

AUTOMATYZACJA TESTÓW

APLIKACJI OKIENKOWYCH

Z UŻYCIEM UIAUTOMATION

PLAN PREZENTACJI

- O nas
 - O projekcie
 - Dlaczego UIAutomation
 - O UIAutomation ogólnie
 - O prostocie użycia
 - O elastyczności
 - Testowalny interfejs
 - Zdalne uruchamianie
 - Architektura i demo przykładowego projektu
 - Pytania
-

O NAS

Milena Sobolewska

sob.milena@gmail.com



- Ponad cztery lata doświadczenia w testowaniu oprogramowania, w tym dwa lata w automatyzacji
- C#, Specflow, Java, JMeter, Sikuli, TCL/Expect
- ISTQB Foundation
- Microsoft Programming in C#



Paweł Maciejewski

pawel@maciejewski.in

- Ponad dwa lata doświadczenia w testowaniu oprogramowania, głównie w automatyzacji
- C#, Specflow, Ruby, Capybara, RSpec
 - ISTQB Foundation
- Microsoft Programming in C#

O PROJEKCIE

- **Klient:** duży brytyjski bank inwestycyjny
 - **Projekt:** okienkowa aplikacja wewnętrzna do edycji i wizualizacji danych bankowych
 - **Zastane:** niskiej jakości framework oparty
 - o TestComplete wykonujący testy półautomatycznie
 - **Obecnie:** framework oparty o UIAutomation wraz z pełnym rozwiązaniem zdalnego wykonania
-

DLACZEGO UIAUTOMATION

- Standardowy element biblioteki .Net (od wersji 3.0)
 - Zaprojektowany w celu automatyzacji użycia interfejsów tworzonych przy użyciu WPF
 - Dodawany do projektu w postaci referencji (biblioteki)
 - Pozwala na pełną kontrolę kodu testów
 - **Prosty w użyciu i elastyczny**
-

PROSTY W UŻYCIU

Pulpit = AutomationElement.RootElement



Kalkulator = dziecko pulpitu o id "Calc"



PROSTY W UŻYCIU C.D.

```
AutomationElement MyDesktop =  
    AutomationElement.RootElement;  
  
AutomationElement Calc = MyDesktop.FindFirst(  
    TreeScope.Children,  
    new PropertyCondition(  
        AutomationElement.AutomationIdProperty,  
        "Calc"));
```

PROSTY W UŻYCIU C.D.

The screenshot displays the Visual UI Automation Verify application. In the center, a Windows calculator window titled "Kalkulator" is open, showing the number "0" on the display. The calculator's "przycisk" (button) "1" is highlighted with a red and blue crosshair. The background shows the "Automation Elements Tree" on the left and the "Properties" window on the right. The "Automation Elements Tree" lists various UI elements, with "przycisk" "1" "131" selected. The "Properties" window shows the following details for the selected element:

Identification	
AutomationId	131
ClassName	Button
ControlType	ControlType.Button
FrameworkId	Win32
hWnd	0xA07A0
IsContentElement	True
IsControlElement	True
IsPassword	False
LocalizedControlType	przycisk
Name	1
ProcessId	9024

Below the "Identification" section, the "Patterns" section is visible, with "InvokePattern" selected, showing the value "VisualUIAutomation.Verify.Features.Automation".

ELASTYCZNY

```
PropertyCondition TypeButton =  
    new PropertyCondition(  
        AutomationElement.ControlTypeProperty,  
        ControlType.Button);  
  
PropertyCondition ElName =  
    new PropertyCondition(  
        AutomationElement.NameProperty, "1");  
  
AutomationElement ButtonOne = Calc.FindFirst(  
    TreeScope.Children,  
    new AndCondition(TypeButton, ElName));
```

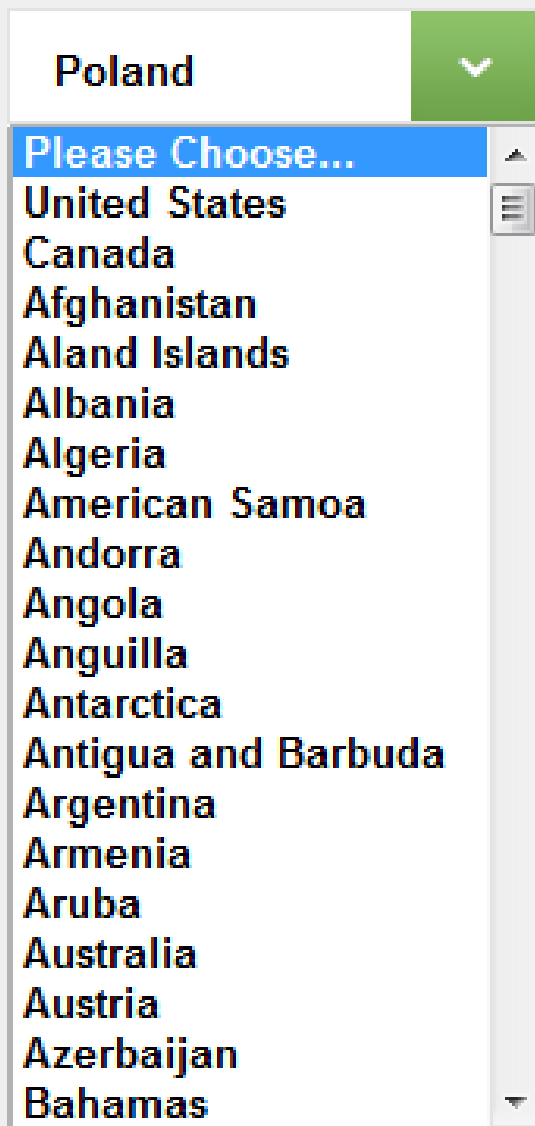
ELASTYCZNY C.D.

The image shows a screenshot of a test automation tool interface. On the left, a window titled "Kalkulator" (Calculator) is displayed. The calculator's display shows the number "2". A red horizontal line and blue vertical lines are drawn around the display area. On the right, a "Properties" window is open, showing a list of properties for the selected element. The "Identification" section is expanded, and the "Name" property is highlighted with a yellow box. The "Patterns" section is also expanded, and the "TextPattern" and "ValuePattern" properties are highlighted with yellow boxes.

Identification	
AutomationId	150
ClassName	Static
ControlType	ControlType.Text
FrameworkId	Win32
hWnd	0x100476
IsContentElement	False
IsControlElement	True
IsPassword	False
LocalizedControlType	tekst
Name	2
ProcessId	7596

Patterns	
DockPattern	
ExpandCollapsePattern	
GridPattern	
InvokePattern	
MultipleViewPattern	
RangeValuePattern	
ScrollItemPattern	
ScrollPattern	
SelectedItemPattern	
SelectionPattern	
TableItemPattern	
TablePattern	
TextPattern	
TogglePattern	
TransformPattern	
ValuePattern	
WindowPattern	

PROSTY + ELASTYCZNY



```
Condition AllListEls = new  
    PropertyCondition(  
        AutomationElement.ControlTypeProperty,  
        ControlType.ListItem);
```

```
Condition NotPleaseChoose = new  
    NotCondition(new  
        PropertyCondition(  
            AutomationElement.AutomationIdProperty,  
            "PleaseChoose"));
```

```
AutomationElementCollection Countries =  
    Calc.FindAll(TreeScope.Children, new  
        AndCondition(AllListEls, NotPleaseChoose));
```

TESTOWALNY INTERFEJS

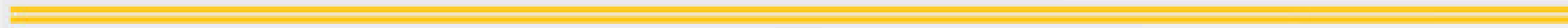
- Charakterystyka testowalnych automatycznie interfejsów
 - Dostępne właściwości oraz akcje dla elementów aplikacji
 - Niestandardowe metody pracy z trudno testowalnymi interfejsami aplikacji
-

ZDALNIE URUCHAMIANIE

- Uruchamianie testów poprzez **połączenie zdalnego pulpitu** i ograniczenia tego rozwiązania
 - Uruchamianie testów poprzez **serwis systemowy** i ograniczenia tego rozwiązania
 - Uruchamianie testów poprzez **połączenie zdalnego pulpitu i serwis systemowy**
-

ARCHITEKTURA I DEMO

PRZYKŁADOWEGO PROJEKTU



PYTANIA?

Kontakt do nas:

sob.milena@gmail.com

pawel@maciejewski.in

