# Specyfikacja zewnętrzna

## *1. Wymagania*

Do prawidłowego działania aplikacji wymagane są:

* Przeglądarka Mozilla Firefox (wersja 61 lub nowsza)
* Java 1.8.0\_181 lub nowsza
* Program Microsoft Excel, lub inny program mogący edytować pliki .xls (do tworzenia skryptów)

## *2. Skróty klawiszowe*

Aplikacja obsługuje kilka standardowych skrótów klawiszowych w celu ułatwienia pracy:

* CTRL + N - tworzy nowy scenariusz
* CTRL + O - otwiera scenariusz
* CTRL + S - zapisuje obecnie wybrany scenariusz
* CTRL + R - uruchamia obecnie wybrany scenariusz

## *3. Dane wejściowe*

Program przyjmuje na wejście trzy rodzaje danych:

* **Scenariusz** specyfikujący, które przypadki testowe powinny być przetestowane, z jakimi danymi i w jakiej kolejności.
* **Przypadki testowe** (ang. *Test cases*) w postaci skryptów, które program interpretuje i przekłada na ciąg czynności wykonywanych w przeglądarce.
* **Dane testowe** zawierające zmienne w postaci „klucz-wartość”, z których korzystają następnie przypadki testowe. Jeden plik może zawierać wiele niezależnych zestawów danych.

Wszystkie dane podawane są w postaci plików .xls, oprócz scenariusza, który jest dokumentem XML.

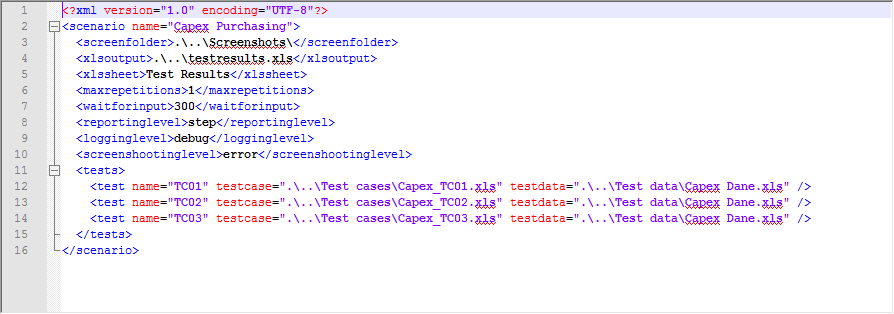
### Scenariusz

Scenariusz przechowywany jest przez program w postaci pliku XML. Oprócz ustawień dotyczących uruchamiania scenariusza, takich jak:

* Ścieżka do folderu ze zrzutami ekranu
* Ścieżkę do pliku wyjściowego z raportem
* Ilość powtórzeń komendy w przypadku wystąpienia błędu
* Czas oczekiwania pomiędzy poszczególnymi komendami
* reportingLevel, loggingLevel, screenshootingLevel (patrz *Dane wyjściowe*)

plik zawiera także listę przypadków testowych, które mają być przetestowane w ramach scenariusza, wraz z zestawami danych. Dany przypadek może być przetestowany wiele razy, za każdym razem z innymi (bądź tymi samymi) danymi. Podobnie ten sam zestaw danych może być wykorzystany wiele razy, w różnych przypadkach testowych.

Jeśli kilka przypadków testowych korzysta z tego samego zestawu danych, to jest on uznawany za współdzielony przez owe przypadki – tj. jeśli pierwszy przypadek ustawi wartość pewnej zmiennej w zestawie danych, kolejny przypadek będzie mógł operować na zmodyfikowanej wartości. Pozwala to na przesyłanie rezultatów pomiędzy poszczególnymi testami.



*Rys. 1 – przykład pliku scenariusza*

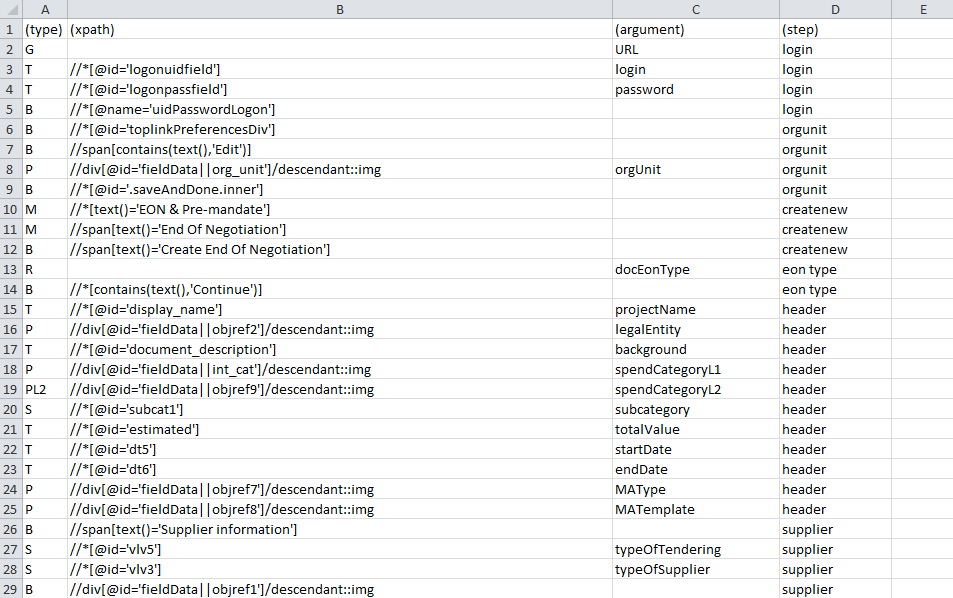
### Przypadek testowy

Skrypt dla pojedynczego przypadku testowego składa się z serii poleceń, z czego każde polecenie ma postać:

type <xpath> <argument> step

Objaśnienie poszczególnych kolumn w skrypcie:

* **type** – komenda, oznaczająca czynność do wykonania przez przeglądarkę. Typ komendy określa też ile i jakich argumentów powinno być podane. Do komend zaliczają się między innymi B (button), T (textbox) i G (go to).
* **xpath** – lokator Xpath wskazujący element na stronie, na którym operuje komenda. Tylko komendy operujące na konkretnym elemencie wymagają podania lokatora.
* **argument** – argument do komendy; zazwyczaj fraza wpisywana lub wyszukiwana na stronie. Może być podawany w postaci zmiennej (*nazwa*) lub stałej (*$wartość*). Tylko niektóre komendy wymagają podania argumentu.
* **step** – krok, w którym wykonana ma być dana komenda. Podział skryptu na kroki jest opcjonalny, ale zalecany, gdyż pozwala na dokładniejsze raportowanie wyników testu.

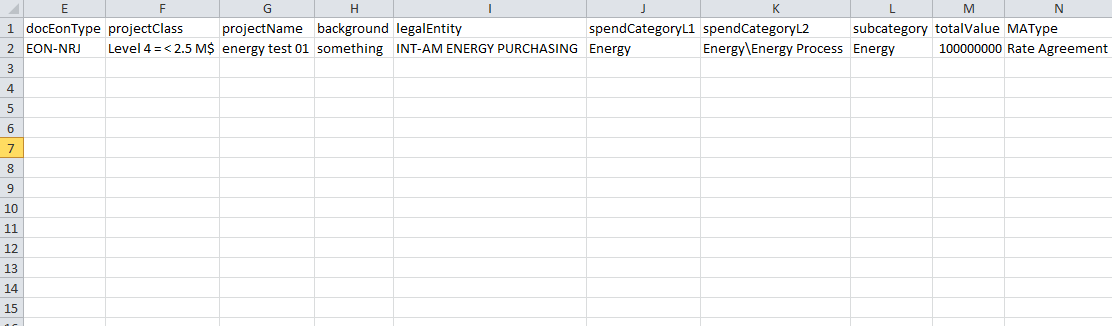


*Rys. 2 – przykład przypadku (skryptu) testowego*

### Dane testowe

Dane do testów podawane są w postaci tabeli, w której pierwszy wiersz zawiera nazwy poszczególnych zmiennych, a każdy kolejny ich wartości. W związku z tym, każdy wiersz poza pierwszym traktowany jest jak osobny zestaw danych.

Nie wszystkie komórki w pliku muszą być wypełnione – puste komórki zostaną zignorowane przez program przy odczycie zestawu danych.



*Rys. 3 – przykładowy plik zawierający pojedynczy zestaw danych*

## *4. Dane wyjściowe*

Program generuje na wyjściu trzy rodzaje informacji: tekstowy log, raport w postaci arkusza .xls, oraz zrzuty ekranu.

Częstotliwość zapisywania informacji wynika z ustawionego poziomu raportowania, reprezentowanego przez trzy zmienne: *reportingLevel*, *loggingLevel* i *screenshootingLevel*. Pierwsza i druga określają odpowiednio częstotliwość zapisów do raportu i loga. Ostatnia zaś określa momenty w których mają być wykonywane zrzuty ekranu (zrzuty wykonują się też zawsze po wystąpieniu komendy SCR w skrypcie testowym).

Istnieje 7 poziomów raportowania. W większości przypadków, poziom niższy zawiera też w sobie możliwości poziomów wyższych (np. loggingLevel = 3 wykonuje wpisy do raportu zarówno na koniec scenariusza, jak i po błędzie). Poziomy to, kolejno:

* DEBUG (0) - wpisy wykonywane po każdym poleceniu.
* STEP (1) – wpisy wykonywane po każdym kroku.
* TEST (2) – wpisy wykonywane po każdym przypadku testowym
* SCENARIO (3) - wpisy wykonywane na koniec każdego scenariusza.
* ERROR (4) - wpisy wykonywane tylko przy wystąpieniu błędu.
* SCR (5) – wpisy do loga i raportu wyłączone; zrzuty ekranu wyłącznie przy poleceniu SCR.
* NEVER (6) – raportowanie całkowicie wyłączone; ignoruje polecenia SCR

Zmienne loggingLevel, reportingLevel i screenshootingLevel mogą być ustawiane niezależnie od siebie, tak więc, na przykład, przy wartościach:  
loggingLevel = 0;  
reportingLevel = 1;  
screenshootingLevel = 4;

Wpisy do loga będą wykonywały się po każdym poleceniu, kroku, teście i scenariuszu, a także przy wystąpieniu błędu, wpisy do raportu po każdym kroku, a zrzuty ekranu tylko przy błędach i komendach SCR.

*Tab. 1 – zapisy do pliku .log w zależności od wartości zmiennej loggingLevel*

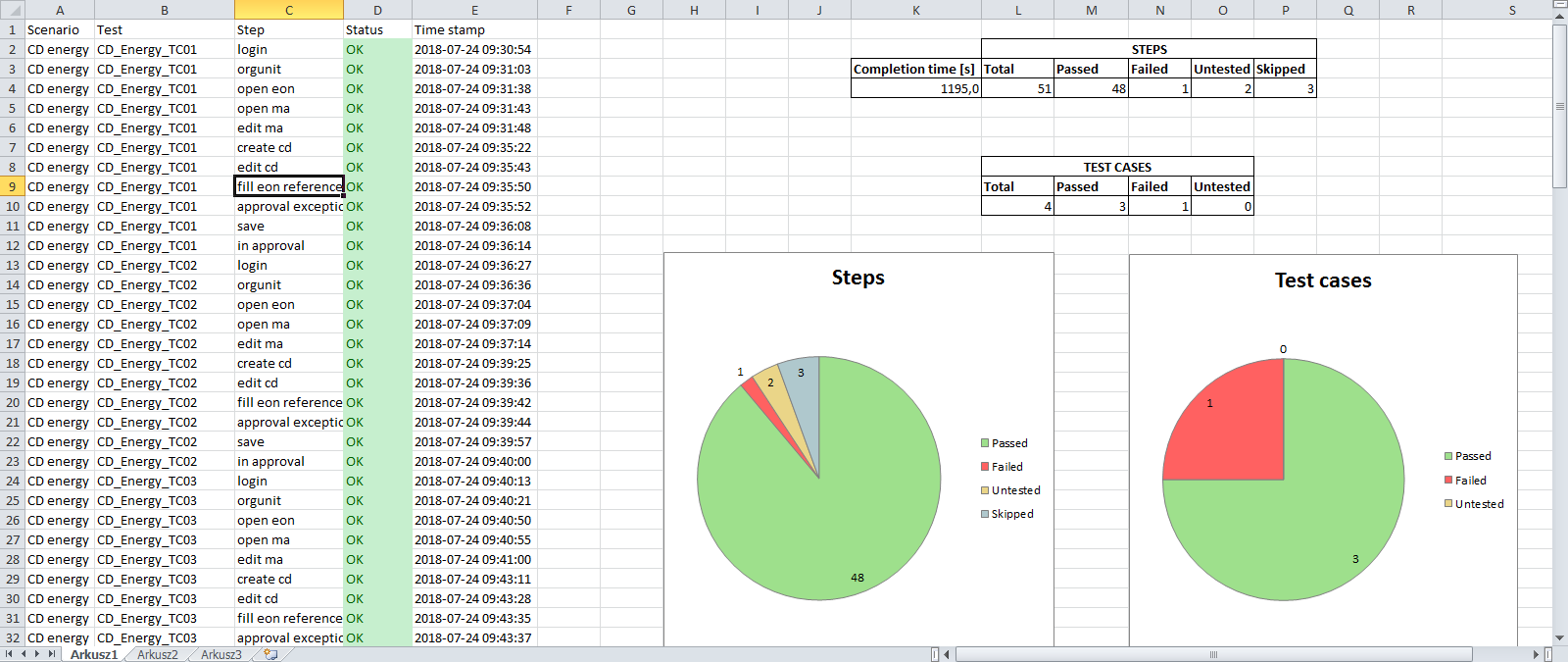
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **loggingLevel** | **Zapisy do loga** | | | | |
| **Po komendzie** | **Po kroku** | **Po teście** | **Po scenariuszu** | **Przy błędzie** |
| DEBUG |  |  |  |  |  |
| STEP |  |  |  |  |  |
| TEST |  |  |  |  |  |
| SCENARIO |  |  |  |  |  |
| ERROR |  |  |  |  |  |
| SCR |  |  |  |  |  |
| NEVER |  |  |  |  |  |

*Tab. 2 – zapisy do pliku .xls w zależności od wartości zmiennej reportingLevel*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **reportingLevel** | **Zapisy do raportu** | | | | |
| **Po komendzie** | **Po kroku** | **Po teście** | **Po scenariuszu** | **Przy błędzie** |
| DEBUG |  |  |  |  |  |
| STEP |  |  |  |  |  |
| TEST |  |  |  |  |  |
| SCENARIO |  |  |  |  |  |
| ERROR |  |  |  |  |  |
| SCR |  |  |  |  |  |
| NEVER |  |  |  |  |  |

*Tab. 3 – zrzuty ekranu w zależności od wartości zmiennej screenshootingLevel*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **screensho-**  **otingLevel** | **Zrzuty ekranu** | | | | | |
| **Po komendzie SCR** | **Po komendzie innej niż SCR** | **Po kroku** | **Po teście** | **Po scenariuszu** | **Przy błędzie** |
| DEBUG |  |  |  |  |  |  |
| STEP |  |  |  |  |  |  |
| TEST |  |  |  |  |  |  |
| SCENARIO |  |  |  |  |  |  |
| ERROR |  |  |  |  |  |  |
| SCR |  |  |  |  |  |  |
| NEVER |  |  |  |  |  |  |

*Rys. 4 – przykładowy raport wygenerowany przez program, a następnie sformatowany w aplikacji Microsoft Excel*

## *5. Tworzenie plików skryptów*

Skrypty można tworzyć w dowolnej aplikacji pozwalającej na edycję plików .xls. Plik należy wypełnić komendami zapisywanymi w kolejnych wierszach arkusza, jedna pod drugą (pierwszy wiersz pozostawiając pusty lub wypełniony nagłówkami).

Każda komenda ma swój skrótowy kod, który należy umieścić w pierwszej kolumnie arkusza ze skryptem. W zależności od komendy, może być też konieczne wypełnienie dwóch kolejnych kolumn (xpath i argument).

### Lista komend dostępnych w programie

*Tab. 4 – komendy dostępne w aplikacji*

|  |  |  |  |
| --- | --- | --- | --- |
| Kod | Objaśnienie | Parametry | Efekt |
| A | Alert | - | Zatwierdza alert (wyskakujące okienko) na stronie. Jeśli nie ma żadnego wyskakującego okna, test zakończy się niepowodzeniem. |
| B | Button | Xpath | Klika w element znajdujący się pod podanym lokatorem Xpath. |
| BTW | Back To Window | - | Wraca z powrotem do oryginalnego okna w przeglądarce. Komendy tej należy używać tylko po przejściu do innego okna (patrz komenda NW); użyta w innej sytuacji nie będzie miała żadnego efektu. |
| CMP | Compare Text | Xpath, Argument | Porównuje tekst elementu pod danym lokatorem Xpath z wartością argumentu. Jeśli są one **różne**, test kończy się niepowodzeniem. |
| DAT | Get DateTime | Argument | Zapisuje do zmiennej o nazwie podanej w argumencie aktualną datę i godzinę (w formacie yyyy-MM-dd HH\_mm\_ss). Jeśli zmienna o podanej nazwie istnieje, jej wartość zostanie nadpisana. Jeśli nie istnieje, zostanie ona utworzona. Jako argument do DAT nie można podać stałej. |
| EE | Element Exists | Xpath | Instrukcja warunkowa. Jeśli element pod podanym lokatorem Xpath **nie istnieje**, test kończy się niepowodzeniem. |
| ENE | Element Not Exists | Xpath | Instrukcja warunkowa. Jeśli element pod podanym lokatorem Xpath **istnieje**, test kończy się niepowodzeniem. |
| G | Go To URL | Argument | Przechodzi pod adres URL podany w argumencie. Adres może być zmienną lub stałą. |
| GET | Get Text | Xpath, argument | Pobiera tekst z elementu pod podanym lokatorem Xpath i zapisuje go do zmiennej o nazwie podanej w argumencie. Jeśli zmienna o podanej nazwie istnieje, jej wartość zostanie nadpisana. Jeśli nie istnieje, zostanie ona utworzona. Jako argument do GET nie można podać stałej. |
| IF | Step Condition | Argument | Instrukcja warunkowa. Jeśli zmienna o nazwie podanej w argumencie istnieje i **nie jest pusta**, nic się nie dzieje. Jeśli zmienna nie istnieje lub jej wartość to pusty ciąg znaków (””), krok natychmiast się kończy – program przechodzi do kolejnego kroku, jeśli takowy istnieje, a obecny oznacza statusem SKIPPED. |
| Xpath | Instrukcja warunkowa. Jeśli na stronie istnieje element o podanym Xpath, nic się nie dzieje. Jeśli nie istnieje, krok natychmiast się kończy – program przechodzi do kolejnego kroku, jeśli takowy istnieje, a obecny oznacza statusem SKIPPED. |
| IFT | Test Condition | Argument | Instrukcja warunkowa. Jeśli zmienna o nazwie podanej w argumencie istnieje i **nie jest pusta**, nic się nie dzieje. Jeśli zmienna nie istnieje lub jej wartość to pusty ciąg znaków (””), cały test natychmiast się kończy – obecny krok i wszystkie kolejne w teście zostają oznaczone statusem UNTESTED, a następnie cały test też otrzymuje status UNTESTED. |
| Xpath | Instrukcja warunkowa. Jeśli na stronie istnieje element o podanym Xpath, nic się nie dzieje. Jeśli nie istnieje, cały test natychmiast się kończy – obecny krok i wszystkie kolejne w teście zostają oznaczone statusem UNTESTED, a następnie cały test też otrzymuje status UNTESTED. |
| M | Move | Xpath | Symuluje przesunięcie kursora nad element znajdujący się pod podanym lokatorem Xpath. Aby zapewnić prawidłową obsługę elementów reagujących na ruch kursora, program automatycznie czeka 1 sekundę po wykonaniu ruchu. |
| NW | New Window | - | Przechodzi do nowo otwartego okna. Komendy tej należy używać tylko po otwarciu nowego okna; jeśli w danej chwili jest otwarte tylko jedno okno, test zakończy się niepowodzeniem. |
| P | Picker | Xpath, argument | Wybiera jedną opcję z okna wyboru. Klika w element znajdujący się pod podanym lokatorem Xpath, przechodzi do nowego okna, zaznacza radiobutton o podanej w argumencie wartości, a następnie wraca.  Identyczny efekt można osiągnąć ciągiem komend:   * B xpath * NW * R argument * B //span[contains(text(),'OK')] * BTW   Argument może być zmienną lub stałą. |
| PL2 | Picker Level 2 | Xpath, argument | Wybiera jedną opcję z wielopoziomowego okna wyboru. Działa identycznie jak komenda P, ale pozwala na obsługę wielopoziomowych drzew z różnymi opcjami. Argument należy podać w postaci Gałąź1\Gałąź2\(…)\Wartość. Komenda najpierw rozwinie gałęzie o podanych nazwach, a następnie zaznaczy radiobutton o podanej wartości. Działa tylko jeśli poszczególne gałęzie rozwija się przyciskiem „+”.  Argument może być zmienną lub stałą. |
| R | Radio button | Argument | Zaznacza radiobutton, którego wartość (value) zawiera w sobie podany argument.  Argument może być zmienną lub stałą. |
| S | Select | Xpath, argument | Wybiera z listy rozwijanej typu drop-down, znajdującej się pod podanym lokatorem Xpath, opcję podaną w argumencie.  Argument może być zmienną lub stałą. |
| SCR | Screenshot | Argument | Jeśli zmienna screenshootingLevel jest ustawiona na wartość 6 (NEVER), nic się nie dzieje. W przeciwnym wypadku wykonuje zrzut ekranu w postaci pliku .png o nazwie podanej w argumencie.  Argument może być zmienną lub stałą. |
| T | Text | Xpath, argument | Wypełnia pole tekstowe znajdujące się pod podanym lokatorem Xpath wartością podaną w argumencie. Argument może być zmienną lub stałą. |
| U | Upload File | Xpath, argument | Klika w element otwierający systemowe okno wyboru pliku, znajdujący się pod podanym lokatorem Xpath, a następnie wpisuje w nim plik o podanej w argumencie ścieżce i zatwierdza. Ścieżka do pliku musi być pełna. Argument może być zmienną lub stałą. |
| W | Wait | Argument | Czeka ilość czasu podaną w argumencie (w milisekundach). Argument może być zmienną lub stałą. Wartość argumentu musi być nieujemną liczbą całkowitą, w przeciwnym wypadku test zakończy się niepowodzeniem. |

### Stałe

Jako argument do wielu komend można podać wartość stałą, poprzedzając ją znakiem dolara ($). Wtedy ta wartość zostanie wykorzystana przez komendę, niezależnie od tego z jakim zestawem danych zostanie uruchomiony test.

### Zmienne lokalne i globalne

Zmienne i ich wartości definiowane są w pliku z danymi. Do ich wartości można się odwoływać podając nazwy zmiennych jako argument do komend w skrypcie testowym.

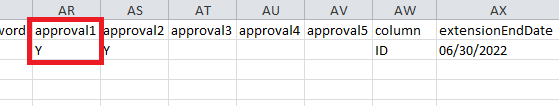
Oprócz listy zmiennych pobieranej z pliku z danymi, skrypty testowe mają też dostęp do **globalnej listy zmiennych**, która jest wspólna dla wszystkich scenariuszy i testów. Domyślnie lista ta jest pusta, ale można wpisywać do niej wartości komendami takimi jak GET lub DAT, poprzedzając nazwę zmiennej globalnej znakiem *&*. Następnie do wartości tych można się odwoływać także wstawiając znak & przed nazwą zmiennej. Pozwala to na przekazywanie danych pomiędzy różnymi przypadkami testowymi w różnych scenariuszach.

Przykład:  
*Na rys. 5, do komendy W (Wait) podana została wartość stała $10000. Oznacza to, że niezależnie od zestawu danych z którym zostanie uruchomiony skrypt, w tym miejscu zawsze nastąpi 10-sekundowe oczekiwanie (10000 ms = 10 s).*

*Komenda IF (Step Condition) odwołuje się zmiennej o nazwie approval1. W zestawie danych (rys. 6) widzimy, że zmienna ta ma wartość „Y”, więc taka wartość zostanie odczytana przez komendę.*

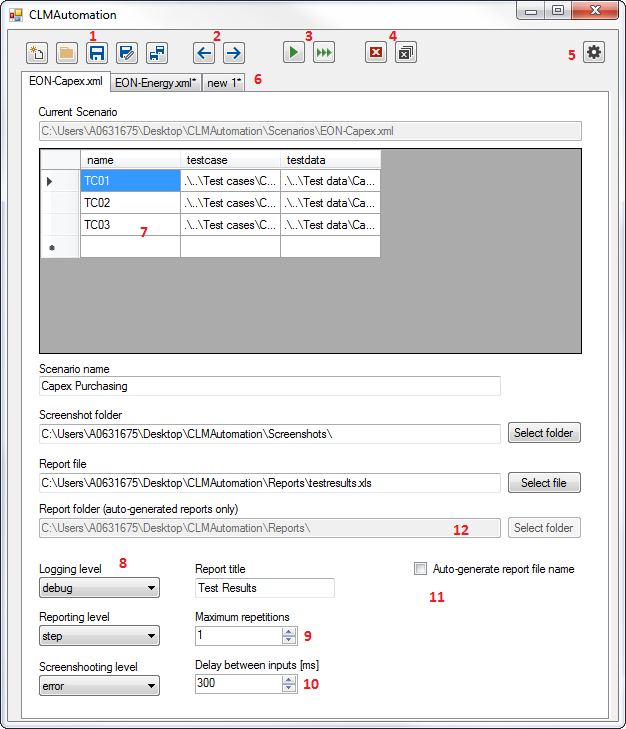


*Rys. 5 – użycie stałych i zmiennych w skrypcie testowym*



*Rys.6 – zmienna z rys. 5 użyta w zestawie danych*

## *6. Obsługa programu*



*Rys. 7 – widok głównego okna aplikacji*

Tworzenie, zapis i edycja scenariuszy testowych odbywa się przy użyciu wbudowanego w program interfejsu graficznego (*Rys. 7*). W lewym górnym rogu okna znajdują się przyciski służące do tworzenia, otwierania i zapisywania scenariuszy (1). Każdy otwarty scenariusz pojawia się jako nowa zakładka w panelu zakładek (6).

Obecnie wybraną zakładkę można przesuwać w lewo bądź w prawo w panelu zakładek korzystając z przycisków strzałek (2).

Wciśnięcie przycisku *Run* (3) uruchomi aktualnie wybrany scenariusz testowy. Wciśnięcie przycisku *Run all* uruchomi wszystkie otwarte scenariusze po kolei, według kolejności zakładek (od lewej do prawej).

Przyciski *Close* i *Close all* (4) zamykają odpowiednio obecnie wybraną, lub wszystkie zakładki. Jeśli wszystkie zakładki zostaną zamknięte, program automatycznie otworzy nową zakładkę z pustym scenariuszem.

Centralne miejsce w oknie aplikacji zajmuje tabela z przypadkami testowymi (7). Po kliknięciu w jedną z komórek tabeli, użytkownik będzie miał możliwość wyboru pliku ze skryptem, który ma zostać dodany do scenariusza.

Poniżej użytkownik ma możliwość zmiany ustawień raportowania (8). Znaczenie zmiennych *Logging level*, *Reporting level*, oraz *Screenshooting level* zostało omówione powyżej, w rozdziale *Dane wyjściowe*.

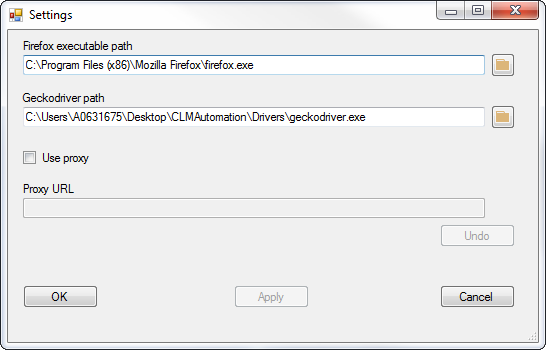
Parametr *Maximum repetitions* (9) decyduje o tym, ile razy program ma próbować wykonywać każdą komendę w przypadku wystąpienia błędu. Domyślnie parametr ten wynosi 1, więc komendy nie będą powtarzane. Parametr *Delay between inputs* (10) oznacza przerwę (w milisekundach) pomiędzy wykonywaniem kolejnych komend przez program. Oba te ustawienia są przydatne w przypadku, gdy niska przepustowość łącza internetowego lub inne czynniki powodują opóźnienia ze strony testowanej strony, co mogłoby doprowadzić np. do uznania testu za nieudany, gdyż przycisk który miał być kliknięty nie załadował się na czas.

Zaznaczenie opcji *Auto-generate report file name* (11) spowoduje wygenerowanie nowej nazwy pliku z raportem dla każdego uruchomienia scenariusza. Nazwa powstaje na bazie nazwy scenariusza, aktualnej daty oraz godziny i ma postać:

*Nazwa\_scenariusza yyyy-MM-dd HH\_mm\_ss.xls*

Opcja ta jest użyteczna, kiedy użytkownik pragnie zachować wyniki z wielu uruchomień scenariusza, na przykład do przyszłej analizy.

Wreszcie, klikając w przycisk *Settings* w prawym górnym rogu (5), użytkownik może dostać się do ustawień aplikacji.



*Rys. 8 – okno z ustawieniami aplikacji*

Okno to (*Rys. 8*) pozwala na dokonanie zmian w ogólnych ustawieniach programu – możliwe jest zmienienie ścieżki do pliku wykonywalnego programu Mozilla Firefox, ścieżki do sterownika Geckodriver (załączonego w folderze z programem), a także ustawień proxy. Zalecane jest sprawdzenie i/lub zmiana tych ustawień przed pierwszym uruchomieniem programu.

Wszystkie ustawienia aplikacji zapisywane są w pliku *properties.xml*.