ISOLUTION Sp. z o. o. S.K.A
ul. Eustachego Tyszkiewicza 21
01-157 Warszawa
NIP: 532-175-44-49
www.isolution.pl

Dokumentacja użytkownika

ATSuite

Automated Test Suite

Wersja dokumentu: 01

ISOLUTION
trust in quality

Spis treści:

1. Wprowadzenie

Dokument zawiera:

- Opis procesu instalacji i uruchomienia interpretera ATSuite,
- Szczegółowy opis instrukcji języka skryptowego Script.ISOLUTION wykorzystywanego w celu utworzenia testu automatycznego uruchamianego i wykonywanego przez interpreter ATSuite.

W rozdziale ***Pierwsze kroki*** opisano instrukcje, dzięki którym użytkownik jest w stanie utworzyć prosty skrypt testu automatycznego, uwzględniając konfigurację interpretera oraz czynności jakie powinien wykonać automat w celu przeprowadzenia testu. Oczywiście zbiór komend w skrypcie jest ściśle uzależniony od aplikacji i od celu (określonego scenariuszami testowymi do pokrycia wymagań funkcjonalnych i niefunkcjonalnych) jaki ma być osiągnięty. Bardziej zaawansowane instrukcje opisano w rozdziale ***Instrukcje zaawansowane.*** Tam również znajdują się instrukcje konfiguracji interpretera i instrukcje języka skryptowego.

Opisy niektórych komend poparte są przykładami prostych skryptów, które znajdują się w opisach komend w sekcjach **Przykłady użycia**. Przykłady skryptów wyróżnione kolorem zielonym były uruchomiane, a ich wynik wykonania był weryfikowany. Dla ujednolicenia i spójności przykładów, komendy w skryptach używają elementów (GUI) znajdujących się na stronie internetowej http://www.isolution.pl/. Przykłady wyróżnione kolorem żółtym to tylko fragmenty skryptów użyte w testach automatycznych (głównie testy regresji aplikacji) zastosowanych w firmie ISOLUTION. Opisywane komendy są pogrubione i w przykładach wyróżnione dodatkowo kolorem niebieskim.

Do pisania skryptów niezbędna jest podstawowa znajomość narzędzi do analizy treści stron wyświetlanych w przeglądarce, np. dla przeglądarki **firefox** wtyczka **firebug** z rozszerzeniem **FirePath**.

Przydatne informacje dla użytkownika o stosowaniu **XPath** można znaleźć na stronie: http://www.w3schools.com/xpath/, o używaniu wyrażeń regularnych na stronie: http://www.regexr.com/. O stosowaniu rozszerzeń dla zaawansowanych użytkowników polecana jest strona http://www.beanshell.org.

2. Instalacja

Ze strony internetowej www.atsuite.com należy pobrać spakowany plik ZIP zawierający archiwum z narzędziem Interpretera. Plik należy zapisać na dysku i go rozpakować do wybranego katalogu (zalecane jest aby nazwa katalogu nie zawierała spacji). Po rozpakowaniu widoczny będzie podkatalog ***lib*** zawierający biblioteki ***jar*** niezbędne do funkcjonowania Interpretera i plik ***jar*** do uruchamiania skryptów. Nie należy modyfikować zawartości podkatalogu ***lib*** ze względu na możliwość błędnego działania Interpretera.
Do niektórych funkcji zaawansowanych interpretera ATSuite konieczne jest utworzenie podkatalogu o nazwie ***ext*** (na tym samym poziomie drzewa co ***lib***), który przewidziany jest na umieszczenie rozszerzeń Interpretera oraz bibliotek ***jar*** niezbędnych dla dodatkowych funkcjonalności. Na przykład aby korzystać z funkcji bazodanowych w podkatalogu tym należy umieścić pliki ***jar*** zawierające sterowniki JDBC dla bazy danych.

Do poprawnego działania ATSuite niezbędne jest zainstalowanie oprogramowania Java. Oprogramowanie to można pobrać ze strony internetowej: https://www.java.com.

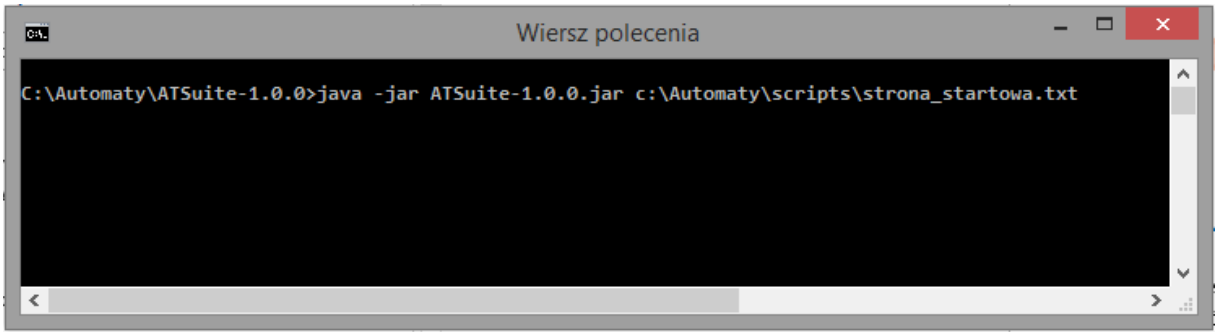
3. Uruchomienie

W celu uruchomienia skryptu poprzez interpreter, należy w wierszu poleceń ***cmd*** z katalogu, w którym znajduje się plik interpretera z rozszerzeniem **.jar** i wykonać komendę:
java -jar ***<nazwa_pliku_interpretera>*.jar** ***<ścieżka_bezwzględna_skryptu>\<nazwa_skryptu>.txt***

gdzie:
<nazwa_pliku_interpretera> – nazwa pliku interpretera z aktualnie posiadaną wersją np.: ATSuite-1.0.0,
<nazwa_skryptu> - to pełna nazwa pliku tekstowego zawierającego skrypt.

Przykłady użycia:
Wykonanie instrukcji uruchomienia interpretera dla skryptu o nazwie ***strona_startowa.txt*** znajdującym się w katalogu ***SCRIPTS*** komenda w wierszu poleceń wygląda następująco:

C:\Automaty\ATSuite-1.0.0.jar>java –jar ATSuite-1.0.0.jar C:\Automaty\scripts\strona_startowa.txt



Rysunek 1 Przykład uruchomienia skryptu z wiersza poleceń *cmd*

4. Pierwsze kroki

W rozdziale tym opisano instrukcje, dzięki którym można napisać prosty skrypt w Script.ISOLUTION i uruchomić go wykorzystując interpreter ATSuite. Instrukcje te dotyczą konfiguracji interpretera, wprowadzania komentarzy w skrypcie, wyszukania obiektu na stronie Web i wykonania na nim akcji.

4.1 Podstawowe instrukcje konfiguracji interpretera

Podrozdział opisuje podstawowe instrukcje konfiguracji interpretera mające na celu określenie rodzaju przeglądarki, na której ma zostać wykonany skrypt, na jaki czas wstrzymać wykonywanie skryptu oraz prezentacji kroków wykonywanego skryptu w oknie konsoli **cmd**.

4.1.1 Określenie przeglądarki dla wykonania testu

W skrypcie wymagane jest podanie dla jakiej przeglądarki ma być wykonany test. Do tego celu służy instrukcja:

config driver

Określenia, która wersja sterownika *WebDriver* dla przeglądarki internetowej zostanie użyta w wykonywanym skrypcie. Instrukcja ta powinna zawsze poprzedzać komendę **open** do otwarcia strony aplikacji internetowej.

O zaawansowanych komendach konfiguracyjnych można przeczytać w podrozdziale ***Zaawansowane instrukcje konfiguracyjne interpretera***.

Składnia:
config driver = *"typ przeglądarki"*

gdzie:
przeglądarka – nazwa przeglądarki internetowej.
Parametrowi *przeglądarka* może być nadana jedna z wartości:

- firefox
- chrome
- ie
- safari

Przykłady użycia:
config driver = "firefox"

Oznacza, że w skrypcie użytkownik zadeklarował korzystanie z przeglądarki **firefox**.

4.1.2 Czasowe zatrzymanie wykonywania skryptu

Interpreter umożliwia zatrzymanie działania skryptu na czas określony parametrem. Do tego celu służy komenda:

wait

Składnia:
wait *czas*
gdzie: *czas* – liczba naturalna określająca czas w milisekundach.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
wait 5000
```

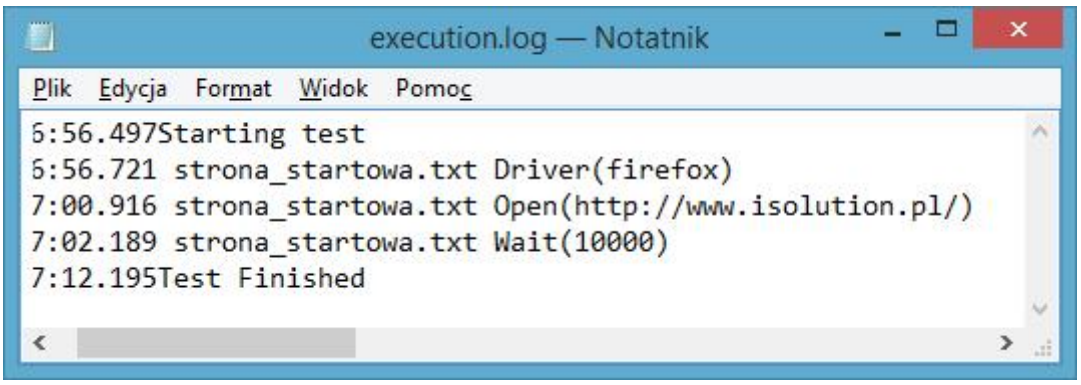
Po otwarciu strony startowej http://www.isolution.pl/ spowoduje zatrzymanie wykonywania skryptu na **5000** milisekund (5 sekund).

4.1.3 Logowanie wykonywania skryptu do konsoli cmd

Domyślnie interpreter nie wypisuje instrukcji na ekranie konsoli. Po użyciu komendy:

verbose

wyświetlone będzie dokładnie to co znajduje się w pliku **execution.log**.
Jest to instrukcja bezparametrowa, która służy do przełączenia interpretera w tryb szczegółowego logowania na konsoli wykonania skryptu.



Rysunek 2 Przykład instrukcji w execution.log

4.2 Podstawowe instrukcje języka skryptowego Script.ISOLUTION

Podrozdział opisuje podstawowe instrukcje języka skryptowego Script.ISOLUTION. Dzięki nim można wykonać prosty w pełni działający automat testujący, który otworzy wskazaną stronę Web, odnajdzie określone w skrypcie obiekty, wykona na nich wybrane operacje oraz sprawdzenie punktów weryfikacji poprawności testu.

W Script.ISOLUTION używane jest kilka rodzajów instrukcji. Są to instrukcje, które posiadają dodatkowe parametry i są nimi np.: **open**

open "parametr_dla_open"

instrukcje bezparametrowe np.: **click**

click

oraz instrukcje posiadające metody np.: **find**

find id "id_elementu"

4.2.1 Dodawanie komentarzy

W skrypcie można umieszczać jedno liniowe komentarze (adnotacje). W skryptach o dużej liczbie komend i złożonej strukturze dodawanie komentarzy jest bardzo użyteczne.
Linia komentarza zaczyna się od znaków **//**.

Przykłady użycia:

```
// To jest komentarz
```

4.2.2 Wskazanie strony internetowej

Do otwarcia w oknie przeglądarki strony internetowej, służy instrukcja:

open

Do niej podawany jest adres strony jako parametr.

Składnia:
open *"www"*
gdzie: *www* – adres strony internetowej.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
```

Przy użyciu przeglądarki **firefox** spowoduje otwarcie strony internetowej: **http://www.isolution.pl/**.

4.2.3 Maksymalizacja okna przeglądarki

Do rozwinięcia okna przeglądarki na cały ekran (spowoduje maksymalizację wielkości okna przeglądarki) wykorzystuje się bezparametrową instrukcję:

maximize

Składnia:
maximize

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
```

Przy użyciu przeglądarki **firefox** spowoduje otwarcie strony internetowej: **http://www.isolution.pl/**. i rozwinięcie jej na cały ekran.

4.2.4 Wyszukiwanie elementów strony

Do wyszukiwania elementów w strukturze strony wykorzystywana jest instrukcja:

find

Wyszukane elementy mogą posłużyć do wykonania kolejnych komend. Wyszukany element pozostaje elementem aktywnym do czasu wywołania kolejnej komendy **find**.

W celu wyszukania elementu należy określić jakie metody użyć:

find id

Komenda służy do wyszukania w strukturze strony elementu na podstawie jego identyfikatora **id**.

Składnia:
find id *"identyfikator"*
gdzie: *identyfikator* – identyfikator *id* elementu na stronie.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find id "products"
wait 5000
```

find xpath

Komenda służy do wyszukania w strukturze strony elementu wskazanego wyrażeniem ***xpath***.

Składnia:
find xpath *"xpath"*

gdzie:
xpath – wyrażenie *xpath* wskazujące element strony.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find xpath "//div[1]/div/div[3]/form/div/input[1]"
wait 5000
```

find name

Komenda służy do wyszukania w strukturze strony elementu z atrybutem *name*.

Składnia:
find name *"name"*

gdzie:
name – wartość atrybutu *name* elementu strony.

Przykłady użycia:

```
find name "login"
write "::login"
find name "password"
write "::password"
```

find linkText

Komenda służy do wyszukania w strukturze strony linku na podstawie jego nazwy.

Składnia:
find linkText *"link"*
gdzie: *link* – pełna nazwa linku.

Przykłady użycia:
```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find linkText "O naszej firmie"
wait 5000
```

find partialLinkText

Komenda służy do wyszukania w strukturze strony linku na podstawie fragmentu jego nazwy.

Składnia:
find partialLinkText *"link"*
gdzie: *link* – fragment nazwy linku.

Przykłady użycia:
```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find partialLinkText "firmie"
wait 5000
```

find className

Komenda służy do wyszukania w strukturze strony elementu na podstawie nazwy klasy CSS.

Składnia:
find name *"class"*
gdzie: *class* – nazwa klasy CSS.

Przykłady użycia:
```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
```

```
find className "homepage"
wait 5000
```

find cssSelector

Komenda służy do wyszukania w strukturze strony elementu na podstawie selektora CSS.

Składnia:
find cssSelector *"selector"*
gdzie: *selector* – nazwa selektora CSS.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find cssSelector "#header"
wait 5000
```

find tagName

Komenda służy do wyszukania w strukturze strony elementu z nazwą znacznika.

Składnia:
find tagName *"tag"*
gdzie: *tag* – nazwa znacznika.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find tagName "h2"
wait 5000
```

4.2.5 Wprowadzanie tekstu w pola tekstowe

Aby wpisać dowolny tekst w pola (typu *text fiedl* i *text area*) należy wykorzystać instrukcję:

write

W tym wypadku tekst podawany jest jako parametr. Użycie komendy **write** musi być poprzedzone wykonaniem komendy **find** z odpowiednim parametrem, wskazując konkretny element znajdujący się w strukturze strony.

Składnia:
write

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find xpath "//div[1]/div/div[3]/form/div/input[1]"
write "automatyzacja"
wait 5000
```

Na stronie startowej http://www.isolution.pl/ spowoduje wpisanie do pola wyszukiwarki (-szukana fraza-) tekstu **automatyzacja**.

Język skryptowy umożliwia wpisanie tekstu do wskazanego wyrażeniem ***xpath*** elementu (pola typu *text fiedl* i *text area*) w jednej linii instrukcji, gdzie tekst podawany jest jako drugi parametr instrukcji.
W tym celu należy użyć:

writex

Składnia:
writex *"xpath" "text"*
gdzie: *xpath* - wyrażenie wskazujące położenie elementu w strukturze strony;
text – ciąg znaków.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
writex "//div[1]/div/div[3]/form/div/input[1]" "projekt"
wait 5000
```

Na stronie startowej http://www.isolution.pl/ spowoduje wpisanie do pola wyszukiwarki (-szukana fraza-) tekstu **projekt**.

4.2.6 Zasymulowanie kliknięcia

Częstym działaniem każdego automatu testującego jest zasymulowanie kliknięcia we wskazany element. Język skryptowy Script.ISOLUTION udostępnia taką możliwość za pomocą instrukcji bezparametrowej:

click

Należy pamiętać, iż użycie tej instrukcji musi być poprzedzone wykonaniem komendy **find** i wyszukaniem elementu.

Składnia:
click

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
find linkText "Produkty"
click
wait 5000
```

Przy użyciu przeglądarki **firefox** spowoduje otwarcie strony internetowej: http://www.isolution.pl/, a po kliknięciu w link **Produkty** wyświetlenie strony z oferowanymi produktami http://www.isolution.pl/produkty.html.

Również i w tym przypadku można użyć instrukcji, która jest połączeniem działania instrukcji **find** z parametrem ***xpath*** i instrukcji **click**. Instrukcją tą jest:

clickx

Składnia:
clickx *"xpath"*

gdzie:
xpath to wyrażenie wskazujące położenie elementu w strukturze strony.

Przykłady użycia:

```
config driver = "firefox"
open "http://www.isolution.pl/"
maximize
wait 2000
//To są trzy linie komentarza
```

```
//Kliknięcie elementu strony o nazwie Produkty
//wskazanego wyrażeniem xpath = "//div[2]/div/div/div[1]/div/div[5]/h1/a"
clickx "//div[2]/div/div/div[1]/div/div[5]/h1/a"
wait 5000
```

Przy użyciu przeglądarki **firefox** spowoduje otwarcie strony internetowej: http://www.isolution.pl/, kliknięcie na element wskazany wyrażeniem ***xpath*** spowoduje wyświetlenie strony http://www.isolution.pl/produkty.html.

4.2.7 Weryfikacja poprawności testu

Podczas wykonywania skryptu można sprawdzić (zweryfikować), czy zostały spełnione określone warunki. Do tego celu służy instrukcja:

verify

Nie spełnienie niektórych z nich powoduje natychmiastowe zakończenie wykonywania skryptu i zwrócenie wyjątku. Każde wejście w blok weryfikacji powoduje wpisy w plikach **verification.log** i **verification.html** utworzonych automatycznie po zakończeniu wykonywania skryptu. Szerszy opis tworzenia tych plików znajduje się w rozdziale **Zaawansowane instrukcje**.

Składnia:
verify ***"Opis weryfikacji"***
warunki weryfikacji
end

gdzie:
end – to słowo kluczowe oznaczające koniec bloku weryfikacji;
warunki weryfikacji:
***"xpath"* must exist** – *element musi być obecny na stronie*
***"xpath"* mustnot exist** – *element nie może być obecny na stronie*
***"xpath"* should exist** – *element powinien być na stronie*
***"xpath"* shouldnot exist** – *element nie powinien być na stronie*
***"xpath"* must be enabled** – *element musi być aktywny*
***"xpath"* mustnot be enabled** – *element nie może być aktywny*
***"xpath"* should be enabled** – *element powinien być aktywny*
***"xpath"* shouldnot be enabled** – *element nie powinien być aktywny*
***"xpath"* must be disabled** – *element musi być nieaktywny*
***"xpath"* mustnot be disabled** – *element nie może być nieaktywny*

***"xpath"* should be disabled** – *element powinien być nieaktywny*
***"xpath"* shouldnot be disabled** – *element nie powinien być nieaktywny*
"xpath"* must be equal *"wartość" – *wartość elementu musi się równać podanej wartości*
"xpath"* mustnot be equal *"wartość" – *wartość elementu nie może równać się podanej wartości*
"xpath"* should be equal *"wartość" – *wartość elementu powinna równać się podanej wartości*
"xpath"* shouldnot be equal *"wartość" – *wartość elementu nie powinna równać się podanej wartości*
"xpath"* must contain *"wartość" – *wartość elementu musi zawierać podaną wartość*
"xpath"* mustnot contain *"wartość" – *wartość elementu nie może zawierać podanej wartości*
"xpath"* should contain *"wartość" – *wartość elementu powinna zawierać podaną wartość*
"xpath"* shouldnot contain *"wartość" – *wartość elementu nie powinna zawierać wartości podanego elementu*

gdzie:
xpath – wyrażenie *xpath* określające położenie elementu w strukturze strony.

Jeżeli warunek zawiera słowo kluczowe **must** a warunek nie zostanie spełniony to wykonanie skryptu zostanie przerwane i zwrócony zostanie wyjątek.

Natomiast, nie spełnienie warunku zawierającego słowo kluczowe **should** spowoduje tylko utworzenie wpisu ostrzeżenia w logu weryfikacji. Brak wpisów w logu dla konkretnej weryfikacji oznacza sukces.

Przykłady użycia:

```
config driver = "firefox"
config overrideGlobals="true"
open "http://www.isolution.pl/"
maximize
wait 2000
verify "Czy pole wyszukiwarki jest aktywne ?"
"//div[1]/div/div[3]/form/div/input[1]" must be enabled
end
verify "Czy jest link SZKOLENIA ?"
"//div[1]/div/div[4]/ul/li[5]/a" must exist
"//div[1]/div/div[4]/ul/li[5]/a" must be equal "SZKOLENIA"
```

end

5. Instrukcje zaawansowane

5.1 Zawansowane instrukcje konfiguracji interpretera

config overrideGlobals

Komenda określa czy w skrypcie możliwe jest nadpisywanie wartości zmiennych globalnych. W skrypcie komendę można używać wielokrotnie.

Uwagi:

- Jeśli wartość parametru ustawiona jest na **false** to próba nadpisania zmiennej globalnej zakończy się błędem.

Składnia:

config overrideGlobals = *"boolean"*

gdzie: *boolean* – jedna z dwóch wartości:

- **true**
- **false**

Domyślną wartością jest **false**.

Przykłady użycia:

```
config overrideGlobals="true"
variable global fraza = "projekt"
variable global fraza = "automatyzacja"
```

config database

Komenda służy do skonfigurowania połączenia do bazy danych. Po wykonaniu tej komendy w pamięci zostanie utworzone połączenie do bazy danych o nazwie odpowiadającej nazwie pliku bez rozszerzenia.

Składnia:

config database = *"nazwa_pliku"*

gdzie: *nazwa_pliku* – to nazwa pliku z rozszerzeniem **properties** zawierającego definicję połączenia do bazy danych. W pliku tym należy umieścić następujące wpisy:

driver.class – *nazwa klasy sterownika jdbc*

username – *nazwa użytkownika*

password – *hasło użytkownika*

connection.string – *URL połączenia do bazy danych*

Na ***Rysunek 3 Przykładowa zawartość pliku definiującego połączenie do bazy danych*** pokazano przykładową zawartość pliku definiującego połączenie do bazy danych.

Uwagi:

- Pliki z rozszerzeniem **jar** zawierające sterowniki niezbędne do nawiązania połączenia z bazą danych (np. sterownik **postgresql.jar**) należy umieścić w katalogu ***ext***;
- Nazwa pliku może zawierać tylko znaki alfanumeryczne (litery i cyfry);

- W komendach, których parametrem jest nazwa połączenia do bazy danych używa się tylko nazwy pliku bez rozszerzenia **properites**

Przykłady użycia:
config database = "dbrozliczenia.properties"
//Poniżej przykład komendy, której parametrem jest nazwa połączenia do bazy danych
query dbrozliczenia "projekt_identyfikator.sql"

```
1  driver.class = org.postgresql.Driver
2  username = isolution
3  password = isolution
4  connection.string = jdbc:postgresql://192.168.0.91:5432/db_rozliczenia
```

Rysunek 3 Przykładowa zawartość pliku definiującego połączenie do bazy danych

timeout

Komenda służy do ustawienia wartości parametru czasowego *WebDriver'a*. Wielkość ta określa po ilu sekundach od pierwszej próby odnalezienia elementu na stronie zostanie ponowiona próba. Nie odnalezienie elementu po upływie podanego czasu spowoduje wystąpienie wyjątku *WebDriver'a* i zakończenie wykonania skryptu.

Składnia:
timeout *czas*
gdzie: *czas* – to liczba naturalna określająca czas w sekundach.

Przykłady użycia:
config driver = "firefox"
timeout 10
Czas odnalezienia elementu na stronie ograniczony został do 10 sekund.

Konfiguracja daty i czasu

config dateFormat

Komenda służy do ustawiania formatu daty, który będzie obowiązywał w wykonywanym skrypcie. Komenda może występować wielokrotnie, obowiązuje format daty po wykonaniu ostatniej komendy **config dateFormat** w uruchomionym skrypcie.

Uwagi:

- Zdefiniowany format daty jest sprawdzany przy wykonywaniu komend, których parametrem jest data i przy pobieraniu danych z bazy danych (kolumny o typie danych DATE)
- Format stosuje notację opisaną w dokumentacji klasy *java.util.SimpleDateFormat*.

Składnia:

config dateFormat = *"pattern"*
gdzie: *pattern* – to format daty, w którym:
- *yyyy* – to cyfry roku,
- *MM* – to cyfry miesiąca,
- *dd*- to cyfry dnia.

Rok, miesiąc i dni mogą występować w dowolnej kolejności, nie każdy człon jest wymagany, znak separatora jest polecany ale nie obowiązkowy. Nie jest zalecane używanie znaku „_” jako separatora.
Domyślną wartością jest ***dd/MM/yyyy***.

Przykłady użycia:

```
config overrideGlobals="true"
variable global data = "___today"
//Data wyprowadzona w formacie domyślnym
log "::data"
config dateFormat = "yyyy-MM-dd"
variable global data = "___today"
log "::data"
config dateFormat = "DD.MM.yyyy"
variable global data = "___today"
log "::data"
config dateFormat = "dd-MM"
variable global data = "___today"
//Tylko dzień i miesiąc
log "::data"
config dateFormat = "MM-YYYY"
variable global data = "___today"
//Tylko miesiąc i rok
log "::data"
config dateFormat = "YYYY"
variable global data = "___today"
//Tylko rok
log "::data"
wait 5000
exit
```

Wykonanie skryptu spowoduje wyprowadzenie na konsoli aktualnej daty w różnych formatach.

config timeFormat

Komenda służy do ustawiania formatu czasu, który będzie obowiązywał w wykonywanym skrypcie. Komenda może występować wielokrotnie, obowiązuje format czasu po wykonaniu ostatniej komendy **config timeFormat** w uruchomionym skrypcie.

Uwagi:

- Zdefiniowany format czasu jest sprawdzany przy wykonywaniu komend, których parametrem jest czas i przy pobieraniu danych z bazy danych (kolumny o typie danych TIME)
- Szablon stosuje notację opisaną w dokumentacji klasy *java.util.SimpleDateFormat*.

Składnia:
config timeFormat = *"pattern"*
gdzie: *pattern* – to format czasu, w którym:
- *hh* – to cyfry godziny,
- *mm* – to cyfry minut,
- *ss* to cyfry sekund.

Godziny, minuty i sekundy mogą występować w dowolnej kolejności, nie każdy człon jest wymagany, znak separatora jest polecany ale nie obowiązkowy.
Domyślną wartością jest ***hh:mm:ss***.

Przykłady użycia:
config timeFormat = "hh:mm"
config timeFormat = "hh/mm"
config timeFormat = "mm-ss"

config dateTimeFormat

Komenda służy do ustawiania formatu daty i czasu, który będzie obowiązywał w wykonywanym skrypcie. Komenda może występować wielokrotnie, obowiązuje format daty i czasu po wykonaniu ostatniej komendy **config dateTimeFormat** w uruchomionym skrypcie.

Uwagi:

- Zdefiniowany format daty i czasu jest sprawdzany przy wykonywaniu komend, których parametrem jest data i czas oraz przy pobieraniu danych z bazy danych (kolumny o typie danych TIMESTAMP)
- Szablon stosuje notację opisaną w dokumentacji klasy *java.util.SimpleDateFormat*.

Składnia:
config dateTimeFormat = *"pattern"*
gdzie: *pattern* – to format czasu, w którym:
- *yyyy* – to cyfry roku,
- *MM* – to cyfry miesiąca,
- *dd*- to cyfry dnia.
- *hh* – to cyfry godziny,
- *mm* – to cyfry minut,
- *ss* to cyfry sekund.

Godziny, minuty i sekundy mogą występować w dowolnej kolejności, nie każdy człon jest wymagany, znak separatora jest polecany ale nie obowiązkowy.
Domyślną wartością jest ***dd/MM/yyyy hh:mm:ss***.

Przykłady użycia:
```
config dateTimeFormat = "yyyy-MM-dd hh:mm:ss"
config dateTimeFormat = "dd hh:mm"
```

config minDate

Komenda służy do ustawiania minimalnej daty jaka może być zwrócona przez komendy do generowania danych losowych (opisane w punkcie ***9.2 Generowanie dat***). Komenda może występować wielokrotnie, obowiązuje ustawienie po wykonaniu ostatniej komendy **config minDate** w uruchomionym skrypcie.

Składnia:
config minDate = *"data"*
gdzie: *data* – to wartość daty podana zgodnie ze zdefiniowanym wzorcem przez komendę konfiguracyjną **config dateFormat**.
Domyślną wartością minimalnej daty jest ***01/01/1970***.

Przykłady użycia:
```
config overrideGlobals="true"
config dateFormat = "yyyy-MM-dd"
config minDate = "2014-01-01"
variable global data = "___randomDate"
log "::data"
config minDate = "1555-01-01"
variable global data = "___randomDate"
log "::data"
exit
```

config maxDate

Komenda służy do ustawiania maksymalnej daty jaka może być zwrócona przez komendy do generowania danych losowych (opisane w punkcie ***9.2 Generowanie dat***). Komenda może występować wielokrotnie, obowiązuje ustawienie po wykonaniu ostatniej komendy **config maxDate** w uruchomionym skrypcie.

Składnia:
config maxDate = *"data"*

gdzie: *data* – to wartość daty podana zgodnie ze zdefiniowanym wzorcem przez komendę konfiguracyjną **config dateFormat**.
Domyślną wartością minimalnej daty jest ***31/12/2199***.

Przykłady użycia:

```
config overrideGlobals="true"
config dateFormat = "yyyy-MM-dd"
config maxDate = "2014-01-01"
variable global data = "___randomDate"
log "::data"
config maxDate = "1900-01-01"
variable global data = "___randomDate"
log "::data"
exit
```

dataset

Komenda umożliwia korzystanie w skrypcie ze zbioru danych testowych zdefiniowanych w pliku z rozszerzeniem **properties**. Klucze zawarte w pliku zostaną użyte jako nazwy zmiennych a ich wartości przyjmą wartości tych kluczy. Domyślnie Interpreter w momencie uruchomienia wyszukuje plik z rozszerzeniem **properties** o nazwie uruchomionego skryptu. Jeśli nie ma takiego pliku to wyszukuje plik o nazwie podanej jako parametr komendy **dataset**.
Jeśli w skrypcie zostanie użyta komenda **dataset** to obowiązujące są dane zdefiniowane w pliku podanym jako parametr tej komendy.

Uwagi:

- Wartości tak zdefiniowanych zmiennych można tylko „przesłaniać” przez użycie w skrypcie komendy **variable local**
- Zmienna ***index***, która jest domyślnie zastosowana i unikalna w komendzie pętli **loop** może zostać „przesłonięta” (wewnątrz ciała pętli) tylko przez komendę **variable local** ale nie ma to wpływu na liczbę wykonań pętli
- Zmiennej ***index*** można używać poza ciałem pętli **loop** tak jak innych zmiennych

Składnia:
dataset *"danetestowe.properties"*
gdzie: *danetestowe* – nazwa pliku bez rozszerzenia; *properties* – nazwa rozszerzenia pliku.

Przykłady użycia:

```
dataset "danetestowe.properties"
```

Poniżej przedstawiono przykładową zawartość pliku ***danetestowe.properties***.

```
1 driver = firefox
2 www = http://www.isolution.pl/
3 user = jkowalski
4 password = kowalski
5 name = Jan
6 surname = Kowalski
```

Rysunek 4 Przykładowa zawartość pliku danych *properties*

Śledzenie wykonywania skryptu

log

Komenda pomocna do śledzenia aktualnych wartości zmiennych podczas wykonywania skryptu. Przy jej użyciu można również wyprowadzać tekst na konsolę np. komentujący wykonywanie skryptu.

Składnia:
log *"tekst* ::*zmienna"*
gdzie: *tekst* – ciąg znaków; *zmienna* – nazwa zmiennej.

Przykłady użycia:

```
variable global nazwa1 = "Automatyzacja"
variable global nazwa2 = "Testowania"
log "::nazwa1 Testowania"
log "Automatyzacja ::nazwa2"
exit
```

Wykonanie skryptu spowoduje wyprowadzenie na konsolę tekstu:
Automatyzacja Testowania
Automatyzacja Testowania

comment

Komenda służy do wyświetla na ekranie użytkownika okienko zawierające komentarz, który jest drugim parametrem komendy. Pierwszy parametr komendy określa czas (w milisekundach) wyświetlania okienka na ekranie.

Składnia:
comment *czas "komentarz"*
gdzie: *czas* – liczba naturalna określająca czas w milisekundach; *komentarz* – tekst.

Przykłady użycia:

```
comment 5000 "TESTY"
```

```
exit
```

W wyniku wykonania skryptu przez 5 sekund będzie wyświetlone poniższe okienko:

Rysunek 5 Przykładowe okno po wykonaniu komendy *comment*

bp

Komenda służy do przerwania wykonywanego skryptu. Spowoduje utworzenie w miejscu jej wystąpienia punktu przerwania (Break Point). Zostanie wyświetlenie okno dialogowe. Wybór przycisku „**Yes**" spowoduje kontynuację wykonywania skryptu, natomiast wybór przycisku „**No**" przerwie jego wykonywanie i zamknie okno przeglądarki.
Poniżej przedstawiono okno dialogowe:

Rysunek 6 Okno dialogowe po wykonaniu komendy przerwania skryptu *bp*

debug

Komenda służy do przełączenia Interpretera w tryb działania ***debug***. W tym trybie dodatkowo generowany jest plik ***debugSession.isds***. Zawiera on dodatkowe informacje niezbędne do analizy potencjalnych błędów Interpretera. W przypadku zgłaszania błędów do funkcjonalności Interpretera należy dostarczyć w/w plik wraz z informacjami wyświetlonymi na konsoli (w szczególności wyjątki).

5.2 Zaawansowane instrukcje języka skryptowego Script.ISOLUTION

import

Komenda służy do zaimportowania do skryptu komend z innego skryptu. Należy zwrócić uwagę, że zaimportowany skrypt posiada swój własny kontekst zmiennych i zmienne te są niedostępne w skrypcie importującym.

Składnia:
import *"nazwa"*
gdzie: *nazwa* – pełna nazwa pliku zawierającego skrypt importowany.

Przykłady użycia:

```
config driver = "firefox"
config overrideGlobals="true"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
variable global fraza = "projekt"
//wartość w zmiennej globalnej "fraza" zostanie użyta
//w zaimportowanym skrypcie "szukaj.txt"
import "szukaj.txt"
wait 3000
back
wait 5000
exit
```

Zawartość skryptu importowanego jako plik **szukaj.txt** (znajduje się w tym samym katalogu co skrypt główny importujący):

```
wait 1000
writex "//div[1]/div/div[3]/form/div/input[1]" "::fraza"
wait 1000
clickx "//div[1]/div/div[3]/form/div/input[2]"
wait 1000
```

Skrypt główny (importujący) wykonuje kolejne komendy i po napotkaniu komendy **import "szukaj.txt"** wykonuje wszystkie komendy zawarte w pliku **szukaj.txt** (uruchamia wyszukiwarkę). Po wykonaniu skryptu zaimportowanego, kolejną jest komenda **back** ze skryptu importującego.

back

Komenda bezparametrowa służy do zasymulowania przejścia (wstecz) do poprzednio wyświetlanej strony w przeglądarce.

Składnia:
back

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
```

```
wait 1000
open "http://www.isolution.pl/kontakt.html"
wait 2000
back
wait 5000
exit
```

Po otwarciu strony http://www.isolution.pl/kontakt.html komenda **back** spowoduje powrót do strony startowej http://www.isolution.pl/ jako poprzedniej (pierwszej) otwartej w skrypcie.

doubleclick

Komenda służy do zasymulowania podwójnego kliknięcia wyszukanego elementu na stronie. Użycie komendy **doubleclick** musi być poprzedzone wykonaniem komendy **find** z odpowiednim parametrem.

Składnia:
doubleclick

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
find linkText "Tester"
doubleclick
wait 5000
exit
```

Przy użyciu przeglądarki **firefox** spowoduje otwarcie strony internetowej: http://www.isolution.pl/, po podwójnym kliknięciu wyszukanego elementu o nazwie **Tester** wyświetlenie strony http://www.isolution.pl/praca-szczegoly/items/tester.html z ofertą pracy dla testera.

doubleclickx

Komenda służy do zasymulowania podwójnego kliknięcia elementu w strukturze strony wskazanego przez wyrażenie *xpath*. Komenda ta łączy w sobie działanie komendy **find** z parametrem ***xpath*** i komendę **doubleclick**.

Składnia:
doubleclickx *"xpath"*
gdzie: *xpath* to wyrażenie wskazujące położenie elementu w strukturze strony.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
//To są trzy linie komentarza
//Podwójne kliknięcie elementu strony o nazwie Produkty
//wskazanego wyrażeniem xpath = "//div[2]/div/div/div[1]/div/div[5]/h1/a"
doubleclickx "//div[2]/div/div/div[1]/div/div[5]/h1/a"
wait 5000
exit
```

Przy użyciu przeglądarki **firefox** spowoduje otwarcie strony internetowej: http://www.isolution.pl/, podwójne kliknięcie na element wskazany wyrażeniem ***xpath*** spowoduje wyświetlenie strony http://www.isolution.pl/produkty.html z oferowanymi produktami.

rclick

Komenda służy do zasymulowania kliknięcia prawym przyciskiem myszy w wyszukany element na stronie. Użycie komendy **rclick** musi być poprzedzone wykonaniem komendy **find** z odpowiednim parametrem.
Komenda **rclick** często jest stosowana do wywołania menu kontekstowego.

Składnia:
rclick

Przykłady użycia:

```
find xpath "//div[3]/div/div/table/tbody[2]/tr/td[1]/div/nobr"
rclick
```

rclickx

Komenda służy do zasymulowania kliknięcia prawym przyciskiem myszy elementu w strukturze strony wskazanego przez wyrażenie ***xpath***. Komenda ta łączy w sobie komendy **find** z parametrem ***xpath*** i komendę **rclick**.
Komenda **rclickx** stosowana jest głównie do wywołania menu kontekstowego wskazanego elementu na stronie.

Składnia:
rclickx *"xpath"*
gdzie: *xpath* - wyrażenie wskazujące położenie elementu w strukturze strony.

Przykłady użycia:

```
rclickx "//div[3]/div/div/table/tbody[2]/tr/td[1]/div/nobr"
```

select

Komenda służy do wyboru jednego z elementów listy typu *select* (jak lista *drop down*). Użycie komendy powinno być poprzedzone wykonaniem komendy **find** z odpowiednim parametrem. Wyszukanie elementu innego typu powoduje wystąpienie wyjątku. Parametrem komendy **select** powinna być wartość atrybutu *value* (wyszukanego elementu *select*), która ma być wybrana.

Składnia:
select *"value"*
value – jedna z wartości elementu typu *select*.

Przykłady użycia:

```
find xpath "//div[7]/div/div/div/div[1]/div[2]/table/tbody/tr[2]/td[1]/select"
select "TCI01"
```

selectx

Komenda służy do wyboru jednego z elementów listy typu *select* wskazanego przez wyrażenie ***xpath***. Komenda ta łączy w sobie działanie komendy **find** z parametrem ***xpath*** i komendę **select**.

Składnia:
selectx *"xpath" "value"*
gdzie: *xpath* - wyrażenie wskazujące położenie elementu typu *select* w strukturze strony;
value – jedna z wartości elementu typu *select*.

Przykłady użycia:

```
selectx "//div[7]/div/div/div/div[1]/div[2]/table/tbody/tr[2]/td[1]/select" "TCI01"
```

5.3 clear

Komenda bezparametrowa służy do wyczyszczenia pola tekstowego typu *text fiedl* i *text area*.
Użycie komendy **clear** musi być poprzedzone wykonaniem komendy **find** z odpowiednim parametrem.

Składnia:
clear

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
```

```
find xpath "//div[1]/div/div[3]/form/div/input[1]"
clear
wait 5000
exit
```

Spowoduje wyczyszczenie pola wyszukiwarki (-szukana fraza-) na stronie startowej http://www.isolution.pl/.

5.4 clearx

Komenda służy do wyczyszczenia pola tekstowego (typu *text fiedl* i *text area*) wskazanego przez wyrażenie *xpath*.

Składnia:

clearx *"xpath"*

gdzie: *xpath* - wyrażenie wskazujące położenie elementu w strukturze strony.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
clearx "//div[1]/div/div[3]/form/div/input[1]"
wait 5000
exit
```

Spowoduje wyczyszczenie pola wyszukiwarki (-szukana fraza-) na stronie startowej http://www.isolution.pl/.

5.5 exit

Komenda bezparametrowa służy do zakończenia wykonywania skryptu i zamknięcia okna przeglądarki.

Składnia:

exit

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
wait 1000
exit
```

6. Komendy z zastosowaniem OCR

Wszystkie opisane poniżej komendy jako pierwszy parametr przyjmują ścieżkę do nazwy pliku graficznego zawierającego fragment ekranu, który ma zostać odnaleziony.

Uwagi:

- Przy przenoszeniu skryptu pomiędzy różnymi systemami, przeglądarkami jak również przy zmianie rozdzielczości ekranu może być konieczna zamiana plików graficznych.
- Mechanizm OCR może nie odnaleźć poszukiwanego elementu w przypadku gdy tło pod półprzezroczystym elementem zmieni się (efekty ekranu *Aero* w systemach Windows i półprzezroczyste elementy okien).

Komendy z użyciem OCR stosowane są głównie w przypadku jeśli testowana aplikacja wywoła okienko systemowe np. odczyt czy zapis pliku (poza aplikacją jest to praktycznie jedyna możliwość obsługi).

6.1 clicks

Komenda służy do zasymulowania kliknięcia myszą na element graficzny znajdujący się na stronie i którego obraz jest zgodny z plikiem graficznym. Lokalizację pliku graficznego określa parametr komendy.

Składnia:
clicks *"path"*
gdzie: *path* – ścieżka określająca lokalizację pliku graficznego.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
clicks "ocr\language.JPG"
wait 5000
exit
```

W katalogu ***ocr*** znajduje plik o nazwie **language.JPG,** którego widok przedstawiono poniżej:

Wykonanie komendy **clicks** spowoduje zmianę treści wyświetlanych stron z języka polskiego (domyślnego) na angielski.

6.2 doubleclicks

Komenda służy do zasymulowania podwójnego kliknięcia myszą na element graficzny znajdujący się na stronie i którego obraz jest zgodny z plikiem graficznym. Lokalizację pliku graficznego określa parametr komendy.

Składnia:
doubleclicks *"path"*
gdzie: *path* – ścieżka określająca lokalizację pliku graficznego.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
find xpath "//div[1]/div/div[3]/form/div/input[1]"
write "automatyzacja"
doubleclicks "ocr\lupa.JPG"
wait 5000
exit
```

W katalogu ***ocr*** znajduje plik o nazwie **lupa.JPG,** którego widok przedstawiono poniżej:

Wykonanie komendy double**clicks** spowoduje uruchomienie wyszukiwarki z kryterium **automatyzacja** i przejście do strony
http://www.isolution.pl/wyszukiwarka.html?keywords=automatyzacja.

6.3 rclicks

Komenda służy do zasymulowania kliknięcia prawym przyciskiem myszy na element graficzny znajdujący się na stronie i którego obraz jest zgodny z plikiem graficznym. Lokalizację pliku graficznego określa parametr komendy.
Komenda **rclicks** stosowana jest głównie do wywołania menu kontekstowego wybranego fragmentu graficznego na stronie.

Składnia:
rclicks *"path"*
gdzie: *path* – ścieżka określająca lokalizację pliku graficznego.

Przykłady użycia:

```
rclicks "ocr\przycisk_Wyloguj.JPG"
```

6.4 writes

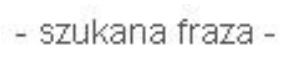
Komenda służy do wpisania tekstu (podanego jako drugi parametr komendy) w pole, którego obraz jest zgodny z plikiem graficznym znajdującym się w lokalizacji (podanej jako pierwszy parametr komendy).

Składnia:
writes *"path" "text"*
gdzie: *path* – ścieżka określająca lokalizację pliku graficznego; *text* – ciąg znaków.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
writes "ocr\pole_wyszukiwarki.JPG" "automatyzacja"
wait 2000
clicks "ocr\lupa.JPG"
wait 5000
exit
```

W katalogu ***ocr*** znajduje się plik o nazwie **pole_wyszukiwarki.JPG,** którego widok przedstawiono poniżej:



Wykonanie komendy **writes** spowoduje wpisanie do pola wyszukiwarki testu **automatyzacja. Wynikiem wykonania skryptu jest przej**ście do strony http://www.isolution.pl/wyszukiwarka.html?keywords=automatyzacja

6.5 pastes

Komenda służy do wpisania tekstu (podanego jako drugi parametr komendy) w pole, którego obraz jest zgodny z plikiem graficznym znajdującym się w lokalizacji (podanej jako pierwszy parametr komendy). Dodatkowo wpisany tekst zostanie skopiowany do schowka systemowego.

Składnia:
pastes *"path" "text"*
gdzie: *path* – ścieżka określająca lokalizację pliku graficznego; *text* – ciąg znaków.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
pastes "ocr\pole_wyszukiwarki.JPG" "kontakt"
wait 2000
clicks "ocr\lupa.JPG"
wait 5000
exit
```

Wynikiem wykonania skryptu jest przejście do strony http://www.isolution.pl/wyszukiwarka.html?keywords=automatyzacja i dodatkowo zapamiętanie tekstu **kontakt** w schowku systemowym.

6.6 dragndrops

Komenda służy do zasymulowania operacji przeciągnięcia (*drag*) i upuszczenia (*drop*) jednego elementu graficznego na inny. Parametrami komendy są lokalizacje plików graficznych zawierających wybrane fragmenty obrazu strony.

Składnia:
dragndrops *"path1" " path2"*
gdzie: *path1, path2* – ścieżki określające lokalizacje plików graficznych.

Przykłady użycia:
dragndrops "ocr\zad1.JPG" "ocr\zad2.JPG"

7. Procedury

W skrypcie możliwe jest stosowanie procedur. **Procedura** to blok komend, który może być wielokrotnie wykonywany w skrypcie. Do zdefiniowania procedury służy komenda **proc.** Do wywołania procedury w wykonywanym skrypcie służy komenda **call**.

7.1 proc

Komenda służy do zdefiniowania procedury. Procedury mogą być definiowane w dowolnym miejscu skryptu i nie jest wymagane zdefiniowanie procedury przed jej wywołaniem.

Uwagi:

- nazwa procedury musi być unikalna
- procedury zdefiniowane w skrypcie widziane są globalnie w całym skrypcie niezależnie od miejsca ich zdefiniowania
- procedura nie powinna być definiowana wewnątrz innej procedury
- próba wywołania procedury, która nie została zdefiniowana spowoduje wystąpienie wyjątku.

Składnia:
proc *nazwa_procedury*
komendy
endp
gdzie: *nazwa_procedury* – nazwa procedury; *komendy* – zbiór kolejnych komend (ciało procedury); **proc**, **endp** – słowa kluczowe określające początek i koniec definicji procedury.

Przykłady użycia:

```
proc wyszukanie
wait 1000
clearx "//div[1]/div/div[3]/form/div/input[1]"
wait 1000
writex "//div[1]/div/div[3]/form/div/input[1]" "::fraza"
wait 1000
clickx "//div[1]/div/div[3]/form/div/input[2]"
wait 1000
back
wait 2000
endp
```

7.2 call

Komenda służy do wywołania (uruchomienia) zdefiniowanej w skrypcie procedury. Procedury w skrypcie mogą być wywoływane wielokrotnie.

Składnia:
call *nazwa_procedury*
gdzie: *nazwa_procedury* – nazwa procedury.

Przykłady użycia:

```
config driver = "firefox"
config overrideGlobals="true"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
variable global fraza = "projekt"
call wyszukanie
variable global fraza = "automatyzacja"
call wyszukanie
wait 5000
exit
//Definicja procedury
proc wyszukanie
wait 1000
clearx "//div[1]/div/div[3]/form/div/input[1]"
wait 1000
writex "//div[1]/div/div[3]/form/div/input[1]" "::fraza"
wait 1000
clickx "//div[1]/div/div[3]/form/div/input[2]"
wait 1000
back
wait 2000
endp
```

Wykonanie skryptu wywołuje dwukrotnie procedurę **wyszukanie**, do której przekazana jest wartość kryterium wyszukiwania przez zmienną **fraza**. Wyszukiwanie rozpoczyna się na stronie http://www.isolution.pl/ .

7.3 Definiowanie zmiennych w skrypcie

W skrypcie istnieje możliwość definiowania zmiennych, które mogą służyć do przechowywania danych, przekazywania parametrów dla niektórych komend (np. w instrukcji warunkowej **if**) czy procedur.
Istnieją dwa typy zmiennych różniące się zasięgiem obowiązywania:

- **zmienne lokalne**:

„widoczne” są jedynie w pliku skryptu, w którym zostały zdefiniowane oraz w każdym importowanym pliku skryptu. Zmienne lokalne dodatkowo wykorzystywane są w pętlach sterowanych danymi (o których mowa dalej w dokumencie) i są one „widoczne” tylko wewnątrz nich.

- **zmienne globalne**:

zdefiniowane w dowolnym miejscu skryptu są „widoczne” w całym skrypcie.

Na zachowanie zmiennych ma wpływ komenda konfiguracyjna **config overrideGlobals**, która określa czy możliwe jest „przesłanianie” wartości zdefiniowanych zmiennych globalnych. Domyślnie jest niedozwolone a każda taka próba spowoduje wystąpienie wyjątku. W celu przekazania aktualnej wartości zmiennej jako parametru dla innej komendy należy użyć składni "***::nazwa***", gdzie *nazwa* – nazwa zmiennej.
Zmienne mogą być umieszczane wewnątrz ciągu tekstowego ograniczonego znakami **""**.

variable local

Komenda służy do deklaracji zmiennej lokalnej.

Składnia:
variable local *nazwa* = "*wartość*"
gdzie: *nazwa* – nazwa zmiennej; *wartość* – wartość nadana zmiennej.

Przykłady użycia:

```
variable local nazwa = "XYZ"
log "::nazwa"
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę tekstu: **XYZ**.

variable global

Komenda służy do deklaracji zmiennej globalnej.

Składnia:
variable global *nazwa* = "*wartość*"
gdzie: *nazwa* – nazwa zmiennej; *wartość* – wartość nadana zmiennej.

Przykłady użycia:

```
variable local nazwa = "XYZ"
log "abc::nazwa def"
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę tekstu: **abcXYZ def**

7.4 Komendy z użyciem zmiennych

increment

Komenda służy do zwiększenia wartości zmiennej o wartość 1 (inkrementacja).

Składnia:
increment "::*nazwa*"
gdzie: *nazwa* – nazwa zmiennej;

Przykłady użycia:

```
variable local test = "100"
log "::test"
increment "::test"
log "::test"
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę w kolejnych wierszach tekstu:
100
101

decrement

Komenda służy do zmniejszenia wartości zmiennej o wartość 1 (dekrementacja).

Składnia:
decrement "::*nazwa*"
gdzie: *nazwa* – nazwa zmiennej;

Przykłady użycia:

```
variable global test = "100"
log "::test"
decrement "::test"
log "::test"
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę w kolejnych wierszach tekstu:
100
99

read

Komenda służy do odczytania wartości tekstowej elementu na stronie wskazanego wyrażeniem ***xpath*** i umieszczenie jej w zmiennej lokalnej lub globalnej.

Składnia:
read local *"::nazwa" "xpath"*
read global *"::nazwa" "xpath"*
gdzie: *nazwa* – nazwa zmiennej; *xpath* - wyrażenie wskazujące położenie elementu w strukturze strony.

Przykłady użycia:

```
variable global xpath1 = "//body/div[1]/div/ul/li/a"
```

```
variable global xpath2 = "//div[1]/div/div[4]/ul/li[5]/a"
config driver = "firefox"
open "http://www.isolution.pl/"
timeout 10
maximize
wait 2000
read local nazwa1 from "::xpath1"
read global nazwa2 from "::xpath2"
log "::nazwa1"
log "::nazwa2"
exit
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę w kolejnych wierszach tekstu:
ISO 9001
SZKOLENIA

extract

Komenda umożliwia zgodnie z podanym wyrażeniem regularnym pobranie ze zmiennej jej fragmentu i zapisaniu go w zmiennej lokalnej lub globalnej.

Składnia:
extract "*wyrażenie regularne*" "*::nazwa*" local "*nazwa_zmiennej*"
extract "*wyrażenie regularne*" "*::nazwa*" global "*nazwa_zmiennej*"
gdzie: *nazwa* to nazwa zmiennej, której wynik działania wyrażenia regularnego zdefiniowanego w *wyrażenie regularne* zostanie zapisany w zmiennej o nazwie *nazwa_zmiennej*.

Przykłady użycia:

```
variable global element = "element01"
extract "^.{2}" "::element" global elem1
extract ".{2}$" "::element" global elem2
log "::elem1::elem2"
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę tekstu (dwa pierwsze i dwa ostatnie znaki ciągu znaków przechowywanego w zmiennej element):
el01

8. Zasilanie skryptu danymi

8.1 Zasilanie z plików zewnętrznych

foreach

Komenda służy do pobierania danych, których źródłem są pliki ***csv***, arkusze programu ***excel*** (zarówno w formacie ***xls*** jak i ***xlsx***). Zasilenie danymi odbywa się w pętli aż do pobrania wszystkich danych z pliku. W zależności od zastosowanego źródła danych sposób użycia instrukcji **foreach** nieznacznie różni się.

Dla pliku csv zawierającego w pierwszym wierszu nazwy kolumn:

Składnia:
foreach in *"nazwa_pliku_csv"*
komendy
next
gdzie: *nazwa_pliku_csv* – nazwa pliku w formacie csv; *komendy* – to zbiór kolejnych komend (ciało pętli); **foreach in**, **next** – to słowa kluczowe określające początek i koniec pętli.
Nazwy kolumn znajdujące się w pierwszym wierszu pliku ***csv*** posłużą jako nazwy zmiennych dostępnych wewnątrz pętli.

Dla pliku csv bez nazw kolumn:

Składnia:
foreach *"lista_zmiennych"* in *"nazwa_pliku_csv"*
komendy
next
gdzie: *lista_zmiennych* – lista zmiennych oddzielonych przecinkami; *nazwa_pliku_csv* – nazwa pliku w formacie csv; *komendy* – to zbiór kolejnych komend (ciało pętli); **foreach**, **in**, **next** – to słowa kluczowe określające początek i koniec pętli.
Ponieważ w pierwszym wierszu pliku ***csv*** nie ma nazw kolumn to należy podać nazwy zmiennych oddzielając je przecinkami.

Dla arkuszy pliku w formacie excel:

Składnia:
foreach *"nazwa_pliku_excel" "nazwa_arkusza"*
komendy
next

gdzie: *nazwa_pliku_excel* – nazwa pliku w formacie excel; ***nazwa_***arkusza – nazwa arkusza w pliku; *komendy* – to zbiór kolejnych komend (ciało pętli); **foreach**, **next** – to słowa kluczowe określające początek i koniec pętli.

W celu użycia arkusza w pliku ***excel*** jako źródła danych musi on zawierać odpowiednią strukturę i niezbędne wpisy jak przedstawiono przykładowo w poniższej tabeli:

	A	**B**	**C**	**D**	**E**
1	**4**	**1**			
2	**A, c, e**				
3					
4	**name**	**second_ame**	**surname**		**age**
5	**test**	**test**	**test**		**1**

W komórce **A1** arkusza należy umieścić numer wiersza, w którym znajdują się nazwy zmiennych, do których będą przypisywane wartości odczytywane z arkusza. W przykładzie powyżej wskazany został wiersz numer **4**. Dane powinny znajdować się bezpośrednio pod wierszem z nazwami zmiennych.
Komórka **B1** zawiera liczbę określającą ile w arkuszu znajduje się wierszy z danymi. W podanym przykładzie liczba **1** określa, że w arkuszu znajduje się tylko jeden wiersz z danymi. Nawet jeśli w arkuszu znajduje się więcej wierszy z danymi, w pętli zostanie wykorzystana tylko zdefiniowana ilość wierszy. Pozostałe wiersze zostaną zignorowane. Niedozwolone jest wprowadzanie pustych wierszy pomiędzy wierszami z danymi. Zalecane jest sformatowanie komórek z danymi jako tekst przed zapisem arkusza w celu uproszczenia interpretacji odczytywanych danych i uniknięcia potencjalnych błędów.
Trzecim wymaganym elementem arkusza jest w komórce **A2** lista nazw kolumn bezpośrednio oddzielonych przecinkami, nie mogą występować dodatkowe znaki oddzielające jak np. spacja czy znak tabulacji np. zawartość komórki **A2 [A,B]** jest poprawna ale **[A, B]** niepoprawna. Interpreter w trakcie odczytywania danych pod uwagę weźmie tylko kolumny wymienione na tej liście, wszelkie pozostałe kolumny zostaną zignorowane bez względu na to czy zawierają jakieś dane czy nie.
Ze względu na możliwość umieszczenia w pliku ***excel*** kilku arkuszy, oprócz nazwy pliku w komendzie pętli należy umieścić również nazwę arkusza, z którego będą pobierane dane. Komenda jest w stanie obsłużyć pliki ***excel*** zarówno w starym formacie ***xls*** jak i nowym ***xlsx***.

Przykłady użycia:

Zawartość pliku **dane1.scv**, który znajduje się w tym samym katalogu co uruchamiany skrypt:

```
login;password
admin;admin
test;test
```

Zawartość pliku **dane2.csv**, który znajduje się w tym samym katalogu co uruchamiany skrypt:

```
admin;admin
test;test
```

Zawartość arkusza o nazwie **Arkusz1** pliku **dane.xlsx**, który znajduje się w tym samym katalogu co uruchamiany skrypt:

3	2
A,B	
login	password
admin	admin
test	test

Zawartość uruchamianego skryptu:

```
config overrideGlobals="true"
foreach in "dane1.csv"
    log "::login"
    log "::password"
next
foreach "login,password" in "dane2.csv"
    log "::login"
    log "::password"
next
foreach in "dane.xlsx" "Arkusz1"
    log "::login"
    log "::password"
next
exit
```

Wynikiem wykonania skryptu jest wyprowadzenie na konsolę tekstu:

admin admin
test test
admin admin
test test
admin admin
test test

8.2 Zasilanie z bazy danych

foreach

Komenda służy do pobierania danych, której źródłem jest baza danych. Zasilenie danymi odbywa się w pętli aż do pobrania wszystkich danych, które są wynikiem zapytania SQL.

Składnia:

foreach *"lista_zmiennych"* in *"zapytanie_SQL" "połączenie_do_bazy"*
komendy
next

gdzie: *lista_zmiennych* – lista zmiennych oddzielonych przecinkami; *zapytanie_SQL* – nazwa pliku z zapytaniem SQL do bazy danych; *połączenie_do_bazy* – nazwa pliku (bez rozszerzenia *properties*) definiująca połączenie do bazy danych; *komendy* – to zbiór kolejnych komend (ciało pętli); **foreach**, **in**, **next** – to słowa kluczowe określające początek i koniec pętli.

Użycie komendy musi być poprzedzone:

- umieszczeniem odpowiedniego sterownika ***jar*** w katalogu ***ext***
- utworzeniem pliku o nazwie ***połączenie_do_bazy.properties***, który zawiera definicję połączenia do bazy danych
- wykonaniem komendy konfiguracji
  config database = *"połączenie_do_bazy.properties"*

Przykłady użycia:

```
config overrideGlobals = "true"
config database = "dbrozliczenia.properties"
foreach "log" in "users.sql" from "dbrozliczenia"
    log "uzytkownik: ::log"
next
exit
```

9. Generowanie danych losowych

9.1 Generowanie liczb

Generator umożliwia generowanie losowych liczb całkowitych. Może być zastosowany w każdym miejscu skryptu, w którym można stosować zmienne.

___randomInt

Opis działania komend:

- **___randomInt** - losowa liczba z zakresu Integer.MIN_VALUE – Integer.MAX_VALUE
- **___randomInt_X** – losowa liczba z zakresu 0 – X
- **___randomInt_X_Y** – losowa liczba z zakresu X – Y

gdzie: X, Y:

- dowolna liczba całkowita,

- Y musi być większe od X (jeśli warunek ten nie zostanie spełniony to wykonanie skryptu zostanie przerwane),

- mogą być zmiennymi.

Składnia:
variable *typ_zmiennej zmienna* = "___randomInt"
variable *typ_zmiennej zmienna* = "___randomInt_X"
variable *typ_zmiennej zmienna* = "___randomInt_X_Y"
gdzie: *typ_zmiennej* to jedna z wartości:

- **local**

- **global**

zmienna – nazwa zmiennej.

Przykłady użycia:

```
config overrideGlobals = "true"
variable global min = "50"
variable global max = "100"
variable global liczba = "___randomInt"
log "::liczba"
variable global liczba = "___randomInt_10"
log "::liczba"
variable local liczba = "___randomInt_::min_::max"
log "::liczba"
exit
```

9.2 Generowanie dat

Generator umożliwia generowanie losowych dat. Może być zastosowany w każdym miejscu skryptu, w którym można stosować zmienne. Daty zostaną wygenerowane zgodnie z formatem ustawionym przez komendę **config dateFormat** (lub formatem domyślnym ***dd/MM/yyyy***) i uzależnione od ustawień dokonanych przez komendy **config minDate** i **config maxDate**.

___today

Komenda służy do wygenerowania aktualnej daty.

Składnia:
variable *typ_zmiennej zmienna* = "___today"
gdzie: *typ_zmiennej* to jedna z wartości:
- **local**
- **global**

zmienna – nazwa zmiennej.

Przykłady użycia:

```
config overrideGlobals="true"
config dateFormat = "yyyy-MM-dd"
variable global data = "___today"
log "::data"
exit
```

___tomorrow

Komenda służy do wygenerowania jutrzejszej daty.

Składnia:
variable *typ_zmiennej zmienna* = "___tomorrow"
gdzie: *typ_zmiennej* to jedna z wartości:
- **local**
- **global**

zmienna – nazwa zmiennej.

Przykłady użycia:

```
config overrideGlobals="true"
config dateFormat = "yyyy-MM-dd"
variable global data = "___tomorrow"
log "::data"
exit
```

___yesterday

Komenda służy do wygenerowania wczorajszej daty.

Składnia:
variable *typ_zmiennej zmienna* = "___yesterday"
gdzie: *typ_zmiennej* to jedna z wartości:
- **local**
- **global**

zmienna – nazwa zmiennej.

Przykłady użycia:

```
config overrideGlobals="true"
config dateFormat = "yyyy-MM-dd"
variable global data = "___yesterday"
log "::data"
exit
```

___randomDate

Opis działania komend:

- **___randomDate** – dowolna data losowa
- **___randomDate_<_data** – losowa data z zakresu *minDate – data*
- **___randomDate_>_data** – losowa data z zakresu *data – maxDate*
- **___randomDate_data1_data2** - data z zakresu *data1 – data2*

Składnia:
variable *typ_zmiennej zmienna* = "___randomDate"
variable *typ_zmiennej zmienna* = "___randomDate_<_data"
variable *typ_zmiennej zmienna* = "___randomDate_>_data"
variable *typ_zmiennej zmienna* = "___randomDate_data1_data2"

gdzie: *typ_zmiennej* to jedna z wartości:
- **local**
- **global**

zmienna – nazwa zmiennej;
data – wartość daty lub jedno ze słów kluczowych: **today**, **tomorrow**, **yesterday**;
data1, **data2** - wartości dat.
Parametry: **data**, **data1**, **data2** mogą być również zmiennymi z przypisanymi wartościami.

Przykłady użycia:

```
config overrideGlobals="true"
config dateFormat = "yyyy-MM-dd"
config maxDate = "1500-01-01"
config maxDate = "2100-01-01"
variable global data1 = "2014-06-01"
variable global data2 = "2014-07-31"
variable global data = "___randomDate"
log "::data"
variable global data = "___randomDate_<_2014-07-31"
log "::data"
variable global data = "___randomDate_>_2014-06-01"
log "::data"
variable global data = "___randomDate_<_tomorrow"
log "::data"
variable global data = "___randomDate_2014-06-01_2014-07-31"
log "::data"
variable global data = "___randomDate_::data1_::data2"
log "::data"
exit
```

9.3 Generowanie danych z pliku tekstowego i bazy danych

___randomDictionaryValue

Komenda umożliwia losowe pobranie wiersza z pliku tekstowego. Tekst z wiersza zostanie przypisany zmiennej typu lokalnego lub globalnego.

Składnia:
variable *typ_zmiennej zmienna* = "___randomDictionaryValue_*path*"
gdzie: *typ_zmiennej* to jedna z wartości:

- **local**
- **globar**

zmienna – nazwa zmiennej; *path* - ścieżka do lokalizacji pliku tekstowego lub nazwa zmiennej z przypisaną wartością lokalizacji.

Przykłady użycia:
Plik **slownik.txt** znajduje się w tym samym katalogu co uruchamiany skrypt a jego zawartość przedstawiono poniżej:

```
elemnt01
elemnt02
elemnt03
```

elemnt04
elemnt05
elemnt06
elemnt07
elemnt08
elemnt09
elemnt10

Zawartość skryptu:

```
config overrideGlobals="true"
variable global nazwa = "slownik.txt"
variable global element = "___randomDictionaryValue_slownik.txt"
log "::element"
wait 1000
variable global element = "___randomDictionaryValue_::nazwa"
log "::element"
exit
```

Wynikiem wykonania skryptu jest wyprowadzenie na konsolę dwóch linii o treści losowo wybranych wierszy z pliku **slownik.txt**.

___randomQueryResult

Komenda umożliwia losowe pobranie danych z pierwszej kolumny wyników zapytania SQL do bazy danych. Pobrana dana zostanie przypisana zmiennej typu lokalnego lub globalnego.

Składnia:
variable *typ_zmiennej zmienna* = "___randomDictionaryValue_*BD_SQL*"
gdzie: *typ_zmiennej* to jedna z wartości:

- **local**
- **globar**;

zmienna – nazwa zmiennej; *BD* - ścieżka do lokalizacji pliku konfiguracyjnego (bez rozszerzenia *properties*) definiującego połączenie z bazą danych lub nazwa zmiennej z przypisaną wartością lokalizacji; *SQL* - ścieżka do lokalizacji pliku z zapytaniem SQL.

Przykłady użycia:

```
config database = "dbrozliczenia.properties"
config overrideGlobals="true"
variable global bd = "dbrozliczenia"
variable global sql = "usersall.sql"
```

```
variable global uzytkownik =
"___randomQueryResult_dbrozliczenia_usersall.sql"
log "::uzytkownik"
variable global uzytkownik = "___randomQueryResult_::bd_::sql"
log "::uzytkownik"
exit
```

10. Sterowanie wykonaniem skryptu

10.1 Instrukcje warunkowe

Instrukcje *if* do porównania wartości

Instrukcja warunkowa pozwalająca wykonać pewien blok kodu w przypadku spełnienia (bądź nie) określonego warunku. W warunku mogą być porównywane wartości zmiennych i stałych. Komendy w blokach **if** I **else** mogą zostać pominięte. Słowo kluczowe **else** jest wymagane nawet w przypadku pominięcia komend w tym bloku.

Składnia:
if *warunek* then *komendy1*
else *komendy2*
endif

if not *warunek* then *komendy1*
else *komendy2*
endif
gdzie: *komendy1*, *komendy2* – to ciąg komend w skrypcie;
warunek:
"wartosc1"* operator *"wartosc2"
gdzie: ***wartosc1***, ***wartosc2*** to porównywane wartości;
operator – to jeden z poniższych kluczy:
eq – *sprawdzenie czy wartosc1 = wartosc2*
ne – *sprawdzenie czy wartosc1 <> wartosc2*
le – *sprawdzenie czy wartosc1 <= wartosc2*
ge – *sprawdzenie czy wartosc1 >= wartosc2*
lt – *sprawdzenie czy wartosc1 < wartosc2*
gt – *sprawdzenie czy wartosc1 > wartosc2*
contains - *sprawdzenie czy wartosc2 jest podciągiem kolejnych znaków w wartosc1*

Uwagi:
Jeśli parametrami warunku (*wartosc1, wartosc2*) instrukcji są liczby to są porównywane liczby. Jeśli jednym z parametrów jest łańcuch znaków zawierający znak różny od cyfry to są porównywane jako teksty np. „999” < „A99”.

Przykłady użycia:

```
variable global liczba1 = "9"
variable global liczba2 = "10"
variable global string1 = "XYZ"
```

```
variable global string2 = "XZ"
if "::liczba1" le "::liczba2" then log "TRUE"
else log "FALSE"
endif
if "::string1" contains "::string2" then log "TRUE"
else log "FALSE"
endif
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę tekstu:
TRUE
TRUE

Instrukcje *if* do sprawdzenia elementu na stronie

W warunku zawarte jest sprawdzenie elementu na stronie.

Składnia:
if *warunek* then *komendy1*
else *komendy2*
endif

if not *warunek* then *komendy1*
else *komendy2*
endif
gdzie: *komendy1*, *komendy2* – to ciąg komend w skrypcie;
warunek:
"*xpath*" is *"stan"*
gdzie: ***xpath*** to wyrażenie wskazujące położenie elementu w strukturze strony;
stan – to jeden z poniższych kluczy:
visible –*sprawdzenie czy element wskazywany wyrażeniem xpath jest widoczny*
enabled –*sprawdzenie czy element wskazywany wyrażeniem xpath jest edytowalny*
selected – *sprawdzenie czy element wskazywany wyrażeniem xpath jest wybrany*

Przykłady użycia:

```
if "//td[text()='Dodaj']" is "visible" then log "TRUE"
else log "FALSE"
endif
```

Wynikiem wykonania powyższego skryptu jest wyprowadzenie na konsolę tekstu **TRUE** – jeśli na stronie jest widoczny tekst **Dodaj** lub **FALSE** jeśli jest niewidoczny.

10.2 Pętle

Komenda pętli służy do powtarzania bloku innych komend znajdujących się w jej wnętrzu (tzw. ciało pętli) aż do spełnienia określonego warunku. Każda pętla posiada swój własny

zasięg zmiennych co oznacza że zmienne zdefiniowane wewnątrz pętli są widoczne tylko wewnątrz tej pętli a także w skryptach importowanych wewnątrz pętli.

loop

Komenda służy do utworzenia prostej pętli wykonywanej ilość razy określonej przez jej parametr. Pętla **loop** posiada specjalną ukrytą zmienną o nazwie ***index*** przechowującej wartość liczbową określającą, która iteracja jest w danym momencie wykonywana. Wartości zmiennej ***index*** liczone są od zera. Wartości zmiennej ***index*** nie można modyfikować wewnątrz pętli.

Składnia:
loop *liczba*
komendy
endloop
gdzie: *liczba* – to liczba naturalna określająca ilość powtórzeń pętli; *komendy* – to zbiór kolejnych komend (ciało pętli); **loop**, **endloop** – to słowa kluczowe określające początek i koniec pętli.

Przykłady użycia:

```
log "Test petli loop"
loop 2
    log ”iteracja numer: ::index”
endloop
```

Wykonanie skryptu spowoduje wyprowadzenie na konsolę tekstu:
Test petli loop
iteracja numer: 0
iteracja numer: 1

while

Komenda służy do utworzenia pętli wykonywanej tyle razy dopóki spełniony jest warunek, który jest sprawdzany przed wykonaniem pierwszej komendy z wnętrza pętli (ciała pętli).

Składnia:
while *warunek*
komendy
endwhile

while not *warunek*
komendy
endwhile

gdzie: *warunek* – definicja warunku jest analogiczna jak dla instrukcji warunkowej **if** (patrz punkt ***10.1 Instrukcje warunkowe***); *komendy* – to zbiór kolejnych komend (ciało pętli); **not** – negacja logiczna**; while**, **endwhile** – to słowa kluczowe określające początek i koniec pętli.

Przykłady użycia:

```
config overrideGlobals="true"
variable global lpetli = "2"
log "Test petli while"
variable global lp = "0"
while "::lp" le "::lpetli" do
    log "iteracja numer: ::lp"
    increment "::lp"
endwhile
log "*****"
variable global lp = "0"
while not "::lp" gt "::lpetli" do
    log "iteracja numer: ::lp"
    increment "::lp"
endwhile
```

Wykonanie skryptu spowoduje wyprowadzenie na konsolę tekstu:

Test petli while
iteracja numer: 0
iteracja numer: 1
iteracja numer: 2
*****
iteracja numer: 0
iteracja numer: 1
iteracja numer: 2

repeat

Komenda służy do utworzenia pętli wykonywanej tyle razy dopóki spełniony jest warunek, który jest sprawdzany po wykonaniu ostatniej komendy z wnętrza pętli (ciała pętli). Pętla **repeat** wykona się zawsze przynajmniej jeden raz.

Składnia:
repeat
komendy
while *warunek* endwhile

repeat
komendy

while not *warunek* endwhile

gdzie: *warunek* – definicja warunku jest analogiczna jak dla instrukcji warunkowej **if** (patrz punkt ***10.1 Instrukcje warunkowe***); *komendy* – to zbiór kolejnych komend (ciało pętli); **not** – negacja logiczna**; repeat**, **endwhile** – to słowa kluczowe określające początek i koniec pętli.

Przykłady użycia:

```
config overrideGlobals="true"
variable global lpetli = "2"
log "Test petli repeat"
variable global lp = "2"
repeat
    decrement "::lp"
    log "iteracja numer: ::lp"
while "::lp" gt "0" endwhile
log "*****"
variable global lp = "0"
repeat
    log "iteracja numer: ::lp"
    increment "::lp"
while not "::lp" gt "::lpetli" endwhile
```

Wykonanie skryptu spowoduje wyprowadzenie na konsolę tekstu:
Test petli repeat
iteracja numer: 1
iteracja numer: 0
*****
iteracja numer: 0
iteracja numer: 1
iteracja numer: 2

10.3 Komendy weryfikacji

Komendy weryfikacji umożliwiają sprawdzenie czy podczas wykonywania skryptu zostały spełnione określone warunki. Nie spełnienie niektórych z nich powoduje natychmiastowe zakończenie wykonywania skryptu i zwrócenie wyjątku. Każde wejście w blok weryfikacji powoduje wpisy w plikach **verification.log** i **verification.html** utworzonych automatycznie po zakończeniu wykonywania skryptu.

dbcheck

Komenda umożliwia sprawdzenie stanu bazy danych z wartościami oczekiwanymi. Pierwszym parametrem komendy jest nazwa połączenia do bazy danych skonfigurowanego przy użyciu komendy **config database**, drugim

parametrem jest ścieżka do pliku zawierającego zapytanie SQL, trzeci parametr to ścieżka do pliku w formacie **csv** z oczekiwanymi danymi.

Składnia:

verify ***"Opis weryfikacji"***
dbcheck ***" połączenie_do_bazy " "zapytanie_SQL" "plik_scv"***
end

gdzie: *połączenie_do_bazy* – nazwa pliku (bez rozszerzenia *properties*) definiująca połączenie do bazy danych; *zapytanie_SQL* – ścieżka do pliku z zapytaniem SQL do bazy danych; *plik_scv* – ścieżka do pliku w formacie **csv** z oczekiwanymi danymi; **end** – to słowo kluczowe oznaczające koniec bloku weryfikacji.

Przykłady użycia:

```
config database = "dbrozliczenia.properties"
verify "Czy istnieje projekt o identyfikatorze ZZY ?"
dbcheck dbrozliczenia "projekt_identyfikator.sql" "wynik_zapytania_1.csv"
end
wait 2000
verify "Czy istnieje projekt o identyfikatorze ZZZ ?"
dbcheck dbrozliczenia "projekt_identyfikator.sql" "wynik_zapytania_2.csv"
end
wait 2000
exit
```

11. Generowane pliki wyjściowe

Pliki tworzone w wyniku wykonania skryptu można podzielić na:

- generowanie automatycznie przez narzędzie ATSuite:
  - ***execution.log***
  - ***verification.log***
  - ***verification.html***
- utworzone w wyniku wykonania w skrypcie następujących komend: **screenshot**, **record**, **dumpvars**, **debug**
- utworzone przez umieszczenie w wykonywanym skrypcie komentarzy dokumentujących: pliki ***scenarios.xml***, ***scenario.xsl***. Zostało to opisane punkcie ***12.2 Komentarze do generowania dokumentacji testowej***.

W nazwie każdego pliku (oprócz pliku utworzonego komendą **record** i w wyniku komentarzy dokumentujących) znajduję się data i czas jego utworzenia w formacie:
DD-MM-hhmmss, gdzie: *D* – cyfra dnia; *M* – cyfra miesiąca; *h* – cyfra godziny; *m* – cyfra minut; *s* – cyfra sekund.

11.1 Pliki generowane automatycznie

execution.log

Dla zakończonego skryptu tworzony jest plik ***execution.log*** zawierający historię wykonania komend wraz ze zmiennymi i stałymi użytymi w skrypcie.

```
15:54:04.453Starting test15:54:04.671 bp_debug.txt Variable(global,xpath1,//body/div[1]/div/ul/li/a)
15:54:04.671 bp_debug.txt Variable(global,xpath2,//div[1]/div/div[4]/ul/li[5]/a)
15:54:04.671 bp_debug.txt Driver(firefox)
15:54:10.203 bp_debug.txt Timeout(10)
15:54:10.203 bp_debug.txt Open(http://www.isolution.pl/)
15:54:10.953 bp_debug.txt Maximize
15:54:11.281 bp_debug.txt Wait(2000)
15:54:13.281 bp_debug.txt Debug
15:54:13.281 bp_debug.txt Read(local,nazwa1,::xpath1)
15:54:13.359 bp_debug.txt Read(global,nazwa2,::xpath2)
15:54:13.390 bp_debug.txt Logg(::nazwa1)
15:54:13.390 bp_debug.txt Logg(::nazwa2)
15:54:13.390 bp_debug.txt Exit
15:54:13.390Test Finished
```

Rysunek 7 Przykładowa zawartość pliku *execution.log*

verification.log i verification.html

Dla zakończonego skryptu tworzone są pliki ***execution.log*** **i** ***verification.html*** zawierające wpisy dotyczące wykonanych komend weryfikacji.

```
1  Verifying
2  Czy pole wyszukiwarki jest aktywne ?
3
4  Verification passed.
5  Verification done.
6  Verifying
7  Czy jest link SZKOLENIA ?
8
9  Verification passed.
10 Verification done.
```

Rysunek 8 Przykładowa zawartość pliku *verification.log*

Verification Log 15-06-2014 16:11:51

Czy pole wyszukiwarki jest aktywne ?		
Verification Check	**Verification Type**	**Result**
Element: //div[1]/div/div[3]/form/div/input[1] was enabled.	Mandatory	SUCCESS

Czy jest link SZKOLENIA ?		
Verification Check	**Verification Type**	**Result**
Element: //div[1]/div/div[4]/ul/li[5]/a did exist.	Mandatory	SUCCESS

Rysunek 9 Przykładowa zawartość pliku *verification.html*

11.2 Pliki generowane przez wykonanie komend w skrypcie

screenshot

Komenda służy do przechwycenia zrzutu ekranu z przeglądarki w momencie wywołania komendy. Jako parametr komendy należy podać ścieżkę do pliku, w którym zostanie zapisany zrzut z okna przeglądarki. Domyślnie do nazwy pliku graficznego zostanie dodane rozszerzenie ***png***.

Składnia:
screenshot *"nazwa_pliku"*
gdzie: *nazwa_pliku* –to ścieżka do pliku.

Przykłady użycia:

```
config driver = "firefox"
config overrideGlobals="true"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
screenshot "\ocr\Isolution"
exit
```

W wyniku wykonania skryptu zostanie utworzony plik o nazwie:
Isolution*dd-MM-hhmmss***.png** w katalogu ***ocr***,

gdzie *DD-MM-hhmmss* – to data i czas utworzenia pliku; **png** – rozszerzenie nazwy pliku.

record

Komenda służy do nagrania filmu z wykonania skryptu.
Jedynym parametrem komendy jest czas nagrywania w sekundach, do którego wlicza się również czas inicjalizacji mechanizmu. Plik z filmem zostanie zapisany w formacie ***mov*** w katalogu głównym. Utworzony plik ma nazwę zgodnie z formatem: **ScreenRecording *RRRR-MM-DD* at *gg.mm.ss*.mov**
gdzie: *RRRR-MM-DD* to data i czas utworzenia pliku z filmem (*R* - cyfra roku, *M* - cyfra miesiąca, *D* - cyfra dnia); *gg.mm.ss* to czas nagrania (*g* - cyfra godziny, *m* - cyfra minut, *s* - cyfra sekund).
Uwagi: Jeśli skrypt skończy się przed zakończeniem działania mechanizmu nagrywającego plik nie zostanie zapisany.

Składnia:
record *"czas"*
gdzie: *czas* – liczba naturalna określająca czas w sekundach nagrywania filmu.

Przykłady użycia:

```
config driver = "firefox"
config overrideGlobals="true"
timeout 10
open "http://www.isolution.pl/"
maximize
record 6
wait 5000
exit
```

W wyniku wykonania skryptu zostanie utworzony plik z filmem o czasie nagrania 6 sekund. Plik znajduje się w katalogu głównym (tam gdzie skrypt). Nazwa pliku to **ScreenRecording *RRRR-MM-DD* at *gg.mm.ss*.mov**
gdzie: *RRRR-MM-DD gg.mm.ss* – data i czas utworzenia pliku.

dumpvars

Komenda służy do zapisania (eksportu) w pliku nazw zmiennych z przypisanymi im wartościami. Wynikiem wykonania komendy jest utworzenie pliku o nazwie podanej jako parametr. Do nazwy pliku automatycznie zostanie dodane rozszerzenie **properties**. Plik zawiera nazwy wszystkich zmiennych występujących w skrypcie wraz z odpowiadającymi im wartościami jakie osiągnęły w momencie zakończenia jego wykonywania.

Składnia:
dumpvars *"nazwa_pliku"*

gdzie: *nazwa_pliku* –to ścieżka do pliku.

Przykłady użycia:

```
variable global xpath1 = "//body/div[1]/div/ul/li/a"
variable global xpath2 = "//div[1]/div/div[4]/ul/li[5]/a"
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
read local nazwa1 from "::xpath1"
read global nazwa2 from "::xpath2"
log "::nazwa1"
log "::nazwa2"
dumpvars "zmienne"
exit
```

```
1  #
2  #Sun Jun 15 18:00:09 CEST 2014
3  xpath2=//div[1]/div/div[4]/ul/li[5]/a
4  xpath1=//body/div[1]/div/ul/li/a
5  nazwa2=SZKOLENIA
6  nazwa1=ISO 9001
```

Rysunek 10 Przykładowa zawartość pliku ze zmiennymi po wykonaniu komendy *dumpvars*

debug.log i debugSession.isds

Pliki ***debug.log*** i ***debugSession.isds*** tworzone są jeśli w skrypcie użyto komendy **debug**. Pliki te zawierają dodatkowe informacje niezbędne do analizy potencjalnych błędów Interpretera.

```
15:54:13.281 bp_debug.txt Read(local,nazwa1,::xpath1)
VARIABLES:
xpath2 -> //div[1]/div/div[4]/ul/li[5]/a
xpath1 -> //body/div[1]/div/ul/li/a

15:54:13.328 class pl.isolution.script.ScriptRunner$$anon$2: finding Element
15:54:13.343 class pl.isolution.script.ScriptRunner$$anon$2: Element found without Exceptions
15:54:13.343 class pl.isolution.script.ScriptRunner$$anon$2: Check if element is displayed.
15:54:13.359 bp_debug.txt Read(global,nazwa2,::xpath2)
VARIABLES:
xpath2 -> //div[1]/div/div[4]/ul/li[5]/a
xpath1 -> //body/div[1]/div/ul/li/a
nazwa1 -> ISO 9001
15:54:13.375 class pl.isolution.script.ScriptRunner$$anon$2: finding Element
15:54:13.375 class pl.isolution.script.ScriptRunner$$anon$2: Element found without Exceptions
15:54:13.375 class pl.isolution.script.ScriptRunner$$anon$2: Check if element is displayed.
15:54:13.390 bp_debug.txt Logg(::nazwa1)
VARIABLES:
xpath2 -> //div[1]/div/div[4]/ul/li[5]/a
nazwa2 -> SZKOLENIA
xpath1 -> //body/div[1]/div/ul/li/a
nazwa1 -> ISO 9001
15:54:13.390 bp_debug.txt Logg(::nazwa2)
VARIABLES:
xpath2 -> //div[1]/div/div[4]/ul/li[5]/a
nazwa2 -> SZKOLENIA
xpath1 -> //body/div[1]/div/ul/li/a
nazwa1 -> ISO 9001
15:54:13.390 bp_debug.txt Exit
VARIABLES:
xpath2 -> //div[1]/div/div[4]/ul/li[5]/a
nazwa2 -> SZKOLENIA
xpath1 -> //body/div[1]/div/ul/li/a
nazwa1 -> ISO 9001
```

Rysunek 11 Fragment przykładowego pliku *debug.log*

12. Komentarze

12.1 Komentarze zwykłe

W skrypcie można umieszczać komentarze (adnotacje). W skryptach o dużej liczbie komend i złożonej strukturze dodawanie komentarzy jest bardzo użyteczne.
Linia komentarza zaczyna się od znaków **//**.

Przykłady użycia:
// To jest komentarz

12.2 Komentarze do generowania dokumentacji testowej

W skrypcie można dodatkowo umieszczać komentarze, które służą do wygenerowania dokumentów zawierających scenariusze testowe. Poniżej znajduje się lista **znaczników**, które można umieszczać w skrypcie w celu utworzenia dokumentacji testowej.
//!scenario – *zaznacza początek scenariusza*
//!identifier – *identyfikator scenariusza*
//!name – *nazwa scenariusza*
//!author – *autor scenariusza*
//!version – *wersja scenariusza*
//!type – *typ scenariusza (podstawowy, alternatywny)*
//!project – *projekt, dla którego jest scenariusz testowy*
//!actor – *aktor (może być wielu, każdy w osobnym komentarzu)*
//!goal – *cel scenariusza*
//!requirement – *wymaganie sprawdzane przez scenariusz (może być ich wiele, każde w osobnym komentarzu)*
//!useCase – *przypadek użycia weryfikowany przez scenariusz (może być ich wiele, każde w osobnym komentarzu)*
//!openingCondition – *wymaganie wstępne dla danego scenariusza (może być ich wiele, każde w osobnym komentarzu)*
//!step – *krok przebiegu scenariusza (może być ich wiele, każde w osobnym komentarzu)*
//!expectedResult – *spodziewany wynik scenariusza (może być ich wiele, każde w osobnym komentarzu)*
//!end – *znacznik określający koniec scenariusza*

W celu utworzenia dokumentu zawierającego scenariusze testowe należy w wierszu poleceń ***cmd*** wykonać komendę:
java -jar "ATSuite-1.0.jar gen" *"nazwa_skryptu"*
gdzie: ATSuite-1.0.jar – to aktualna wersja Interpretera; *nazwa_skryptu* – nazwa skryptu zawierająca odpowiednie komendy dokumentujące.

W wyniku wykonania komendy powstaną dwa dodatkowe pliki ***scenarios.xml*** i ***scenario.xsl***.
Wszystko co znajduje się pomiędzy znacznikami **scenario** i **end** należy do jednego scenariusza, chyba że wewnątrz będzie kolejna para **scenario** i **end**, która ogranicza kolejny osobny scenariusz.

Uwagi:

- Wykonanie komendy ***java -jar "ATSuite-1.0.jar gen" "nazwa_skryptu"*** nie uruchamia skryptu (komendy inne niż komentarze dokumentujące nie są wykonywane)
- Po znacznikach **//!scenario** i **//!end** nie wolno umieszczać komentarzy
- Umieszczanie komentarzy dokumentujących w procedurach nie jest wskazane

Składnia:
//znacznik *"komentarz"*
gdzie: znacznik – to jeden z wymienionych znaczników; *komentarz* – adekwatny do znacznika komentarz.

Przykłady użycia:
Postać skryptu znajdującego się w pliku **ISOL-PT-0001.txt**:

```
//!scenario
//!author "ISOLUTION"
//!project "ATSuite"
//!identifier "ISOL-PT-0001"
//!version "Wersja 3"
//!type "podstawowy"
//!name "Test wyszukiwarki"
//!actor "SYSTEM"
//!actor "UZYTKOWNIK"
//!goal "Wyszukiwanie tekstu na podstawie kryterium -szukana fraza-"
//!requirement "Możliwość wyszukiwania podanej frazy na stronie http://www.isolution.pl/"
//!useCase "Uruchomienie wyszukiwrki"
config driver = "firefox"
config overrideGlobals="true"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
//!openingCondition "Znany jest adres strony internetowej"
//!step "1. Otwórz stronę internetową http://www.isolution.pl/"
variable global fraza = "projekt"
```

```
call wyszukanie
//!step "2. Do pola wyszukiwarki "-szukana fraza-" wprowadź tekst"
//!step "3. Kliknij ikonę "lupy" uruchamiając wyszukiwanie"
variable global fraza = "automatyzacja"
call wyszukanie
wait 5000
//!expectedResult "Prezentacja strony z wyszukanym tekstem"
//!end
exit
proc wyszukanie
wait 1000
clearx "//div[1]/div/div[3]/form/div/input[1]"
wait 1000
writex "//div[1]/div/div[3]/form/div/input[1]" "::fraza"
wait 1000
clickx "//div[1]/div/div[3]/form/div/input[2]"
wait 1000
back
wait 2000
endp
```

W wyniku uruchomienia komendy z wiersza poleceń *cmd*:
java -jar "ATSuite-1.0.jar gen" *"nazwa_skryptu"*
w katalogu głównym zostaną utworzone pliki ***scenarios.xml*** i ***scenario.xsl***.
Poniżej przedstawiono przykładową zawartość pliku ***scenarios.xml***.

```xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="scenario.xsl"?>
<scenarios>
<scenario>
<identifier>"ISOL-PT-0001"</identifier>
<name>"Test wyszukiwarki"</name>
<author>"ISOLUTION"</author>
<project>"ATSuite"</project>
<version>"Wersja 3"</version>
<goal>"Wyszukiwanie tekstu na podstawie kryterium -szukana fraza-"</goal>
<scenarioType>"podstawowy"</scenarioType>
<actors>
<actor>"SYSTEM"</actor>
<actor>"UZYTKOWNIK"</actor>
</actors>
<requirements>
<requirement>"Możliwość wyszukiwania podanej frazy na stronie http://www.isolution.pl/"</requirement>
</requirements>
<useCases>
<useCase>"Uruchomienie wyszukiwrki"</useCase>
</useCases>
<openingConditions>
<openingCondition>"Znany jest adres strony internetowej"</openingCondition>
</openingConditions>
<expectedResults>
<expectedResult>"Prezentacja strony z wyszukanym tekstem"</expectedResult>
</expectedResults>
<steps>
<step>"1. Otwórz stronę internetową http://www.isolution.pl/"</step>
<step>"2. Do pola wyszukiwarki "-szukana fraza-" wprowadź tekst"</step>
<step>"3. Kliknij ikonę "lupy" uruchamiając wyszukiwanie"</step>
</steps>
</scenario>
</scenarios>
```

Rysunek 12 Zawartość przykładowego pliku *scenarios.xml*

13. Uruchomienie Interpretera z pliku wsadowego *bat*

W celu uruchomienia Interpretera z pliku wsadowego należy zdefiniować plik z rozszerzeniem ***bat*** zawierający polecenie uruchomienia skryptu i następnie umieścić go w katalogu ***SCRIPTS***.
Poniżej przedstawiono zawartość przykładowego pliku o nazwie **run_test.bat**:
Przykłady użycia:
Dla przygotowanego środowiska jak na **Błąd! Nie można odnaleźć źródła odwołania.** w katalogu ***SCRIPTS*** należy umieścić pliki: **run_test.bat** ***,scripts_01.txt***, ***danetestowe.properties***.

Zawartość przykładowego pliku **run_test.bat** znajdującym się w katalogu ***SCRIPTS***:

```
@echo off
cls
echo TESTY AUTOMATYCZNE
java -jar "..\conf\ATSuite-1.0.jar" "script_01.txt"
echo KONIEC
pause
```

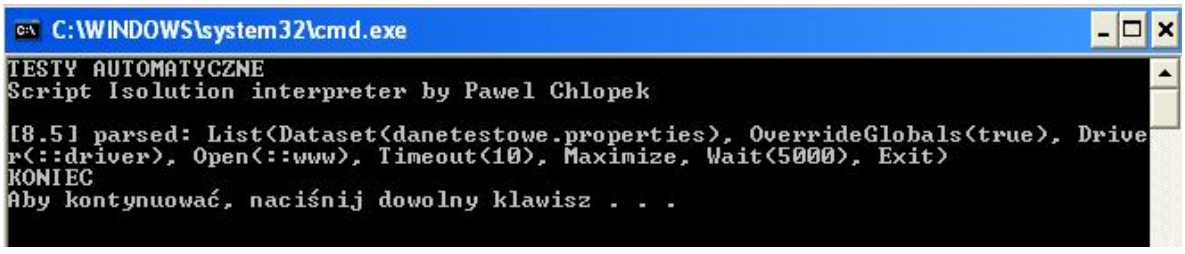
Zawartość przykładowego pliku **scripts_01.txt** znajdującym się w katalogu ***SCRIPTS***:

```
dataset "danetestowe.properties"
config overrideGlobals = "true"
config driver = "::driver"
open "::www"
timeout 10
maximize
wait 5000
exit
```

Zawartość pliku **danetestowe.properties** znajdującym się w katalogu ***SCRIPTS***:

```
driver = firefox
www = http://www.isolution.pl/
```

W celu wywołania polecenia z pliku wsadowego należy podwójnie kliknąć nazwę pliku **run_test.bat**. Poniżej przedstawiono log z wyniku wykonania skryptu uruchomionego z pliku wsadowego ***bat***. Wynikiem uruchomienia jest pokazanie przez 5 sekund internetowej strony http://www.isolution.pl/ firmy Isolution.



Rysunek 13 Log z wykonania przykładowego skryptu z pliku wsadowego *bat*

14. Inne komendy

14.1 scrollup

Komenda służy do przewinięcia strony w górę o liczbę pikseli podaną w parametrze komendy.

Składnia:
scrollup *piksele*
gdzie: *piksele* – liczba naturalna określająca liczbę pikseli.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
scrollup 50
wait 1000
scrollup 50
wait 5000
exit
```

14.2 scrolldown

Komenda służy do przewinięcia strony w dół o liczbę pikseli podaną w parametrze komendy.

Składnia:
scrolldown *piksele*
gdzie: *piksele* – liczba naturalna określająca liczbę pikseli.

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
scrolldown 50
wait 1000
scrolldown 50
wait 5000
exit
```

14.3 specialkey

Komenda służy do zasymulowania wciśnięcia na klawiaturze jednego ze specjalnych klawiszy.

Uwagi:

- Użycie komendy musi być poprzedzone wykonaniem komendy **find.**

Składnia:
specialkey *"nazwa_przycisku"*
gdzie: *nazwa_przycisku*:
enter - *przycisk Enter*
tab - *przycisk Tab*
pgup - *przycisk PgUp*
pgdown - *przycisk PgDn*
arrowup - *przycisk* ↑
arrowdown - *przycisk* ↓
arrowleft - *przycisk* ←
arrowright - *przycisk* →
home - *przycisk Home*
end - *przycisk End*

Przykłady użycia:

```
config driver = "firefox"
timeout 10
open "http://www.isolution.pl/"
maximize
wait 2000
find //body/div[2]
specialkey "pgdown"
wait 1000
find //body/div[2]
specialkey "pgpup"
wait 1000
find //body/div[2]
specialkey "arrowdown"
wait 1000
find //body/div[2]
specialkey "arrowup"
wait 5000
exit
```

14.4 dragndrop

Komenda służy do zasymulowania operacji przeciągnięcia (*drag*) i upuszczenia (*drop*) jednego elementu graficznego na inny. Parametrami komendy są wyrażenia ***xpath***, które wskazują na wybrane fragmenty obrazu strony.

Składnia:
dragndrop ***"xpath1" "xpath2"***
gdzie: x*path1, xpath2* – wyrażenia *xpath* wskazujące wybrane fragmenty obrazu strony.

Przykłady użycia:
dragndrop "//tr[7]/td[1]/div/nobr/table/tbody/tr/td[2]/img[2]" "//tr[5]/td[1]/div/nobr/table/tbody/tr/td[2]/img[2]"

14.5 submit

Komenda służy do zatwierdzenia formularza wyszukanego przy użyciu komendy **find** lub formularza, do którego należy wyszukany element. Przed użyciem komendy **submit** konieczne jest wykonanie komendy **find**, której parametrem jest wskazanie na formularz lub element formularza.

Składnia:
submit

Przykłady użycia:

```
dataset "danetestowe.properties"
config driver = "::driver"
timeout 10
open "::www"
maximize
wait 2000
maximize
find name "j_username"
write "::login"
find name "j_password"
write "::password"
submit
wait 3000
import "wylogowanie.txt"
wait 3000
import "wylogowanie.txt"
exit
```

14.6 submitx

Komenda jest odpowiednikiem komendy **submit** ale wymaga podania parametru będącego wyrażeniem *xpath*, które wskazuje na formularz lub element formularza.

Składnia:
submit *"xpath"*
gdzie: *xpath* – wyrażenie *xpath* wskazujące na formularz lub element formularza w strukturze strony.

Przykłady użycia:

```
dataset "danetestowe.properties"
config driver = "::driver"
timeout 10
open "::www"
maximize
wait 2000
maximize
find name "j_username"
write "::login"
find name "j_password"
write "::password"
submitx "//div[2]/div[1]/div/div/div/div[2]/form/div"
wait 3000
import "wylogowanie.txt"
exit
```

14.7 query

Komenda służy do wykonania zapytania SQL. Nie zwraca ona żadnych wyników dlatego sugerowane jest używanie jej do wykonywania zapytań typu *INSERT* i *UPDATE*. Jako pierwszy parametr komendy należy podać nazwę połączenia do bazy danych skonfigurowanego poleceniem **config database**. Drugi parametr to nazwa pliku z zapytaniem SQL.

Składnia:
query *połączenie_do_bazy "zapytanie_SQL"*
gdzie: *połączenie_do_bazy* – nazwa pliku (bez rozszerzenia properties) definiująca połączenie do bazy danych; *zapytanie_SQL* – nazwa pliku z zapytaniem SQL do bazy danych.
Przykłady użycia:

```
query dbrozliczenia "users.sql"
```

14.8 beanshell

Zaawansowany użytkownik może sam zdefiniować w pliku dodatkową funkcjonalność korzystając ze skryptu **beanshell. Taki skrypt następnie może uruchomić komendą, której parametrem jest nazwa pliku.**

Składnia:
beanshell *"nazwa pliku"*
gdzie: *"nazwa pliku"* – to nazwa pliku zdefiniowanego przez użytkownika.