Narzędzia do testowania

Wykorzystanie White UI oraz Selenium

Dominika Fusek, Michał Ocios

Agenda

- Omówienie narzędzia White UI
  - Co to jest
  - Co jest potrzebne do korzystania
  - Podstawowe operacje, które umożliwia narzędzie(np. podpięcie aplikacji, obsługa okna, obsługa elementów okna, kryteria wyszukiwania, obsługa klawiatury)

- Omówienie narzędzia Selenium
  - Co to jest
  - Przegląd narzędzi
  - Selenium IDE (instalacja, przykłady, wady i zalety)
  - Selenium WebDriver (uruchamianie sterownika, zamykanie aplikacji, przejeście do konkretnej strony, identyfikatory, przydatne metody, asercje)
- Page Object Pattern (co to jest?, po co to jest?, jak się tego używa?)

1 Narzędzie White UI

White UI - wstęp

Co to jest?

Narzędzie pozwalające automatyzować testowanie aplikacji desktopowych poprzez wywoływanie

https://github.com/TestStack/White

http://white.teststack.net/docs/getting-started

http://teststackwhite.readthedocs.org/en/latest/

Narzędzia niezbędne do korzastania z White UI

Aplikacja do przeglądania UI, taka jak UI Verify lub AutoIT, umożliwiająca pobieranie poszczególnych elemetów testowanie aplikacji

UI Verify: https://uiautomationverify.codeplex.com/releases/view/11366

AutoIT: https://www.autoitscript.com/site/autoit/downloads/

TestStack.White

Podpięcie aplikacji

- Uruchomienie

  Application app = Application.Launch(params); - Uruchamia podaną aplikację
- Podpięcie do procesu

  Application app = Application.Attach(params); - White łączy się z podanym procesem
- Uruchomienie lub podpięcie

  Application app = Application.AttachOrLaunch(params) – White uruchamia podaną aplikację po czym łączy się z procesem

```
//Przykład:
Application app = Application.AttachOrLaunch(new ProcessStartInfo("calc"));
```

CleanUp

- Wyłączenie aplikacji

  Application.Kill(); - zabija process aplikacji

  Application.Close(); - zamyka główne okno aplikacji

```
//Przykład:
app.Close();
```

TestStack.White – TestStack.White.UIItems

Obsługa okna

- Znalezienie okna w procesie

  Application. WindowItems.GetWindow(params) – zwraca okno o podanym tytule

  Application. WindowItems.GetWindows() – zwraca wszystkie okna aplikacji

```
//Przykład:
Window win = app.GetWindow("Calculator");
```

- Znalezienie okna dialogowego

  Window.ModalWindow(params) – zwraca modalne okno o podanym tytule lub spełniającym podane kryteria

  Window.ModalWindows() – zwraca wszystkie okna modalne

```
//Przykład:
Window modalWindow = win.ModalWinow(„Warning");
```

TestStack.White – TestStack.White.UIItems

Obsługa elementów okna

- Znalezienie danego elementu okna

  Window.Get<Typ>(SearchCriteria) – zwraca element okna danego typu (np. przycisk, listę itp.), która spełnia podane kryteria

```
//Przykład:
Button button = win.Get<Button>(SearchCriteria.ByAutomationId("btnOK"));
var checkbox = win.Get(SearchCriteria.ByText("Checkbox1"));
```

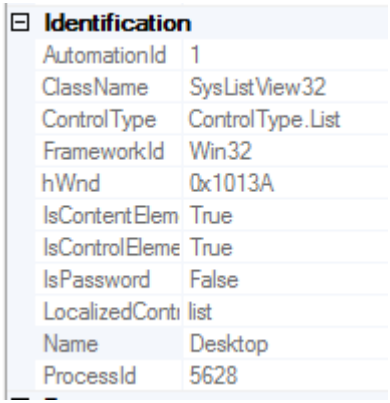
- Obsługa Menu
  - Window.MenuBars;
  - Window.MenuBar;

```
//Przykład:
MenuBar menuBar = win.MenuBar; - pobiera główne menu danego okna
Menu menu = menuBar.MenuItemBy(SearchCriteria.ByText("Menu name"); - pobiera dane menu, np.: File
Menu subMenu = menu.SubMenu(SearchCriteria.ByText("Submenu name"); pobiera pod-menu
```

TestStack.White – TestStack.White.UIItems

- Kryteria wyszukiwania elementów
  - (And)ByText()
  - (And)ByClassName()
  - (And)ByAutomationID()
  - (And)ByControlType()

```
Menu menu = menuBar.MenuItemBy(SearchCriteria.ByText("Menu name"); - pobiera dane menu, np.: File
Menu subMenu = menu.SubMenu(SearchCriteria.ByText("Submenu name"); pobiera pod-menu
```



- Typy UIItems – najczęściej używane typy:

UIA ControlType	White's UIItem
List	ListBox
DataGrid	ListView
Edit	TextBox
Text	Label
ComboBox	ComboBox
Button	Button
ListItem	ListItem

Więcej typów na:

http://teststackwhite.readthedocs.org/en/latest/UIItems/#controltype-to-uiitem-mapping-for-primary-uiitems

TestStack.White – TestStack.White.UIItems

Obsługa klawiatury

Keyboard.Instance.Enter(string) - Naciśniecie dowolnego klawisza

Keyboard.Instance.PressSpecialKey(SpecialKeys) - Naciśnięcie specjalnego klawisza

Keyboard.Instance.HoldKey(SpecialKeys) – Przytrzymanie klawisza

Keyboard.Instance.LeaveKey(SpecialKeys) - Zwolnienie klawisza

Przykłady niektórych klawiszy specjalnych:

KeyboardInput.SpecialKeys.ALT;

KeyboardInput.SpecialKeys.CONTROL;

KeyboardInput.SpecialKeys.RETURN;

KeyboardInput.SpecialKeys.SHIFT;

KeyboardInput.SpecialKeys.ESCAPE

```
//Przykład:
win.Keyboard.HoldKey(KeyboardInput.SpecialKeys.CONTROL);
win.Keyboard.Enter("c");
win.Keyboard.LeaveKey(KeyboardInput.SpecialKeys.CONTROL);
```

2 Narzędzie Selenium

Selenium- wstęp

Co to jest?

Narzędzie pozwalające automatyzować testowanie aplikacji internetowych poprzez wywoływanie, dzieli się na:

- Selenium IDE
- Selenium WebDriver

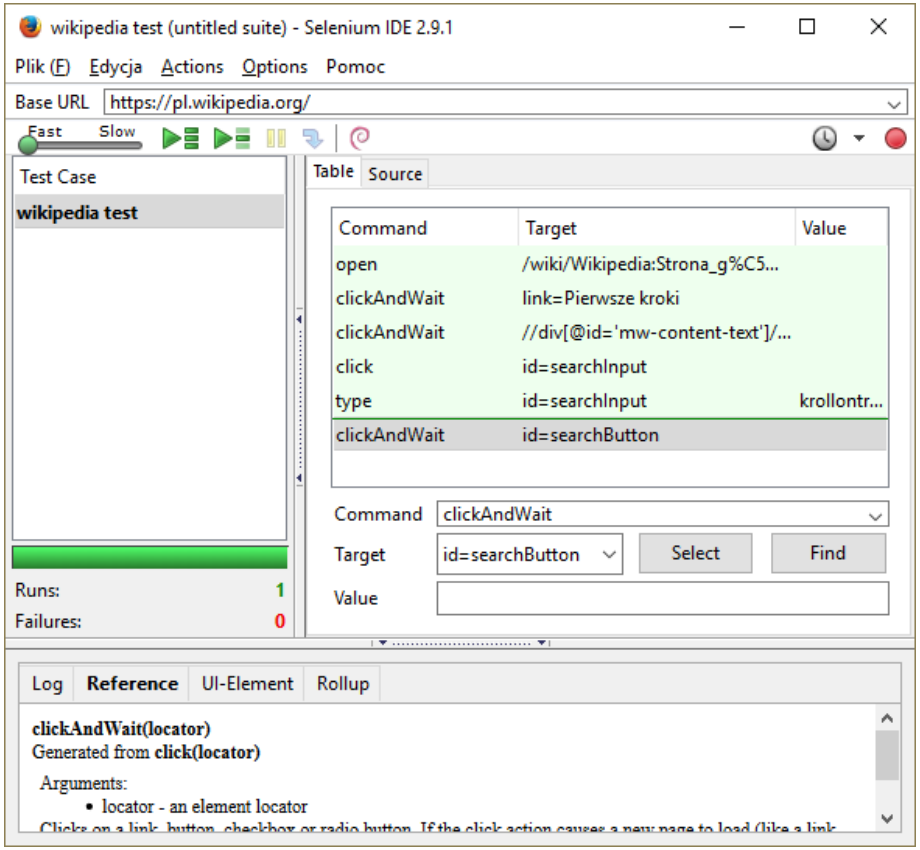
Narzędzia wspomagające korzastanie z Selenium

- Chrome Dev Tools
- Firebug
- IE developer tools

Selenium IDE – wprowadzenie do automatyzacji

Co to jest?

- Najprostrze narzędzie do automatyzacji testów
- Wtyczka do przegladarki Firefox
- Nagrywanie i odtwarzanie interakcji użytkownika z przegladarką



Selenium IDE – wady i zalety

Zalety	Wady
Narzędzie jest bezpłatne	Testy można odtwarzać tylko na przeglądarce Firefox
Łatwa obsługa, nie trzeba posiadać umiejętności programowania	Konieczność przerabiania/nagrywania testów od nowa przy najdrobniejszej zmianie funkcjonalności
Akcje wykonywanie przez użytkownika są automatycznie nagrywane	
Możliwość rozszerzenia o dodatkowe komponenty (TestResults – zapis przebiegu testów, Selenium IDE: flow Control – dodawanie prostych instrukcji sterujących, File Logging – zapis logów, ScreenShot on Fail)	
Pozwala tworzyć Test Case i Test Suite	

Selenium IDE – instalacja

- Wchodzimy na stronę: http://docs.seleniumhq.org/download/
- Znajdujemy aktualną wersję Selenium IDE i ją instalujemy
- Po zainstalowaniu Selenium IDE staje się dostępne w postaci wtyczki Firefox'a
- Uruchamiamy ponowanie przeglądarkę Firefox i uruchamiamy Selenium IDE
- Instalujemy Firebuga

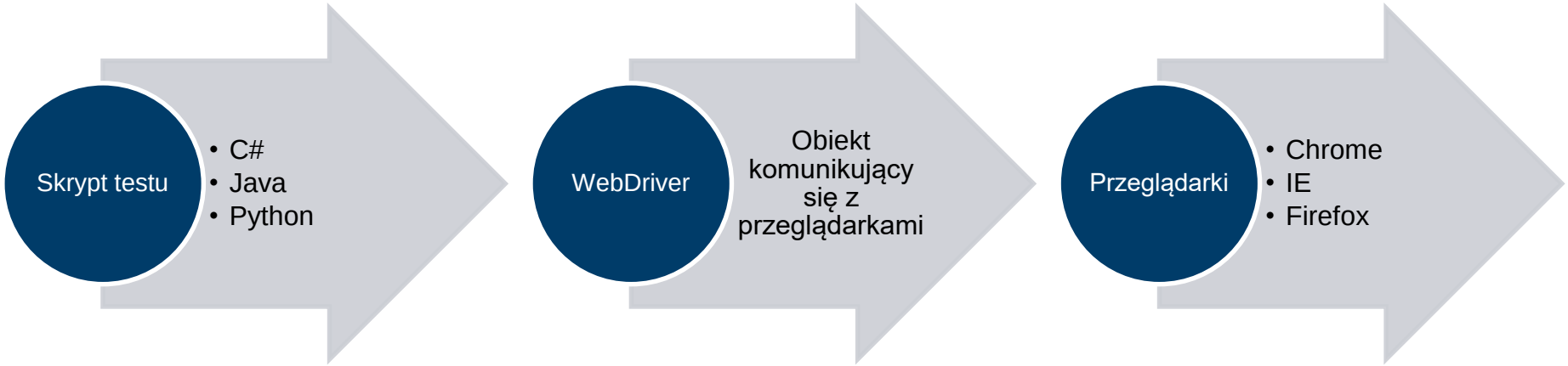
Selenium IDE – identyfikatory

- ID
- Name
- Link Text
- CSS Selector
  - Tag and ID
  - Tag and class
  - Tag and attribute
- Tag, class, and attribute
- Inner text

- DOM (Document Object Model)
  - getElementById
  - getElementsByName
  - dom:name
- dom: index
- XPath

Selenium WebDriver

Narzędzie bardzo wygodne dla programisty API pozwalające na interakcję z przeglądarką



Selenium WebDriver

Inicjalizacja sterownika

```
//Przykład
public IWebDriver driver;
driver = new FirefoxDriver();
```

Wyłączenie aplikacji

```
//Przykład:
        driver.Quit();
        driver.Dispose();
```

Przejście do adresu strony

```
//Przykład:
         driver.Navigate().GoToUrl(adres strony);.
```

Selenium WebDriver– identyfikatory

- ID
- Name
- LinkText
- CSSSelector
  - Tag and ID
  - Tag and class
  - Tag and attribute
- TagName,
- ClassName,
- XPath
  - *XPath("//div[@id='p-personal']/ul/li[5]")*

- z klasy IWebDriver
  - Title
  - PageSource
  - Url

Selenium WebDriver– przydatne metody

- FindElement();

  FindElement(By.Id("searchInput”));

- Clear();

- SendKeys();

  SendKeys(„coś tam");

- Click();

  FindElement(By.Id("searchButton")).Click();

Selenium WebDriver– asercje

- **Assert**
  - AreEqual()

    Assert.AreEqual(wartość_oczekiwana, wartość_pobrana);
  - IsTrue()

    Assert.IsTrue(driver.PageSource.Contains(„coś tam"));
  - IsFalse()
- **StringAssert**
  - Contains()

    StringAssert.Contains(wartość_oczekiwana, wartość_pobrana);

3 Page Object Pattern

Page Object Pattern

definicja

jest to wzorzec projektowy skupiony głównie na obsłudze interfejsu. Polega on na przypisaniu to każdej strony w aplikacji osobnego obiektu, który będzie odpowiedzialny za obsługę wszystkich funkcji danej strony. Z tych obiektów korzysta się przy pisaniu testów automatycznych.

po co?

Używa się głównie przy pisaniu testów, żeby zwiększyć ich czytelność oraz zmniejszyć ilość powtórzeń tego samego kodu. Co więcej jeżeli zmieni się sposób uzyskania efektu końcowego funkcji na danej stronie, wystarczy zaktualizować obiekt odpowiedzialny za jej obsługę, a nie wszystkie testy dotyczące tej strony.

Page Object Pattern

jak?

Dla każdej strony tworzy się osobną klasę. Klasa ta zawiera metody odpowiedzialne za uruchamianie wszystkich możliwych funkcji na obsługiwanej stronie. Dodatkowo klasa ta posiada również metody sprawdzające stan obsługiwanej strony, dzięki temu wszelkie assercje w testach wystarczy zapisywać jako: ***Assert.IsTrue(obiektStrony.SprawdzStan());***

Dana klasa zajmuje się tylko obsługą możliwości danej strony, nie powinna natomiast zajmować się obsługą części wewnętrznej (np. WebDrivera dla aplikacji Webowych).

Dana klasa nie musi odzwierciedlać całej strony, może być również odpowiedzialna za sekcję która powtarza się na wielu stronach w aplikacji.

Page Object Pattern

podsumowanie

- Publiczne metody obiektu przedstawiają możliwości danej strony
- Klasa nie musi zawsze odzwierciedlać całej strony
- Metody mogą zwracać inne obiekty
- Różne rezultaty dla tej samej akcji są obsłużone przez różne metody tego samego obiektu

Przydatne linki

White UI

https://github.com/TestStack/White

http://teststackwhite.readthedocs.org/en/latest/

https://uiautomationverify.codeplex.com/releases/view/11366

Selenium

https://www.kainos.pl/blog/pierwsze-kroki-z-selenium-%E2%80%93-selenium-ide/

http://learnseleniumtesting.com/basic-webdriver-and-c-sharp-script-for-beginners/

Page Object Pattern

http://martinfowler.com/bliki/PageObject.html

http://www.assertselenium.com/automation-design-practices/page-object-pattern/

http://docs.seleniumhq.org/docs/06_test_design_considerations.jsp#page-object-design-pattern

Q&A

Laboratorium

https://github.com/DosiaKroll/UsClass