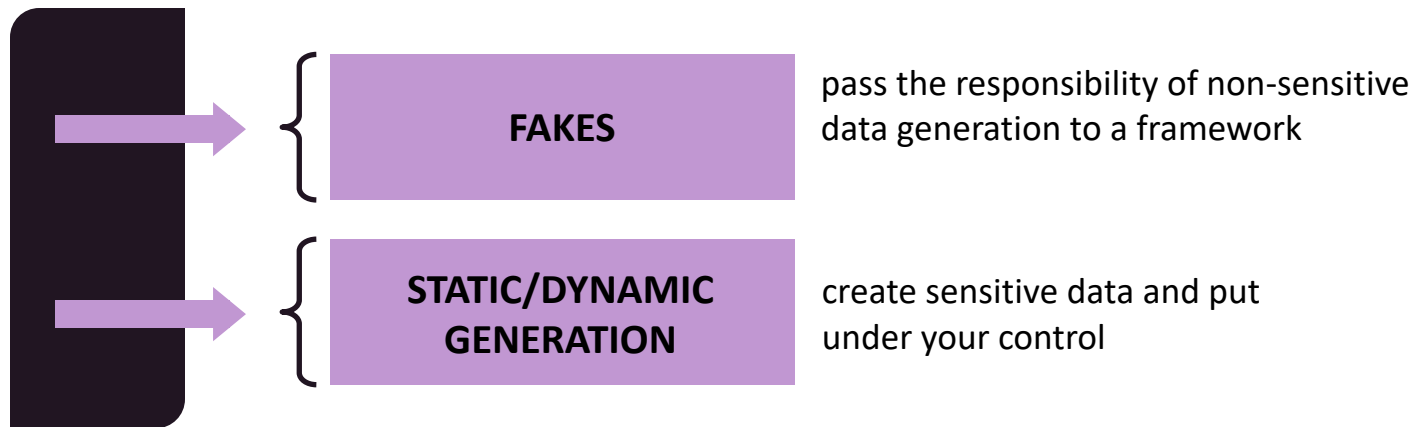
Create an executive report to provide information and evidence about the test execution.

This report may contain screenshots when an error occurs to help to analyze the root cause of a problem.



DATA GENERATION

solve one of the biggest problems



FAKE GENERATION

Ability to create an approach to generate non-sensitive data for your test without the necessity to manually change the test data in each execution.

There're a lot of tools to create this type of data.

Example with *javafaker*

```
Faker faker = new Faker(new Locale("pt-BR"));
```

```
faker.name().fullName();  
faker.address().fullAddress();  
faker.internet().emailAddress();  
faker.business().creditCardNumber();  
faker.date().birthday();
```

STATIC / DYNAMIC GENERATION

When the data cause different behaviors in your application.

A **Static** approach can be implemented with any kind of solution, like:

- Files
 - CSV | JSON | TXT | YML
- Database
- Mock

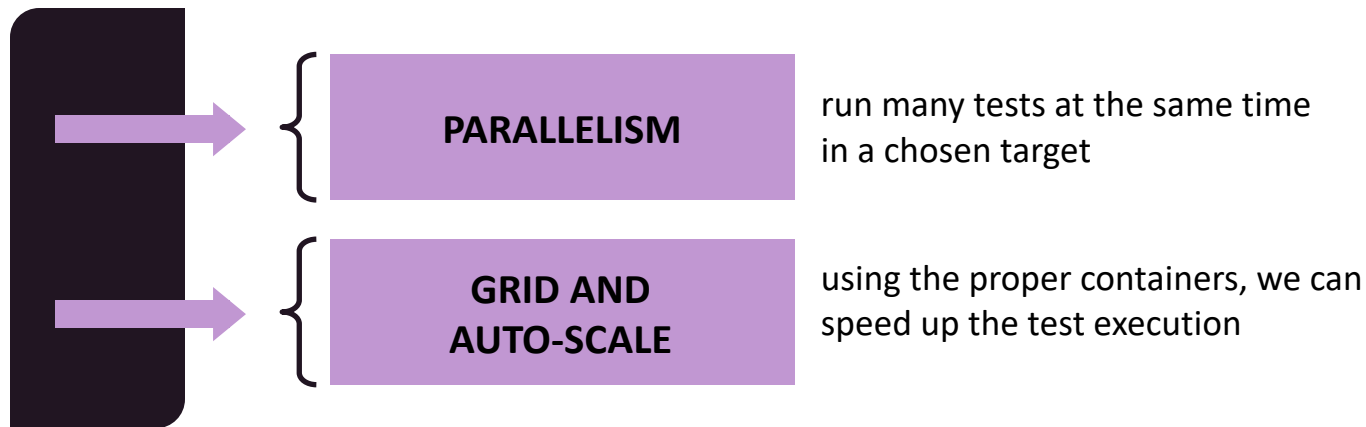
A **Dynamic** approach can be created according to your context.

Used to remove the maintenance of test data

- Queries in a database
- Consume data from a static poll

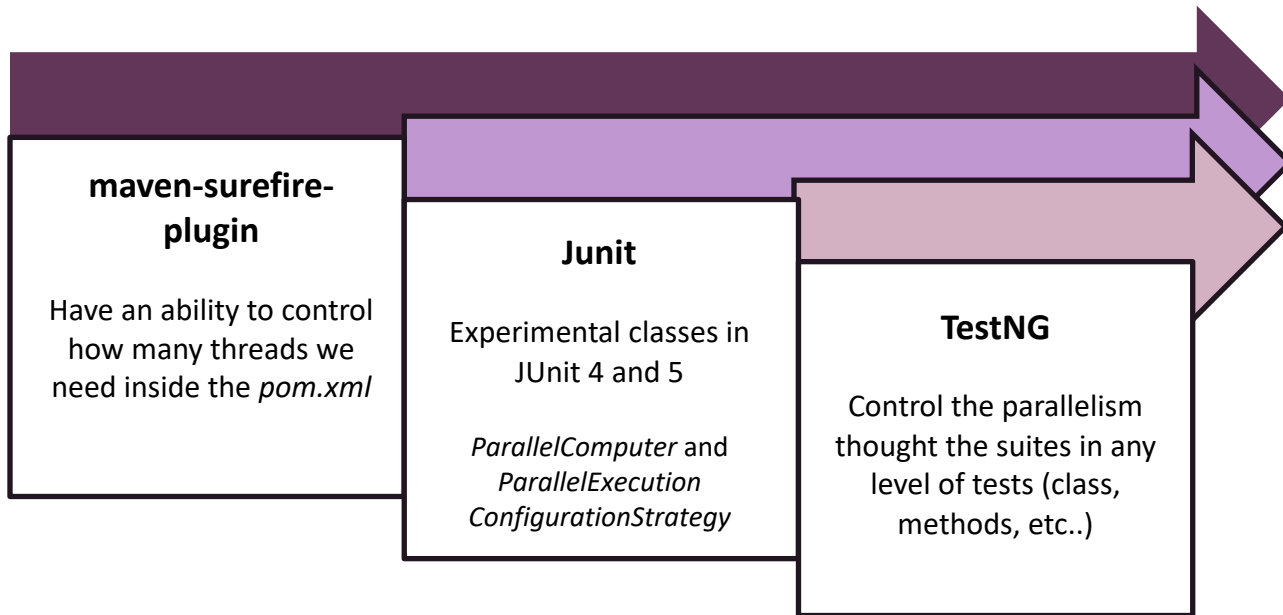
PARALLEL EXECUTION

to speed up your test execution

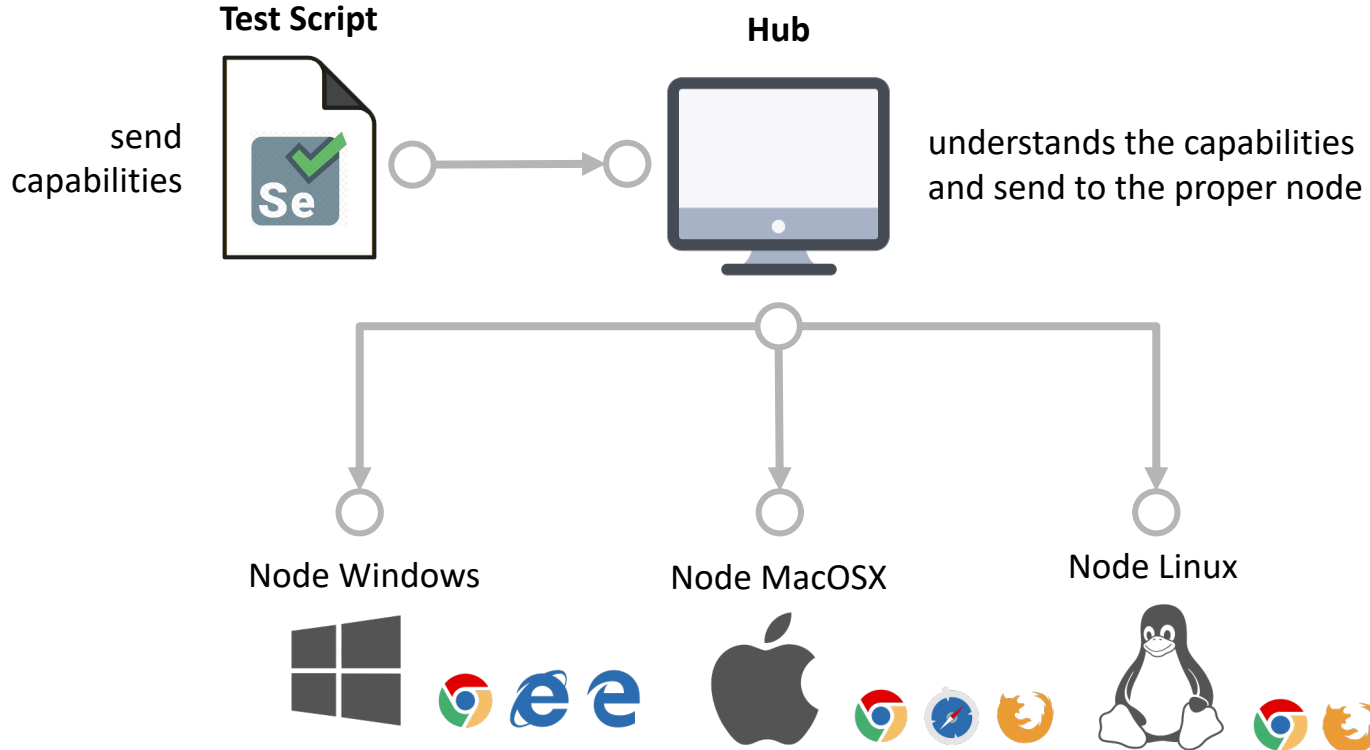


PARALLELISM

Parallelism, under test, is the ability to perform the same test in different conditions (browser, devices, etc...) or different tests at the same time.



GRID SCHEMA



WAYS TO CREATE A GRID



LOCAL

Uses machines inside an infrastructure.

Can be a bare-metal desktop or a virtual machine



CONTAINERS

Uses containers (locally or cloud-based) to create the infrastructure and support orchestration



CLOUD

Uses a cloud infrastructure platform to create virtual machines

CONTAINERS TO AUTO-SCALE

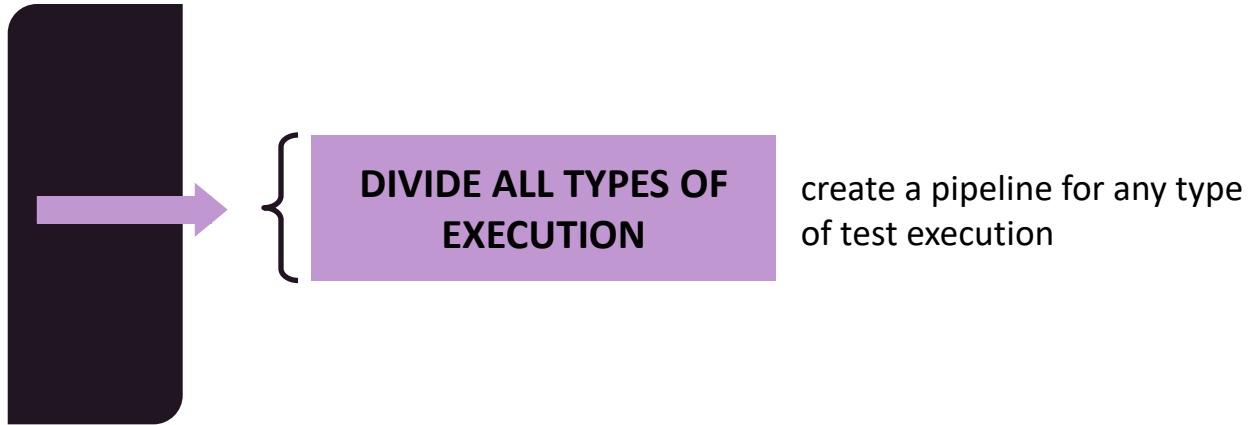


- Uses a custom container *elgalu/selenium* that provides:
 - live Preview with VNC
 - video recording
 - dashboard
- automatic auto-scale containers based on the number of tests

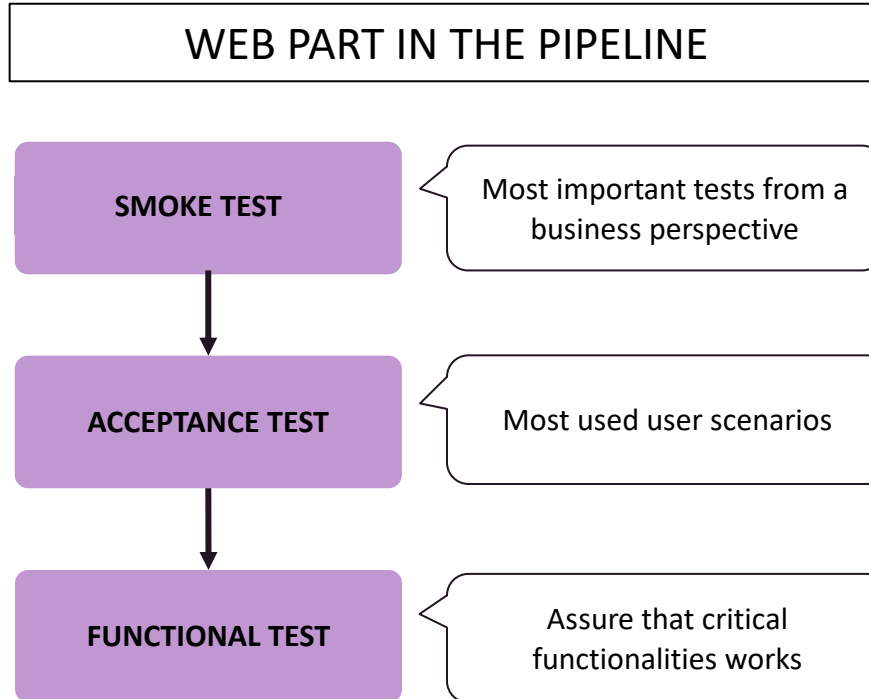
CURRENTLY NOT BEING DEVELOPED ANYMORE IN ORDER TO SELENIUM GRID 4 ADOPTION

PIPELINE

make the execution process clear



DIVIDE ALL TYPES OF EXECUTION



THANK YOU!

<https://github.com/eliasnogueira/public-speaking>

