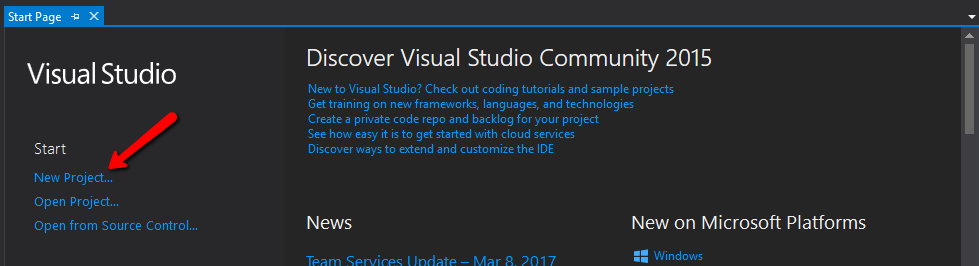
Selenium Tutorial

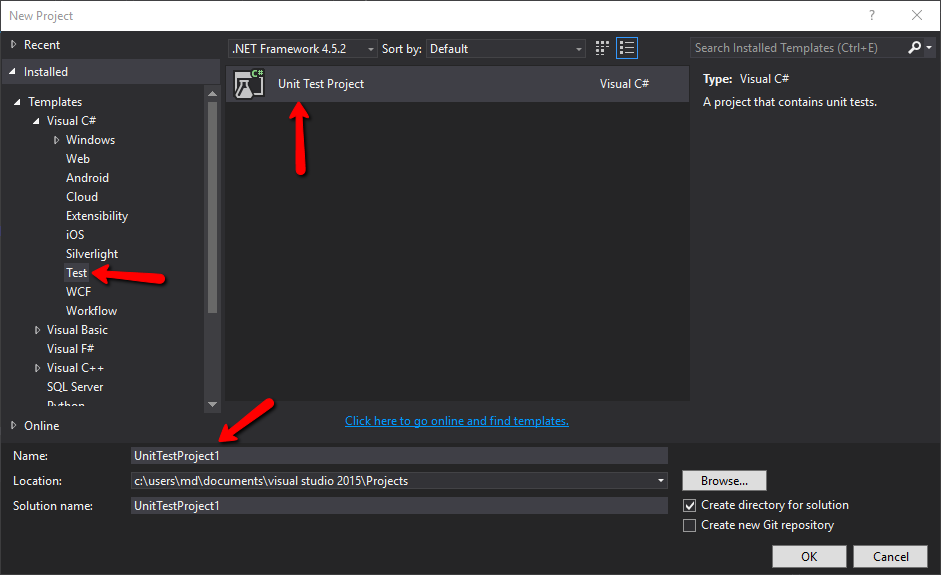
# Pierwszy projekt testowy

Wybieramy New project:

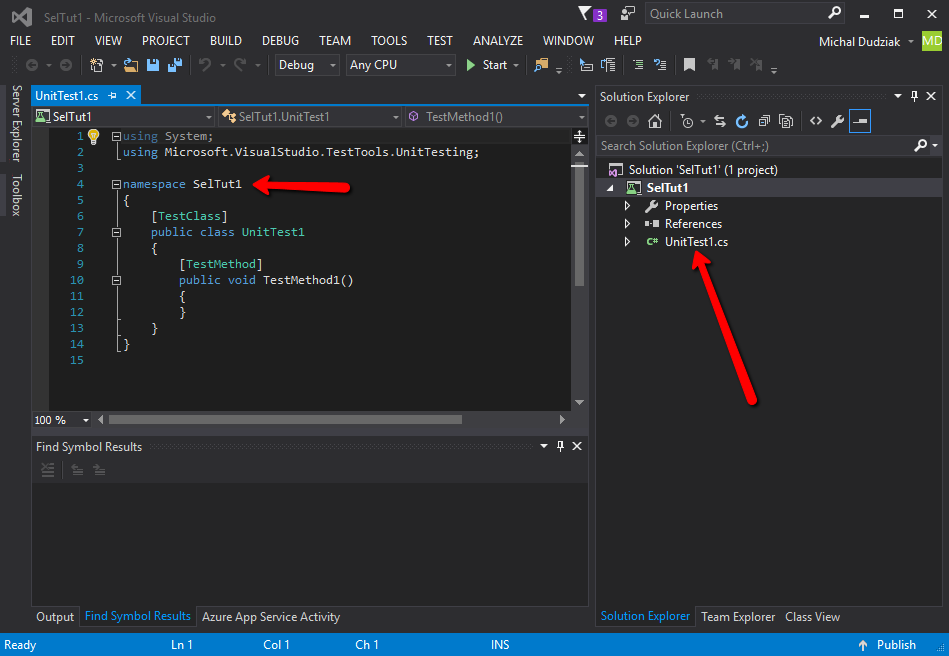


Test -> Unit Test Project

Wprowadzamy nową nazwę i klikamy OK



Mamy gotowy projekt testowy. Strzałki pokazują nazwę projektu oraz pierwszy domyślny plik testowy.

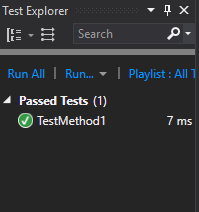


Wybierając z menu kolejno:

Test -> Run -> All tests

Możemy uruchomić pełny zestaw testów – w tym przypadku wystartuje oczywiście tylko jeden test utworzony automatycznie razem z projektem.

Po lewej stronie okna widoczny będzie rezultat uruchomienia testów:



Sekcja Passed Tests pokazuje ile testów zakończyło się sukcesem (1). Poniżej nazwy sekcji znajduje się lista metod testowych, które przebiegły pomyślnie.

## Prosty test weryfikacyjny

W celu sprawdzenia prawidłowego działania naszego domyślnego testu (aktualnie pustego), dopiszmy krótki kod.

Stworzenie dwóch zmiennych oraz obliczenie ich sumy:

int x = 5;

int y = 10;

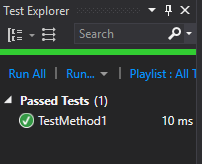
int sum = x + y;

Zweryfikowanie, czy wynik jest równy 15:

Assert.AreEqual(15, sum);

Ponownie uruchamiamy nasz test. Tym razem z przybornika Test Explorer wybieramy polecenie Run All.

W tym przypadku test potrwał nieco dłużej niż poprzednio, jednak wynik pozostaje cały czas ten sam:



Spróbujmy nieco namieszać w treści naszego kodu. Wprowadzając jedną małą zmianę, tak jak poniżej:

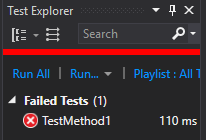
int x = 5;

int y = 10;

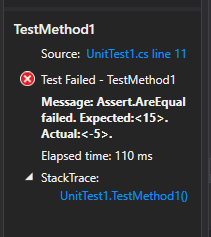
int sum = x - y;

Oczekujemy, że wynik obliczenia nie będzie tym razem równy 15. Jednak chwilowo pozostawiamy tę wartość w naszej asercji.

Uruchamiamy ponownie test (Run All) i widzimy następujący wynik:



Oczywiście w tym przypadku łatwo domyślić się, dlaczego test się nie powiódł, jednak w trudniejszych sytuacjach będziemy musieli skorzystać z podpowiedzi. Po lewej stronie w sekcji Test Explorer przy nazwie testu wyświetla się informacja, co poszło nie tak. Jeśli jest ona ukryta, należy kliknąć na nazwę testu:



Wiadomość (Message) pokazuje, iż nasza weryfikacja (Assert) oczekuje (Expected) wartości 15, jednak otrzymuje zupełnie inną (Actual), mianowicie -5.

# Instalacja i konfiguracja Selenium

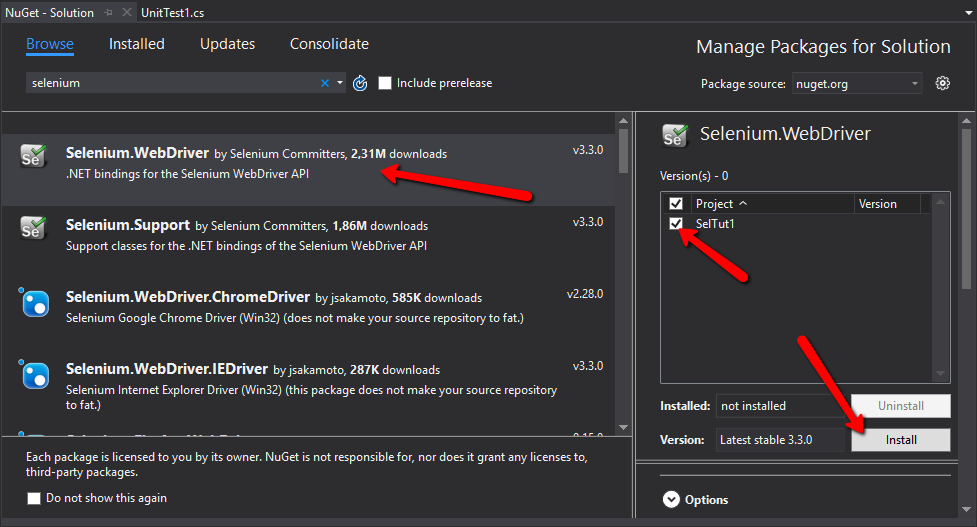
Instalacja i konfiguracja pod VS 2015

Klikamy prawym przyciskiem na projekt albo solucję i wybieramy Manage Nuget Packages.

W sekcji Browse wpisujemy pakiety jakie chcemy dodać. W naszym przypadku będą to następujące pakiety:

1. NUnit
2. NUnit3TestAdapter
3. Selenium.WebDriver
4. Selenium.WebDriver.ChromeDriver
5. Selenium.Support

Po wyszukaniu pakietu klikamy Install aby dodać go do projektu:



VS może zapytać nas, czy faktycznie chcemy dokonać takich zmian. Klikamy OK i czekamy, aż instalacji zostanie zakończona. Możemy po tym przejść do wyszukania pozostałych pakietów i po kolei dodawać je do projektu testowego.

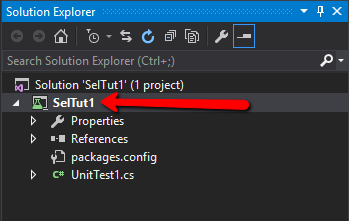
## Prosty test weryfikacyjny

W celu sprawdzenia prawidłowego działania dodanych rozszerzeń, dopiszmy krótki kod.

### Nowy plik

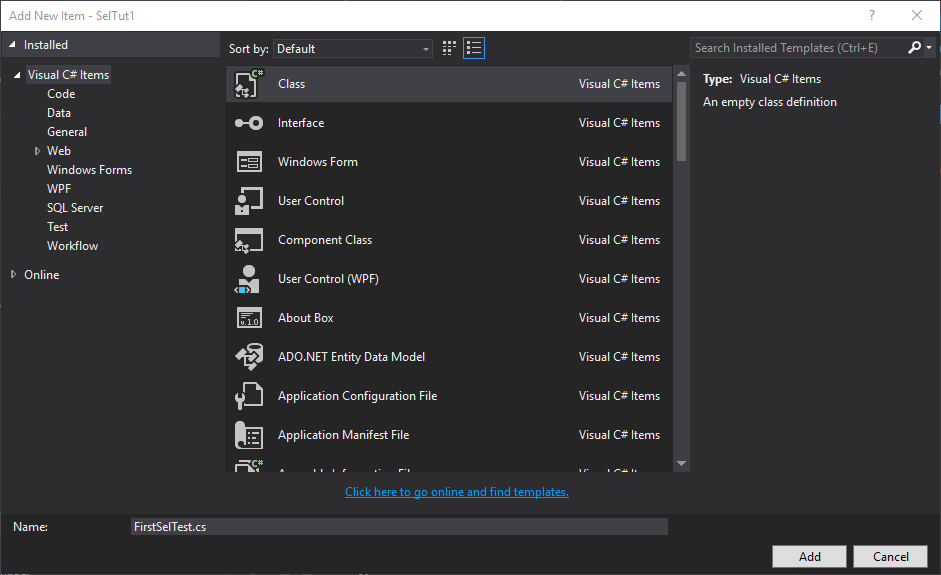
Zacznijmy od dodania nowego, oddzielnego pliku, żeby wyodrębnić testy wykonane przy pomocy Selenium, od tych, które nie bazują na przeglądarce, czyli nie wymagają dostępu do interfejsu użytkownika.

W sekcji Solution Explorer klikamy prawym przyciskiem myszy na nazwę projektu:

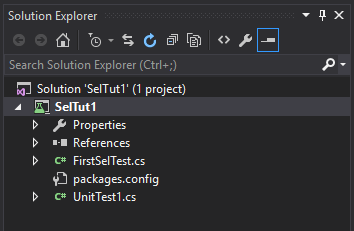


Wybieramy Add -> New Item

Wybieramy Class, wprowadzamy nową nazwę i klikamy Add:



Za chwilę możemy zobaczyć, że plik został dodany zgodnie z naszymi oczekiwaniami:



Zawartość takiego pliku jest następująca:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SelTut1

{

class FirstSelTest

{

}

}

Na razie nie znajdziemy tutaj nic ciekawego, większość pracy dopiero przed nami.

### Wnętrze testu

Dodajmy pierwszą metodę stanowiącą nowy test. W treści klasy FirstSelTest należy dopisać:

[Test]

public void FirstTest()

{

}

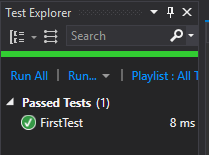
Visual Studio prawdopodobnie podpowie nam, iż w celu skorzystania z flagi [Test] należy dołączyć NUnit. W przypadku, gdy tego nie zrobi, albo zapomnimy kliknąć i dodać, należy dopisać na samym początku pliku:

using NUnit.Framework;

Tak samo jak w poprzednim rozdziale, uruchamiany utworzony test:

Test -> Run -> All Tests

Widać, że test przez nas utworzony kończy się wynikiem pozytywnym:



### Więcej Selenium

Teraz konfigurujemy już to, co najważniejsze, czyli połączenie z Selenium WebDriver.

Na początku pliku dodajemy dwie linijki:

using OpenQA.Selenium;

using OpenQA.Selenium.Chrome;

które wskazują na użycie przeglądarki Chrome do naszych dalszych testów. Oczywiście, można zamiast niej wykorzystać inną dowolną, na przykład FireFox albo IE. Na razie pozostańmy jednak przy Chrome.

W treści uzupełniamy już konkretny przebieg testu:

1. Zainicjowanie przeglądarki (driver)  
   IWebDriver driver = new ChromeDriver();
2. Wczytanie konkretnego adresu www  
   driver.Navigate().GoToUrl("https://pl.wikipedia.org/");
3. Zamknięcie przeglądarki  
   driver.Quit();

Po wprowadzeniu tych wszystkich danych, nasza metoda testowa wygląda następująco:

class FirstSelTest

{

[Test]

public void FirstTest()

{

IWebDriver driver = new ChromeDriver();

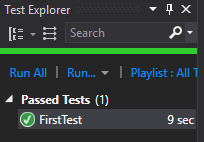
driver.Navigate().GoToUrl("https://pl.wikipedia.org/");

driver.Quit();

}

}

Warto już teraz uruchomić taki test, aby zobaczyć, czy działa poprawnie. Im szybciej wychwycimy wszelkie problemy, tym lepiej.



Test przeszedł prawidłowo. Powinniśmy zobaczyć uruchomioną przeglądarkę Chrome oraz przez chwilę załadowaną stronę Wikipedii. Po zamknięciu przeglądarki kończy się także nasz test.

To co teraz zostało nam do zrobienia to dodanie polecenia sprawdzającego, czy na pewno załadowała się strona, której oczekujemy. Aktualnie możemy to zrobić jedynie naocznie. W przypadku gdy wyświetli się strona inna niż Wikipedia, nasz test tego nie pokaże, nadal wynik pozostanie pozytywny. Musimy w takim razie sprawdzić jakiś element na stronie, wskazujący na tę konkretną stronę. Sposób na zrobienie tego zostanie pokazany w kolejnym rozdziale.

# Lokalizatory

Istnieją różnego rodzaju selektory, czyli punkty zaczepienia, po których możemy znajdywać elementy na stronie internetowej, które chcemy w jakiś sposób wykorzystać (kliknąć, wpisać tekst, wybrać jedną z opcji).

Lista selektorów:

* Id
* Name
* TagName
* CssSelector
* LinkText
* PartialLinkText
* XPath
* ClassName

## Tekst na stronie

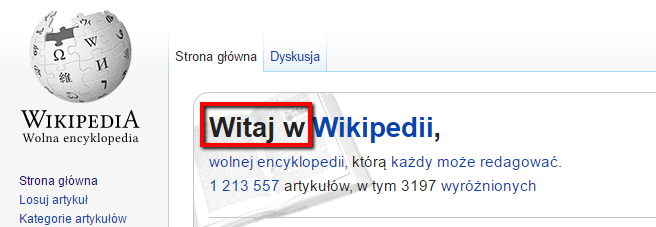
Jednym ze sposobów sprawdzenia, czy strona załadowała się prawidłowo, jest weryfikacja wyświetlenia konkretnego tekstu.

W tym celu musimy wykonać następujące kroki:

1. Wyszukać na stronie interesujący nas element
2. Pobrać tekst w nim zawarty
3. Sprawdzić czy ten tekst zawiera określony przez nas wzorzec

### Szukanie elementu

Znalezienie elementu możemy zrealizować na kilka sposobów. Na początku spróbujmy odszukać ID dla interesującego nas elementu, na przykład:



Podglądając budowę strony internetowej realizujemy na przykład poprzez kliknięcie prawym przyciskiem myszy na interesujący nas element i wybór opcji Zbadaj.

Wyświetlone zostanie dodatkowe okno z następującą zawartością:



Na szaro podświetlona została sekcja, która najbardziej nas interesuje. Widać też, że zawiera ona ID, które możemy łatwo wykorzystać

Teraz musimy poprosić przeglądarkę o znalezienie elementu bazując na ID pola:

driver.FindElement(By.Id("main-page-welcome"))

Polecenie takie szuka oraz wskazuje element na stronie www.

### Weryfikacja zgodności ze wzorcem

Po znalezieniu elementu na stronie, musimy pobrać tekst w nim zawarty. Wystarczy w tym celu dodać .Text do polecenia opisanego powyżej.

driver.FindElement(By.Id("main-page-welcome")).Text;

Polecenie to zwraca napis, a więc pozostaje nam tylko przypisać je do jakiejś zmiennej:

string welcome = driver.FindElement(By.Id("main-page-welcome")).Text;

Dzięki temu zmienna welcome zawiera teraz napis pobrany z elementu.

# Narzędzia

Wszystkie narzędzia przedstawione wcześniej są dostępne w wersji darmowej bądź community, do zastosowania niekomercyjnego.

W przypadku posiadania aplikacji w wersji komercyjnej, na przykład dla Visual Studio, wszystkie działania i komendy zawarte w tym dokumencie powinny działać identycznie.

## Visual Studio 2017

W wersji community do pobrania ze strony:

<https://www.visualstudio.com/thank-you-downloading-visual-studio/?sku=Community&rel=15>

# TODO:

1. Page Object Pattern
2. BDD