Selenium

Testowanie automatyczne

Zajęcia I

Wprowadzenie do automatyzacji

Marek Konitz

CODE:ME

UCZYMY PROGRAMOWANIA

CODE:ME
UCZYMY PROGRAMOWANIA

Co będziemy robili na kursie

Zajęcia I

- Po co automatyzujemy i czy zawsze się to opłaca
- Dlaczego nie omawiamy tylko Selenium
- Co to jest egzamin A4Q Certified Selenium Tester Foundation
- Jak jest zbudowana strona www i jak znajdujemy elementy na stronie
- Narzędzia developerskie w Chrome i przydatne wtyczki

Egzamin A4Q Selenium Tester Foundation Level

Certyfikat potwierdza, że osoba go posiadająca:

- Zna i potrafi zastosować zasady testowania automatycznego do utworzenia poprawnych i łatwych w utrzymaniu testów
- Potrafi wybrać odpowiednie narzędzia to automatyzacji i zastosować je w praktyce
- Potrafi zaimplementować skrypty z użyciem Selenium WebDriver do przeprowadzenia testów funkcjonalnych aplikacji webowych

Automatyzacja testowania

Automatyzować można praktycznie każdy aspekt testowania aplikacji, na potrzeby tego kursu ograniczymy się do wybranego zakresu:

Automatyczne **wykonanie** zaprojektowanych przypadków testowych dla **testów funkcjonalnych** w sposób symulujący **działanie człowieka**

Co chcemy (i możemy) osiągnąć

- Zmniejszenie kosztów wykonania testów
- Przetestowanie większego zakresu przypadków testowych niż byłoby to możliwe lub praktyczne w sposób manualny
- Skrócenie procesu testowania - szybszy feedback
- Zwiększenie częstotliwości wykonania testów
- Zwiększenie powtarzalności testów
- Odciążenie pracowników od żmudnych i powtarzalnych czynności

Czego nie chcemy (ale możemy) osiągnąć

- Zwiększenie kosztów testowania (zwłaszcza na początku projektu)
- Zbyt wielką złożoność testów i trudność w utrzymaniu
- Zwiększenie ryzyka wystąpienia paradoksu pestycydów
- Trudności z wykryciem przypadków fałszywie pozytywnych testów i fałszywie negatywnych
- Zbyt dużą koncentrację na utrzymaniu skryptów automatycznych względem testowanej funkcjonalności
- Fałszywe poczucie bezpieczeństwa

Czynniki wpływające na sukces automatyzacji

- planowanie długoterminowe powiązane z celami biznesowymi
- mądre zarządzanie
- pilnowanie detali
- dojrzałość procesu wytwarzania oprogramowania
- sformalizowanie architektury i frameworków do testowania
- odpowiednie przeszkolenie
- właściwe utrzymywanie dokumentacji

Architektura Testów Automatycznych

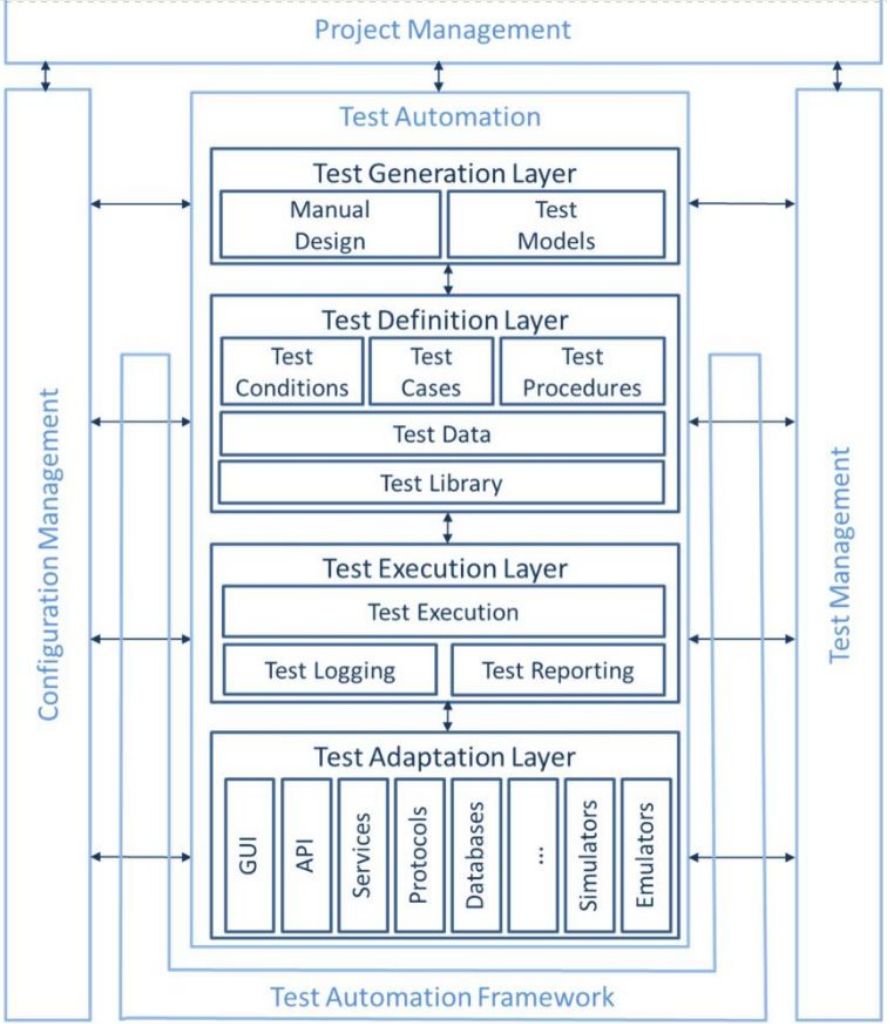


Figure 1: A generic TAA (from TAE syllabus)

Jakie interfejsy możemy automatyzować

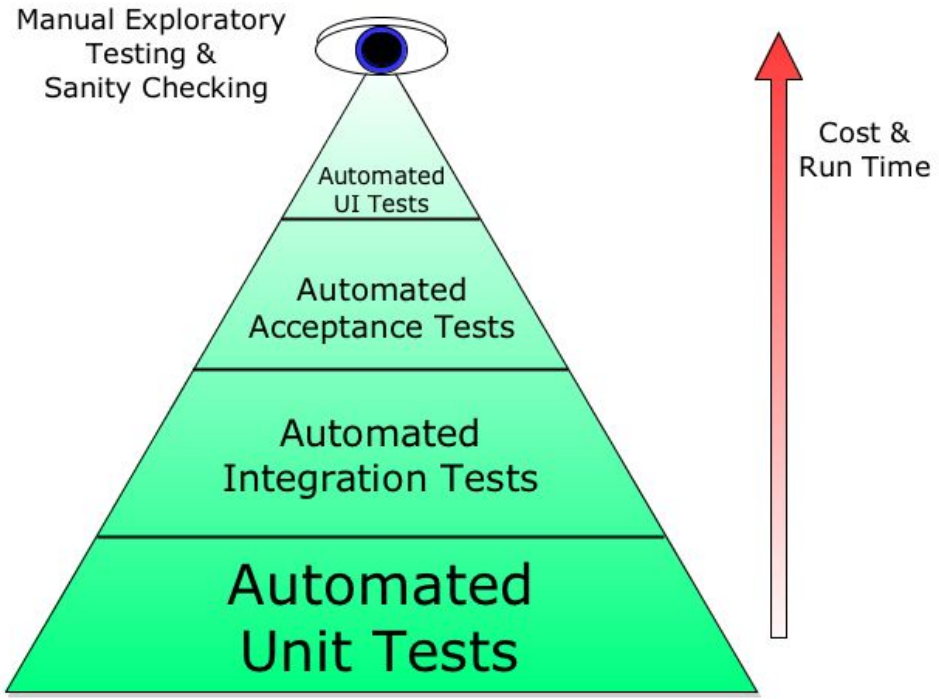
- GUI level
  - Graphical User Interface
- API level
  - Application Programming Interface
- Private hooks (Test APIs)
  - Test API
- Service level (SOAP, REST, etc.)
  - Application protocol level
- Protocol level (HTTP, TCP, etc.)
  - Transport

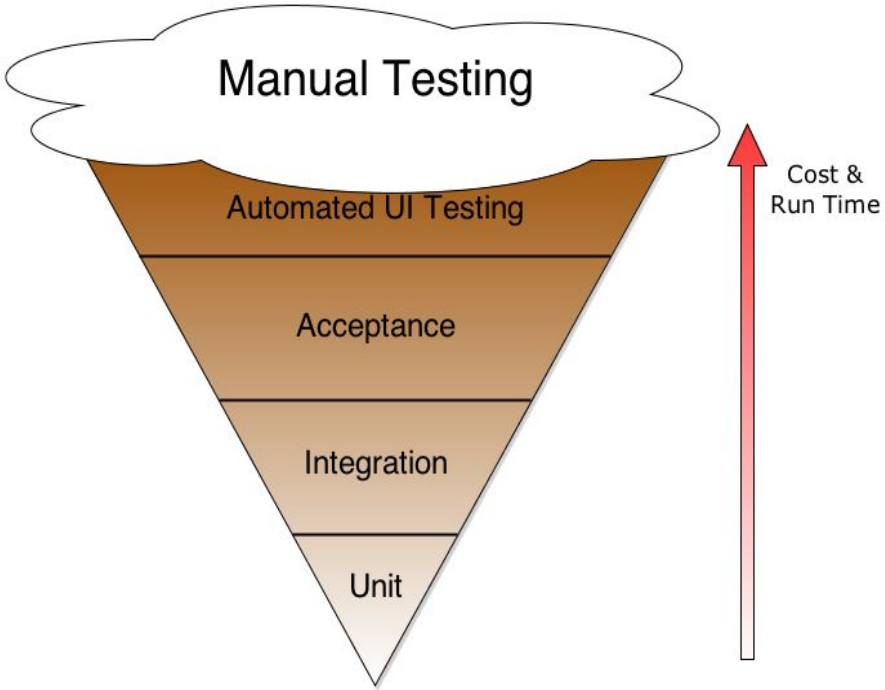
Czym ryzykujemy stosując Selenium

- Testy GUI wydają się intuicyjnym wyborem
- Zwłaszcza w środowiskach CI zajmują więcej czasu niż oczekujemy
- Zmiany w UI powodują konieczność zmiany testów
- Testerzy manualni często znajdują błędy o wiele bardziej efektywnie niż testy automatyczne
- Gdy aplikacja jest stosunkowo stabilna, testy automatyczne mogą “nie dać zwrotu z inwestycji”

Czy unikać testów na poziomie GUI

Mimo swoich wad - testy GUI są najbardziej zbliżone do rzeczywistego użycia aplikacji


Manual Exploratory
Testing &
Sanity Checking
Automated
UI Tests
Automated
Acceptance Tests
Automated
Integration Tests
Automated
Unit Tests
Cost &
Run Time

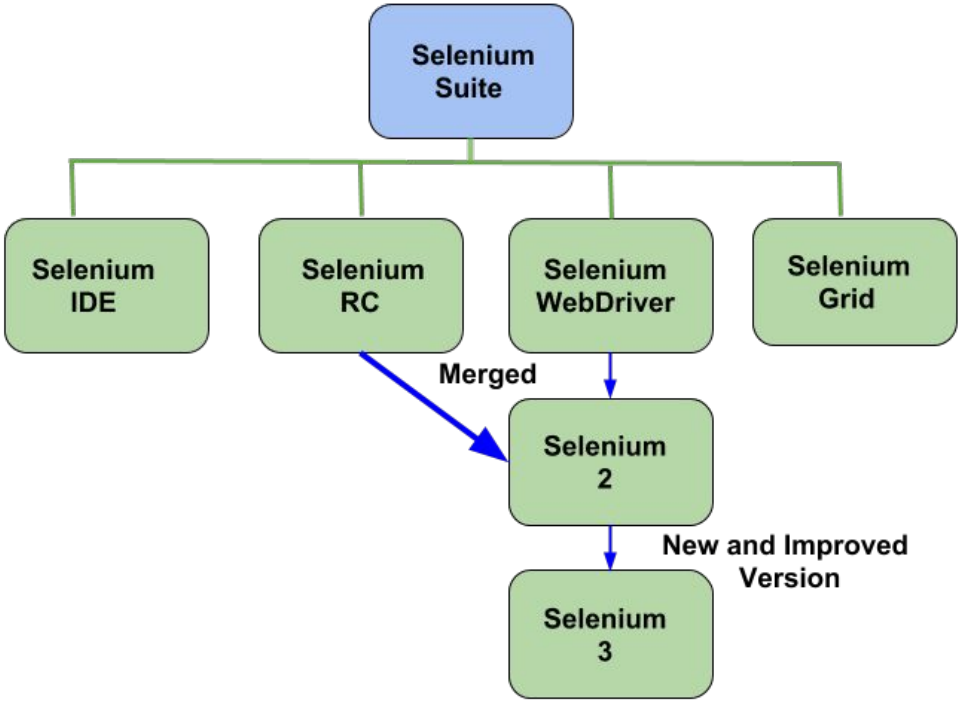

Manual Testing
Automated UI Testing
Acceptance
Integration
Unit
Cost &
Run Time

Selenium

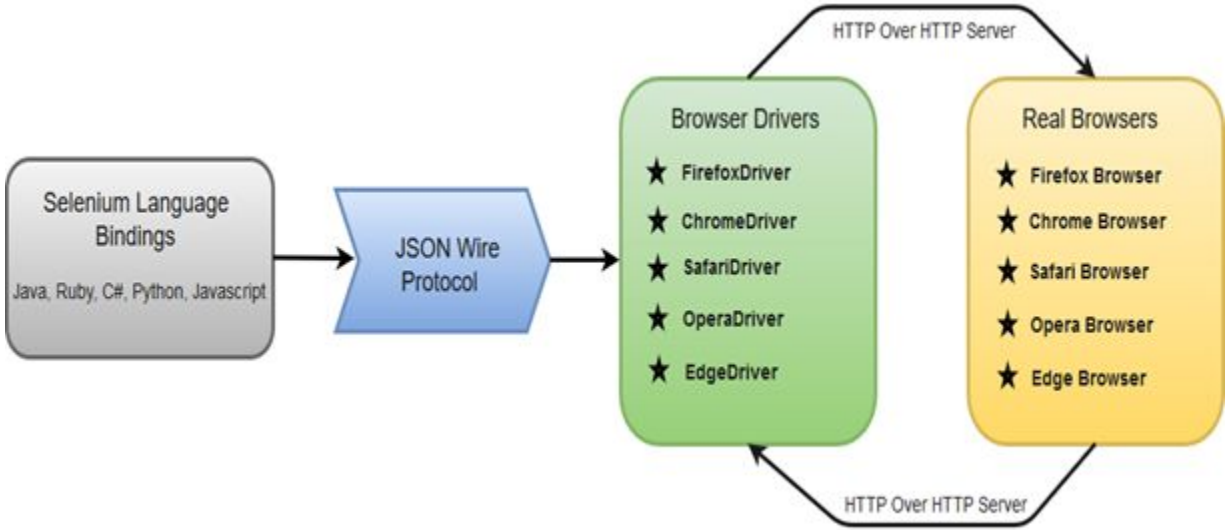
Testowanie za pomocą Selenium

- testy automatyczne
- aplikacje webowe
- na poziomie interfejsu użytkownika
- testy funkcjonalne
- zestaw komend Selenese
- użycie jednego z wielu języków programowania

Selenium - architektura



Selenium - jak działa “pod spodem”



Selenium 4

Udoskonalono

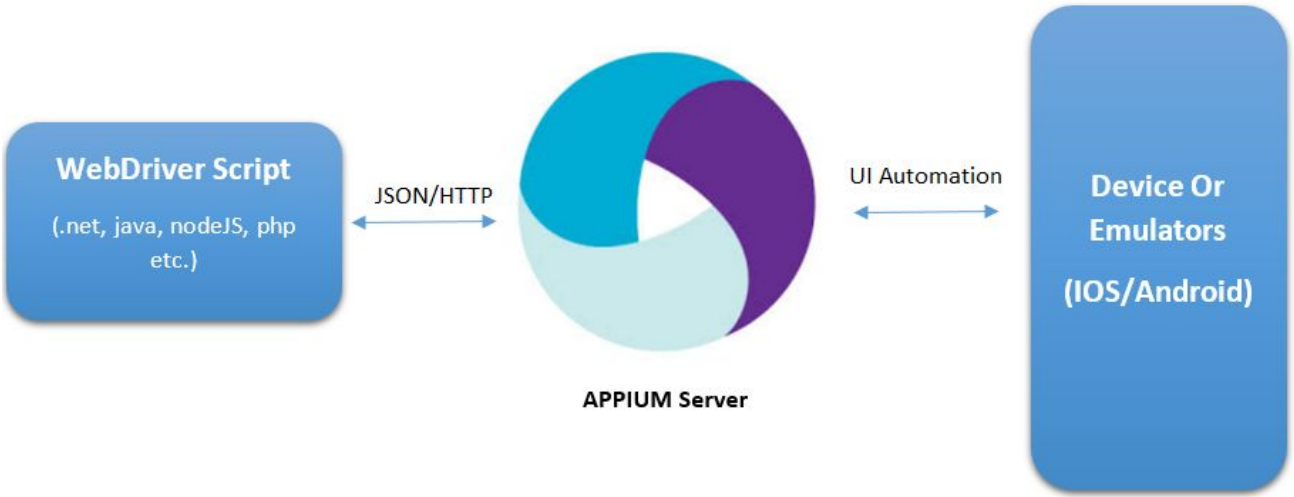
- Wsparcie dla Dockera
- Różne koncepcje Selenium Grid

Nowe cechy

- Relative locators - lokatory względne
- Obsługa wielu przeglądarek, okien i zakładek
- Natywne wsparcie dla Chrome Dev Tools

Testowanie aplikacji mobilnych

appium



XML a HTML

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<note>
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this
weekend!</body>
</note>
<note>
      <title>Favorite food</title>
      <body>Shrimps with butter</body>
</note>
```

HTML

```html
<!DOCTYPE html>
<html>
      <head>
            <title>Page Title</title>
      </head>
      <body>

            <h1>My First Heading</h1>
            <p>My first paragraph.</p>

      </body>
</html>
```

Podstawowe znaczniki HTML

Table 2: Basic HTML Tags

Tag	Used for
<html> ... </html>	Signifies root of HTML document
<!DOCTYPE>	Defines document type (not needed in HTML 5)
<head> ... </head>	Definition and meta data for document
<body> ... </body>	Defines main content for the document
...	Defines a paragraph
 	Inserts a single line break
...	Defines a section in the document
<-- ... -->	Defines a comment (may be multi-line)

Podstawowe znaczniki HTML - listy

Table 4: List and Table Tags

Tag	Used for
...	Defines an unordered (bulleted) list
...	Defines an ordered (numbered) list
...	Defines a list item (for or)

Podstawowe znaczniki HTML - tabele

<table> ... </table>	Defines an HTML table
<tr> ... </tr>	Defines a table row
<th> ... </th>	Defines the column header for a table
<td> ... </td>	Defines a table data cell
<tbody> ... </tbody>	Groups body content in an HTML table
<thead> ... </thead>	Defines an HTML table header
<tfoot> ... </tfoot>	Defines an HTML table footer
<colgroup> ... </colgroup>	Groups table columns for formatting

Lokalizowanie elementów

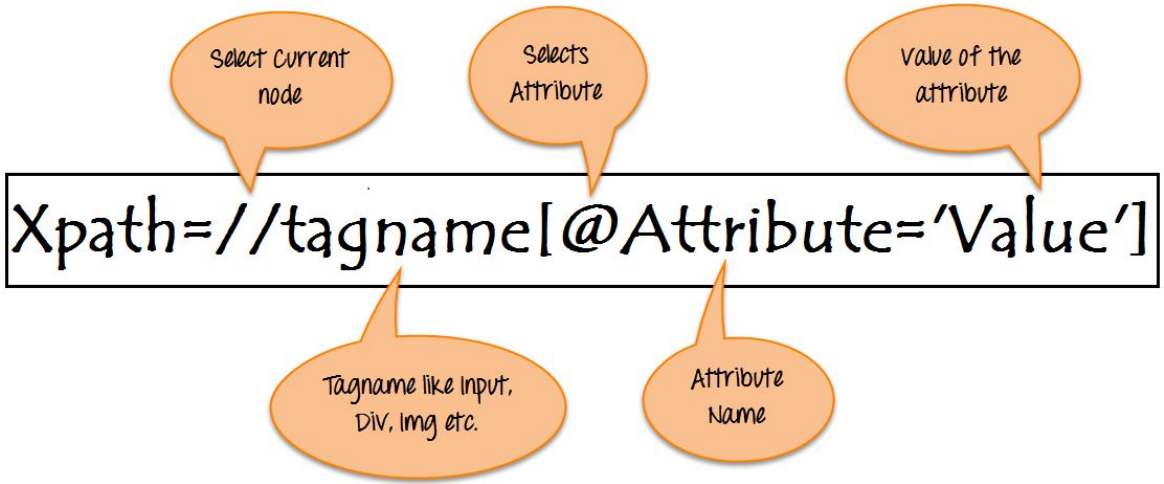
Method	Target Syntax	Example
By ID	id= *id_of_the_element*	id=email
By Name	name=*name_of_the_element*	name=userName
By Name Using Filters	name=*name_of_the_elementfilter=value_of_filter*	name=tripType value=oneway
By Link Text	link=*link_text*	link=REGISTER
Tag and ID	css=*tag#id*	css=input#email
Tag and Class	css=*tag.class*	css=input.inputtext
Tag and Attribute	css=*tag*[*attribute=value*]	css=input[name=lastName]
Tag, Class, and Attribute	css=*tag.class*[*attribute=value*]	css=input.inputtext[tabindex=1]

XPath (1)

XML Path Language

+ Najbardziej uniwersalny sposób lokalizowania elementów

- Najbardziej skomplikowany - jest mnóstwo reguł i sposobów dotarcia do tego samego elementu



XPath (2)

- XPath absolutny

xpath=/html/body/h1

xpath=/html/body/div/p[1]

- XPath względny

xpath=//p

xpath=//p[1]

xpath=//div/p[1]

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>

    <h1>This is a Heading</h1>
    <div>
         <p>This is a paragraph.</p>
         <p>This is a second paragraph
     </div>

  </body>
</html>
```

XPath (3)

XPath -> XML Path language

- Wyszukiwanie węzłów

/	ścieżka bezwzględna
//	ścieżka względna
. ..	bieżący węzeł / rodzic
@	atrybut
*	
@*	
node()	

- Predykaty

zawężanie zbioru do węzłów spełniających warunek

[ ]

//node[last()]

//node[@attr = “”]

//node[text() > 100]

//node[contains(text(), “piece of text”]