

# TESTY END-TO-END

Programowanie wizualne

Wojciech Frohmberg, Instytut Informatyki, Politechnika Poznańska

2022



# **TESTY JEDNOSTKOWE VS TESTY INTEGRACYJNE**

Tak jak testy jednostkowe skupiają się na weryfikacji poprawności działania wyodrębnialnych części kodu tj. tzw. jednostek (i stąd też nazwa – jednostkowe) tak testy integracyjne skupiają się na współdziałaniu tych jednostek ale też nadbudowanych na nich modułów czy bibliotek (nazwa zatem odnosi się do zestawiania ze sobą funkcjonalności i testów tej współpracy).

# **TESTY INTEGRACYJNE VS TESTY END-TO-END**

Testy end-to-end (e2e) to zasadniczo testy integracyjne – weryfikują czy poszczególne fragmenty kodu współpracują ze sobą. Ich specyfika jednak polega na badaniu wielu takich "integracji" na raz. Właściwie przeprowadzamy tu testowanie aplikacji włąb po wszystkich jej warstwach – począwszy od warstwy interfejsu użytkownika i wypełniania formularzy skończywszy na warstwie danych i dostępu do bazy.

## **TESTY END-TO-END**

Tworząc testy e2e skupiamy się na funkcjonalności aplikacji, tym samym możemy przeprowadzać w ramach aplikacji całe przypadki jej użycia. Nie jesteśmy tutaj ograniczeni do testowania w ramach pojedynczej roli. Co jest jednak istotne testy e2e to nadal testy automatyczne. Oczywiście, z racji na charakter testów, dozwolone jest tutaj ich dłuższe, w porównaniu do standardowych testów integracyjnych, działanie.



The background of the entire image is a repeating pattern of stylized flowers and swirls. The pattern is composed of white and black elements on a dark background. The flowers are stylized with multiple layers of petals, and the swirls are elegant, flowing lines. The pattern is dense and covers the entire area.

**SELENIUM**



## **TESTY E2E Z BIBLIOTEKĄ SELENIUM**

Biblioteka Selenium stanowi zestaw narzędzi do automatyzacji korzystania z aplikacji przeglądarkowych. Wbrew powszechnej opinii nie musi służyć tylko i wyłącznie do przeprowadzania testów, jednak najczęściej pożytkuje się ją właśnie w tym zastosowaniu. Biblioteka deklaruje ujednolicony interfejs dostawania się do akcji przeglądarki. Poszczególne przeglądarki muszą udostępniać do tego celu dedykowane sterowniki.

## **TESTY E2E Z BIBLIOTEKĄ SELENIUM**

Biblioteka współpracuje z wieloma językami programowania oraz technologiami. W każdym języku biblioteka dostosowana jest do konwencji wynikających z tego języka. Z reguły jednak określa interfejs, przy użyciu którego możemy abstrahować od sterownika konkretnej przeglądarki.

# **NAJWAŻNIEJSZE INTERFEJSY BIBLIOTEKI SELENIUM W JĘZYKU C#**

- **IWebDriver** – główny interfejs biblioteki stanowiący pewnego rodzaju jej punkt wejścia, przy użyciu którego uzyskujemy kontrolę nad instancją przeglądarki
- **IWebElement** – podczas korzystania ze sterownika przeglądarki uzyskujemy dostęp do poszczególnych elementów znajdujących się na stronie IWebElement stanowi interfejs do właśnie takich elementów
- **ISearchContext** – zarówno IWebDriver jak i IWebElement dziedziczą właśnie z tego interfejsu, przy jego użyciu jesteśmy w stanie odnajdywać poszczególne elementy na stronie zgodnie z ich pozycją w ramach drzewa elementów (struktury DOM)
- **IJavaScriptExecutor** – punkt wejścia, przy użyciu którego możemy strzykiwać skrypty do strony



# **NAJWAŻNIEJSZE KLASY BIBLIOTEKI SELENIUM W JĘZYKU C#**

- **WebDriverWait** – udostępnia uogólniony sposób oczekiwania na zaistnienie określonego zdarzenia w ramach aktualnie uruchomionej strony
- **Actions** – stanowi agregat kolejnych kroków, które chcielibyśmy zaaplikować w ramach strony, dzięki wykorzystaniu obiektu tej klasy jesteśmy w stanie sterować zdarzeniami uruchomionymi na elementach strony (np. najechanie myszką na element), ale również kontrolować ich czas trwania

# ISTOTNE INTERFEJSY

## IWEBDRIVER (1)

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

using (IWebDriver driver = new ChromeDriver())
{
    driver.Url = "https://google.com";
    driver.Navigate().GoToUrl("https://cuda.cs.put.poznan.pl");
    Console.WriteLine($"cuda source: {driver.PageSource}");
    driver.Navigate().Back();
    Console.WriteLine(driver.Title);
}
Console.ReadKey();
```

W ramach interfejsu IWebDriver dostępnych jest szereg elementów służących do nawigacji po adresie strony jej frame'ach oraz wyłuskiwania poszczególnych właściwości strony:

- **Url** – właściwość za pomocą której możemy przeładować stronę na określony adres bądź uzyskać informację o aktualnym adresie
- **Title** – właściwość, za pomocą której możemy odczytać tytuł aktualnej strony
- **PageSource** – właściwość, za pomocą której możemy odczytać źródło aktualnej strony
- **Navigate** – metoda służąca do wyłuskiwania obiektu nawigującego po stronie za jego pomocą będziemy mogli poruszać się po historii przeglądania stron w przeglądarce, odświeżyć stronę czy też przejść do zadanego adresu

# ISTOTNE INTERFEJSY

## IWEBDRIVER (2)

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

IWebDriver driver = new ChromeDriver();
driver.Url = "https://google.com";
foreach (var cookie in driver.Manage().Cookies.AllCookies)
    Console.WriteLine($"{cookie.Name} = {cookie.Value}");
driver.Quit();
Console.ReadKey();
```

- **Manage** – metoda pobierająca obiekt służący do zarządzania opcjami strony, pozwala np. na dostęp do ciasteczek strony (właściwość Cookies) czy logów strony (właściwość Logs)
- **SwitchTo** – metoda służąca do przechodzenia pomiędzy aktywnymi frame'ami strony
- **Quit** – metoda służąca do zamykania okna przeglądarki

# ISTOTNE INTERFEJSY

## ISEARCHCONTEXT (1)

- **FindElement** - służy do znajdowania pierwszego wystąpienia elementu na stronie spełniającego założone predykatem kryteria.
- **FindElements** - służy do znalezienia wszystkich wystąpień elementu spełniających określone predykatem kryteria.

**Predykat, który przyjmują powyższe metody powinien być typu By.**



# ISTOTNE INTERFEJSY

## ISEARCHCONTEXT (2)

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

IWebDriver driver = new ChromeDriver();
driver.Url = "https://google.com";
driver.FindElement(
    By.XPath("//*[text()='Zaakceptuj wszystko']"))
    .Click();
driver.FindElement(By.TagName("input"))
    .SendKeys("google pacman" + Keys.Enter);
Console.ReadKey();
driver.Quit();
```

Możliwe rodzaje predykatu By:

- **ClassName** – gdy szukamy po klasie css elementu
- **CssSelector** – gdy szukamy po dowolnym selektorze css
- **Id** – gdy szukamy po htmlowym identyfikatorze elementu
- **LinkText** – gdy szukamy równościowo po tekście linku znajdującym się na stronie
- **PartialLinkText** – gdy szukamy po fragmencie linku ze strony
- **Name** – gdy szukamy po atrybucie Name stosowanym w formularzach strony
- **TagName** – gdy szukamy element po tagu
- **XPath** - najpotężniejszy z predykatów, pozwala zdefiniować dostosowane zapytanie wynikające ze struktury elementów drzewa DOM (właściwie to XML)

app/abstract

```
<app>
  <welcome-message>Hi! This is xpather beta...</welcome-message>
  <abstract>
    This web app enables you to query XML/HTML documents with your
    browser in real time. It can generate queries for you too!
  </abstract>
  <description>
    <subject>
      You can enter your xpath query in the top-left panel
      and it will be instantly executed against this document.
      Once some results are displayed on the right, you can
      scroll to them by clicking on them.
    </subject>
    <subject>
      To generate an xpath query for a specific element,
      please hold CTRL and hover over it.
      An xpath is generated heuristically with the aim
      to be unambiguous and the shortest possible.
    </subject>
  </description>
  <extra-notes>
    <note>
      None of entered documents leave your computer because all
      the processing is done by your powerful browser!
      (of course as long as you do not save your input)
    </note>
  </extra-notes>
</app>
```

//book/price

```
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <cover>
      <price>2.99</price>
    </cover>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
    with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <cover>
      <price>2.99</price>
    </cover>
    <price>5.95</price>
    <publish_date>2000-12-15</publish_date>
  </book>
</catalog>
```

# XPATh 1.0 W PRZYKŁADACH \*

Selenium obsługuje XPath w wersji wynikającej z wersji wspieranej natywnie przez przeglądarki (tj. najczęściej 1.0).  
Najważniejsze elementy składni XPath 1.0:

- pełną ścieżkę elementów podajemy przez oddzielenie każdego z tagów slashem
- rozpoczęcie poszukiwania od dowolnego miejsca w strukturze osiągamy korzystając z podwójnego slashu

\* Przykłady zrealizowane przy użyciu strony: <http://xpather.com/>

```
//test_run/@status
```

```
<test_result>
  <build build_id="31" job_id="Test-REST" server_id="12345678"/>
  <test_runs>
    <test_run started="STARTED_TS"
              status="Skipped"
              duration="14"
              name="bandTestA"
              class="BandTest"
              package="com.mycomp.devops.demoapp"
              module="webapp"/>
    <test_run started="STARTED_TS"
              status="Passed"
              duration="0"
              name="bandTestB"
              class="BandTest"
              package="com.mycomp.devops.demoapp"
              module="webapp"/>
    <test_run started="STARTED_TS"
              status="Passed"
              duration="1"
```

```
//book[cover/price<2.8]
```

```
<catalog>
  <book id="bk101">
    <author>Gambardella, Matthew</author>
    <title>XML Developer's Guide</title>
    <genre>Computer</genre>
    <cover>
      <price>2.79</price>
    </cover>
    <price>44.95</price>
    <publish_date>2000-10-01</publish_date>
    <description>An in-depth look at creating applications
    with XML.</description>
  </book>
  <book id="bk102">
    <author>Ralls, Kim</author>
    <title>Midnight Rain</title>
    <genre>Fantasy</genre>
    <cover>
      <price>2.89</price>
    </cover>
    <price>5.95</price>
    <publish_date>2000-12-16</publish_date>
    <description>A former architect battles corporate zombies,
```

# XPATH 1.0 W PRZYKŁADACH \*

- celem dostania się do atrybutu danego elementu korzystamy ze znaku @.
- **UWAGA!** w Selenium tylko pełne tagi mogą być zwracane przez wyrażenie XPath – nie można zwracać atrybutów a podany przykład w Selenium będzie zwracał błąd, oczywiście nic nie stoi na przeszkodzie żeby atrybuty wykorzystywać w procesie wyłaniania pożądanego tagu
- warunek użycia danego tagu określamy w nawiasach kwadratowych (w jego ramach można m.in. używać elementów składowych tagu)

\* Przykłady zrealizowane przy użyciu strony: <http://xpather.com/>

```
//song[number(substring(@length, 1, 1))>4]
```

```
<music>
<artist name="Radiohead">
  <album title="The King of Limbs">
    <song title="Bloom" length="5:15"/>
    <song title="Morning Mr Magpie" length="4:41"/>
    <song title="Little by Little" length="4:27"/>
    <song title="Feral" length="3:13"/>
    <song title="Lotus Flower" length="5:01"/>
    <song title="Codex" length="4:47"/>
    <song title="Give Up the Ghost" length="4:50"/>
    <song title="Separator" length="5:20"/>
    <description link="http://en.wikipedia.org/wiki/The_King_of_Limbs">
      The King of Limbs is the eighth studio album by English rock band Radi
    </description>
  </album>
  <album title="OK Computer">
    <song title="Airbag" length="4:44"/>
    <song title="Paranoid Android" length="6:23"/>
    <song title="Subterranean Homesick Alien" length="4:27"/>
    <song title="Exit Music (For a Film)" length="4:24"/>
    <song title="Let Down" length="4:59"/>
    <song title="Karma Police" length="4:21"/>
    <song title="Fitter Happier" length="1:57"/>
    <song title="Electioneering" length="3:50"/>
    <song title="Climbing Up the Walls" length="4:45"/>
    <song title="No Surprises" length="3:48"/>
    <song title="Lucky" length="4:19"/>
  </album>
</music>
```

```
//value[position()=//value[last()]]
```

```
<valuepack>
  <values>
    <value>5</value>
    <value>3</value>
    <value>2</value>
  </values>
</valuepack>
```

# XPATH 1.0 W PRZYKŁADACH \*

- przykładowe funkcje w XPath 1.0:
  - contains(haysack[string], needle[string])
  - ⑩ normalize-space([string]) - usuwa zbędne spacje
  - ⑩ substring([string], start[number], length[number])
  - ⑩ string-length([string])
  - ⑩ position() - pozycja elementu na liście wszystkich dopasowanych elementów
  - ⑩ last() - liczba wszystkich dopasowanych pozycji
  - ⑩ count([path]) - liczba dopasowujących się pozycji
  - ⑩ text() - wyłuskuje literał tekstowy z aktualnego tagu
  - ⑩ number([string]) - dokonuje konwersji ciągu znaków na liczbę

\* Przykłady zrealizowane przy użyciu strony: <http://xpather.com/>



```
//*[@title='Paranoid Android']/..
```

```
<music>
  <artist name="Radiohead">
    <album title="The King of Limbs">
      <song title="Bloom" length="5:15"/>
      <song title="Morning Mr Magpie" length="4:41"/>
      <song title="Little by Little" length="4:27"/>
      <song title="Feral" length="3:13"/>
      <song title="Lotus Flower" length="5:01"/>
      <song title="Codex" length="4:47"/>
      <song title="Give Up the Ghost" length="4:50"/>
      <song title="Separator" length="5:20"/>
      <description link="http://en.wikipedia.org/wiki/The_King_of_Limbs">
        The King of Limbs is the eighth studio album by English rock band Radiohead
      </description>
    </album>
    <album title="OK Computer">
      <song title="Airbag" length="4:44"/>
      <song title="Paranoid Android" length="6:23"/>
      <song title="Subterranean Homesick Alien" length="4:27"/>
      <song title="Exit Music (For a Film)" length="4:24"/>
      <song title="Let Down" length="4:59"/>
      <song title="Karma Police" length="4:21"/>
      <song title="Fitter Happier" length="1:57"/>
      <song title="Electioneering" length="3:50"/>
      <song title="Climbing Up the Walls" length="4:45"/>
      <song title="No Surprises" length="3:48"/>
      <song title="Lucky" length="4:19"/>
      <song title="The Tourist" length="5:24"/>
      <description link="http://en.wikipedia.org/wiki/OK_Computer">
        OK Computer is the third studio album by the English alternative rock band Radiohead
      </description>
    </album>
  </artist>
```

```
//*[@2]
```

```
<valuepack>
  <values>
    <value>5</value>
    <value>3</value>
    <value>2</value>
  </values>
</valuepack>
```

# XPATH 1.0 W PRZYKŁADACH \*

- wzorzec dopasowujący się do wszystkich tagów to znak jokera "\*", z kolei wzorzec dopasowujący się do dowolnego atrybutu to "@\*"
- wzorzec "." dopasowuje się do bieżącego węzła, wzorzec ".." dopasowuje się do elementu rodzica
- warunek liczbowy np. [1] jest równoznaczny z [position()=1]

\* Przykłady zrealizowane przy użyciu strony: <http://xpather.com/>

# ISTOTNE INTERFEJSY

## IWEBELEMENT

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

IWebDriver driver = new ChromeDriver();
driver.Url = "https://google.com";
driver.FindElement(
    By.XPath("//*[@text()='Zaakceptuj wszystko']")
).Click();
driver.FindElement(By.TagName("input"))
    .SendKeys("keanu reeves bacon number" + Keys.Enter);
var xpathBaconPattern = "//*[contains(text(),'Bacon number is ')]";
var baconElement = driver.FindElement(By.XPath(xpathBaconPattern));
var baconText = baconElement.Text;
var baconPosition = baconText.IndexOf("Bacon number is ") + 16;
var baconNumber = baconText.Substring(baconPosition, 1);

Console.WriteLine($"Keanu Reeves Bacon number is {baconNumber}");
Console.ReadKey();
driver.Quit();
```

Interfejs IWebElement pozwala na odczytywanie własności poszczególnych elementów struktury DOM strony oraz na interakcje podobną do interakcji użytkownika z elementami strony. Dostępne funkcje interfejsu:

- **GetCssValue** – zwraca element określający styl obiektu
- **GetAttribute, GetDomAttribute** – zwraca wartość podanego w parametrze atrybutu
- **Text** – stanowi właściwość, za pomocą której można uzyskać ciąg znaków znajdujący się w ramach tagu elementu
- **Click** – uruchomienie funkcji imituje naciśnięcie elementu lewym przyciskiem myszy
- **SendKeys** – przekazujemy funkcji ciąg znaków które chcemy wysłać do danego elementu. Można tutaj skorzystać również z zestawu znaków zgromadzonych w klasie Keys.

# ISTOTNE INTERFEJSY

## IJAVASCRIPTEXECUTOR

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

IWebDriver driver = new ChromeDriver();
driver.Url = "https://stackoverflow.com";
IJavaScriptExecutor executor = (IJavaScriptExecutor)driver;
executor.ExecuteScript($"$('a:contains(\"Products\")')\" +
    ".text('My injected text')");

Console.ReadKey();
driver.Quit();
```

Interfejs IJavaScriptExecutor udostępnia możliwość wstrzykiwania kodu Javascript do istniejącej strony. Kontekst, w który wstrzykujemy kod, wynika z dostępnych w ramach strony skryptów tj. np. jeśli na stronie będzie załadowana biblioteka jquery będziemy mogli w ramach skryptu z niej korzystać. Do uruchamiania skryptu służy metoda ExecuteScript.

# ISTOTNE KLASY

## WEBDRIVERWAIT

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;

IWebDriver driver = new ChromeDriver();
driver.Url = "https://interactive-examples.mdn.mozilla.net/"
            + "pages/js/statement-async.html";
WebDriverWait wait = new WebDriverWait(driver,
    TimeSpan.FromSeconds(10));

driver.FindElement(By.Id("execute")).Click();

wait.Until(condition =>
    driver.FindElements(
        By.XPath("//code[contains(text(), 'resolved')]")
    ).Count > 0);
Console.WriteLine("Finally found resolved text!");

Console.ReadKey();
driver.Quit();
```

WebDriverWait jest odpowiedzią Selenium na asynchronicznie doładowywany контент do strony. Metoda Until obiektu klasy zamraża działanie bieżącej procedury aż do momentu spełnienia warunku dotyczącego kontentu strony albo przekroczenia czasu timeout'u podawanego w ramach konstruktora klasy ewentualnie później zmienionego przy użyciu właściwości Timeout. Weryfikowanie warunku odbywa się poprzez odpytywanie driver'a co określony właściwością PollingInterval (domyślnie 0.5 sekundy) czas.



# ISTOTNE KLASY

## ACTIONS

```
using OpenQA.Selenium;  
using OpenQA.Selenium.Chrome;  
using OpenQA.Selenium.Interactions;
```

```
WebDriver driver = new ChromeDriver();  
driver.Manage().Window.Maximize();  
driver.Url = "https://yaireo.github.io/tagify/";  
var tags = driver.FindElement(  
    By.XPath("//input[@name='input']/..")  
);  
tags.Click();
```

```
Actions actions = new Actions(driver);  
actions.Pause(TimeSpan.FromMilliseconds(500));  
actions.MoveToElement(tags);  
actions.SendKeys("C+");  
actions.Pause(TimeSpan.FromMilliseconds(1500));  
actions.SendKeys(Keys.ArrowDown);  
actions.Pause(TimeSpan.FromMilliseconds(500));  
actions.SendKeys(Keys.Enter);  
actions.Perform();
```

```
Console.ReadKey();  
driver.Quit();
```

Actions jest klasą służącą do ustalania kolejkwania i przebiegu czasowego wywoływanych zdarzeń. Za pomocą klasy jesteśmy w stanie tworzyć automatyczne przebiegi akcji w ramach aplikacji posiadającej szereg asynchronicznych wywołań. Najistotniejsze metody klasy:

- **Pause** – pozwala określić czas pomiędzy kolejnymi akcjami
- **MoveToElement** – pozwala zakolejkować przeniesienie focusu/kursora na określony w parametrze obiekt typu `IWebElement`
- **MoveByOffset** – pozwala zakolejkować akcję przeniesienia wirtualnego kursora myszy o zadany offset
- **SendKeys** – pozwala zakolejkować akcję wpisania określonej sekwencji klawiszy
- **Click** – pozwala zakolejkować akcję kliknięcia prawym przyciskiem myszy w określone wirtualnym kursorem miejsce
- **ClickAndHold** – pozwala zakolejkować akcję kliknięcia i przytrzymania prawego przycisku myszy w określonym wirtualnym kursorem miejscu
- ...