[Skip to end of metadata](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt" \l "page-metadata-end)

* Created by [Kumar, Neeraj](https://devcloud.swcoe.ge.com/devspace/display/~212588212), last modified by [Kumar, Vinod](https://devcloud.swcoe.ge.com/devspace/display/~212608875) on [Dec 10, 2019](https://devcloud.swcoe.ge.com/devspace/pages/diffpagesbyversion.action?pageId=1499887298&selectedPageVersions=243&selectedPageVersions=244)

[Go to start of metadata](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#page-metadata-start)

Overview

**ProUI-Nxt is a Keyword Driven Library for UI Automation. Keyword Driven is a type of Functional Automation Testing Library which is also known as Action Word based testing. Library is divided into four major sections as Action, Assertion, Element and Wait. Tests can be developed with minimal programming knowledge.**

**ProUI-Nxt library supports typescript, flexible folder structure i.e. one can use folder structure of their choice, in-line locator storage, wide variety of assertions, exception logging & handling mechanism, backward compatibility with ProUI and Extensions and many more. ProUI is open for everyone to contribute.**

* [Overview](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Overview)
* [System Requirements](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-SystemRequirements)
* [Prerequisites](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Prerequisites)
* [Current Version](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-CurrentVersion)
* [Installation & Execution](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Installation&Execution)
* [TypeScript Config File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-TypeScriptConfigFile)
* [Package.json File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Package.jsonFile)
  + [For backward compatibility(proui-utils/proui-extension)](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Forbackwardcompatibility(proui-utils/proui-extension))
  + [For new version(ui-nxt)](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Fornewversion(ui-nxt))
* [Project File Details](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-ProjectFileDetails)
  + [Conf File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-ConfFile)
  + [Feature File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-FeatureFile)
  + [Locator File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-LocatorFile)
  + [Page Objects File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-PageObjectsFile)
  + [Step definition File](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-StepdefinitionFile)
* [IDE configuration](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-IDEconfiguration)
  + [Visual Studio Code](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-VisualStudioCode)
  + [Webstorm](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Webstorm)
* [Reports](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Reports)
  + [Configuration of Reports](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-ConfigurationofReports)
* [Types of Locators](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-TypesofLocators)
  + [Core Utility](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-CoreUtility)
  + [Different Ways of passing Element Details to core functionalities](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-DifferentWaysofpassingElementDetailstocorefunctionalities)
  + [Actions](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Actions)
  + [Assertions](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Assertions)
  + [Elements](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Elements)
  + [Wait](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Wait)
  + [Logger](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Logger)
  + [Exception Handling](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-ExceptionHandling)
    - [Resolve Exception](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-ResolveException)
* [Propel Integration](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-PropelIntegration)
* [Propel Job Creation](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-PropelJobCreation)
* [Rally Posting](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-RallyPosting)
  + [.env Reference](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-.envReference)
  + [For backward compatibility(proui-utils/proui-extension)](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Forbackwardcompatibility(proui-utils/proui-extension).1)
  + [For new version(ui-nxt)](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-Fornewversion(ui-nxt).1)
* [Release Notes](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-ReleaseNotes)

System Requirements

| **Software** | **Version** |
| --- | --- |
| Java | 1.8.0 or greater |
| Node | 8.0 or greater |
| Visual Studio Code/Webstorm | 2019 Community Edition |

Prerequisites

* System should be ready with all the above mentioned software requirements
* Should have Github account
* Make sure to have SSH key generated as part of your GitHub account (<https://help.github.com/articles/generating-an-ssh-key/>)

Current Version

* 1.0.4

Installation & Execution

* Open command prompt to clone below git repository to getting started with ProUI-NXT

|  |
| --- |
| git clone [git@github.build.ge.com](mailto:git@github.build.ge.com):ToolsEngineering/ProUINxt.git |

* One can also clone project as shown below in case of having issue with using SSH key

|  |
| --- |
| git clone <https://github.build.ge.com/ToolsEngineering/ProUINxt.git> |

* Install all the required npm packages mentioned in package.json by using the following command:

|  |
| --- |
| npm install |

* If you get proxy related issue run all the below commands and re-run **npm install**command again.

|  |
| --- |
| npm config set proxy <http://sjc1intproxy02.crd.ge.com:8080/>  npm config set http-proxy <http://sjc1intproxy02.crd.ge.com:8080/>  npm config set https-proxy <http://sjc1intproxy02.crd.ge.com:8080/>  npm config set registry <http://registry.npmjs.org/> |

* Transpile project's \*.ts files by adding [TypeScriptConfigFile](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt" \l "ProUI-Nxt-TypeScriptConfigFile) and execute below command.

|  |
| --- |
| //To transpile all the files mentioned in include section of tsconfig.json  tsc  //To transpile specific ts file (Eg: stepDef.ts)  tsc stepDef |

               Below command will install web-drivers and run the basic tests

|  |
| --- |
| grunt test |

**Note:** To restore the npm registry, one need to execute the below command

|  |
| --- |
| npm config set registry <http://registry.npmjs.org/> |

TypeScript Config File

Add tsconfig.json file[ if not available in the cloned repository ] in root folder and specifies the root files and the compiler options required to compile the project. Below are some of the compiler options and the file include or exclude options provided for reference.

**tsconfig.json file**

|  |
| --- |
| {    "compilerOptions": {      "target": "es6",      "module": "commonjs",      "moduleResolution": "node",      "inlineSourceMap": true,      "declaration": false,      "noImplicitAny": false,      "rootDir": "Projects"    },    "exclude": [      "node\_modules"    ],    "include": [        ".\\Projects"    ]  } |

Package.json File

NPM uses the package.json file to specify the version of a package that your app depends on. The version number is broken into three sections separated by a dot. i.e., "major.minor.patch" .

For backward compatibility(proui-utils/proui-extension)

To get latest functionality with older cucumber version, package.json should have the below dependencies. Users can execute the existing test cases as it is.

**package.json**

|  |
| --- |
| "dependencies": {      "@ge-tools/ui-nxt": "1.0.4",      "@ge-tools/ui-bl-apm\_m": "1.0.5",      "babel-runtime": "6.26.0",      "bluebird": "3.5.4",      "chai": "4.2.0",      "chai-as-promised": "7.1.1",      "cucumber": "0.10.2",      "cucumber-html-report": "0.2.5",      "cucumber-junit": "1.5.0",      "dotenv": "8.0.0",      "grunt": "1.0.4",      "grunt-contrib-jshint": "2.1.0",      "grunt-execute": "0.2.2",      "grunt-protractor-runner": "5.0.0",      "grunt-protractor-webdriver": "0.2.5",      "grunt-shell-spawn": "0.4.0",      "highcharts": "7.1.2",      "mkdirp": "0.5.1",      "protractor": "5.4.2",      "through2": "3.0.1",      "protractor-cucumber-framework": "0.5.0",      "protractor-html-screenshot-reporter": "0.0.21",      "protractor-image-comparison": "3.1.1",      "proui-extension": "1.0.12",      "proui-utils": "1.1.100",      "rally": "2.1.3",      "request": "2.88.0",      "request-promise": "4.2.4",      "superagent": "5.0.5",      "superagent-proxy": "2.0.0",      "winston": "3.2.1"    },    "author": "GE Digital",    "license": "ISC",    "repository": {      "type": "git",      "url": "git@github.build.ge.com:ToolsEngineering/ProUINxt.git"    },    "devDependencies": {      "@types/chai": "4.1.7",      "@types/chai-as-promised": "7.1.0",      "@types/cucumber": "4.0.6",      "@types/node": "12.0.10"    } |

## For new version(ui-nxt)

To get latest functionality with latest cucumber version, package.json should have the below dependencies.

**package.json**

|  |
| --- |
| "dependencies": {      "@ge-tools/ui-nxt": "1.0.4",      "@ge-tools/ui-bl-apm\_m": "1.0.5",      "babel-runtime": "6.26.0",      "bluebird": "3.5.4",      "chai": "4.2.0",      "chai-as-promised": "7.1.1",      "cucumber": "5.1.0",      "cucumber-html-report": "0.6.5",      "cucumber-junit": "1.7.1",      "dotenv": "8.0.0",      "grunt": "1.0.4",      "grunt-contrib-jshint": "2.1.0",      "grunt-execute": "0.2.2",      "grunt-protractor-runner": "5.0.0",      "grunt-protractor-webdriver": "0.2.5",      "grunt-shell-spawn": "0.4.0",      "highcharts": "7.1.2",      "mkdirp": "0.5.1",      "protractor": "5.4.2",      "through2": "3.0.1",      "protractor-cucumber-framework": "6.1.1",      "protractor-html-screenshot-reporter": "0.0.21",      "protractor-image-comparison": "3.1.1",      "protractor-multiple-cucumber-html-reporter-plugin": "1.8.1",      "rally": "2.1.3",      "request": "2.88.0",      "request-promise": "4.2.4",      "superagent": "5.0.5",      "superagent-proxy": "2.0.0",      "typescript": "3.5.2",      "winston": "3.2.1"    },    "author": "GE Digital",    "license": "ISC",    "repository": {      "type": "git",      "url": "git@github.build.ge.com:ToolsEngineering/ProUINxt.git"    },    "devDependencies": {      "@types/chai": "4.1.7",      "@types/chai-as-promised": "7.1.0",      "@types/cucumber": "4.0.6",      "@types/node": "12.0.10"    } |

# Project File Details

This library provides flexible folder structure. Below are some file names used in document for reference.

## Conf File

A **Conf File** is protractor configuration file of the project that we are using(starting point of execution). Every project can have its own configuration file.

Step definition file used to run test script should be javascript file. Configuration of step definition file in as shown below

**Configuration**

|  |
| --- |
| cucumberOpts: {    // define your step definitions in this file     require: [      "../step\_definitions/sampleTest.js",           //OR      "../step\_definitions/\*.js"     ],  } |

Feature File

A **Feature File** is an entry point to the Cucumber tests. This is a file where you will describe your tests in Descriptive language

Locator File

A **Locator File** is meant for storing all the locators required for the project as JSON.

Page Objects File

A **Page Object File**contains page object.

Step definition File

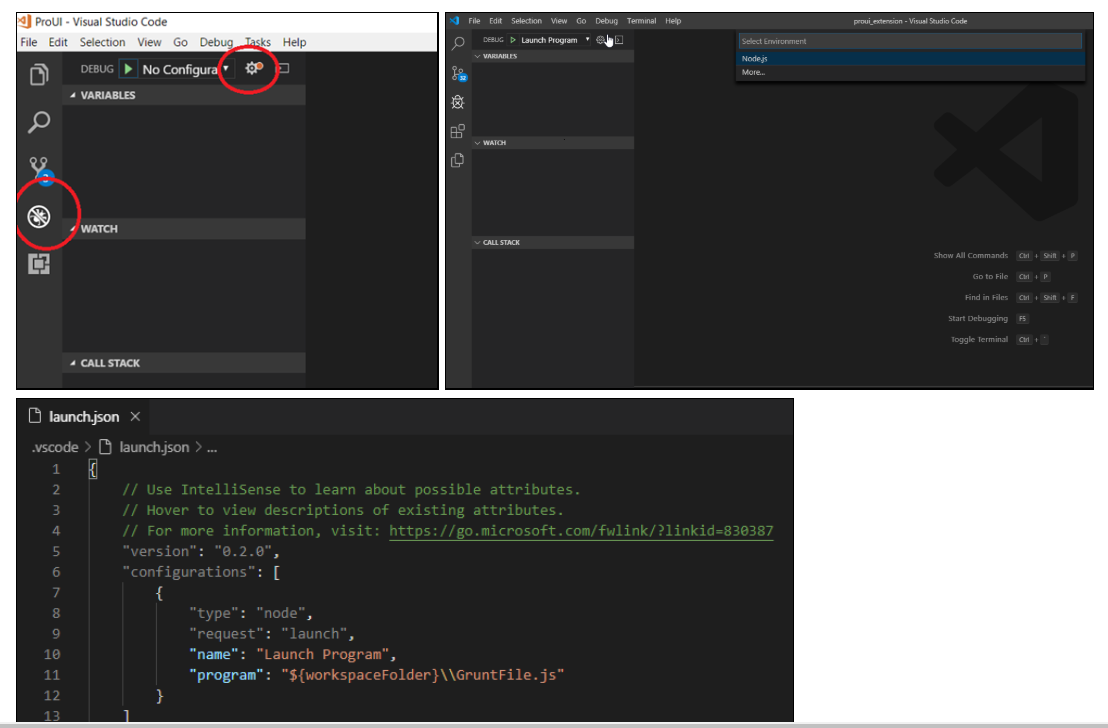
A **Step Definition File** maps the Test Case Steps in the feature files(introduced by Given/When/Then) to code. It which executes the steps on Application Under Test and checks the outcomes against expected result.

IDE configuration

Visual Studio Code

Below is the visual studio code configuration for running any ProUI-NXT projects.

* Open project using Visual Studio Code. Click on **Debug** and **Settings** icon respectively. select environment as "**Node.js**" You can see the**launch.json** file as below.



* Paste the following sample code in launch.json file to enable the debugging support for your respective projects. One can modify the **program** attribute and can add **args** attribute as per the project.

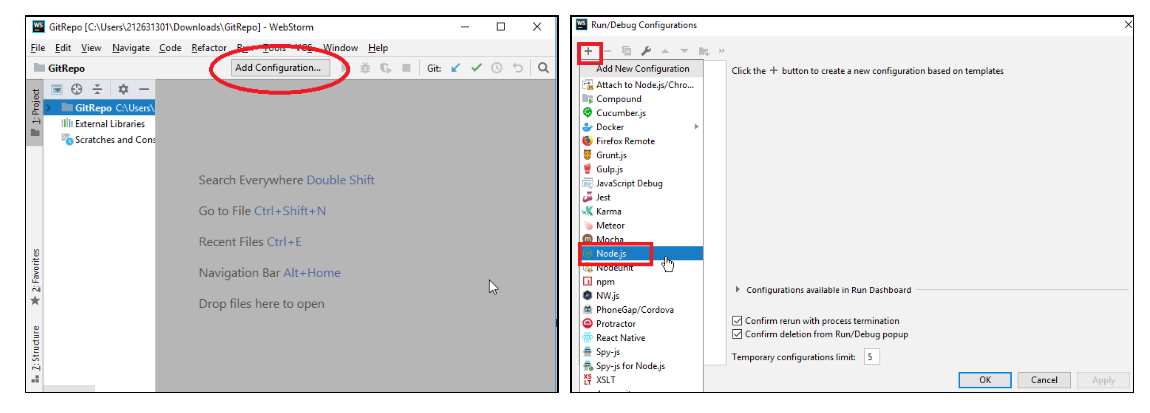
**Cucumber report configuration**

|  |
| --- |
| {      "version": "0.2.0",      "configurations": [          {              "type": "node",              "request": "launch",              "name": "ProUI-NXT",              "program": "C:\\ProUI-NXT\\node\_modules\\protractor\\built\\cli.js",              "stopOnEntry": false,              "args": ["${workspaceRoot}/Projects/Sample/Conf/Config.js","--suite","Test"]          }      ]  } |

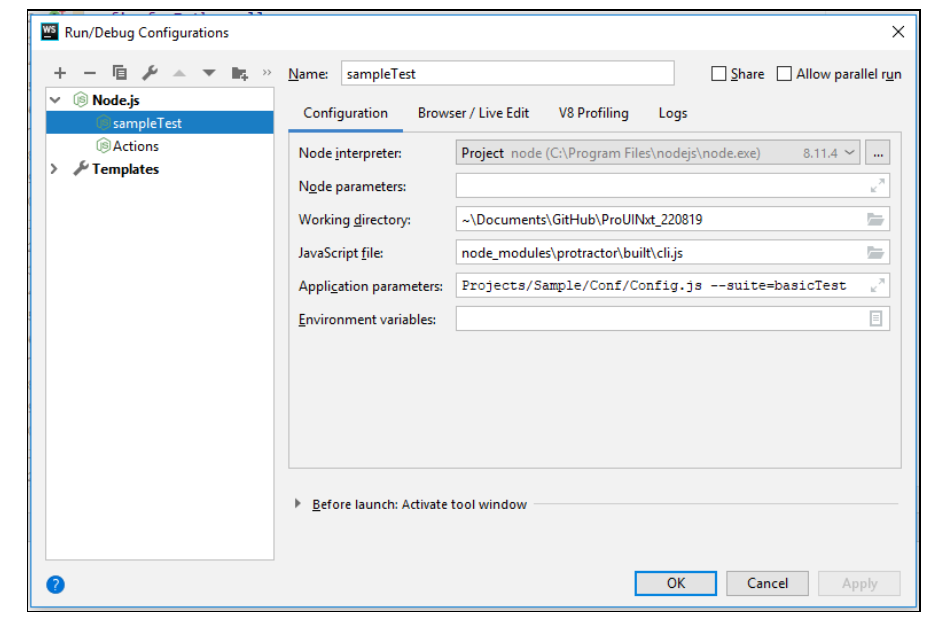
Webstorm

Below is the Webstorm configuration for running any ProUI-NXT projects.

* Open project using webstorm. Click on **Add Coniguration...**icon which will open configuration window. On the open window click on **+** button to select the **Node.js** configuration type as shown in below screenshots.

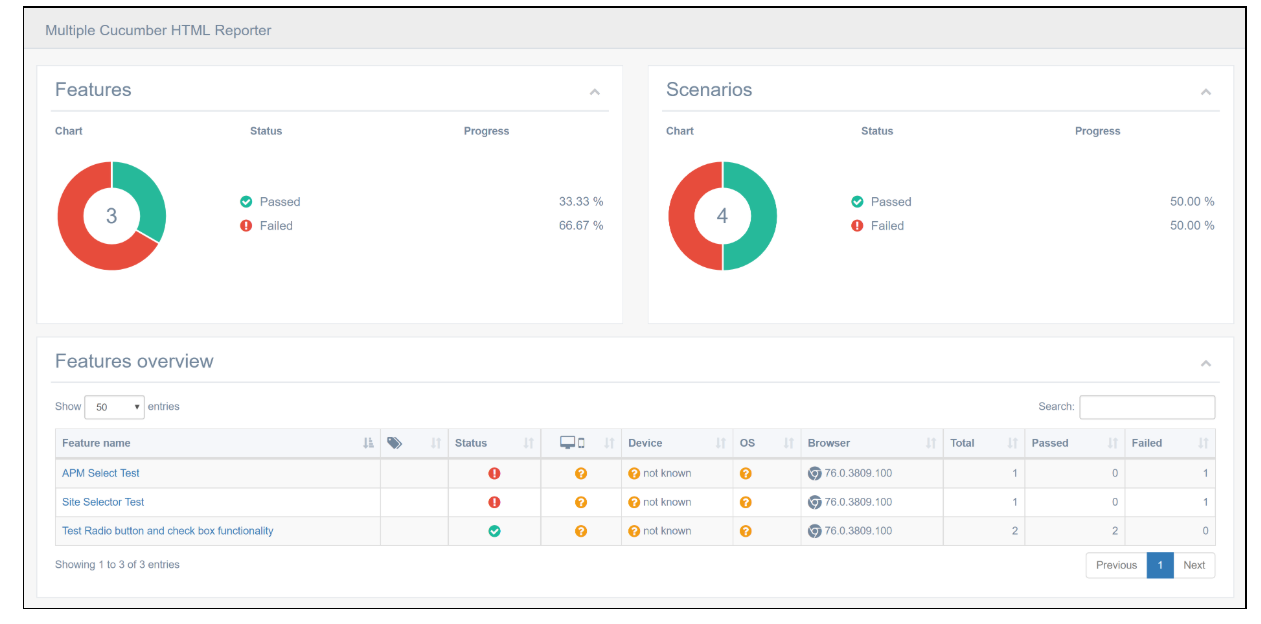


Add the configuration parameters such as JavaScript\_file, Application parameters as per the project requirements and Apply.



# Reports

After execution, reports are generated under **Reports** folder within root directory. Below is the screenshot of HTML cucumber report



Configuration of Reports

In protractor configuration file add the below mentioned configuration to generate the HTML report. By default, this configuration is available in the configuration file once the user clones the PROUI-NXT repository.

Sample test cases provided in ProUINxt repo also generating new cucumber report. For report configuration you can refer the configuration file for same. <https://github.build.ge.com/ToolsEngineering/ProUINxt/blob/master/Projects/Sample/Conf/Config.ts>

**Cucumber report configuration**

|  |
| --- |
| plugins: [{       package: 'protractor-multiple-cucumber-html-reporter-plugin',       options:{              // Read the options part for more options              automaticallyGenerateReport: true,              removeExistingJsonReportFile: true      }  }],  cucumberOpts: {            format: 'json:Reports/results.json', // Specify the output format            strict: true // Fail if there are any undefined or pending steps  } |

Types of Locators

Different locator types are supported to find elements. One can use required type of locator as per need. The list of supporting locator types are as shown below

| **Locator Types** | **Description** |
| --- | --- |
| id | Locates an element by its ID |
| css | Locate element using a CSS selector |
| cssAll | Locates all CSS element using CSS selector |
| cssContainingText | Finds elements by CSS which contain a certain string |
| deepCss | Finds an element by css selector within the Shadow DOM |
| name | Locates elements whose name attribute has the given value |
| className | Locates elements that have a specific class name |
| xpath | Locates elements matching a XPath selector |
| linkText | Locates link elements whose visible text matches the given string |
| partialLinkText | Locates link elements whose visible text contains the given substring |
| buttonText | Finds a button by text |
| partialButtonText | Finds a button by partial text |
| binding | Finds an element by text/partialText  binding |
| exactBinding | Finds an element by exact binding |
| model | Finds an element by ng-model expression |
| repeater | Finds elements inside an ng-repeat. For element at specific row and  column, locatorValue can be given as a json with value,row,column keys  **repeaterLocatorExample** Expand source |
| exactRepeater | Finds an element by exact repeater |
| options | Finds an element by ng-options expression |
| tagName | Locates elements with a given tag name |

Library Details

Core Utility

It consists of all the core utilities like Actions, Assertions, Waits, Elements and Exception Handlers required for supporting library. One can utilize Core Utility by importing it from ui-nxt npm package and creating object of the **CoreUtility.**

As shown in the code blocks below, CoreUtility object is created by using default as well as parameterized constructor. Both are having its own usage.

* Default constructor is used, when we don't have any locator files and want to use in-line locators. And we can also use **ChangeJsonFile()** functionality to set the JSON file later on as shown in [Nxt-SetorChangeJSONfile](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt" \l "ProUI-Nxt-Nxt-SetorChangeJSONfile)

**CoreUtility object using default constructor**

|  |
| --- |
| import {CoreUtility} from '@ge-tools/ui-nxt';    //without passing json details.  let CoreUtilityObj =  new CoreUtility(); |

* Parameterized constructor is used to set the locator JSON file by using parameters **jsonFileRelPath**(Relative path of the JSON file) and **pageName** (page name for key:value pair)

**CoreUtility object using parameterized constructor**

|  |
| --- |
| import {CoreUtility} from '@ge-tools/ui-nxt';    //passing json file details  let CoreUtilityObj =  new CoreUtility(jsonFileRelPath, pageName); |

**Set or Change JSON file**

ChangeJsonFile function can be used to set the JSON file, if default constructor of CoreUtility is called or ChangeJsonFile is used to switch between multiple JSON files.

Below is the syntax for ChangeJsonFile function where, parameter ***jsonFileRelPath***can be passed as either json file name directly or relative path of json file w.r.t Project repo.

**changeJsonFile functionality**

|  |
| --- |
| await CoreUtilityObj.changeJsonFile(jsonFileRelPath, pageName);    //Example    await CoreUtilityObj.changeJsonFile("common-element-repo", "navigation");    //OR    CoreUtilityObj.changeJsonFile("Projects\SampleTest\sample.json", "test"); |

Different Ways of passing Element Details to core functionalities

Parameter **elementDetails**can be passed as **key name** or **type locator details** or an **element** itself. Below are the examples that shows the usage of same.

**Example1 : elementDetails as locatorDetails object**

|  |
| --- |
| import { CoreUtility, locatorDetails } from '@ge-tools/ui-nxt';    let element: locatorDetails = {      locatorType: "xpath",      locatorValue: "//a[contains(text(),'Resources')]"  };    CoreUtilityObj.Action.scrollIntoView(element, 30000); |

**Example2 : elementDetails as Key Name**

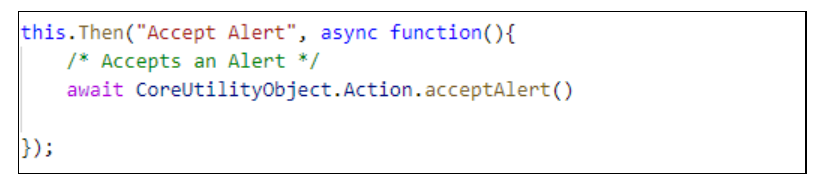
|  |
| --- |
| let KeyName= "'scrollResource'"; //key name from json file    CoreUtilityObj.Action.scrollIntoView(KeyName, 30000); |

**Example3: elementDetails as ElementFinder**

|  |
| --- |
| //Here elementDetails can be either of type locator Details or KeyName from json file  let element: ElementFinder=  await CoreUtilityObj.Element.getElement(elementDetails,"", 30000);    CoreUtilityObj.Action.scrollIntoView(element); |

Actions

It consists of all the necessary actions required to perform on applications. Before using the methods of **Action,** make sure to import the file **[C](https://github.build.ge.com/ToolsEngineering/ui-nxt/blob/master/Core/CoreUtility.ts" \o "CoreUtility.ts)oreUtility.ts** and create an object of it in the config file. One can use the methods of Action by using core utility object. Below is the basic example to demonstrate Action usage.



Parameter **elementDetails** used in methods listed below is either key name, locator details type or an element. Refer [DifferentWaysofpassingElementDetailstocorefunctionalities](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-DifferentWaysofpassingElementDetailstocorefunctionalities) for examples

Below are the list of Actions:

1. **scrollIntoView:**Scrolls until the element is in view

**scrollIntoView**

|  |
| --- |
| CoreUtilityObj.Action.scrollIntoView(elementDetails, optionalWaitTime); |

1. **acceptAlert:**Accepts Alert

|  |
| --- |
| CoreUtilityObj.Action.acceptAlert(); |

1. **acceptAlertandGetText:** Accepts Alert and returns text in the alert box

|  |
| --- |
| CoreUtilityObj.Action.acceptAlertandGetText(); |

1. **dismissAlert:** Dismisses Alert

|  |
| --- |
| CoreUtilityObj.Action.dismissAlert(); |

1. **dismissAlertandGetText:**Dismisses Alert and returns text in the alert box

|  |
| --- |
| CoreUtilityObj.Action.dismissAlertandGetText(); |

1. **click**: Clicks on the element

|  |
| --- |
| CoreUtilityObj.Action.click(elementDetails, optionalWaitTime); |

1. **jsClick**: Clicks on the element using javascript executor.

|  |
| --- |
| CoreUtilityObj.Action.jsClick(elementDetails, optionalWaitTime); |

1. **doubleClick**: Performs double click operation on the element

|  |
| --- |
| CoreUtilityObj.Action.doubleClick(elementDetails, optionalWaitTime); |

1. **clear**: clears the text in the input element

|  |
| --- |
| CoreUtilityObj.Action.clear(elementDetails, optionalWaitTime); |

1. **enterData**: Enters data to the input element

|  |
| --- |
| CoreUtilityObj.Action.enterData(elementDetails, optionalWaitTime); |

1. **appendText**: Appends the given text to an existing string

|  |
| --- |
| CoreUtilityObj.Action.appendText(elementDetails, textToAppend, optionalSpaceFlag); |

1. **prependText**: Prepends the given text to an existing string

|  |
| --- |
| CoreUtilityObj.Action.prependText(elementDetails, textToPrepend, optionalSpaceFlag); |

1. **mouseHover**: Performs mouse hover operation on the element

|  |
| --- |
| CoreUtilityObj.Action.mouseHover(elementDetails, optionalWaitTime); |

1. **browseForward**: Move forward in the browser history

|  |
| --- |
| CoreUtilityObj.Action.browseForward(); |

1. **browseBackWard**: Move backward in the browser history

|  |
| --- |
| CoreUtilityObj.Action.browseBackWard(); |

1. **browseRefresh**: Refresh the current page

|  |
| --- |
| CoreUtilityObj.Action.browseRefresh(); |

1. **switchToDefaultContent**: Switch back to default window, after dealing with some framed elements

|  |
| --- |
| CoreUtilityObj.Action.switchToDefaultContent(); |

1. **takeScreenshot**: Takes the screenshot of entire page and save the same with given screenShot FileName at provided Directory path

|  |
| --- |
| CoreUtilityObj.Action.takeScreenshot(screenShotFileName, optionalDirectoryPath); |

1. **takeElementScreenShot**: Takes screenshot of the passed element

|  |
| --- |
| CoreUtilityObj.Action.takeElementScreenShot(elementDetails, screenShotFileName, optionalWaitTime); |

1. **selectDropdownByIndex**: Selects drop-down item by index

|  |
| --- |
| CoreUtilityObj.Action.selectDropdownByIndex(elementDetails, index, optionalWaitTime); |

1. **selectDropdownByText**: Selects drop-down item by Text

|  |
| --- |
| CoreUtilityObj.Action.selectDropdownByText(elementDetails, itemText, optionalWaitTime); |

1. **getElementsCount**: gets the elements count matching the locator

|  |
| --- |
| CoreUtilityObj.Action.getElementsCount(elementDetails); |

1. **getElementsText**: gets text of all elements matching locator

|  |
| --- |
| CoreUtilityObj.Action.getElementsText(elementDetails); |

1. **getElementsAttributeValue**: gets attribute values of all elements matching locator

|  |
| --- |
| CoreUtilityObj.Action.getElementsAttributeValue(elementDetails, attribute); |

1. **switchToFrame**: Switches to the given frame element

|  |
| --- |
| CoreUtilityObj.Action.switchToFrame(elementDetails); |

1. **selectParentFrame**: Select's parent frame

|  |
| --- |
| CoreUtilityObj.Action.selectParentFrame(); |

1. **getWindowHandle**: Get current window handle

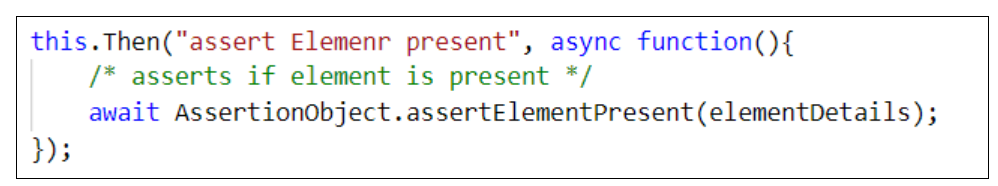
|  |
| --- |
| CoreUtilityObj.Action.getWindowHandle(); |

1. **switchToWindow**:  Switchs to the given window handle

|  |
| --- |
| CoreUtilityObj.Action.switchToWindow(windowId); |

Assertions

It consists of assertions to perform check on elements and verify the same. Before using the methods of **Assertion,** make sure to import the file **CoreUtility.ts** and create an object of it in the config file. One can use the methods of Assertion by using core utility object.  Below is the basic example to demonstrate Assertion usage.



Parameter **elementDetails** used in methods listed below is either key name, locator details type or an element. Refer [DifferentWaysofpassingElementDetailstocorefunctionalities](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-DifferentWaysofpassingElementDetailstocorefunctionalities) for examples

Below are the list of Assertions:

1. **assertElementPresent:**Asserts if the element is present

|  |
| --- |
| CoreUtilityObj.Assertion.assertElementPresent(elementDetails); |

1. **assertElementNotPresent:** Asserts if the element is not present

|  |
| --- |
| CoreUtilityObj.Assertion.assertElementNotPresent(elementDetails); |

1. **assertElementText**: Asserts the exact text on given element

|  |
| --- |
| CoreUtilityObj.Assertion.assertElementText(elementDetails, expectedText, optionalWaitTime); |

1. **assertElementAttributeValue**: Asserts the exact value on provided element's attribute. Attrubute defaults to 'value' attribute

|  |
| --- |
| CoreUtilityObj.Assertion.assertElementAttributeValue(elementDetails, expectedText, optionalWaitTime, attribute); |

1. **assertElementContainsText**: Asserts if element contains the sub text provided

|  |
| --- |
| CoreUtilityObj.Assertion.assertElementContainsText(elementDetails, subtextToMatch, optionalWaitTime); |

1. **assertElementContainsAttributeValue**: Asserts if element contains the sub text provided in the value of provided attribute. Attribute defaults to 'value' attribute

|  |
| --- |
| CoreUtilityObj.Assertion.assertElementContainsAttributeValue(elementDetails, subtextToMatch, optionalWaitTime, attribute); |

1. **verifyElementEnabled**: Verifies whether element is enabled

|  |
| --- |
| CoreUtilityObj.Assertion.verifyElementEnabled(elementDetails, optionalWaitTime); |

1. **verifyElementDisplayed**: Verifies whether selected element is displayed

|  |
| --- |
| CoreUtilityObj.Assertion.verifyElementDisplayed(elementDetails, optionalWaitTime); |

1. **verifyURLContainsText**: Verifies specific text contained in URL

|  |
| --- |
| CoreUtilityObj.Assertion.verifyURLContainsText(textToMatch); |

1. **verifyAttrContains**: Verifies attribute value contains expected text

|  |
| --- |
| CoreUtilityObj.Assertion.verifyAttrContains(elementDetails, attributeToVerify, expectedText, optionalWaitTime); |

1. **verifyAttrEquals**: Verifies attribute value has expected text

|  |
| --- |
| CoreUtilityObj.Assertion.verifyAttrEquals(elementDetails, attributeToVerify, expectedText, optionalWaitTime); |

1. **verifyDropDownLoaded**: Verify is drop-down loaded

|  |
| --- |
| CoreUtilityObj.Assertion.verifyDropDownLoaded(elementDetails, optionalWaitTime); |

1. **assertDropDownOptionsCount**: Asserts on available options in dropdown

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownOptionsCount(elementDetails, expectedItemCount, optionalWaitTime); |

1. **assertDropDownInAlphabaticOrder**: Asserts drop down is in alphabetical order

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownInAlphabaticOrder(elementDetails, SortingOrder); |

1. **assertDropDownSelectedOption:** Asserts if the expected text is selected in the dropdown

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownSelectedOption(elementDetails, expectedText, optionalWaitTime); |

1. **assertDropDownHasOption**: Asserts if the exact expected text is present in the dropdown

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownHasOption(elementDetails, expectedText, optionalWaitTime); |

1. **assertDropDownSameOptionsCount**: Asserts on count of items with exact text as provided

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownSameOptionsCount(elementDetails, exactTextToMatch, expectedItemCount, optionalWaitTime); |

1. **assertDropDownPartialOptionsCount**: Asserts count of options with sub text provided

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownPartialOptionsCount(elementDetails, subTextToMatch, expectedItemCount, optionalWaitTime); |

1. **assertDropDownContainOptionWithPartialText**: Asserts if expected partial text is present in any of the dropdown option text

|  |
| --- |
| CoreUtilityObj.Assertion.assertDropDownContainOptionWithPartialText(elementDetails, subTextToMatch, expectedItemCount, optionalWaitTime); |

1. **assertCheckBoxSelected:**Asserts whether the checkbox element is selected

|  |
| --- |
| CoreUtilityObj.Assertion.assertCheckBoxSelected(elementDetails, optionalWaitTime); |

1. **assertCheckBoxNotSelected:** Asserts whether the checkbox element is not selected

|  |
| --- |
| CoreUtilityObj.Assertion.assertCheckBoxNotSelected(elementDetails, optionalWaitTime); |

1. **assertRadioButtonNotSelected:** Asserts whether the radio button element is not selected

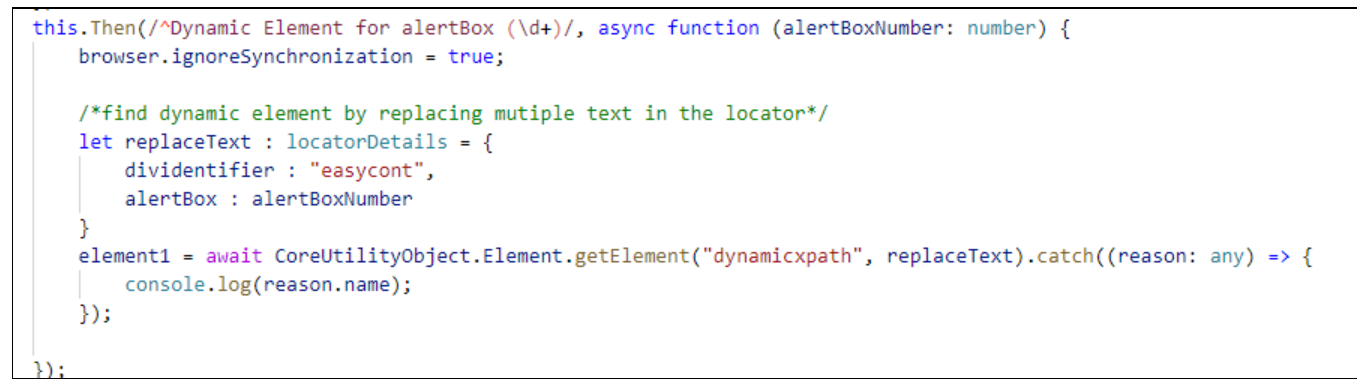
|  |
| --- |
| CoreUtilityObj.Assertion.assertRadioButtonNotSelected (elementDetails, optionalWaitTime); |

1. **assertRadioButtonSelected:**Asserts whether the radio button element is selected

|  |
| --- |
| CoreUtilityObj.Assertion.assertRadioButtonSelected(elementDetails, optionalWaitTime); |

Elements

It consists of all the functionalities related to elements like, finding element or getting element by its visibility or presence and so on. On successful operation it will return the required element. Before using the methods of **Element,** make sure to import the file **CoreUtility.ts** and create an object of it in the config file. One can use the methods of Element by using core utility object.



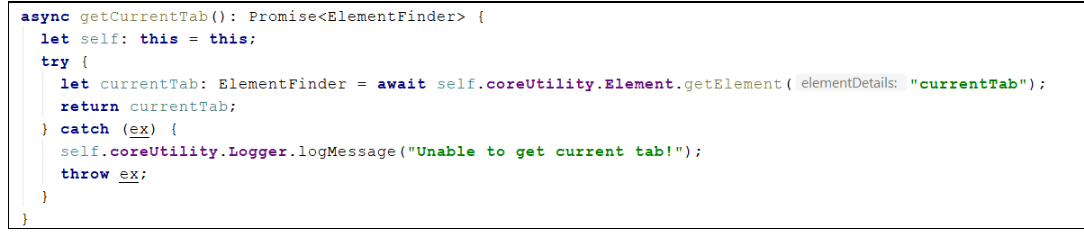
Below are the list of Elements functionalities:

* Parameter **elementDetails** used in methods listed below is either key name, locator details type or an element. Refer [DifferentWaysofpassingElementDetailstocorefunctionalities](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-DifferentWaysofpassingElementDetailstocorefunctionalities) for examples.
* Parameter **oTextToReplace**is an optional paramater which is used to replace text in the locatorValue. It can be a string or an object with key as replacable-text and value as text-to-be-replaced-with.
* Parameter **oTimeWait**optional wait-time to wait for an element to meet the expected condition provided.
* Parameter **oElementExpectedCondition**Expected Condition to check on the element. Accepts a value from enum "*ElementExpectedCondition*" . "ElementExpectedCondition" can be imported from '@ge-tools/ui-nxt'
* Parameter **oElementText** is consider for specific expected Conditions like "TextToBePresentInElement" and "TextToBePresentInElementValue"

1. **getElement:** An expectation for waiting and returning an element that meets the specified Expected Condition. Default expected condition is  "**visibilityOf**'. "TimeOutError" is thrown when element doesn't meet the expected condition provided. getElement method provides the first element when there are multiple matches for the locator passed.

CoreUtilityObj.Element.getElement(elementDetails, oTextToReplace,  oTimeWait , oElementExpectedCondition, oElementText);

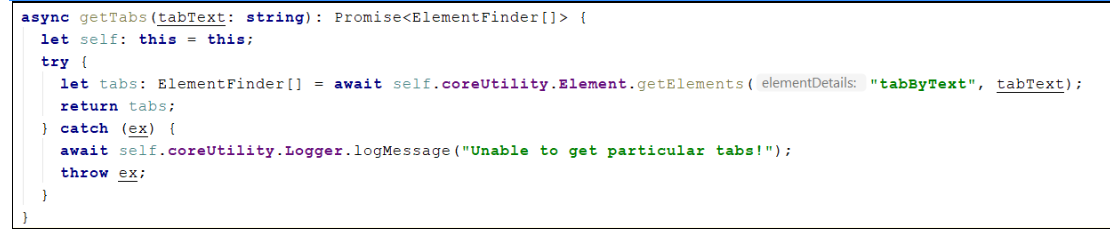
1. **Example**: Below example demonstrates the usage of getElement method, which will return first matched element



**getElements:** An expectation for waiting and returning multiple element which match the locator given and also meets the specified Expected Condition. Default expected condition is  "**visibilityOf**'. Empty array is returned if no element meets the expected condition.

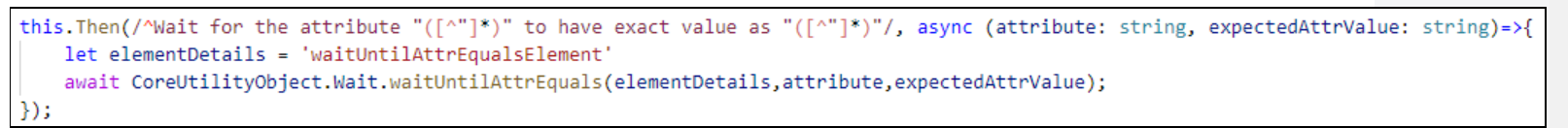
CoreUtilityObj.Element.getElements(elementDetails, oTextToReplace,  oTimeWait , oElementExpectedCondition, oElementText);

**Example**: Below example demonstrates the usage of getElements method, which will return multiple matched elements



## Wait

It consists of functionalities that provides wait mechanism. Functionalities to schedule a command to wait for a condition to hold or promise to be resolved. Before using the methods of **Wait**, make sure to import the file **CoreUtility.ts** and create an object of it in the config file. One can use the methods of Wait by using core utility object.



Parameter **elementDetails** used in methods listed below is either key name, locator details type or an element. Refer [DifferentWaysofpassingElementDetailstocorefunctionalities](https://devcloud.swcoe.ge.com/devspace/pages/viewpage.action?spaceKey=KKQTL&title=ProUI-Nxt#ProUI-Nxt-DifferentWaysofpassingElementDetailstocorefunctionalities) for examples

Below are the list of Wait functionalities:

1. **waitForThenClick:** Wait for specified/default time for element to be clickable and then click

|  |
| --- |
| CoreUtilityObj.Wait.waitForThenClick(elementDetails, optionalWaitTime); |

1. **waitUntilClickable:** Wait for specified/default time for element to be clickable

|  |
| --- |
| CoreUtilityObj.Wait.waitUntilClickable(elementDetails, optionalWaitTime); |

1. **waitUntilAttrContains:** waits for specified/default time for a attribute to contain expected value

CoreUtilityObj.Wait.waitUntilAttrContains(elementDetails, actualAttribute, expectedAttribute, optionalWaitTime);

1. **waitUntilAttrEquals:** waits for specified/default time for a attribute to get exact expected value

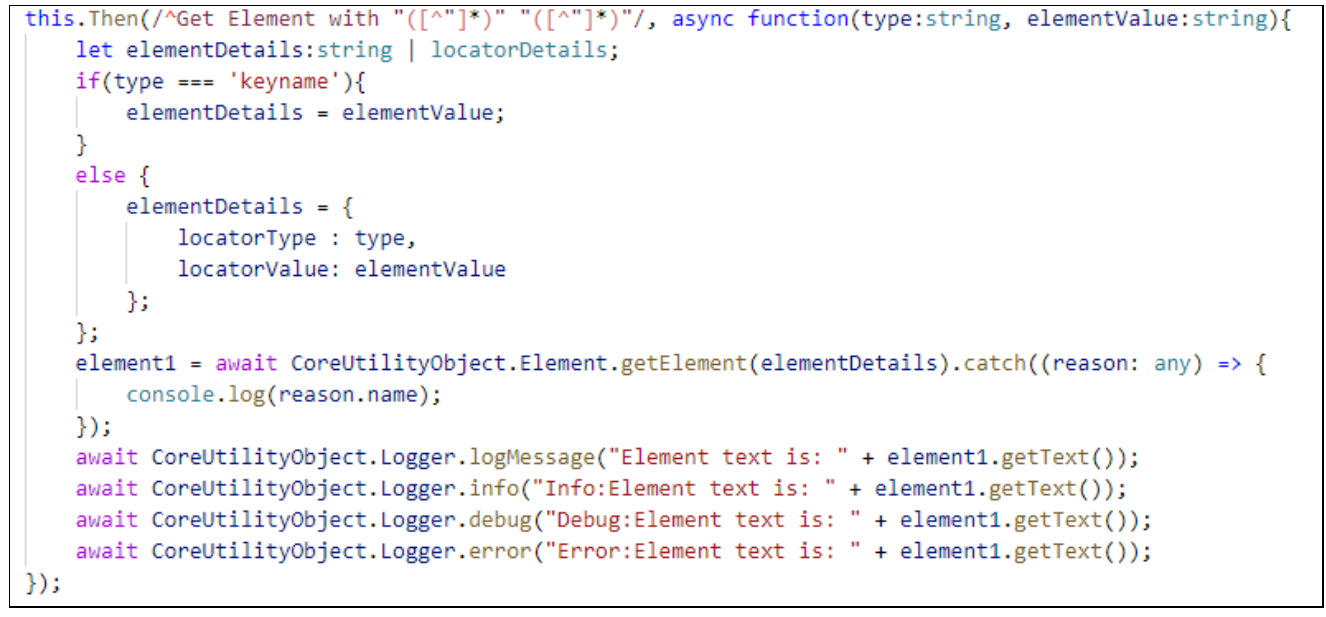
CoreUtilityObj.Wait.waitUntilAttrEquals(elementDetails, actualAttribute, expectedAttribute, optionalWaitTime);

1. **waitForElementToDisappear:** waits for specified/default time for the provided element to get disappered.

|  |
| --- |
| CoreUtilityObj.Wait.waitForElementToDisappear(elementDetails, optionalWaitTime); |

Logger

Logger consists of functionalities that provide logging mechanism. Functionalities allow the user to write the information to log as well as the console. Before using the methods of **Logger**, make sure to import the file **CoreUtility.ts** and create an object of it in the config file. One can use the methods of Logger by using core utility object.



Below are the functions of Logger

1. **logMessage:**write information to the log and console

|  |
| --- |
| CoreUtilityObj.Logger.logMessage("message to log to console as well as logger"); |

1. **info:** write information to log at INFO level

|  |
| --- |
| CoreUtilityObj.Logger.info("message to log at info level"); |

1. **debug:**write information to log at DEBUG level

|  |
| --- |
| CoreUtilityObj.Logger.debug("message to log at debug level"); |

1. **error:**write information to log at ERROR level

|  |
| --- |
| CoreUtilityObj.Logger.error("message to log at error level"); |

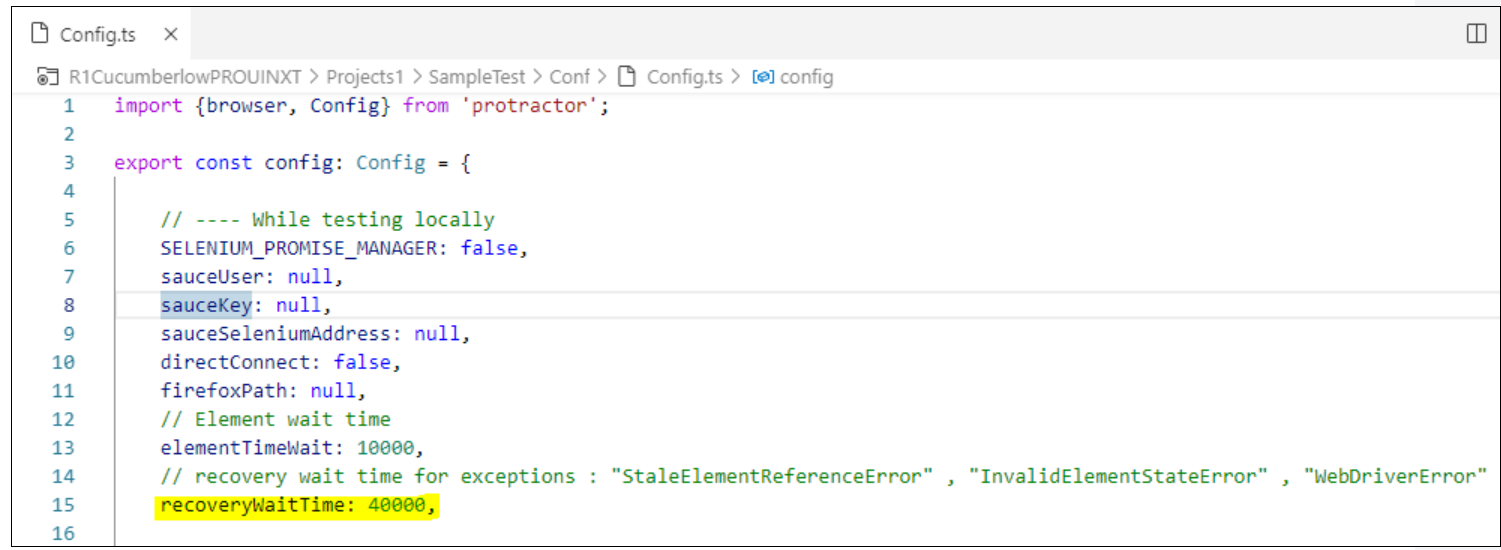
## Exception Handling

Exception handler will log custom messages for the exception occurred and also tries to resolve certain exceptions.

### Resolve Exception

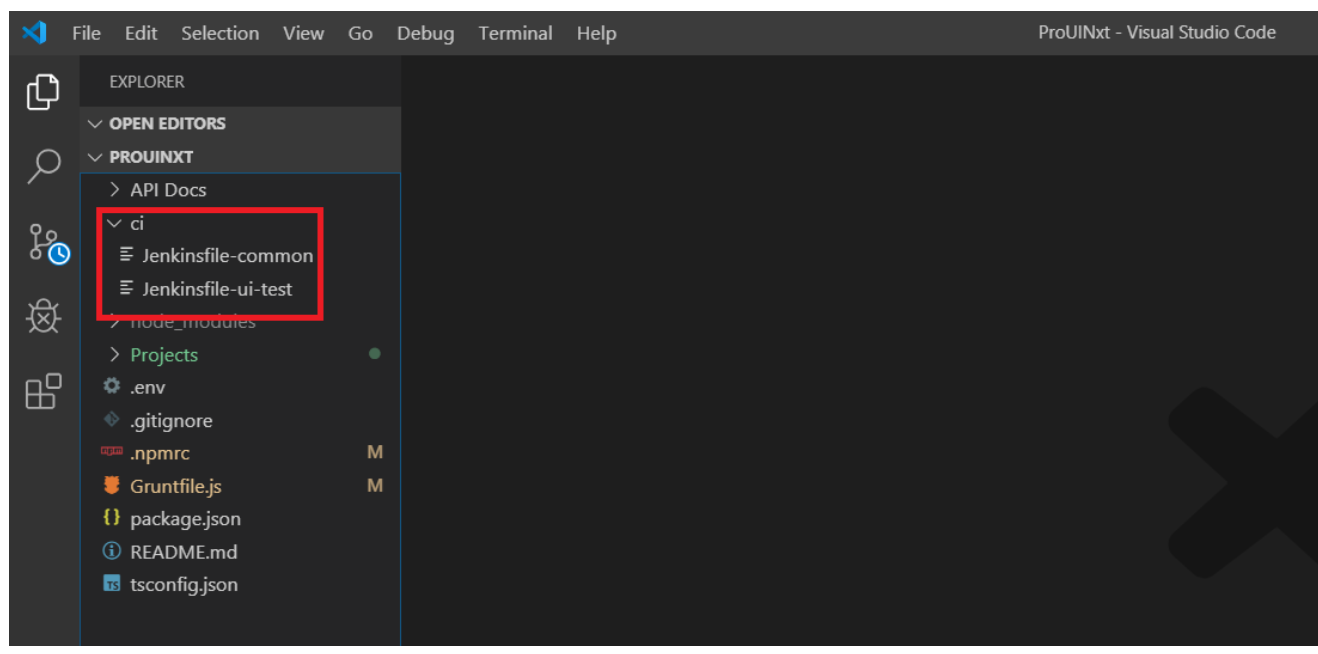
Exception handler also tries to resolve exceptions like "**StaleElementReferenceError**" , "**InvalidElementStateError**" , "**WebDriverError**" based on the "**recoveryWaitTime**" mentioned in protractor Configuration file. "**recoveryWaitTime"**attribute accepts wait time in milliseconds. By default **recoveryWaitTime** is considered as "40000"ms. Exception is thrown if the above mentioned exceptions are still not recovered in the specified time.

Below is an example for defining recoveryWaitTime.:



Propel Integration

* In root directory of ProUINxt repo you can find **"ci"** folder which contains the jenkins files used for propel integration.



* Below you can find the contents of both Jenkins files which contains jenkins pipeline script. Replace "YOUR\_EMAIL\_ID" text with your email id "Jenkinsfile-common" file.

**Jenkinsfile-common**

|  |
| --- |
| EMAIL\_LIST = "YOUR\_EMAIL\_ID"  ATTACHMENTS\_PATTERNS = "Reports/report/index2.html"    /\*   \*  stage: setup   \*/  def setup() {      cleanCheckout()      npmInstall()  }    def cleanCheckout() {      sh 'rm -rf ./\*'      checkout scm  }    def npmInstall() {      try {          npmConfig()          npmInstallGlobals()          npmInstallLocals()      } catch (ex) {          clearRegistry()          npmInstallGlobals()          npmInstallLocals()      }  }    def npmConfig() {      def CACHING\_REPO\_URL = '<http://propel.ci.build.ge.com/cache/repository/npm-virtual/'>      sh 'npm config set strict-ssl false'      sh "npm config set registry ${CACHING\_REPO\_URL}"  }    def npmInstallGlobals() {      sh 'npm install findup-sync nopt grunt-known-options -g'      sh 'npm install grunt -g'      sh 'npm install typescript@3.5.2 -g'  }    def npmInstallLocals() {      sh 'npm install'  }    def clearRegistry() {      sh 'npm config delete registry'  }    def compile() {      sh 'tsc'  }    /\*   \*  stage: test   \*/  def test(conf, suite, tags) {       sh "grunt test --conf=${conf} --suite=${suite} --tags=${tags}"  }    def testNoSuite(conf) {      sh "grunt noSuite --conf=${conf}"  }    /\*   \*  post: success   \*/  def success(currentBuild) {      notifySuccess(currentBuild)      cleanWorkspace()  }    def notifySuccess(currentBuild) {      if (isBackToNormal(currentBuild)) {          echo 'Your pipeline is back to normal'          emailext (            subject: "Build is back to normal: ${env.JOB\_NAME}",            attachmentsPattern: ATTACHMENTS\_PATTERNS,            body: "Your pipeline: '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]': was successful. Check console output at ${env.BUILD\_URL}/console",            to: EMAIL\_LIST          )      }  }    def isBackToNormal(currentBuild) {      currentBuild.previousBuild != null && currentBuild.previousBuild.result != 'SUCCESS';  }    /\*   \*  post: failure   \*/  def failure() {      notifyFailure()      cleanWorkspace()  }    def notifyFailure() {      echo "Your pipeline failed sending notification"      emailext (          subject: "Build failed: ${env.JOB\_NAME}",          attachmentsPattern: ATTACHMENTS\_PATTERNS,          body: "Your pipeline: '${env.JOB\_NAME} [${env.BUILD\_NUMBER}]': has an error. Check console output at ${env.BUILD\_URL}/console",          to: EMAIL\_LIST      )  }    /\*   \*  post: always   \*/  def always() {      reportResults()      archiveLogs()  }    def reportResults() {       System.setProperty("hudson.model.DirectoryBrowserSupport.CSP", "")       publishHTML (target: [        allowMissing: false,        alwaysLinkToLastBuild: false,        keepAll: true,        reportDir: 'Reports/report',        reportFiles: 'index.html',        reportName: "Cucumber Report"      ])  }    def archiveLogs() {      archiveArtifacts artifacts: 'Reports/all-logs.log', allowEmptyArchive: true  }    def cleanWorkspace() {      cleanWs notFailBuild: true  }    return this; |

**Jenkinsfile-ui-test**

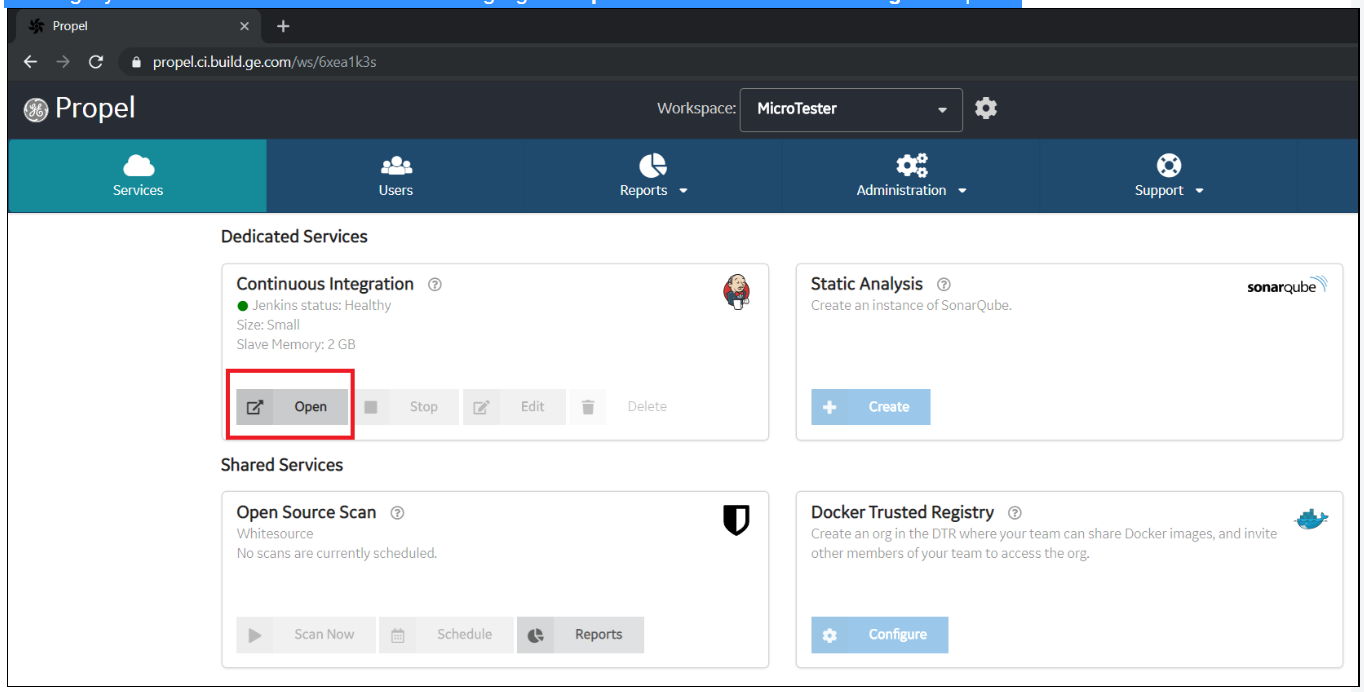
|  |
| --- |
| #!groovy    def DAYS\_TO\_KEEP = '5'  def BUILDS\_TO\_KEEP = '10'  def DOCKER\_IMAGE = 'registry.gear.ge.com/dig-apmdocker/apm-proui-node6-chrome-canary'  def DOCKER\_LABEL = 'dind'  def ENV\_NODE\_PROXY = 'NONE'  def ENV\_CHROME\_DRIVER = '2.45'  def ENV\_CHROME\_HEADLESS = 'true'  def JENKINSFILE\_COMMON = './ci/Jenkinsfile-common'  def common = null;    pipeline {  agent {  docker {  image DOCKER\_IMAGE  label DOCKER\_LABEL  }  }    options {  buildDiscarder(logRotator(  daysToKeepStr: DAYS\_TO\_KEEP,  numToKeepStr: BUILDS\_TO\_KEEP,  artifactDaysToKeepStr: DAYS\_TO\_KEEP,  artifactNumToKeepStr: BUILDS\_TO\_KEEP  ))  }    environment {  chromedriver\_ver = "${ENV\_CHROME\_DRIVER}"  CHROME\_HEADLESS = "${ENV\_CHROME\_HEADLESS}"  }    stages {  stage('setup') {  steps {  script {  common = load JENKINSFILE\_COMMON  common.setup()  }  }  }    stage('compile') {  steps {  script {  common.compile()  }  }  }    stage('test') {  steps {  script {  common.test("${env.conf}", "${env.suites}", "${env.tags}")  }  }  }  }    post {  success {  script {  sh 'node node\_modules/@ge-tools/ui-nxt/Core/CdnFilePathMerger.js'  common.success(currentBuild)  }  }    failure {  script {  sh 'node node\_modules/@ge-tools/ui-nxt/Core/CdnFilePathMerger.js'  common.failure()  }  }    always {  script {  common.always()  }  }  }  } |

Propel Job Creation

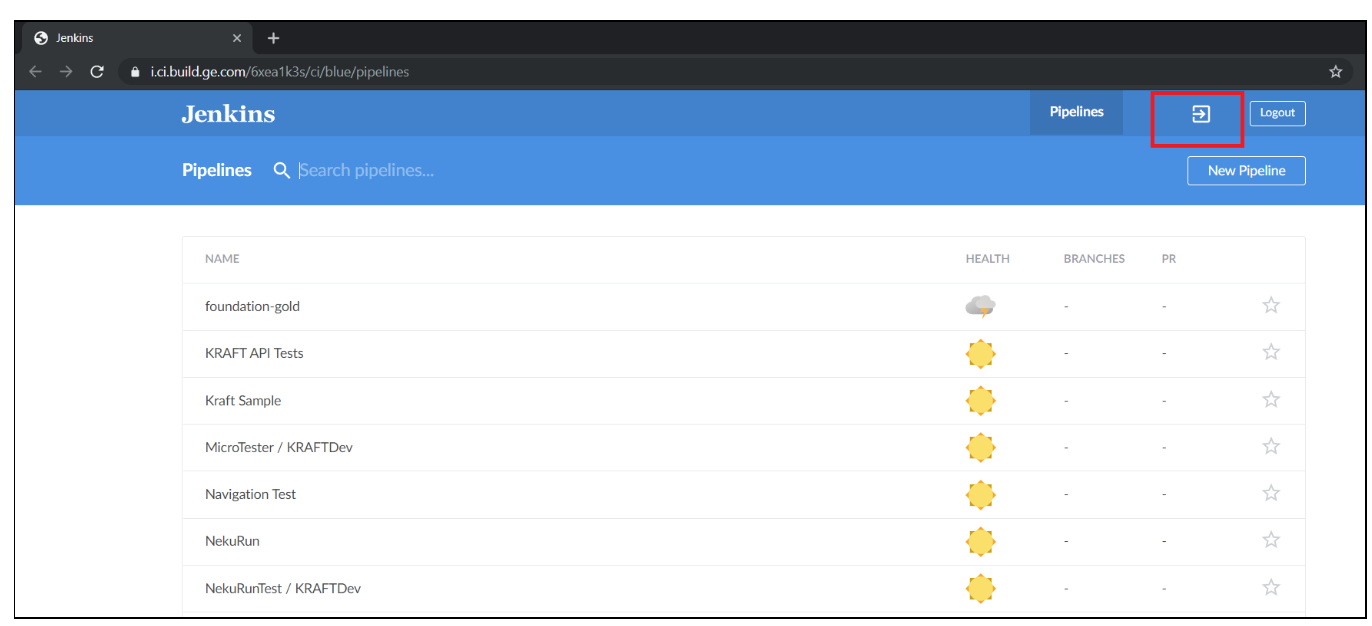
* Using below url login to propel account. You can use use your **SSO** and **Password** for credential.

<https://propel.ci.build.ge.com/welcome/>

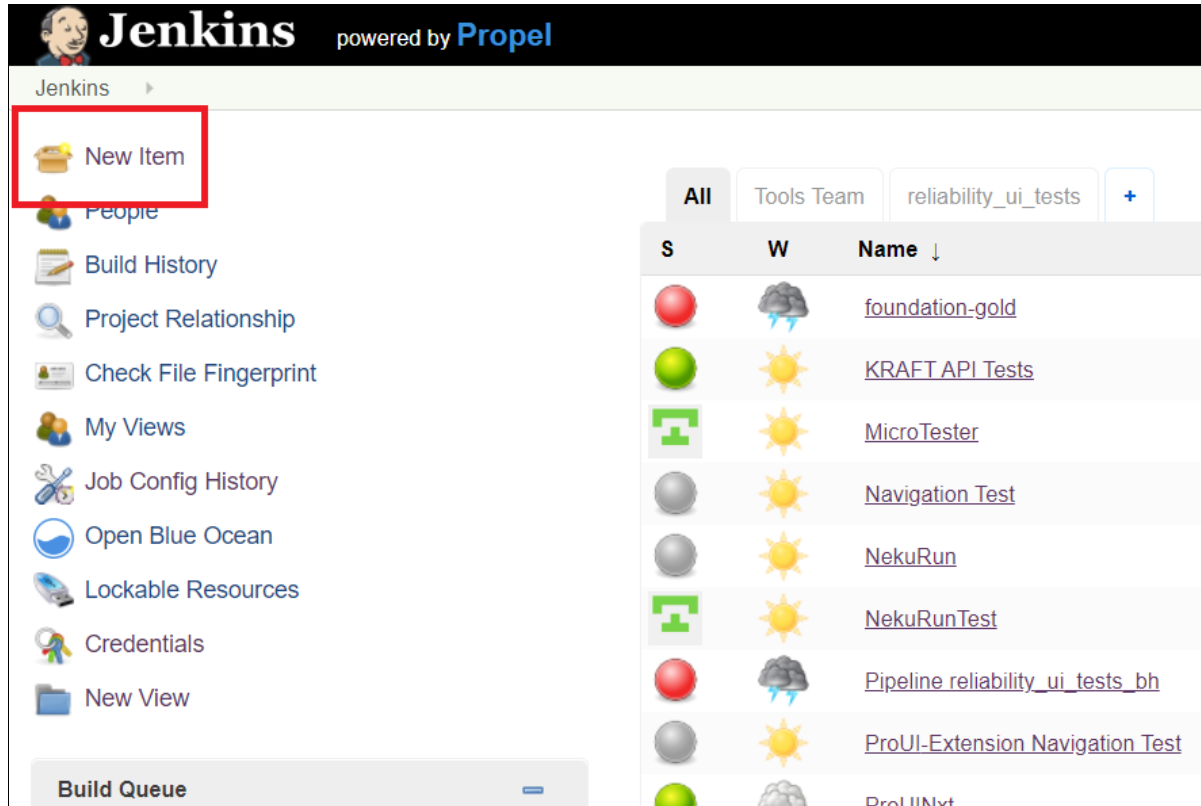
* After login you can see the below UI. Click on the highlighted **Open** button for **Continuous Integration**panel.



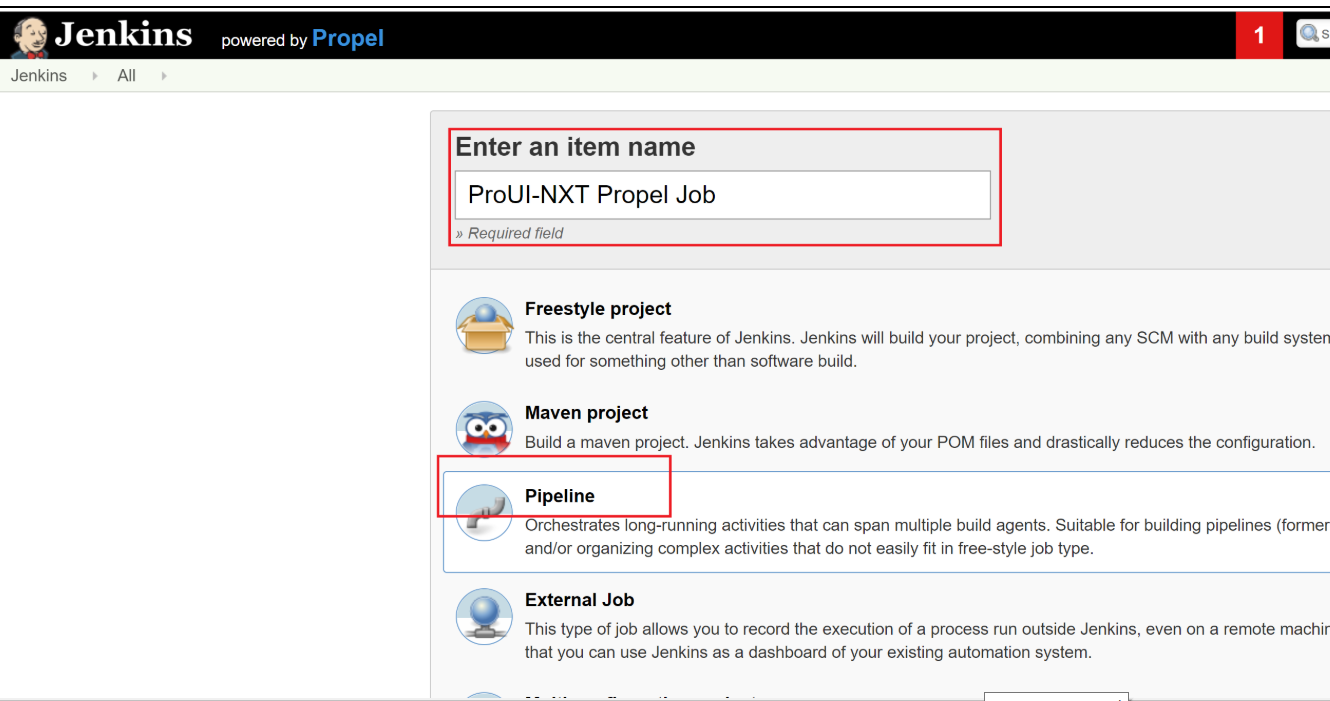
Click on the highlighted **Arrow** button for classic view.

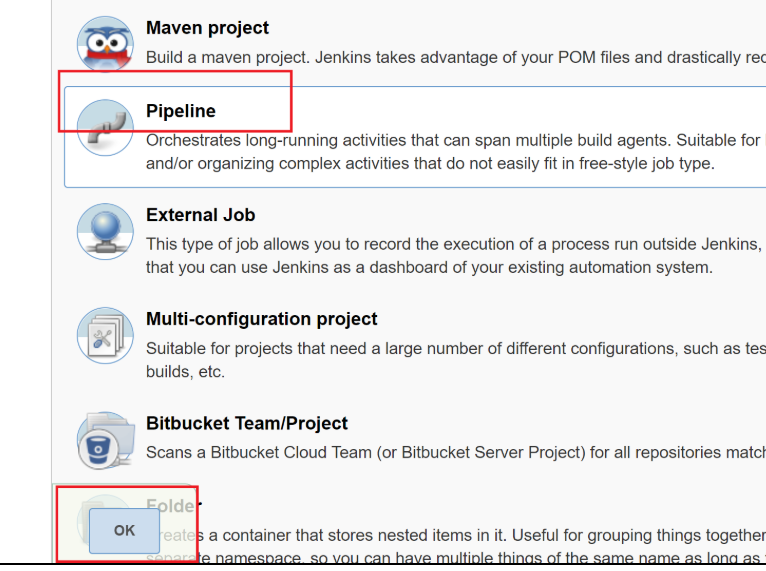


Now click on the **New Item** link button to create a new propel job.

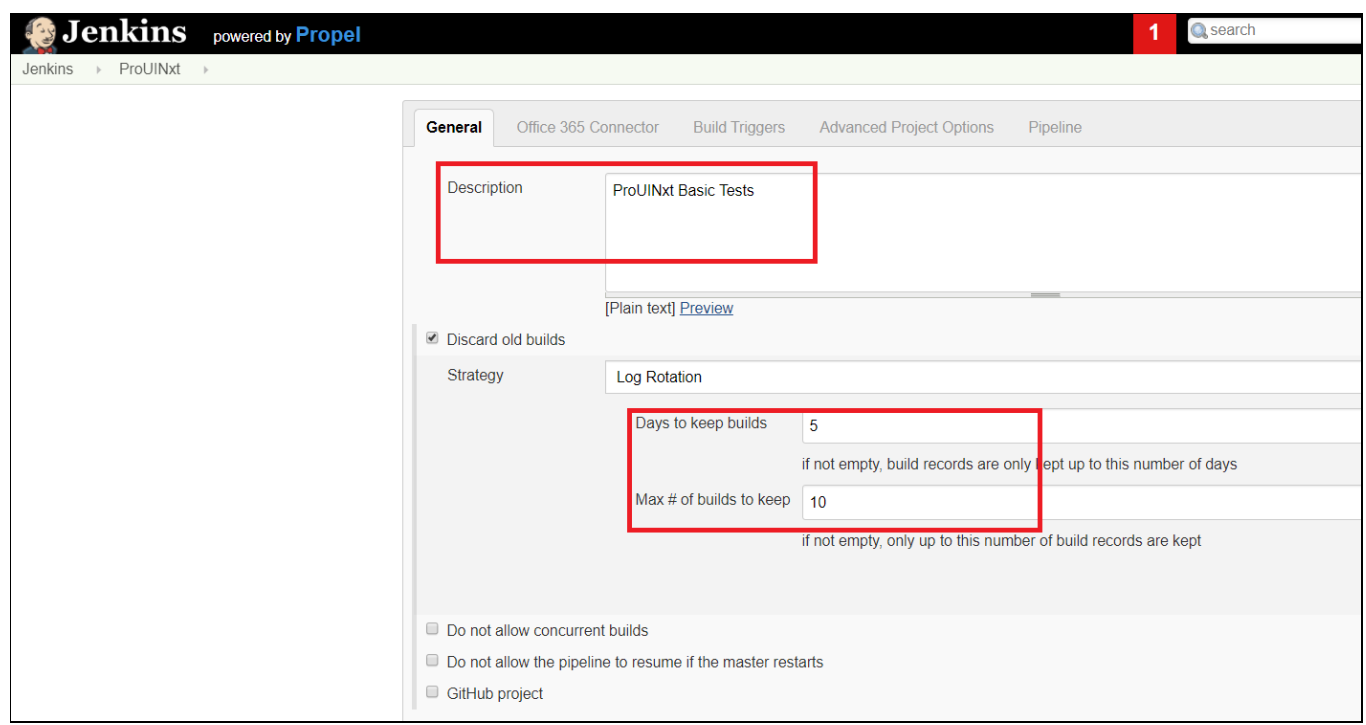


Enter Propel Job name in highlighted text box. Also select the **Pipeline** option and click on **OK** button.

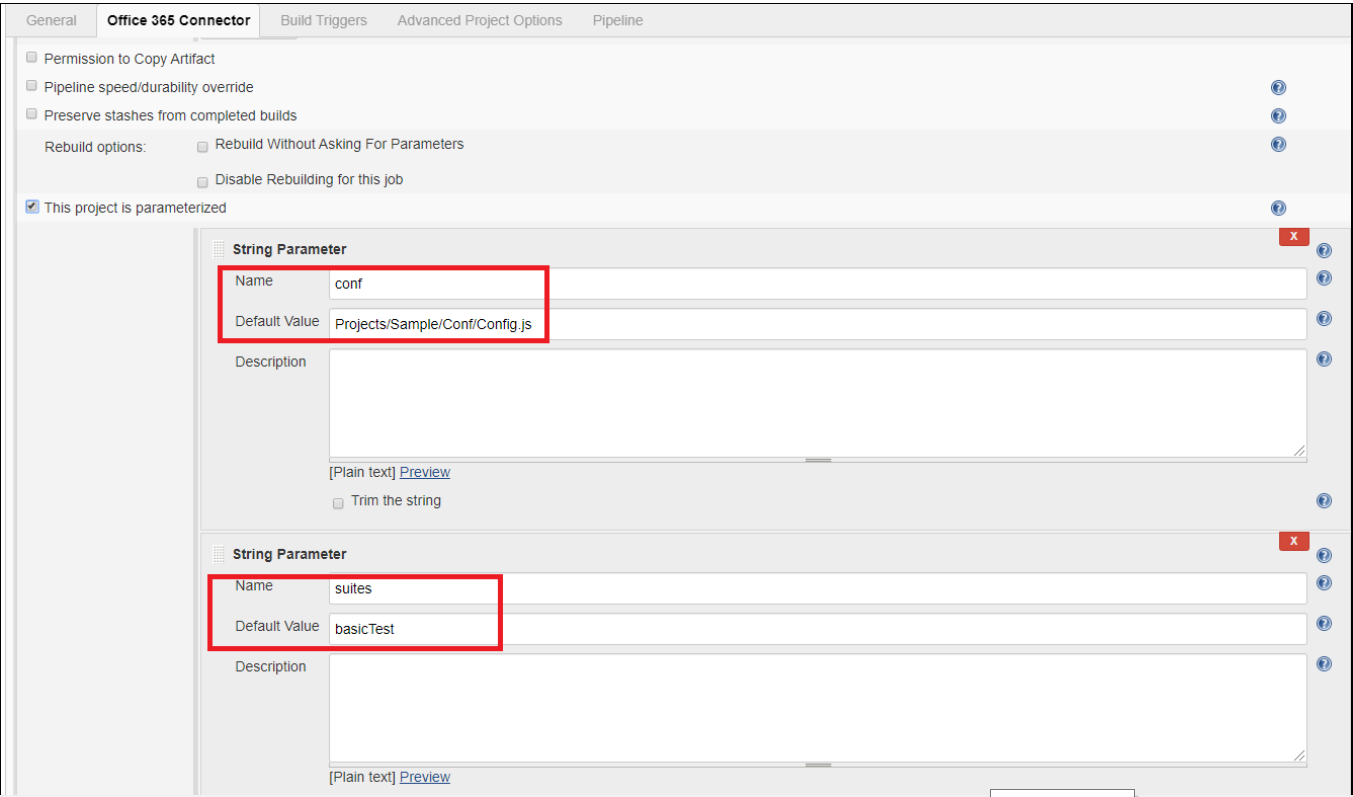




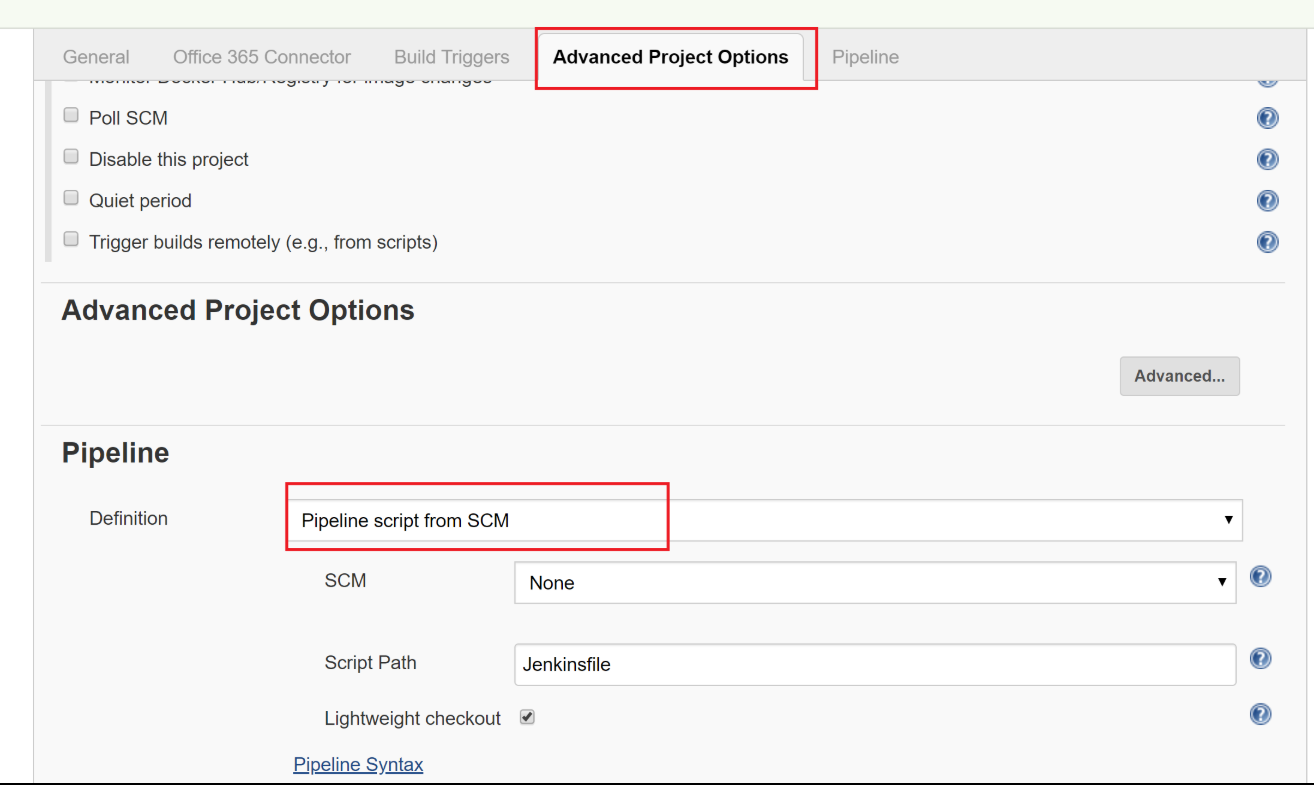
Enter the propel job description in the highlighted **Description** text box. You can also discard the old builds by selecting checkbox and set the strategy for log rotation by adding the parameters as shown below.



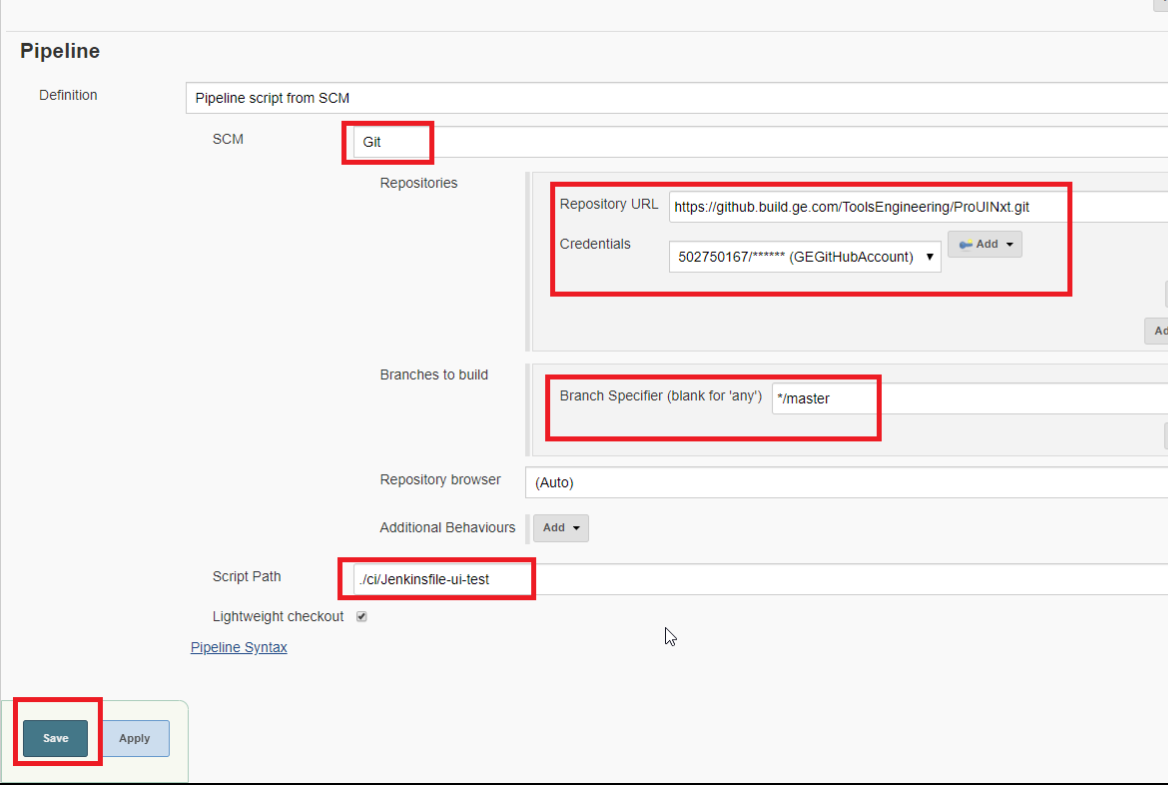
One can set the parameterized project by providing config file, test suites or tags(if any) details



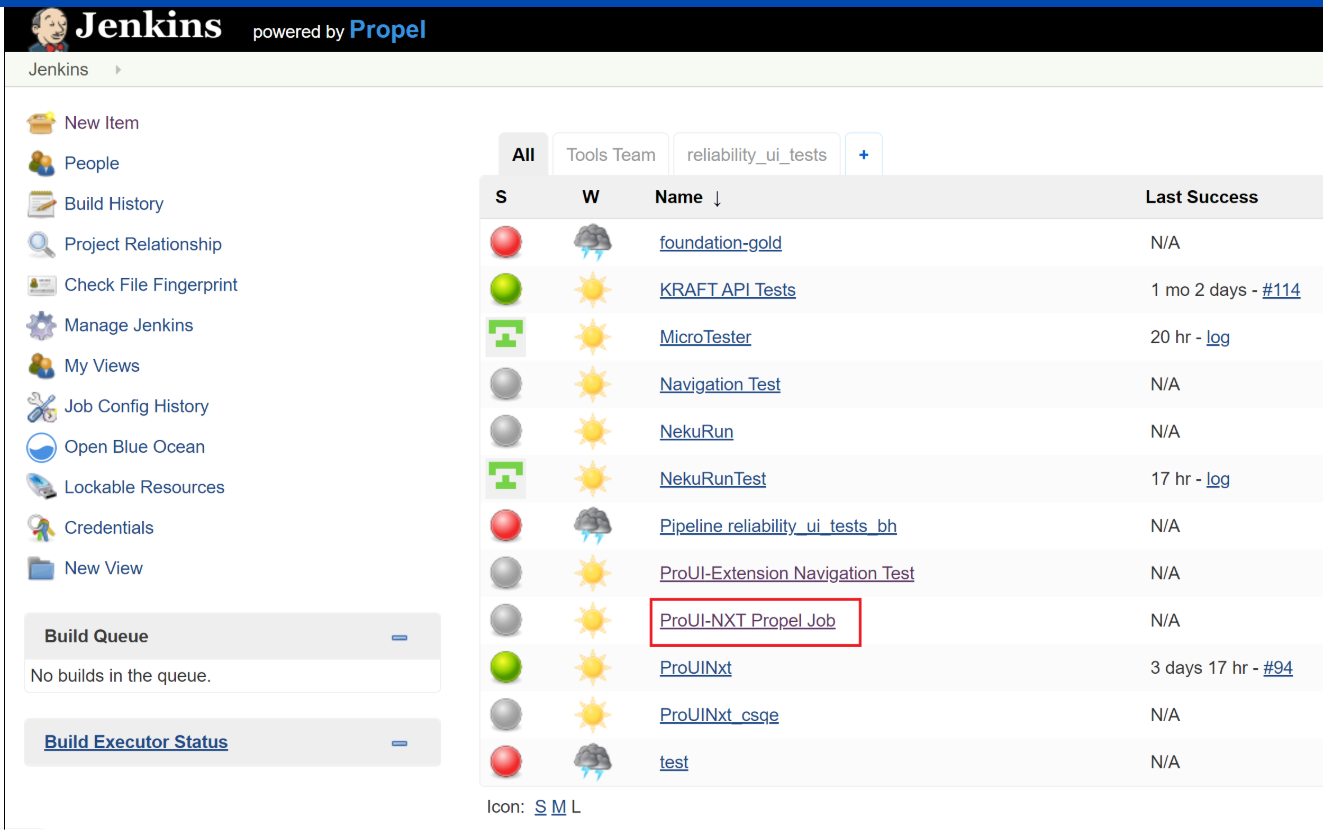
Click on the **"Advanced Project Options"** tab and choose **"Pipeline script from SCM"** option from the highlighted **Definition** drop down.



Choose the**"Git"** option from the highlighted **SCM** drop down. Enter git repo url in **Repository URL** text box, branch name in **Branch Specifier** text box. From **Credentials** drop down choose **"GEGitHubAccount"** user account to connect with your git repo and artifactory. If **Credential** drop down is not showing **"GEGitHubAccount"**user account you can check with **Devops team**. Specify Jenkins file name in **Script Path** text box. Jenkins file should be available in your git repo.



By clicking on **"Save"** button your propel job will be created. You can view the created propel job in dashboard.



Rally Posting

Results of the executed test cases can be directly posted to Rally based on the details provided in .env file present at the root level of the directory. For each feature executed, a folder is created with name of feature provided in feature file. Each scenario is considered as a Testcase, All the examples in scenario outline are considered as different testcases when the scenario name is dynamic ( i.e., Scenario name should contain headers of the Examples that makes it unique for each row in example).

.env Reference

Below are the mandatory keys that are needed for Rally posting. The attribute " postToRally" should be true to start rally posting

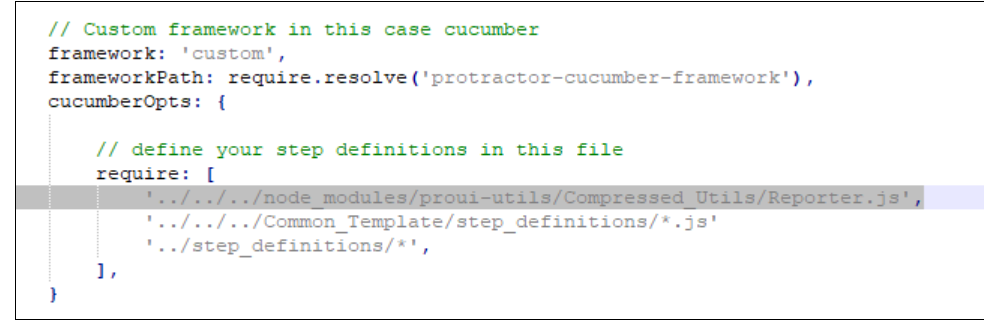
**.env reference**

|  |
| --- |
| chromedriver\_ver=78.0.3904.97    httpProxy=http://americas-cincinnati-internal.proxy.corporate.ge.com:80    rallyProxy=http://502635365:RIU.8272.ge@PITC-Zscaler-Americas-Alpharetta3pr.proxy.corporate.ge.com:80    apiKey=User\_Rally\_Api\_Key    postToRally=true    rallyworkspace=GE Digital    project=User\_Project\_Name    //Rally certificate handling  NODE\_TLS\_REJECT\_UNAUTHORIZED=0 |

For backward compatibility(proui-utils/proui-extension)

Below are the steps to follow for posting results to rally using lower version of cucumber( **cucumber**version**below 2.0.0**)

1. In protractor configuration.js file, include ''**node\_modules/proui-utils/Compressed\_Utils/Reporter.js'**' in "require" of "cucumberOpts" as shown below. This will generate reports in "Reports" folder under root directory with failure screenshot and based on details in .env file, Test results are posted to Rally.



2. Execute "grunt test" or "grunt <user\_defined\_registerTask\_name>" in the terminal

**Reference Gruntfile.js for backward compatibility(proui-utils/proui-extensions)** Expand source

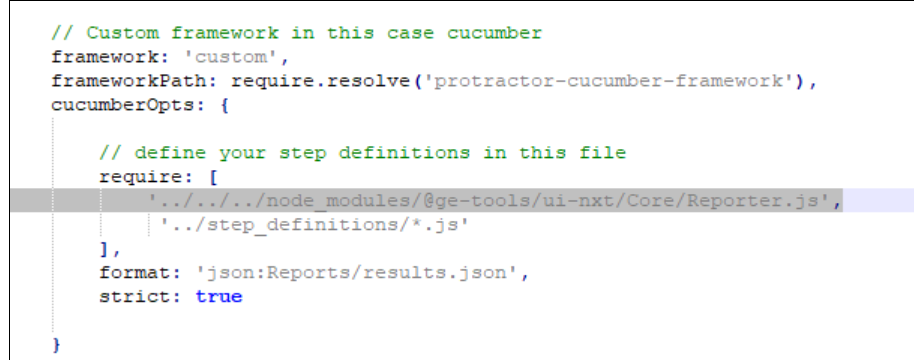
**Reference Gruntfile.js for backward compatibility(proui-utils/proui-extensions)** Collapse source

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174  175  176  177  178  179  180  181 | module.exports = function(grunt) {      'use strict';          grunt.initConfig({          pkg: grunt.file.readJSON('package.json'),            jshint: {              files: ['Gruntfile.js', 'step\_definitions/\*.js'],              options: {                  // options here to override JSHint defaults                  esversion: 6,                  globals: {                      jQuery: true,                      console: true,                      module: true,                      document: true                  }              }            },            execute: {              target: {                  src:['./node\_modules/proui-utils/updateChromeDriver.js']              }          },            shell: {              options: {                  stdout:true              },              disableSSL:              {                  command:'npm config set strict-ssl false'              },              npm\_install: {                  command: 'npm install --only=dev'// --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/)'              },              npm\_update: {                  command: 'npm update --only=dev'// --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/)'              },              protractor\_install: {                  command: 'node ./node\_modules/protractor/bin/webdriver-manager update'// --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/) --ignore\_ssl'              },              // ie\_install: {              //     command: 'node ./node\_modules/protractor/bin/webdriver-manager update --ie --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/)'              // }              },              protractor: {              default: {                  options: {                      keepAlive: false,                      configFile: 'Test\_Modules/Config/protractor.cucumber.common.conf.js',                      args: {suite: 'asset',                          params:{"login": {                          "baseUrl":grunt.option('baseUrl'),                          "username":grunt.option('username'),                          "password":grunt.option('password'),                          "adminusername":grunt.option('adminusername'),                          "adminpassword":grunt.option('adminpassword')                          }                        }}                  },              },            test: {              options: {                  keepAlive: false,                  configFile: grunt.option('conf'),                  args: {cucumberOpts:grunt.option('spec'),suite: grunt.option('suite')}              },          },            browser:{              options: {                  keepAlive: false,                  configFile: grunt.option('conf'),                  args: {cucumberOpts:grunt.option('spec'),suite: grunt.option('suite'), browser: grunt.option('browser')}              },          },            noSuite: {                  options: {                      keepAlive: false,                      configFile: grunt.option('conf'),                  },              },              singlerun: {},              auto: {                  keepAlive: false,                  options: {                      args: {                          seleniumPort: 4444                      }                  }              }          },          });        var target = grunt.option('target') || 'def';        grunt.loadNpmTasks('grunt-contrib-jshint');        grunt.loadNpmTasks('grunt-execute');        grunt.loadNpmTasks('grunt-contrib-jshint');        grunt.loadNpmTasks('grunt-shell-spawn');        grunt.loadNpmTasks('grunt-protractor-runner');        //grunt.loadNpmTasks('perfjankie');        grunt.registerTask('default',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:default');            } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });        grunt.registerTask('test',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              //grunt.task.run('shell');              grunt.task.run('protractor:test');            } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });        grunt.registerTask('noSuite',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:noSuite');            } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });          grunt.registerTask('browser',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:browser');            } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });          grunt.registerTask('run', ['protractorperf']);  }; |

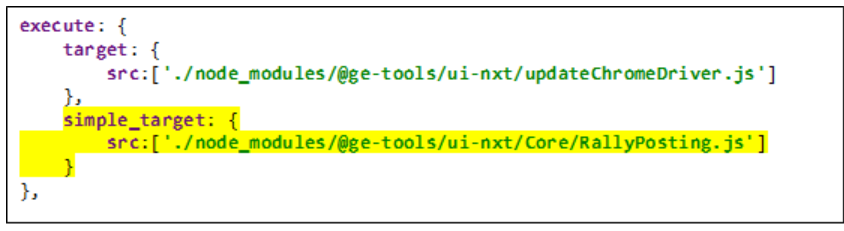
For new version(ui-nxt)

Below are the steps to follow for rally posting using latest cucumber. (**cucumber**version **above 2.0.0**).

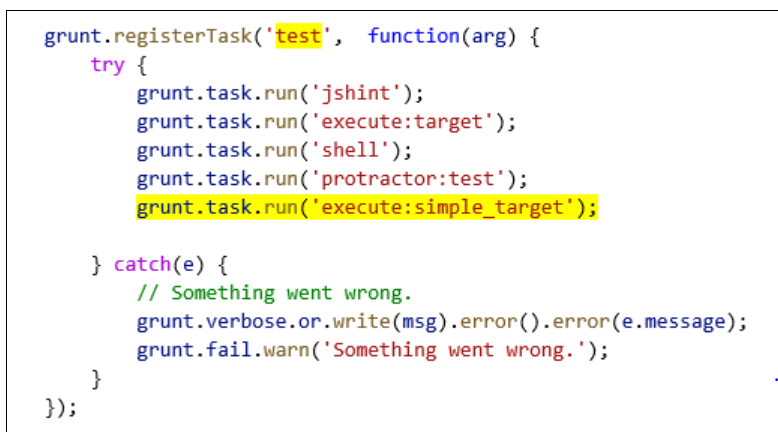
1. In protractor configuration.js file, include ''**node\_modules/@ge-tools/ui-nxt/Core/Reporter.js'**' in "require" of "cucumberOpts" as shown in the picture below. This will create reports with failure screenshot in "Reports" folder.



 In Gruntfile.js of PROUINXT project update **execute** of **grunt.initConfig** as below



 In GruntFile.js, add "**grunt.task.run('simple\_target')**" next to "**protractor:test**" task. i.e., update **grunt.registerTask** with name **'test'** or **<any\_task\_with\_protractor\_test>** as below



1. Execute command “**grunt test --force**” or “**grunt <user\_specified \_task\_name> --force**” in **terminal.**The “--**force**” argument is **manditory**, if the protractor test fails, then grunt gets aborted if “—force” argument is not given and rally posting task will not be executed.
2. **Reference Gruntfile.js for new version (ui-nxt)** Collapse source

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128  129  130  131  132  133  134  135  136  137  138  139  140  141  142  143  144  145  146  147  148  149  150  151  152  153  154  155  156  157  158  159  160  161  162  163  164  165  166  167  168  169  170  171  172  173  174 | module.exports = function(grunt) {      'use strict';          grunt.initConfig({          pkg: grunt.file.readJSON('package.json'),            jshint: {              files: ['Gruntfile.js', 'step\_definitions/\*.js'],              options: {                  // options here to override JSHint defaults                  globals: {                      jQuery: true,                      console: true,                      module: true,                      document: true                  }              }            },            execute: {              target: {                  src:['./node\_modules/@ge-tools/ui-nxt/updateChromeDriver.js']              },              simple\_target: {                  src:['./node\_modules/@ge-tools/ui-nxt/Core/RallyPosting.js']              }          },            shell: {              options: {                  stdout:true              },              disableSSL:              {                  command:'npm config set strict-ssl false'              },              npm\_install: {                  command: 'npm install --only=dev'// --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/)'              },              npm\_update: {                  command: 'npm update --only=dev'// --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/)'              },              protractor\_install: {                  command: 'node ./node\_modules/protractor/bin/webdriver-manager update'// --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/) --ignore\_ssl'              },              // ie\_install: {              //     command: 'node ./node\_modules/protractor/bin/webdriver-manager update --ie --proxy=[http://sjc1intproxy02.crd.ge.com:8080](http://sjc1intproxy02.crd.ge.com:8080/)'              // }              },          protractor: {              default: {              options: {                  keepAlive: false,                  configFile: 'Projects/Sample/Conf/Config.js',                  args: {suite: 'basicTest'}                  },              },                test: {                  options: {                      keepAlive: false,                      configFile: grunt.option('conf') || 'Projects/Sample/Conf/Config.js',                      args: {suite: grunt.option('suite') || 'basicTest'}                  },              },            browser:{              options: {                  keepAlive: false,                  configFile: grunt.option('conf'),                  args: {cucumberOpts:grunt.option('spec'),suite: grunt.option('suite'), browser: grunt.option('browser')}              },          },            noSuite: {                  options: {                      keepAlive: false,                      configFile: grunt.option('conf'),                  },              },              singlerun: {},              auto: {                  keepAlive: false,                  options: {                      args: {                          seleniumPort: 4444                      }                  }              }          },          });        var target = grunt.option('target') || 'def';        grunt.loadNpmTasks('grunt-contrib-jshint');        grunt.loadNpmTasks('grunt-execute');        grunt.loadNpmTasks('grunt-contrib-jshint');        grunt.loadNpmTasks('grunt-shell-spawn');        grunt.loadNpmTasks('grunt-protractor-runner');        //grunt.loadNpmTasks('perfjankie');        grunt.registerTask('default',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:default');              grunt.task.run('execute:simple\_target');          } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });        grunt.registerTask('test',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:test');              grunt.task.run('execute:simple\_target');          } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });        grunt.registerTask('noSuite',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:noSuite');              grunt.task.run('execute:simple\_target');            } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });          grunt.registerTask('browser',  function(arg) {          try {              grunt.task.run('jshint');              grunt.task.run('execute:target');              grunt.task.run('shell');              grunt.task.run('protractor:browser');              grunt.task.run('execute:simple\_target');            } catch(e) {              // Something went wrong.              grunt.verbose.or.write(msg).error().error(e.message);              grunt.fail.warn('Something went wrong.');          }      });          grunt.registerTask('run', ['protractorperf']);  }; |

# Release Details

| **Sr No.** | **Feature** | **Description** | **Release** | **Version** |
| --- | --- | --- | --- | --- |
| **Sr No.** | **Feature** | **Description** | **Release** | **Version** |
| 1 | Action | Provides set of methods for action functionality. | R1 | 1.0.1 |
| 2 | Assertion | Provides set of methods for assertion functionality. | R1 | 1.0.1 |
| 3 | Wait functionality | Provides set of methods for wait functionality. | R1 | 1.0.1 |
| 4 | Logger | Provides logging mechanism. | R1 | 1.0.1 |
| 5 | Exception handling | Central Exception handling mechanism implemented. | R1 | 1.0.1 |
| 6 | Latest version of Npm packages | Existing NPM packages updated for latest version. | R1 | 1.0.1 |
| 7 | TypeScript support | Framework is developed in TypeScript. | R1 | 1.0.1 |
| 8 | Typescript support for ProUI-Extension | TypeScript version of ProUI-Extension. | R1 | 1.0.1 |
| 9 | Zero or minimal learning curve | Keyword Driven Implementation. | R1 | 1.0.1 |
| 10 | Backward compatibility | Provides backward-compatibility with older version of framework. | R1 | 1.0.1 |
| 11 | Flexible folder structure | Can be plugged into any repository with any kind of folder structure. | R1 | 1.0.1 |
| 13 | In-line locator storage | Three ways to create Locator Repository. | R1 | 1.0.1 |
| 14 | Custom Assertions types | Wide variety of Assertions are provided so that user does not have to design them again and again. | R1 | 1.0.1 |
| 16 | New cucumber reports | New cucumber report integrated which also provides background steps generation and search functionality. | R1 | 1.0.1 |
| 17 | Artifactory integration | NPM packages for ProUI-Nxt and UI-bl deployed on Artifactory. | R1 | 1.0.1 |
| 18 | Typescript support for ProUI-Extension | TypeScript version of ProUI-Extension. | R2 | 1.0.2 |
| 19 | Retry Logic implementation based on specific Exceptions | Retry logic implemented at the framework layer to deal with specific exceptions without user intervention. Exceptions like StaleElementReferenceError, InvalidElementStateError, WebDriverError etc will be resolved by the framework. | R2 | 1.0.2 |
| 20 | Propel job Integration | Propel job Integration. | R2 | 1.0.2 |
| 21 | New features in Element class | New feature addition. | R2 | 1.0.2 |
| 22 | Get multiple elements | Find multiple elements. | R2 | 1.0.2 |
| 23 | Rally posting | Posting result to rally using latest version of cucumber. | R3 | 1.0.3 |
| 24 | Failure screenshot in cucumber Reports | With cucumber latest version, failure Test case's screenshot is attached to cucumber reports. | R3 | 1.0.3 |
| 25 | Removing continuous log while recovery | While recovering from exception, no more continuous log till exception recovery. | R3 | 1.0.3 |
| 26 | Added new Assertion functions | assertElementContainsAttributeValue, assertElementAttributeValue functions are added. | R3 | 1.0.3 |
| 27 | Requested reusable components | Part of R3, details available with Confluence page | R3 | 1.0.3 |
| 28 | Fix Rally posting | Rally posting for multiple After Steps in a scenario is fixed | R4 | 1.0.4 |
| 29 | New Functions in Action class | switchToFrame,selectParentFrame,getWindowHandle,switchToWindow | R4 | 1.0.4 |