JDI UI Test Framework Documentation

JDI (Just Do It) – is the test Framework for automation of UI applications testing. It extends Page Objects design pattern and introduces lot of additional elements with implementation of its common usages.

Framework used concept “Easy things should be easy, and hard things should be possible” Larry Wall (c).

So all elements and abilities has default realization that mostly used, but in case your application used some actions in different way, you can override this behavior on any level (just for this element, for all elements with same type or even all elements actions scenarios usage).

Similar you can use any kind of external tools and frameworks for related functionality (different loggers, reporting tools, drivers test runners and asserters)

We try to do test process easier and full of joy☺

# Framework Settings

Framework have following settings for integration with external tools. All this settings have default implementation but you can use any tool you want. Just implement interface for use it in Framework

* driverFactory – see details Driver Factory. By default used SeleniumWebDriver
* logger – see details in Loggers. By default used TestNG+Log4j loggers
* asserter – see details in Asserters. By default used TestNG with screenshots at exception
* testRunner - see details in Test Runners. By default used TestNG
* timeouts – timeouts used in framework. By default used:
  + waitElementSec = 3 (seconds) – wait element on already loaded page
  + waitPageLoadSec = 20 (seconds) – wait element if page reloads (including ajax load)
  + retryMSec = 100 (milliseconds) – retry timeout for fast access needs

# Driver Factory

Now Used Only Selenium Drivers Factory. Ability to have different driver factories not implemented now.

Idea is to add ability to use different Drivers in framework for accessing to Application

Possible drivers (in future):

* Appium Driver – implements work with mobile applications
* WinApi Driver – implements with win desktop applications using WinApi
* JS Driver – implements all access to web elements using JavaScript
* Autoit Driver – implements with win desktop applications using Autoit framework
* RobotFramework – implements with win desktop and web applications using Robot Framework
* Selenium Coordinates driver – using selenium Actions to navigate elements by coordinates (no locators)
* Selenide Driver -
* Sikuli Driver

# Test Runners

Not implemented now. Idea to add possibility do additional action before/after test suit/scenario by default.

E.g.

* Log Test Suit started.
* Log Test started with params.
* Process test scenario failure
* Log TestSuit finished with some statistic of test Run.
* …

# Asserters

You can use different ways to throw assertions by framework.

Default assertions are:

* TestNG
* JUnit

\* For default asserters you can do screenshot if exception thrown (see constructor param)

But you can use your own way for example by throwing new Exception() or something else

You just need implement IAsserrter interface:

* exception – for throwing exceptions
* silent – for avoid adding throws Exception in methods signature but you can throw it (e.g.using assertTrue(false) from testNG or Junit) or really hide and just add message in log for example
* areEqualsuals – for comparing object values (used in framework)
* matches – for check is actual string matches to regEx (used in framework)
* contains – for check is actual contain expected string (used in framework)

# Loggers

Framework provides list of default loggers but you can add your logger if you have better or more habitual for you. You just need implement ILogger interface. But if you want to add your logger the better way is just extend AbstractLogger where you just implement Inlog method in this case you save all logic related to showing log strings for different loglevels and types

## Log methods info

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Log method | Description | Log Level | Log Prioroty | Log Info Type |
| init | Used in test suit init block or for any other test data initialization goals like dataProviders or driver initialization | FATAL | 0 | BUSINESS |
| suit | Used for logging features description | FATAL | 0 | BUSINESS |
| test | Used for logging test start action (like test name and arguments) | FATAL | 0 | BUSINESS |
| step | Used for logging Steps of tests | FATAL | 0 | BUSINESS |
| fatal | Used for logging Unexpected Errors | FATAL | 0 | BUSINESS |
| error | Used for loggind critical expected errors (bad elements initialization) | ERROR | 3 | Custom |
| warning | Using for not critical errors as info | WARNING | 4 | Custom |
| info | Used for simple information | INFO | 6 | FRAMEWORK |
| debug | Used for technical messages | DEBUG | 7 | TECHNICAL |

Default loggers (extends AbstractLogger):

* ConsoleLogger – writes log only in console
* Log4JLogger – writes log in log4J log (you can tune work of log4J in standard way)
* TestNGLogger – adds info to testNG executing log

In addition, you can use ListLogger if wants to use set of loggers.

For loggers inherited from AbstractLogger you also can set following features:

* Remove duplicated lines from log – if this feature turned on (by default it is) then similar log lines from similar code line not printed in log. This is useful if logger falls in cycle and prints in log same lines

You can turn off this feature using forbidDuplicatedLines()

e.g.

You have following code:

|  |  |  |
| --- | --- | --- |
| Code | Log view  Remove duplicates On | Log view  Remove duplicates Off |
| for (int i = 1; I <=3; i++)  logger.info(“some text”) | some text | some text  some text  some text |
| Line 10: logger.info(“some text”)  Line 20: logger.info(“some text”) | some text  some text | some text  some text |
| Class A: logger.info(“some text”)  Class B: logger.info(“some text”) | some text  some text | some text  some text |
| for (int i = 1; I <=3; i++) {  logger.info(“some text”)  logger.info(“other text”)  } | some text  other text | some text  other text  some text  other text  some text  other text |

* LogSettings – you can set different setting for your loggers:

\* Logic below implemented only for Loggers based on AbstractLogger

* + LogLevel – is the lowest level of allowed logging

\* Highest priority is -1 and lowest is 100

No log inputs with lower priority added in log stream

|  |  |  |  |
| --- | --- | --- | --- |
| Priority | Log level | Description | Allowed methods |
| -1 | OFF | No logging |  |
| 0 | FATAL | Unexpected errors | Init, suit, test, step, fatal |
| 3 | ERROR | Critical errors | Init, suit, test, step, fatal, error |
| 4 | WARNING | Erorrs due to wrong params | Init, suit, test, step, fatal, error, warning |
| 6 | INFO | Actions Info | Init, suit, test, step, fatal, error, warning, info |
| 7 | DEBUG | Debug info (not for prod) | Init, suit, test, step, fatal, error, warning, info, debug |
| 100 | ALL | All log messages | Init, suit, test, step, fatal, error, warning, info, debug |

* + LogInfoType – is the type of logged information

|  |  |
| --- | --- |
| Type | Description |
| BUSINESS | Used for business messages like features or step invocations |
| FRAMEWORK | Used for Framework messages like click, getText, getColumn |
| TECHNICAL | Used for low level useful in debug info like getWebElement or driver init/quit |

You can set list of messages type in any order for logging

\* By default logged all types

See LogInfoType to log methods mapping here

|  |  |
| --- | --- |
| Log method | Value |
| init, suit, test, step, fatal | BUSINESS |
| error, warning, toLog | Custom |
| info | FRAMEWORK |
| debug | TECHNICAL |

# Elements

## Common

#### Text

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| getText() | Get element’s text | String |
| waitText(String) | Wait while element’s text contains expected text. Returns element’s text | String |
| waitMatchText(String) | Wait while element’s text matches regEx. Returns element’s text | String |
| HaveValue; Element | See methods by link |  |

#### Button / Label

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| click() | Click on button |  |
| Text; HaveValue; Element | See methods by link |  |

#### Link

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| click() | Click on link |  |
| getReference() | Get link destination | String |
| waitReference(String) | Wait while link destination contains expected text. Returns link destination | String |
| waitMatchReference(String) | Wait while link destination contains expected text. Returns link destination | String |
| getTooltip | Get links tooltip |  |
| Text; HaveValue; Element | See methods by link |  |

#### Image

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| click() | Click on image |  |
| getSource() | Get image source | String |
| getTip() | Get image tip/alt/hint text | String |
| Element | See methods by link |  |

#### Checkbox

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| check() | Set checkbox checked |  |
| uncheck() | Set checkbox unchecked |  |
| isChecked() | Verify is checkbox checked | Boolean |
| click() | Change checkbox state |  |
| SetValue; HaveValue; Element | See methods by link |  |

#### TextField / DatePicker / FileInput

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| input(String) | Input text in textfield |  |
| sendKeys(String) | Input text in textfield |  |
| newInput(String) | Clear and input text in textfield |  |
| clear() | Clear textfield |  |
| focus() | Focus(click) on textfield |  |
| SetValue; Text; HaveValue; Element | See methods by link |  |

#### TextArea

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| inputLines(String…) | Clear textarea and Input several lines of text in textarea | String[] |
| addNewLine(String) | Add text in textarea from new line |  |
| getLines() | Get lines of text in textarea | String[] |
| getLines().length | Get count of lines in textarea |  |
| TextField; SetValue; Text; HaveValue; Element | See methods by link |  |

## Complex

#### Selector<TEnum> / RadioButtons<TEnum> / Dropdown<TEnum> / Tabs<TEnum> / Menu<TEnum>

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| select(TEnum)  select(String)  select(int) | Select element with name (use enum/text/index) from list |  |
| getOptions()  getNames()  getValues() | Get labels of all options | List<String> |
| getOptionsAsText() | Get all options labels in one string separated with “; ” | String |
| getSelected() | Get name of the selected element | String |
| getSelectedIndex() | Get index of the selected element | Int |
| isSelected(TEnum)  isSelected(String) | Is option (from enum/text/int) selected? |  |
| waitSelected(TEnum)  waitSelected(String) | Wait while option (from enum/text/int) selected. Returns false if this not happens | Boolean |
| SetValue; HaveValue; | See methods by link |  |

#### CheckList / DropList

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| select(TEnum …)  select(String…)  select(int…) | Select options with name (use enum/text/index) from list (change their state selected/deselected) |  |
| check (TEnum …)  check (String…)  check (int…) | Check only specified options (use enum/text/index) from list (all other options unchecked) |  |
| uncheck(TEnum …)  uncheck (String…)  uncheck (int…) | Uncheck only specified options (use enum/text/index) from list (all other options checked) |  |
| checkAll()  selectAll() | Set all options checked |  |
| uncheckAll()  clear() | Set all options unchecked |  |
| getOptions()  getNames()  getValues() | Get labels of all options | List<String> |
| getOptionsAsText() | Get all options labels in one string separated with “; ” | String |
| areSelected(); | Get names of checked options | List<String> |
| areDeselected(); | Get names of unchecked options | List<String> |
| waitSelected(TEnum …)  waitSelected(String…) | Wait while all options with names (use enum/text) selected. Returns false if this not happens | Boolean |
| waitDeselected(TEnum …)  waitDeselected(String …) | Wait while all options with names (use enum/text) deselected. Returns false if this not happens | Boolean |
| SetValue; HaveValue; | See methods by link |  |

#### ComboBox

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| Dropdown; TextField; SetValue; Text; HaveValue; Element | See methods by link |  |

#### Table<TClickableText>

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| cell(Column, Row) | Get Cell value by column/row index (Int) or name(String) | Cell |
| cells(String)  cellsMatch(String) | Get all Cells with values equals/matches to searched value/regex | List<Cell> |
| cell(String)  cellMatch(String) | Get first Cell with equals/matches to searched value/regex | Cell |
| rows(String…)  columns(String…) | Searches Rows/Columns in table matches specified criteria  colNameValues/rowNameValues - list of search criteria in format <columnName>=<columnValue>  e.g. rows("Name=Roman", "Profession=QA")  Each Row/Column is map: column/row-Name:cell | Map<String, Map<String, Cell>> |
| waitValue(String, Row)  waitValue(String, Column) | Waits while value appear in Row/Column  e.g. waitValue("Roman", column("Name"))  or waitValue("Roman", column(3)) | Boolean |
| isEmpty() | Indicates are any rows in table. Check immediately | Boolean |
| waitHaveRows() | Wait while at least one row appear in table | Boolean |
| waitRows(Int) | Wait while at least count of rows appear in table | Boolean |
| cell(String, Row)  cell(String, Column) | Get first Cell with searched value in column/row by index (Int) or name(String)  e.g. cell("Roman", column("Name"))  or cell("Roman", column(3)) | Cell |
| cellsMatch (String, Row)  cellsMatch (String, Column) | Get all Cells with values matches to searched in Row/Column by index (Int) or name(String) | List<Cell> |
| column(String, Column)  row(String, Row) | Get Column/Row cells for Cell with searched value in Row/Column by index(Int) or name(String)  Each Row/Column is map: column/row-Name:cell | Map<String, Cell> |
| column(Int)  column(String) | Get Column with index(Int) or name(String)  Each Column is map: rowName:cell | Map<String, Cell> |
| columns().count() | Get Columns count | Int |
| columns().headers() | Get Columns headers | String[] |
| row(Int)  row(String) | Get Row with index(Int) or name(String)  Each Row is map: columnName:cell | Map<String, Cell> |
| rows().count() | Get Rows count | Int |
| rows().headers() | Get Rows headers | String[] |
| header() | Get Column’s header of table as list of elements | Map<String, SelectElement> |
| header(String) | Get Column header with name | SelectElement |
| headers() | Get Column’s header of table as text | String[] |
| footer() | Get Footer | String[] |
| cell(“Name”, 1).select()  cell(“Name”, 1).click() | Selects cell in table |  |
| cell(“Name”, 1).getText() | GetText from cell in table | String |
| cell(“Name”, 1). waitText(String) | Wait while cell’s text contains expected text. Returns element’s text | String |
| cell(“Name”, 1). waitMatchText(String) | Wait while cell’s text matches regEx. Returns element’s text | String |
| cell(“Name”, 1).get() | Get Cell element to perform any proper action for type TClickableText | TClickableText |
| Text; HaveValue; Element | See methods by link |  |

## Composite

#### Section

Page object element. Consist elements of any type.

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| Element | See methods by link |  |

#### Page

Page object element. Consist elements of any type. Can operate with browser page

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| open() | Opens url specified for page |  |
| static openUrl(String) | Opens expected url |  |
| refresh() | Refreshes page |  |
| back() | Press browser “Go back” |  |
| forward() | Press browser “Forward” |  |
| addCookie(Cookie) | Adding cookie in browser |  |
| clearCache() | Clears cache |  |
| checkOpened() | Apply suitable check for page (url/title) |  |
| url().check() | Checks that current page’s url is correct |  |
| title().check() | Checks that current page’s title is correct |  |
| url().match() | Checks that current page’s url matches to urlMatcher |  |
| title().match () | Checks that current page’s title matches to urlMatcher |  |
| url().contains() | Checks that current page’s url contains urlMatcher |  |
| title().contains () | Checks that current page’s title contains urlMatcher |  |

#### Search

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| chooseSuggestion(String, String) | Input text in search and then select value from suggestions |  |
| chooseSuggestion(String, int) | Input text in search and then select suggestions by index |  |
| find(String) | Input text in search field and press search button |  |
| getSuggesions(String) | Select all suggestions for text | List<String> |
| TextField; SetValue; Text; HaveValue; Element | See methods by link |  |

#### Pagination

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| next() | Choose Next page |  |
| previous() | Choose Previous page |  |
| first() | Choose First page |  |
| last() | Choose Last page |  |
| selectPage(int) | Choose page by index |  |

#### Form<T>

Page object element. Consist of elements based on SetValue interface and one button with function “submit”

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| fill(T) | Fills all elements on the form which implements SetValue interface and can be matched with fields in input entity |  |
| submit(T) | Fill all SetValue elements and click on Button “submit” <br>  \* To use this option Form pageObject should have only one IButton element |  |
| submit(T, String) | Fill all SetValue elements and click on Button specified button e.g. "Publish" or "Save" <br>  \* To use this option Form pageObject should have button names in specific format <br>  e.g. if you call "submit(user, "Publish") then you should have element 'publishButton'. <br>  \* Letters case in button name no matters |  |
| submit(T, Enum) | Similar to submit(T, String) use enum.toString() to get ButtonName |  |
| verify(T) | Check is form elements similar to entity fields |  |
| Element | See methods by link |  |

#### Popup

Page object element. Consist of Text element and buttons “Ok“, “Cancel”, “Close”

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| getText() | Get text from first element of type Text |  |
| ok() | Click on Button marked with annotation @OkButton or named "okButton" |  |
| cancel() | Click on Button marked with annotation @CancelButton or named "cancelButton" |  |
| close() | Click on Button marked with annotation @CloseButton or named "closeButton" |  |
| Text; HaveValue; Element | See methods by link |  |

#### PopupForm<T>

Page object element. Consist of elements based on SetValue interface and buttons “Ok“, “Cancel”, “Close”

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| Form<T>; Popup; Text; HaveValue; Element | See methods by link  \* Use “ok” button instead of “submit” |  |

## Base

#### Element

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| getWebElement() | Returns Selenium webElement for this Element | WebElement |
| waitAttribute(String, String) | Waits while attribute gets expected value. Return false if this not happens | Boolean |
| setAttribute(String, String) | Sets attribute value for element |  |
| waitDisplayed() | Waits while element becomes visible | Boolean |
| waitVanished() | Waits while element becomes invisible | Boolean |
| wait(JFuncTT<WebElement, Boolean>) | Waits while condition with happens WebElement and returns wait result | Boolean |
| wait(JFuncTT<WebElement, TResult>, JFuncTT<TResult, Boolean>) | Waits while condition with WebElement happens and returns result using resultFunc | TResult |
| wait(JFuncTT<WebElement, Boolean>, int) | Waits while condition with WebElement happens during specified timeout and returns wait result | Boolean |
| wait(JFuncTT<WebElement, TResult>, JFuncTT<TResult, Boolean>, int) | Waits while condition with WebElement happens during specified timeout and returns result using resultFunc | TResult |
| getDriver() | Get WebDriver associated with element | WebDriver |
| getLocator() | Get Element’s locator | By |
| getName() | Get Element’s name | String |

#### HasValue

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| getValue() | Get value of element | String |

#### SetValue

|  |  |  |
| --- | --- | --- |
| Method | Description | Result |
| setValue(String) | Set value to element |  |