# Summary

Important notes about the automation framework. What parts are not developed, what are the restrictions, how to set the parameters in certain methods. It will be divided in general notes, classes notes and possible plan in continuation development for this automation project.

# General notes

## Miscellaneous

* Apache POI is used for reading excel files and extracting their data from cells
* Selenium: web UI testing framework for sending commands where to navigate to
* TestNG: testing framework used for asserting content on the web pages. Also generates a report by default. It’s an extension of JUnit
  + Place all the test cases in test/testCases folder.
  + The TestNG framework is configured to run all test cases in the folder test/testCases

## Restrictions:

# Classes:

Specific things to know about the class and methods within them.

## lib/graphValidator.GraphValidator

### General idea

* Read information from excel file as an answer key. Gets compared with the values in the graph
  + Two rows get read.
    - Date: currently the cells must be formatted in a date in order to read it in Java and manipulate it
    - Value for the date: must be a format of number with two decimal places. This will make it easier to read information and manipulate it.
* The dates are read **only** in a monthly format.
  + Ex: Jan 2014, Feb 2014.
* It shouldn’t be hard to incorporate daily and yearly dates.
  + Use an enum to differentiate between different date types, daily, monthly, yearly and possibly more in the future.
  + Depending on the date format, set the Date Formatter accordingly.
* Numbers read from the excel files are formatted with commas.
  + Ex: 1000 in excel will become 1,000
  + Ex: 100000 in excel will become 100,000
* The parameter HighChart in validateGraph are only meant to be used by the subclasses. The subclasses will contain their own definition of hoverOver in order to use polymorphism and reduce repeating code so it can be easily managed. When calling the method, make sure the HighChart parameter are it’s subclass. It’s also used so more charts can be added and be validated in the same way (hopefully)

### getDateAndValue(int, int, int) and truncateColumns(ArrayList<Object[]>) method

* **NOTE**: currently the method getDataAndValue and truncateColumns will ignore cells that are empty AND cells that contains **0.00** regardless where they’re located. This may be important because some import files may contain and esi.activity ignores leading and trailing empty cells and 0.00, **BUT** it keeps 0.00 between cells that are non 0.00 cells.
  + If a test case is to be made, make sure the 0.00 is NOT between non 0.00 cells.
  + Refactor code to handle this case.

#### How getDateAndValue method works

* The method hasNext is used to check if there is a value next to the current cell (to the right)
* The user specifies where the starting row and column is for the date. It’s represented as an int for row and column. Also the starting row and column **MUST** contain a date, or the whole method will be confused and probably throw an error.
* The user specifies where the starting row is for the value. It’s represented as an int for row. The method is assuming the column for the values will be the **same** as the one for date.
* Starting at the date row and column, it checks if there is a date next to it (to the right), if there is, then the loop variable will be set to true; if not, it will get the date for the current cell and the value corresponding to the date and end the loop. This will loop until there is no value in the date row.

Ex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Jan 2014 | Feb 2014 | Mar 2014 |  |
| 10 | 20 | 30 | 40 | 50 |

* + Starting at Jan 2014 date cell, the method will check if there is a date next to the current cell (to the right), since there is it will get the date and corresponding value and store it in an Object[], the first element will be the date in a string and the second will be the corresponding value (the idea is to use some sort of a pair) and added to an ArrayList <Object[]>. Once the information extraction is complete for the current cell, it moves onto the next cell, which in this cases is Feb 2014. Same thing happens, checks if there a value next to the current cell, since there is, it gets the information for the current column and moves onto the next cell (Mar 2014). It checks if there is a value next to the current cell, and there is not since it’s empty. It will extract information for the current cell and end the loop, returning the ArrayList<Object[]>. The reason of using an ArrayList and not a HashTable, is because the validation method needs the values extracted in order. The method ignored the cells in red, and since they were ignored, the cells in orange were also ignored.

#### How truncateColumns works

* **NOTE**: this method will also ignore values that contain 0.00 or empty cells, whether it’s trailing or leading as well in between non-zero cells.

Ex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jan 2014 | Feb 2014 | Mar 2014 | April 2014 | May 2014 |
| <empty cell> | 10 | 0.00 | 20 | 0.00 |

* + Cells in orange whether the value for the corresponding date is an empty cell or 0.00
* **NOTE**: The truncateColumns method’s intention is to remove leading and trailing zeros along with empty cells. It will also ignore values that are before the corresponding date

Ex:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Jan 2014 | Feb 2014 | Mar 2014 |  |
| 10 | 20 | 30 | 40 | 50 |

* When the truncateColumns method is called, it will only extract the cells in green (in this case) and ignore the cells in orange. Even if the cells in orange has zero or non-zero values. The truncateColumns method must keep date and corresponding value unchanged. In other words, for this case, Jan 2014 must have the value 20, NOT 30 or 40 or etc. Same for Feb 2014, it corresponds to the value 30, NOT 20 or 40. The same idea is applied for the data returned by getDateAndValue
  + The cell color has no impact when calling truncateColumns. It’s used to make it easier to understand how it works.

#### -DexitOnFail ANT argument

* When running the tests from the command line using ANT file, an argument –DexitOnFail
  + Is used for this particular instance. When the argument is set to true, the assertions in validateGraph will throw an AssertionError if the assertions fails, and not is handled anywhere else, this will case the test case to stop executing and output the report at that point.
  + This is used to indicate where the assertion failed much more clearly, when the report is generated.
  + If the argument was not specified or was set to false, the failed assertion will NOT throw an AssertionError, it will log it and move on to the next column/bar/point.
  + When the ant file run, the exitOnFail is set as a property, it can contain either true, false or null.
    - At the beginning of each test case, this property will be checked. If the value is true, then GraphValidator.exitOnFail is set to true. It doesn’t do anything if it’s false or null. The default value for exitOnFail in GraphValidator is set to false.
  + The same idea is to be used for MetricValidator.

## lib/metricValidator.MetricValidator

* Used for validating metrics on dashboard.
* Meant to be used to reduce repeating code that is used to validate Metrics on a dashboard.
* At the moment it is being done hardcoded in the test cases.
* Also to be used to handle multiple metrics in a single dashboard object.
* Has NOT been implemented yet.
  + It should be similar to GraphValidator.validateGraph method. It should be a little shorter.

## lib/seleniumTools.WebDriverTools

### General idea

* Class used to setup browser drivers.
  + Supports Google Chrome, Mozilla Firefox and Internet Explorer
  + This can be used to expand to more web browsers.
  + If more web browsers are being added, the only thing to do is update BrowserType enum & add a switch case for the new BrowserType in WebDriverTools.getWebDriver(BrowserType) method

### Using WebDriverTools

* When instantiating WebDriver object, use WebDriverTools to do it, there are some configurations that a specific browser needs to have in order to perform correctly.
  + Ex: Mozilla Firefox needs to run with a specific profile, in the framework, it uses the default profile. The profile contains the login credentials for esi.activity (AutoAuth add-on is used to fill them out automatically)

### Taking screenshots

* Has functionality to take screenshots.
  + Meant to be used when an assertion fails or a web element cannot be found.
  + Currently not being used anywhere.
  + To be used in later versions of the automation framework

## lib/seleniumTools.highcharts.HighCharts

### General idea

* HighCharts is used for creating interactive JavaScript charts
* HighCharts class is used for extracting information from the graph on the web page and hovering over certain elements on the graph.
* HighCharts, BarChart, ColumnChart and LineChart are obtained from [GitHub](https://github.com/Ardesco/Powder-Monkey/tree/master/src/main/java/com/lazerycode/selenium/graphs) in order to be able to interact with the graphs on esi.activity.
* The main idea for using this is to validate tooltips whenever hovering over a column/bar/point on a graph.

### How the graphs work

* In order to understand how the graph worked, a lot of investigation in the web browser was required.
  + Tools to investigate with
    - Google Chrome: Developer tools. Ctrl + Shift + I
    - Firefox: Firebug. Firefox Add-on
  + The graph is contain within **svg** HTML tag
  + Within svg, tag **g** with class **highcharts-series highcharts-tracker** contains **rect** tag.
  + When the mouse hovers over one of the columns/bar/point, the corresponding **rect** tag’s attribute **fill** changes to some other value
  + Also when a mouse hovers over one the column/bar/point, a tool-tip is displayed for the corresponding column/bar/point.
  + In order to get the tool-tip to display, we need the framework to hover over the column/bar/point. To do that, we need the **rect** tag which will be used to hover over using JavaScript in the automation framework.
  + The idea is to get all the **rect** tags in svg > g (class: highcharts-series highcharts-tracker) and hover over them sequentially.
  + This is the main idea behind validating charts.
  + So far, I have checked for Column, Bar and Line graphs. It may or may/may not be different from Column, Bar or Line graphs.

### How it’s going to be used

* When validating a graph [using GraphValidator.validateGraph() method], one of the parameters is HighCharts, it is meant to pass in its subclass object types in order to be able validate different types of charts such as column, bar or line. This also applies polymorphism, so if more chart types are going to be validated, just create a subclass of HighCharts and must contain hoverOver(WebElement) method which will contain its own definition of how it hovers.
* BarChart, ColumnChart and LineChart are all subclasses of HighCharts.
  + These subclasses must contain a method called hoverOver(WebElement) in order to be able to hover a column/bar/point.
  + Each subclass has their own definition of how the hover takes place.

### Difference between code from github and local

* Only few methods were modified in order to use it with the graphs on esi.activity. The rest were untouched for possible future development
  + getXAxisLabelsText()
    - Changed the parameter value for By.cssSelector(String) method to “g.highcharts-series.highcharts-tracker > rect”
  + getXAxisLabelsWebElementList()
  + isChartDisplayed()
  + isLegendDisplayed()
  + isTooltipDisplayed()
  + hoverOver(WebElement)

## lib/seleniumTools.highcharts.BarChart

* A subclass of HighCharts
* Defines how the bars in the graph gets hovered
* Meant to be used in a polymorphic format in GraphValidator.validateGraph method

## lib/seleniumTools.highcharts.ColumnChart

* A subclass of HighCharts
* Defines how the columns in the graph gets hovered
* Meant to be used in a polymorphic format in GraphValidator.validateGraph method

## lib/seleniumTools.highcharts.LineChart

* A subclass of HighCharts
* Define how the line points in the graph gets hovered
* Meant to be used in a polymorphic format in GraphValidator.validateGraph method

## lib/excelUtils.ExcelUtils

### General idea

* Uses Apache POI open source framework
* Used to get information from a excel cell
* Can get information in String, double, Date
* Can get a cell object that can manipulated

## lib/esiActivity.EsiActivity

### General idea

* Create a semi-keyword driven framework so it’s easier to execute common tasks
  + Ex: creating workspace, deleting workspace
* Meant to reduce code in the test cases and to make it more manageable for large test cases.
  + Being able to not execute certain instructions by commenting out the method call in the test case to debug

### What the methods do

#### goTo methods

* Meant for navigating between pages.
* Ex: going from Dashboard to Calculations
* NOTE: will not be able to navigate if a drill down is displayed.
  + Ex: An Opportunity is opened and then the goToDashboard command is being executed, it will not be able to find the Dashboard link because the drill down window is open

#### load methods

* Load esiActivity is used for going to esiActivity web page. Probably change it to goTo method
* Load opportunity, measureGraph, and dashboardGraph
  + Are used for either creating or opening an opportunity/measure with certain filer criteria
    - Opening an Opportunity and view the production of the oil in bbl displayed in a column graph
      * The filter criteria is a list of xpath/css/linktext web elements that get clicked
        + Go to opportunity
        + Open opportunity

#### create methods

#### delete methods

#### validate methods

### Areas that required work

* Deleting workspace
  + Being able to delete a workspace by name and NOT by which workspace is first available.
* Validating and deleting dashboard graphs
  + Being able to delete a graph in dashboard that has multiple graphs.
  + Being able to validate a graph in dashboard that has multiple graphs
    - Validate the graph by specifying the name of the graph.
  + At the moment in order to validate and delete graphs, only **one** graph must be present in order to be able to validate and delete them properly.
    - If there’s more than one graph, the framework may get confused and validate/delete the wrong graph.
* Validating and deleting measures
  + Being able to delete a measure by name and NOT by whatever measure is first.
  + Being able to recognize all the measures and interacting with the desired measure.
  + At the moment, in order to validate and delete a measure, only one measure by present in order to make it work properly
    - If there’s more than one measure present, the framework may get confused and interact/validate/delete the wrong measure
* Validating and deleting dashboard metrics
  + Same idea as dashboard graphs
    - Being able to delete a metric in dashboard that contains multiple metrics or graphs
    - Being able to validate a metric in dashboard that contains multiple metrics or graphs
    - At the moment the framework can validate metrics, IF that is the only thing present in the dashboard, otherwise it will get confused and fail the test case.

# Possible development plans

* Currently this is what I have planned for it’s future development. It all depends on how successful is this automation project.
  + **Phase 1**: get the automation framework working
    - Get the basics working, recording user actions using Selenium IDE (Firefox plugin) and exporting it Java and refactored to manageable code.
  + **Phase 2**: Making it easier to use
    - Instead of having same set of steps being hardcoded, create a method that can be called over and over again and still perform the same functions.
      * Ex: Loading esi.activity
      * Ex: creating workspace using a method that takes workspace name, and excel File Location.
  + **Phase 3**: handling most (if not all) error, detecting web elements and can be interacted with easily and develop keyword driven testing
    - If an error is displayed the workspace already exits, the framework should detect it and modify the workspace name and continue on like normal
    - Recognizing multiple dashboard graphs and metrics.
      * Instead of having a single dashboard graph to be validated, it should have multiple dashboard graphs and can validate a specific one based on users request.
      * Same applies for metrics. There can be multiple metric values in the metric.
      * This applies to recognizing elements in terms of the name of the element not the xpath version of it.
        + Ex: recognizing measure types production in measure types when adding a new graph, rather than defining the
        + Ex: recognizing attributes from measures when adding a new chart. Rather than defining them through xpath, which can break if something changed.
    - Keyword driven testing will allow non-technical people to create a test case and run them with little or no trouble
      * Ex: Having a set of commands in an excel file. Having them read by the framework and execute commands accordingly

|  |  |  |
| --- | --- | --- |
| Action | Sub-action | Value |
| Open | Browser | Chrome |
| Create | Workspace | New Workspace |
|  | File | File location |
| Open | Opportunity | Well 1 Production |
| Apply | Filters |  |
|  | Measure | Production |
|  | Attribute | Oil |
|  | Graph | Column |
| Validate | Graph |  |
| Close | Opportunity |  |
| Go to | Workspaces |  |
| Delete | Workspace | New Workspace |
| Close | Browser |  |

* + **Phase 4**: Stay with Java or develop in easier programming language (Ruby)
    - This is for maintaining the framework and adding new improvements.
      * Generating better reports.
      * Having email sent out about the status of test cases.
      * Having test cases run on multiple computer using different browsers for a more realistic testing.
      * Create a better version of Selenium IDE
        + The idea is to remove writing commands in excel files and replacing with an interactive GUI to give a better context on what’s going on.
        + Making it easy as possible for the tester to create test cases. Possibly the business department create a specific test case
        + Able to record and playback user actions
        + When validating a graph, a prompt is displayed to define the excel file to be compared against

Display the excel and ask where the date row and column is. Same for the value corresponding to the date.

* + - Switching over to another easier language such as Ruby, so not so technical people can maintain it as well.