Selenium Training

Framework (Python)

Guide

Document Version: *1.0*
Last Modified: *2019-05-20*

Confidentiality Notice

The information in this document is confidential and it must not be disclosed to any person other than an employee or advisor of Andea Solutions sp. z o.o. and its affiliates. It may not be reproduced in whole, or in part, nor may any of the information contained therein be disclosed without the prior consent of Andea Solutions sp. z o.o.

The contents of this document represent the views and opinions of Andea Solutions sp. z o.o., and has not been independently verified to be comprehensive and complete. No statement made of information given herein amounts to a commitment, representation or warranty by Andea Solutions sp. z o.o.

Any form of reproduction, dissemination, copying, disclosure, modification, distribution and or publication of this material is strictly prohibited.

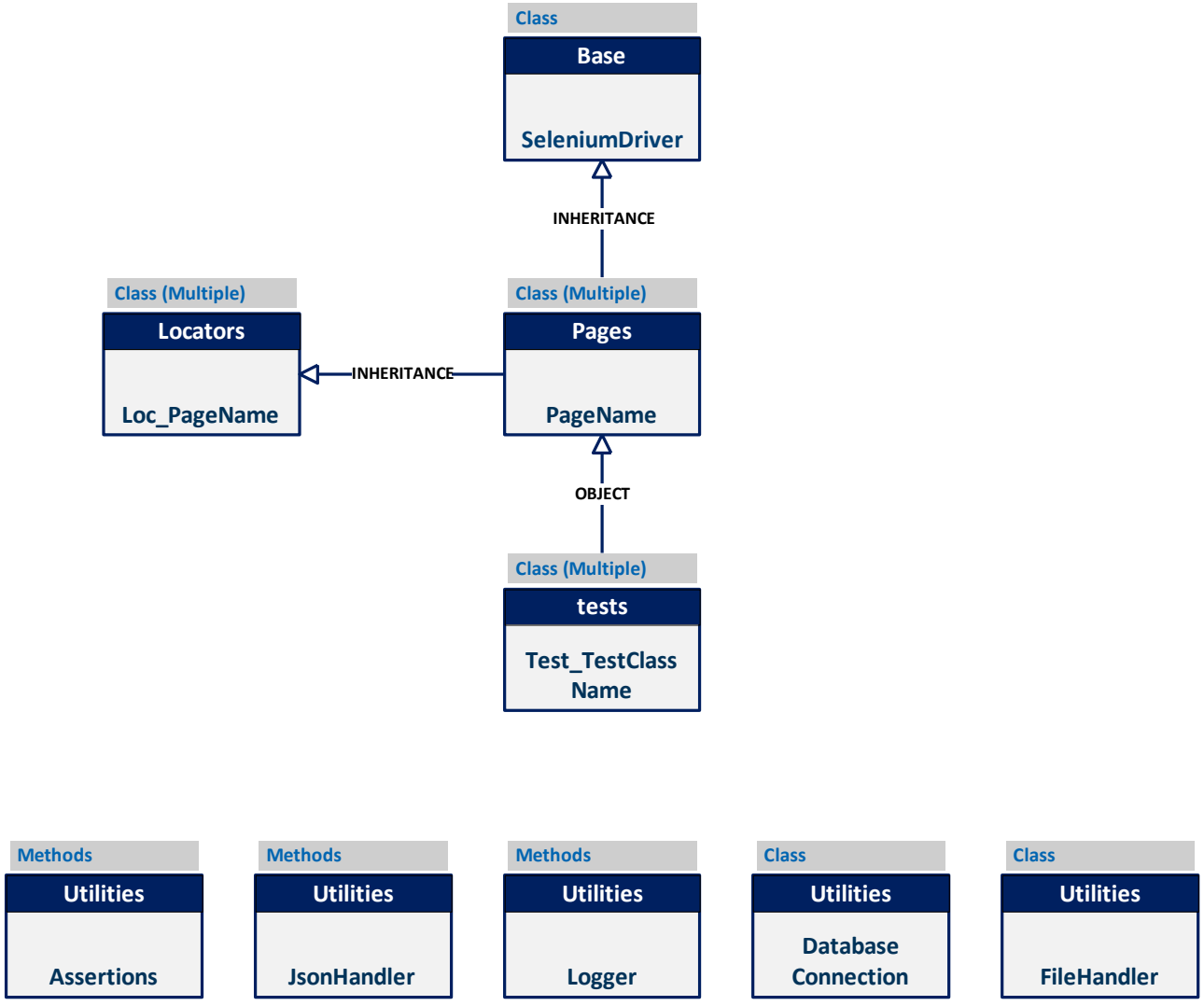
Contents

1 Framework Architecture

1.1 Fundamental information

- Programming language: Python
- Test automation framework: Selenium WebDriver
- Objects of tests: Web Applications
- IDE used in this guide: PyCharm
- Version Control System: GitLab

1.2 Structure



1.2.1 SeleniumDriver

Description

This class contains all basic methods, that can be used in pages methods to do some action on web pages.

Details

- Type: Class inside the file with the same name
- Catalog: Base
- Amount: One per framework
- Inherits from: N/A
- Example methods:
  - elementClick()
    Custom method that is used to click web page element
  - sendKeys()
    Custom method that is used to send text value to web page element

1.2.2 Loc_PageName

Description

These classes contain all locators that can be used in Pages classes to get element from specific page.

Details

- Type: Class inside the file with the same name
- Catalog: Locators
- Amount: One per each web page
- Inherits from: N/A
- Example locator:
  - BTN_LOGIN = By.XPATH, "//input[@type='login]"
    Locator for login button that uses XPATH to find it on web page
- Example dynamic locator:
  - def BTN_RESET_DYNAMIC(self, equipment_name):
    return By.XPATH, "//button[contains(@value, '" + equipment_name + "') and starts-with(@value, 'RESET')]"
    Method that returns locator for reset button depending on equipment name included in different element

1.2.3 PageName

Description

These classes contain all methods that can be used in test methods to do some action on specific page.

Details

- Type: Class inside the file with the same name
- Catalog: Pages
- Amount: One per each web page
- Inherits from: SeleniumDriver, related Locators class
- Example methods:
  - login()
    Method that is used to log into web application (includes actions like e.g. send login, send password, click login button)
  - getEquipmentState()
    Returns text value from element that indicates state of equipment

1.2.4 test_TestName

Description

These are main framework classes. They contain fixtures (Set up and tear down methods) and test methods.

Details

- Type: Class inside the file with the same name
- Catalog: tests
- Amount: One per each test scenario
- Inherits from: unittest2.TestCase (standard class from unittest2 module)
- Each test method = one test case. Example test method:

```
def test_TemplateTestClass_TC1(self):
    # From home page go to 'Equipment Monitor' screen
    self.homepage.searchForScreen(self.equipment_monitor_page.TITLE)

    # Get current state of equipment
    machine_state = self.equipment_monitor_page.getEquipmentState(self.test_data.EQUIPMENT_NAME)

    # Assert if current state of equipment = 'StandBy'
    Assertions.lastAssert(actual=machine_state, expected='StandBy', message="Checking if state is equal to StandBy")
```

1.2.5 Assertions

Description

These methods allow to create custom assertions in test methods. Assertions are captured by custom logger.

Details

- Type: Methods inside Assertions file
- Catalog: Utilities
- Amount: One per framework
- Inherits from: N/A
- Methods:
  - setResults()
    Appends the result of assertion (PASS/FAIL) to the list. It is being used in other Assertions methods. No need to use it outside the Assertions file.
  - addAssertion()
    Compare expected result with actual result and send it to the list using setResults method. It should be used, if it is NOT the last assertion in a test case.
  - lastAssertion()
    Compare expected result with actual result and send it to the list using setResults method. It should be used, if it is the last assertion in a test case.

1.2.6 JsonHandler

Description

These methods allow to get data from json files. JsonHandler object can be created everywhere we need to read json data.

Details

- Type: Methods inside JsonHandler file
- Catalog: Utilities
- Amount: One per framework
- Inherits from: N/A
- Methods:
  - getDataFromJsonFile()
    Returns object with all data from given json file.
    Example call:
    data = getDataFromJsonFile(data.json)
    value = data.KeyName
  - getDatabaseStatement()
    Returns database statement for given json key (it looks for this key in DatabaseStatements.json file)

1.2.7 Logger

Description

This method returns logger object, that should be used everywhere we want to track some action. Each log is being put into SeleniumLogs.log file.

Details

- Type: Method inside Logger file
- Catalog: Utilities
- Amount: One per framework
- Inherits from: N/A
- Method:
  - getLogger()
    Returns the logger object. It should be created in classes with common methods like SeleniumDriver.
    Example of logger object usage:
    loggerObject.info('message')

1.2.8 DatabaseConnection

Description

This class creates database connection and contains methods that allow to handle database operation. It should be used everywhere we need to get/change some data in/from database.

Database statements should be placed in the DatabaseStatements.json file (catalog: Configuration).

Details

- Type: Class inside the file with the same name
- Catalog: Utilities
- Amount: One per framework
- Inherits from: N/A
- Methods:
  - select()
    Returns all rows of given database query.

1.2.9 FileHandler

Description

This class contains methods that allow to handle operations on files like moving, deleting, checking content etc. It should be used everywhere we need to perform some operation on files like XML, PAS, TXT.

Details

- Type: Class inside the file with the same name
- Catalog: Utilities
- Amount: One per framework
- Inherits from: N/A
- Methods:
  - copyFile()
    Copies given file to different place
  - deleteFile()
    Deletes given file
  - checkIsFileGenerated()
    Checks if file has been generated in provided path. Additional parameters that generated file should contain inside can be passed.

2 Naming Convention

2.1 Locators

UI/Control Type	Prefix	Example
Button	BTN	BTN_EXIT
Check box	CHK	CHK_OPTION
Label	LBL	LBL_MESSAGE
Dropdown	DDN	DDN_ANSWERS
Tab	TAB	TAB_WORK_CENTER
IFrame	FRA	FRA_CONTENT
Table	TBL	TBL_REPORT
Text box	TXT	TXT_PASSWORD
Image	IMG	IMG_LANDSCAPE
Radio button	RDO	RDO_ADVANCED
Link / Anchor tag	LNK	LNK_GO_TO_ANDEA_PAGE

2.2 Miscellaneous

Type	Pattern	Example
File in “tests” catalog	**test_**TestClassName**.py**	test_MachineState.py
File in “Locators” catalog	**Loc_Page**PageName**.py**	Loc_PageHome.py
File in “Page” catalog	**Page**PageName**.py**	PageHome.py
Configuration file (json) for tests	**test_**TestClassName**.json**	Test_MachineState.json
Other file	FileName.extension	BaseConfiguration.json
Method name in test class	**test_**TestClassName_**TC**number()	test_MachineState_TC1()
Other method	methodName()	getEquipmentState()

2.3 Other

- Each class should have the same name as its file name.
  - **PageHome.py** file -> **PageHome** class
- Dynamic locators should have suffix = “_DYNAMIC”
  - Def LBL_EQUIPMENT_STATE_DYNAMIC()

3 Creating automated tests

- Create new tests class (you can copy “test_TemplateTestClass.py” from the framework)
- Create Locators class for each page you are going to go through during the tests (you can look at the “Loc_PageTemplateLocators.py”) with locators for page elements
- Create Page class for each created Locators class with methods that will be used to perform action on these pages (you can look at the “PageTemplatePages.py”)
- Create Configuration file with data for your tests
- Fill your tests class with action using methods from created pages classes
- Run your tests

www.andea.com
Andea Solutions Sp. z o.o.
ul. Kapelanka 26, 30-347 Kraków, tel. +48 12 266 33 55
NIP 676-245-53-86, KRS 0000419451, kapitał zakładowy 212 750 zł