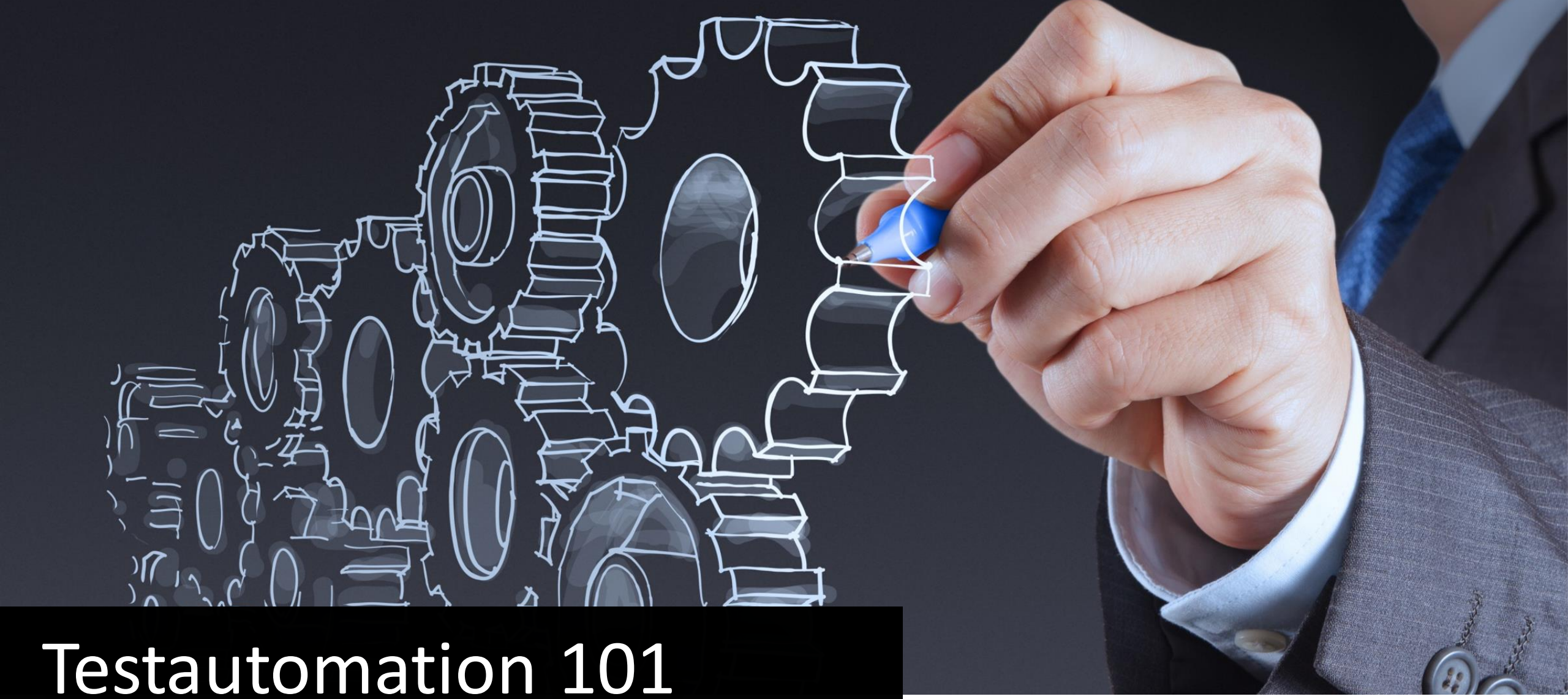


Testautomation 101

Boris
Wrubel

DI Boris Wrubel

Softwaretest

Self-employed - 2004

Soccer referee

Test-automation Engineer

Travel

Golf

ISTQB AL (TM,TA)
CMAP CAT

Business Informatics

PESO – Software Testing

Vienna University of Technology

Timetable

PART I

- Introduction
- 60min Challange

PART II

- Identify your WebElements
- PageObject Pattern

PART III + IV

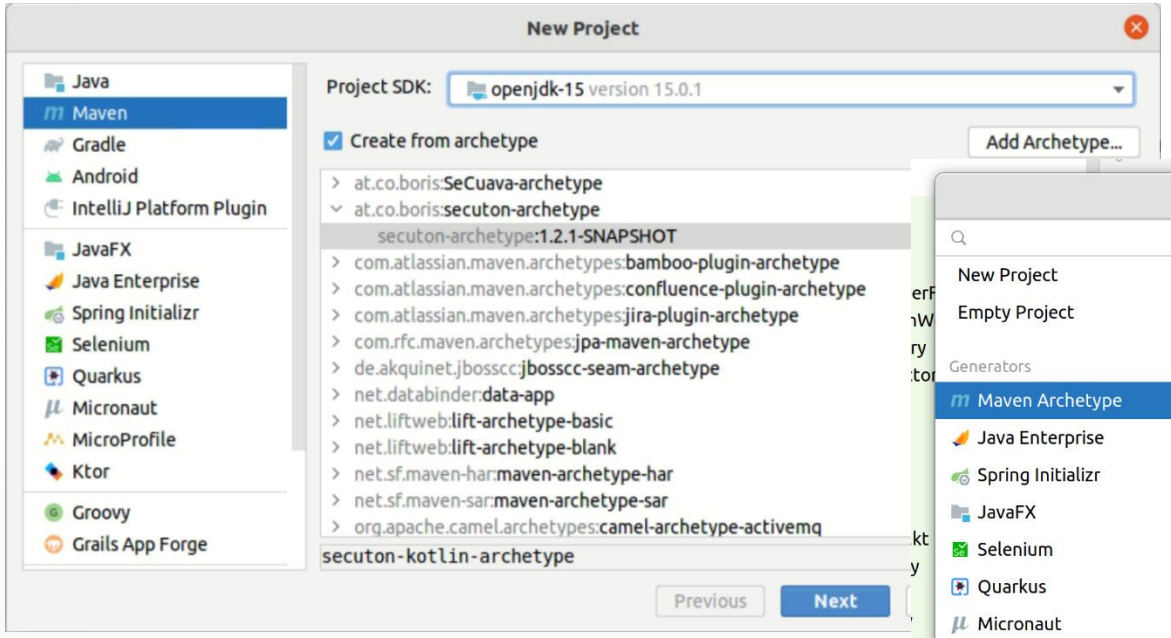
- Let the IDE work for you
- Reporting

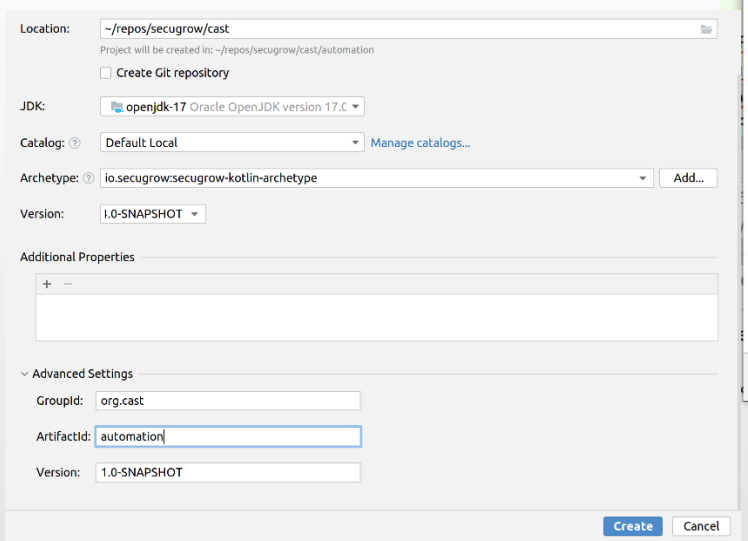
Why did we founded SeCuGrow?

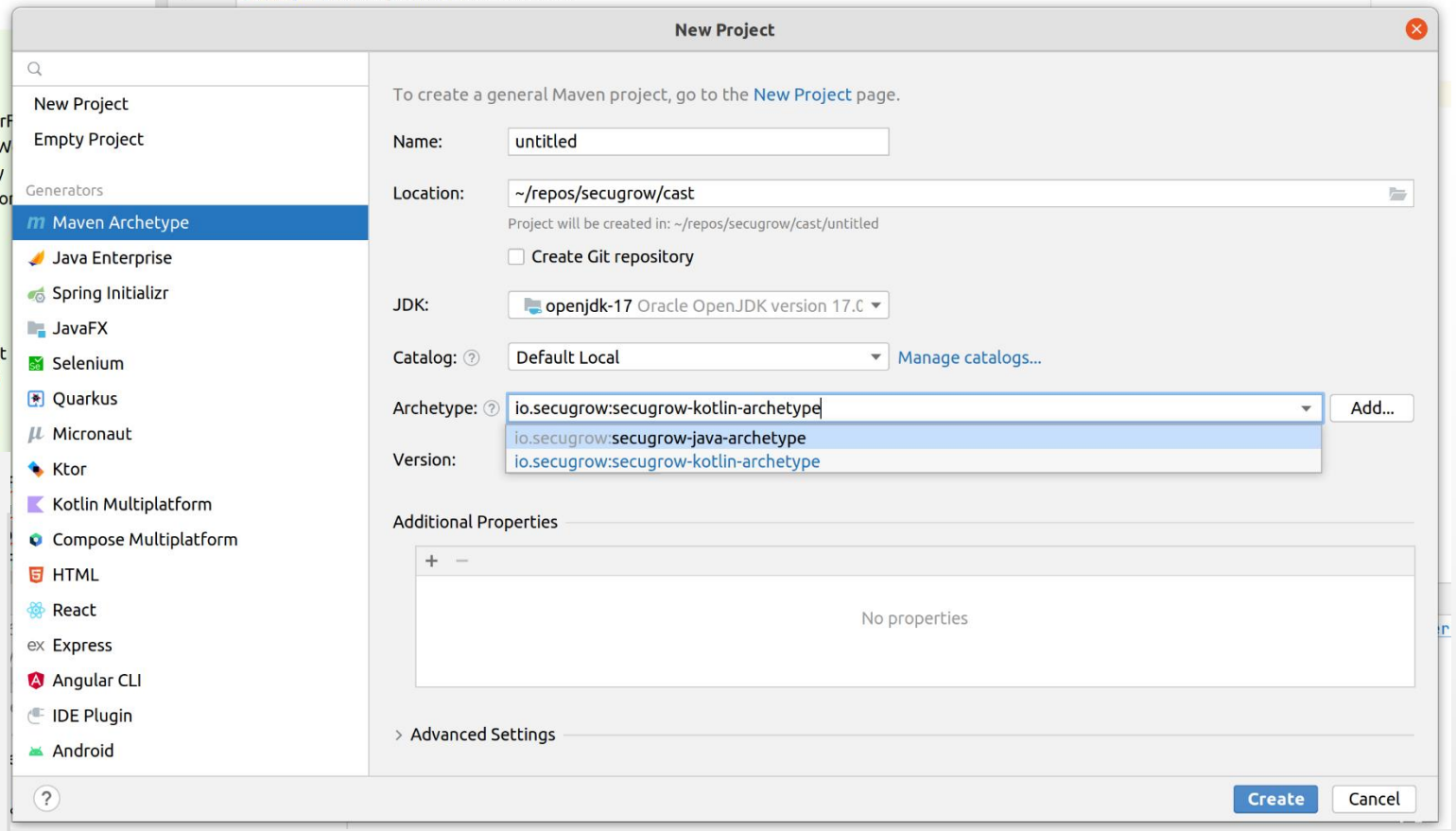
Selenium Cucumber Project Generator

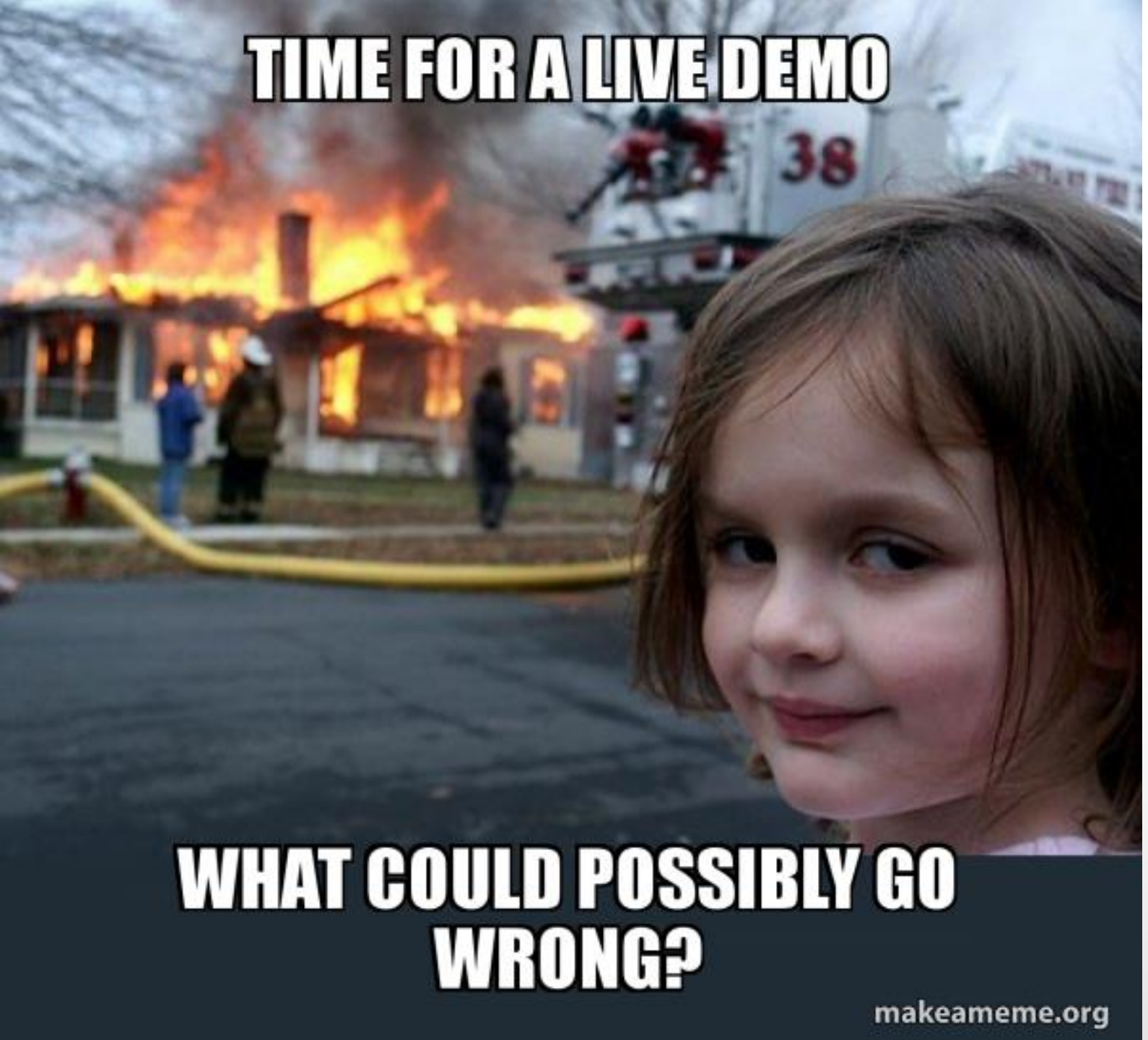
Maven Archetype

Project generator







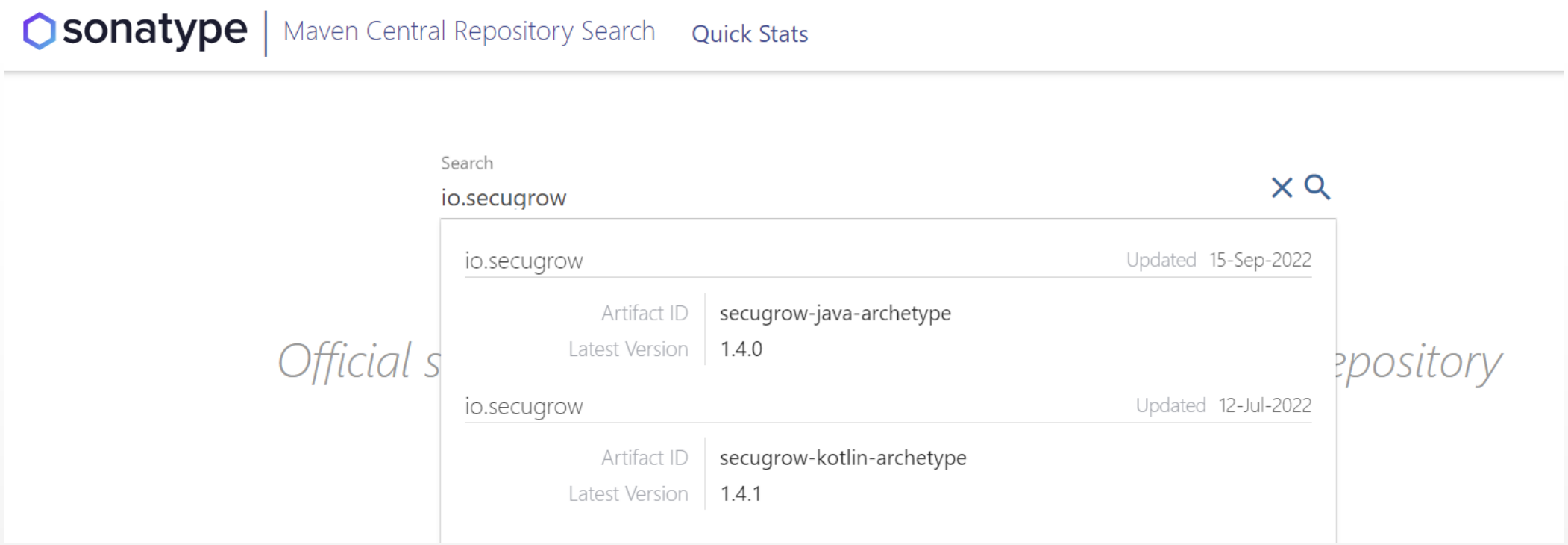


SecuGrow

secugrow.io

https://boris-wrubel.gitbook.io/cast-2022/

It‘s open source, it‘s in the repo



Identify your WebElements

One of the most important things

- Good identifications
  - Stable
  - Relyable
  - Maintaince

Most low-Code/no-Code tools have their strength with the element localization.

Element identification

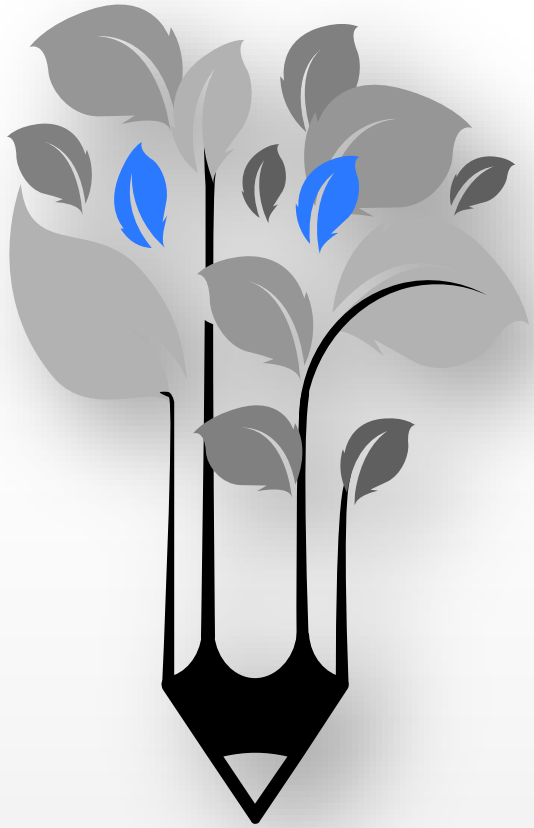
If you want to interact with an element, you will need to first identify the element. You can also call it by locating an element or finding an element. Besides, you can find multiple elements at once. Let us start with a non-automation example:

What are the ways for a teacher to identify a student in a classroom?

- *Each table corresponds to a team.*
- *Discuss and brainstorm.*
- *Take notes on all the identification strategies.*
- *Also discuss the limitations of a given identification strategy.*

Identification of participant

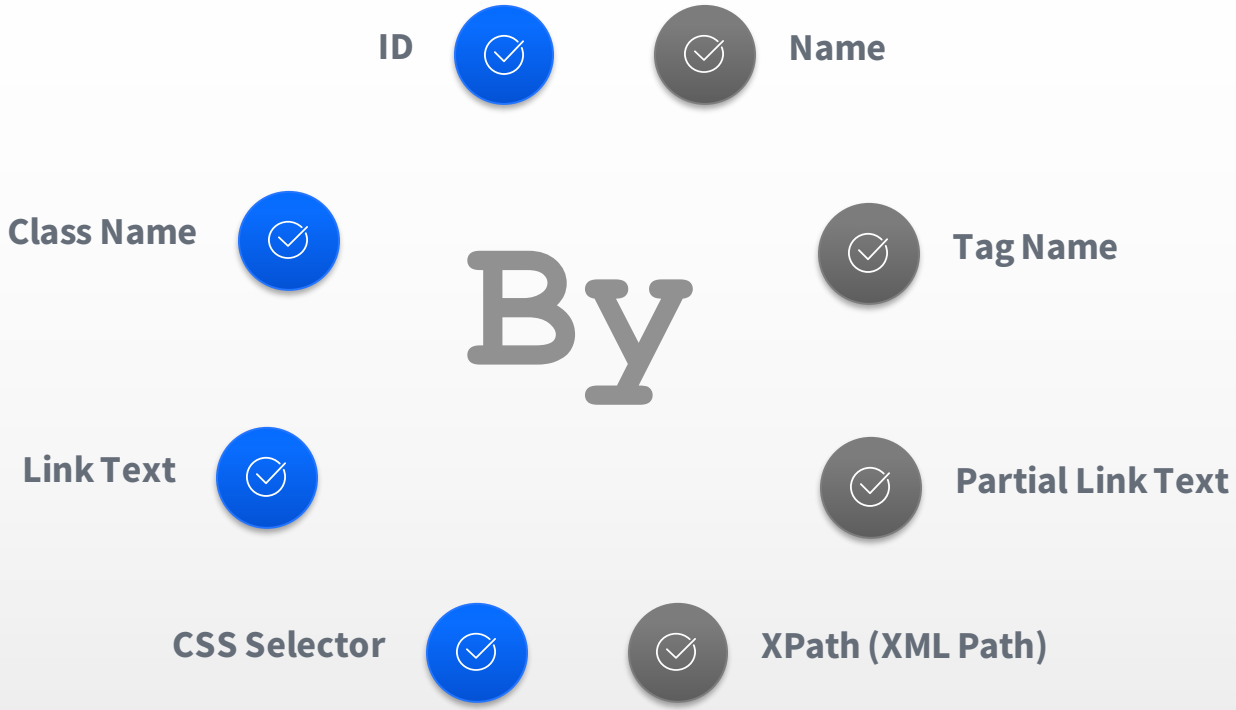
- Name
- Association
- Looks



- ID/Roll No.
- Category
- Other Attributes

Descripting identification types in Selenium

A `By` object is used to represent an element locator strategy in Selenium. A `By` object knows its locator type. In some language implementations, it also knows the value of the locator used for identification. The following are different locator strategies supported by `By`:



Relative Locators (since Selenium 4)

above
below
toLeftOf
toRightOf
near

```
val password = driver.findElement(By.id("password"))
val email = driver.findElement(with(By.tagName("input")).above(passwordField))
```

```
WebElement password = driver.findElement(By.id("password"));
WebElement email = driver.findElement(with(By.tagName("input")).above(passwordField));
```

CSS

- Tag
- Tag with Attribute

- Tag with Attribute Value
- ID
- Class
- Name

Special Symbols
- # for ID
- . for Class
- * for any tag

Relationships
- > for Parent to Child
- + for Following Sibling

Partial Match
- *= for Contains
- ^= for Starts With
- $= for Ends With
- ~= for Contains Word

Logical Operators
- , for OR
- . and # give AND condition in combination with element (filtering)

:not() function

- Multiple Attributes
- Multi Attribute Values

(Act as AND)

There are many more variants. Continue exploring after the workshop.

Xpath

- Tag (Absolute) – NOT Recommended
- Tag (Relative)
- Tag with Attribute

- Tag with Attribute Value
- ID
- Class
- Name

Special Symbols
- * for any tag

- text() for visible text

Relationships
- / for Parent to Child
- /following-sibling::

Partial Match
- contains() – Attr, Text
- starts-with() – Attr, Text
- ends-with() -> Currently not supported by browsers

Work with text() as well

Logical Operators
- or
- and

not() function

More Relationships (Axes)
- .. For Parent
- // for descendant
- /ancestor::
- /preceding-sibling::

- [number] for position
- (<xpath>)[number]

- Multiple Attributes
- Multi Attr Values

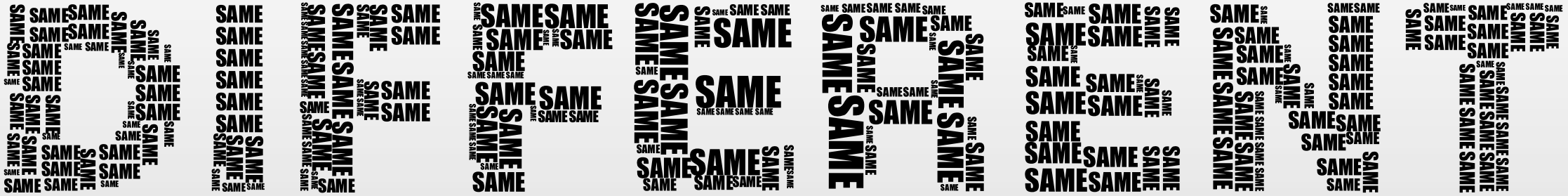
There are many more variants. Continue exploring after the workshop.

CSS vs Xpath

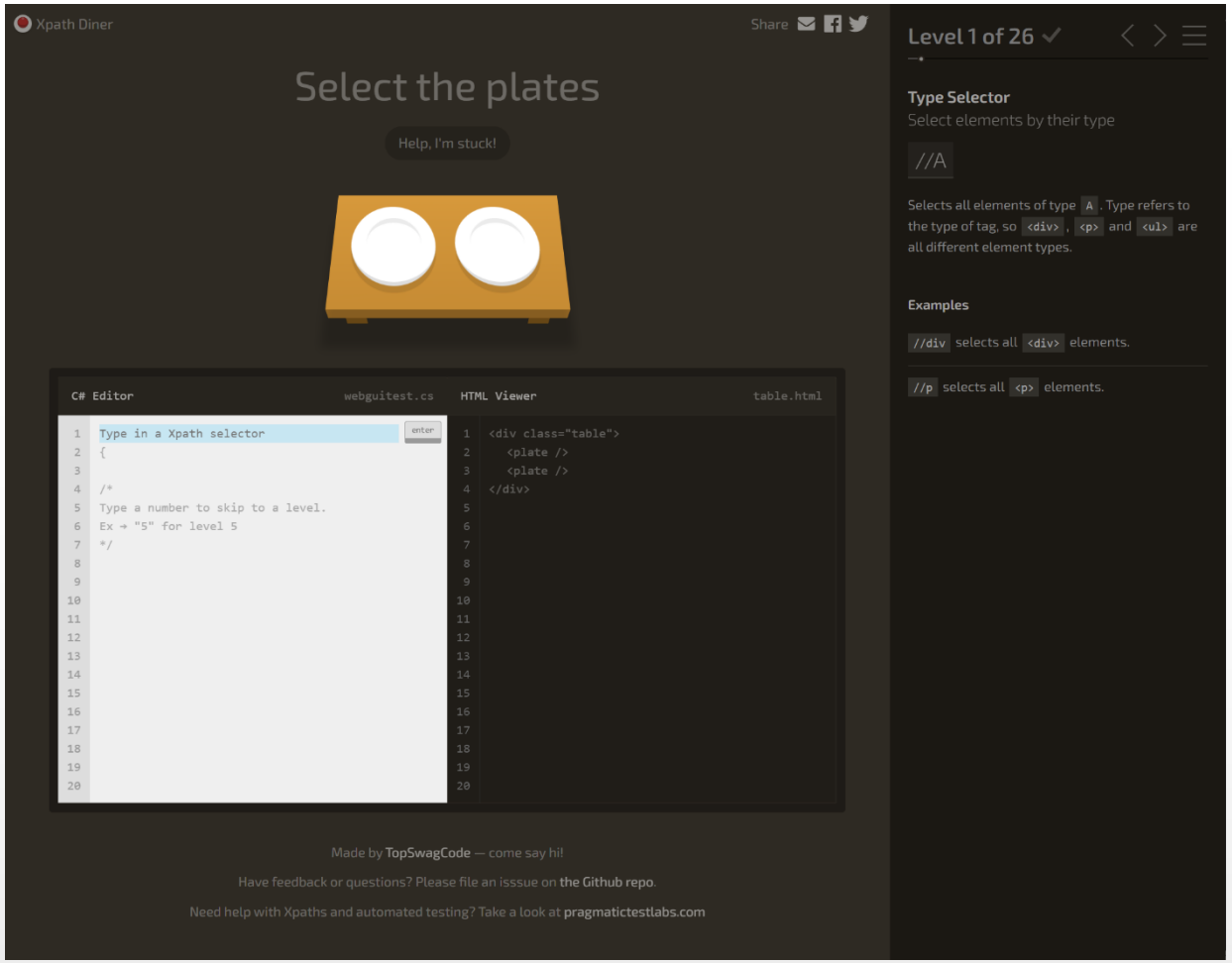
CSS does not load the whole tree, so it consumes less resources.

- Search backwards / tree up
- Text – search

```
▼<div class="col-xs-6 col-lg-7">
    <h4>Your profile is 85% complete.</h4> == $0
    <a class="btn btn-primary" href="/members/620a6c208446c9000787926f">
    Edit Profile</a>
  </div>
```



Go out for dinner tonight



https://topswagcode.com/xpath/

https://flukeout.github.io/

Pageobject Pattern

Pageobject Pattern explained

Idea:

- Pages provide functionalities
- Just call the functionality with data
- Separation of testcode and locators

e.g.
LoginPage.doLogin(username, password)
DocumentPage.downloadInvoice(“August 2022”)

Write code only once (easy to maintain)

https://www.selenium.dev/documentation/test_practices/encouraged/page_object_models/

IDE is coding for you

Fast testcreation

With the help of IDE

- Write step (reuse!)
- Generate step
- Call page methods
- Generate methods
- Write code

- getPage(*expectedPageClass*).methodCall(optional parameters)

- assertThat(actual).is(expected)

Fast testcreation

Syntax

Page functions

- getPage(*expectedPageClass*).functionCall(optional parameters)

Assertions

- assertThat(actual).is(expected)

Reporting

Reporting out of the box

- Cucumber plugins
- Cucumber report-service
- Integrate your reports in CI (Jenkins)

Cheatsheet

- Write your steps (don‘t reinvent things)
- Let the IDE help you
- Write your page functions
  - Reuse or create with IDE

Use the syntax from framework

- for Page functions:
  getPage(*expectedPageClass*).functionCall(optional parameters)

- For assertions (Then Steps)
  assertThat(actual).is(expected)

- testdataContainer for saving data (testdata, environment, expected data, etc.)
  testdatacontainer.setTestdata(key, value)

- get data
  val data = testdatacontainer.getData(key)

Testautomation 101

Let it grow… fast forward

Summary

- Start a Selenium Cucumber project very quickly
- Using a maven archetype is easy to learn
- Good locators for success
- Any browser, any device possible