目录

[Intro about the excel file. 1](#_Toc1164064327)

[Create a testcase. 6](#_Toc1438056071)

[Create a new template. 10](#_Toc1212447238)

[Short Intro about selenium framework 13](#_Toc1692128792)

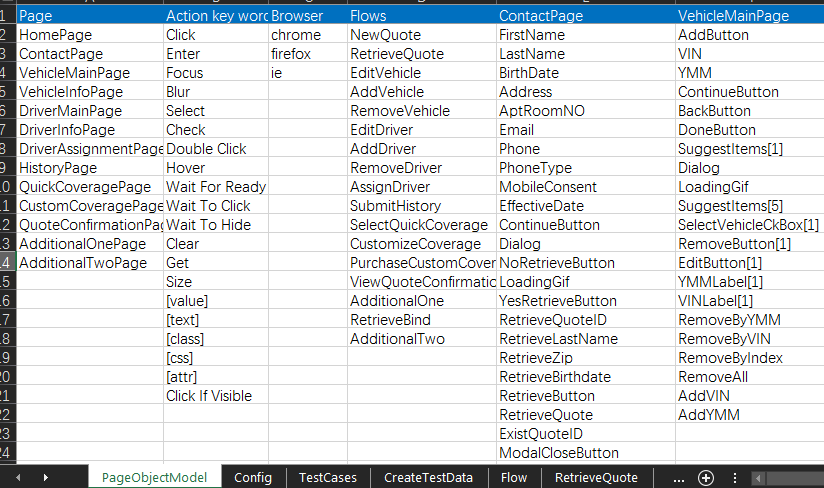
[SendKey(keyname) 17](#_Toc1721513778)

## Intro about the excel file.

1. **PageObjectModel**

List all related elements, or options here, just for convenient and maintainable when creating testcases.

1. Action key words like [value], [text], [class] are to get the value or text of one web element, and compare it with the given value. TestNG Asserts will take effect when meeting [value] actions.
2. SuggestItems[1] stands for the first item of various suggestItems.



1. **Config**

Config the running timeout, wait time, and retry limit, etc.

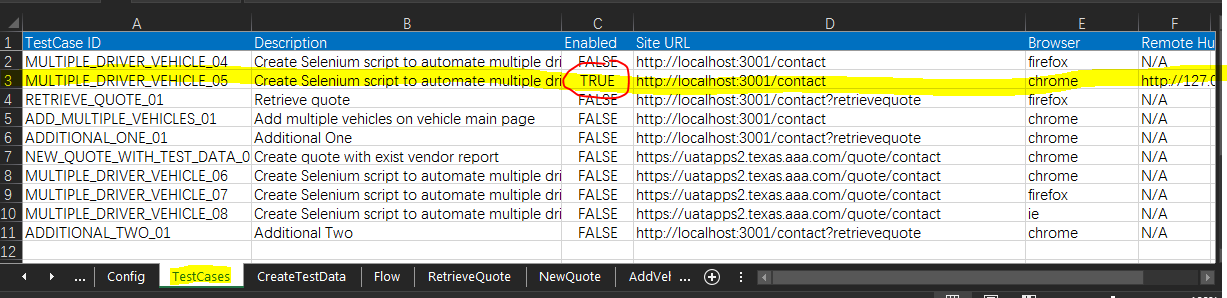
1. *Operation* is now used to wait an element to be visible, clickable or hidden, etc.
2. *Async* is now used to wait a specific time when an element is entered or clicked and so on. Especially like the Address, VIN and YMM fields, it will take some time to display the child elements.
3. *Implicit* is now used to set the implicitlyWait.
4. *Redirect* is now used to set the pageLoadTimeout.
5. *Retry* is now used to rerun the testcase for specific times when a testcase failed.



1. **TestCases**

The overall testcases, collect all needed info such as which URL site or browser each testcase will run on.

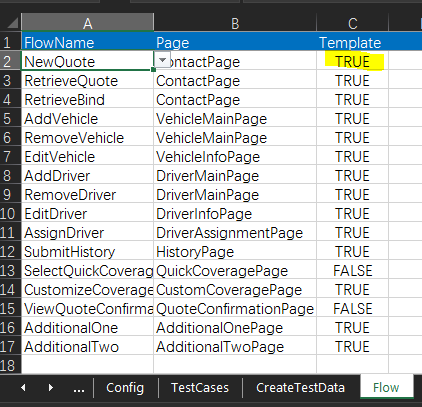
1. The TestCase ID should be unique.
2. Set “Enabled” to TRUE if you want to run this testcase this time.
3. Fill in the Remote Hub if the testcase runs on remote site.



1. **Flow**

Set the related templates to true.

1. if you are going to create new-a-quote-flow testcase using the NewQuote sheet template, then set the template to TRUE. Otherwise you should create the test steps on ContactPage sheet.
2. For more details, please see the following *two methods to create the test steps*.

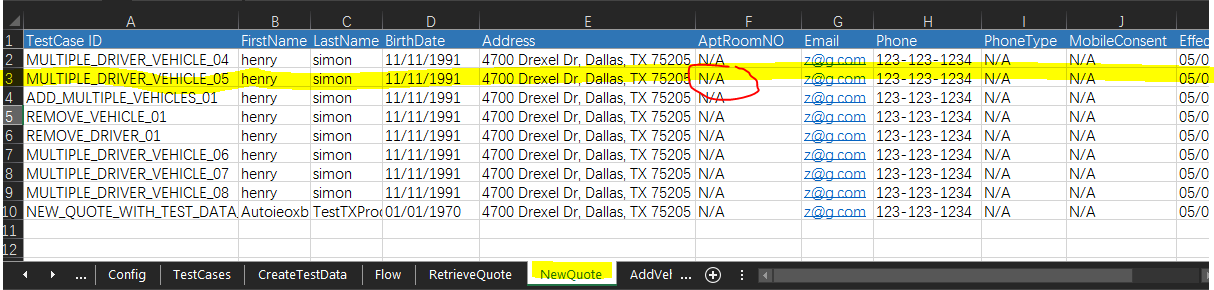


1. **Two methods to create the test steps.**

**5.1 Template (convenient)**

If you are using the template to create related test steps within one specific flow, you should list all the test steps in this corresponding flow template.

1. Each test step should have a TestCase ID, which is consistent with the TestCase ID of the testcase in TestCases sheet.
2. Set the invisible or hidden fields to N/A.
3. Also for visible fields, set it to N/A if you want to ignore this field (optional fields can often be ignored)
4. Leave the filed to blank will still cause corresponding behavior for the element, will either enter nothing for input filed or click the button.



**5.2 Page (flexible)**

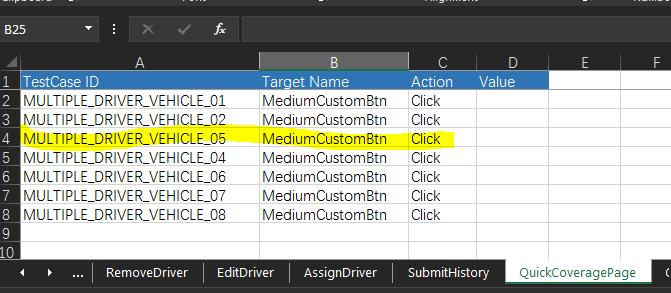
If you are not using template to create related test steps within some specific flow, then you should find the corresponding page the workflow is on. List all the related test steps in this specific page.

List the detailed test steps line by line.

Each step will be detailed described as the targetName, action (click, enter, select...) and value (what you want to enter, select, or compare with).

Leave the value blank if no parameters are needed.

(note: In the page sheet, all test steps can be highly customized, which is more flexible than using the template. But you should be more familiar with the business info or rules of the website, also familiar with the framework.)



## Create a testcase.

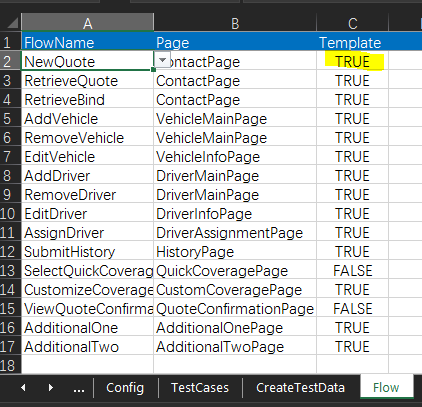
E.g.: create a multiple vehicle and driver workflow testcase. In this testcase, we will go throgh the following flows step by step:

*NewQuote, AddVehicle, EditVehicle, AddDriver, EditDriver, Assign Driver, Submit History, SelectQuickCoverage, CustomizeCoverage, and ViewQuoteConfirmation.*

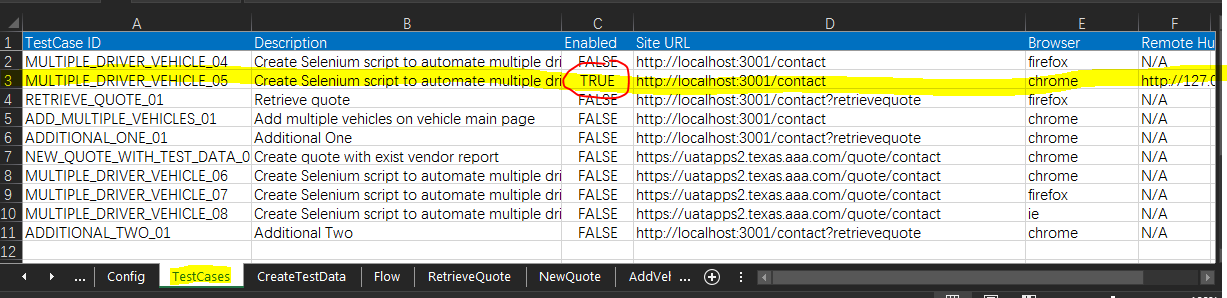
1. **Preparation:**

Make sure the related templates within the work flow are set to TRUE in Flow Sheet.

(note: SelectQuickCoverage and ViewQuoteConfirmation is not using template, so the test steps will be created in QuickCoveragePage sheet and QuoteConfirmationPage sheet. Otherwise, the test steps will be created in the template like NewQuote sheet and AddVehicle sheet).

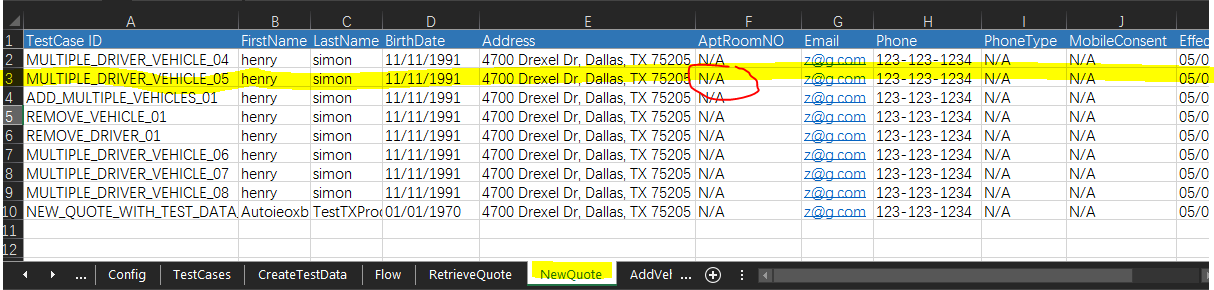


1. **Find the TestCases sheet, fill in the needed info**, such as the TestCase ID (unique), site URL, Browser, or simple description.



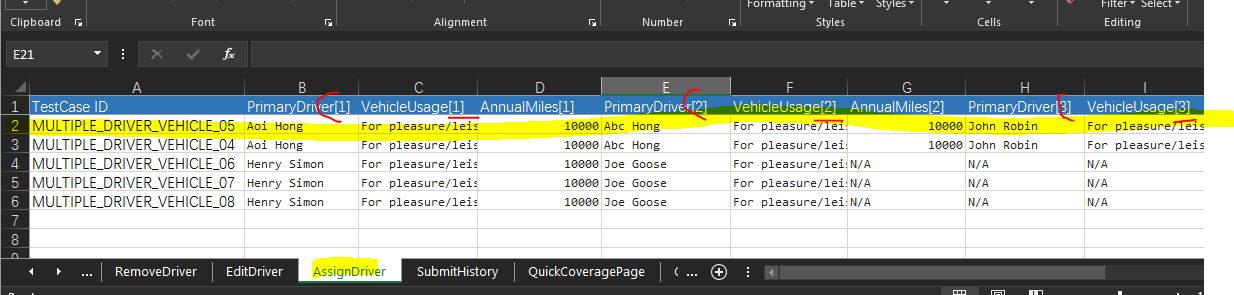
1. **Follow the testcase workflow and fill in the corresponding template or page step by step)**
2. Find the NewQuote sheet, fill in the test data.

*Note*: The TestCase ID should be consistent with the TestCase ID in step 2).

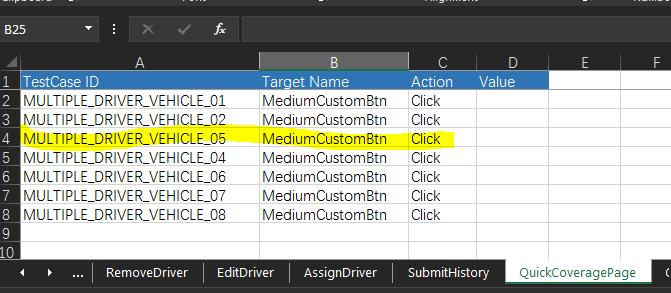


1. Find the AddVehicle sheet, fill in the test data.
2. Find the EditVehicle sheet, fill in the test data.
3. Find the AddDriver sheet, fill in the test data.
4. Find the EditDriver sheet, fill in the test data.
5. Find the AssignDriver sheet, fill in the test data.

*Note:* On DriverAssignment page, you will see multiple primary driver or vehicle usage questions. So if you want to select the first vehicle’s primary to Aoi Hong, fill it in PrimaryDriver[1]. Fill in the AnnualMiles[2] if you want to enter text in second AnnualMiles input filed.



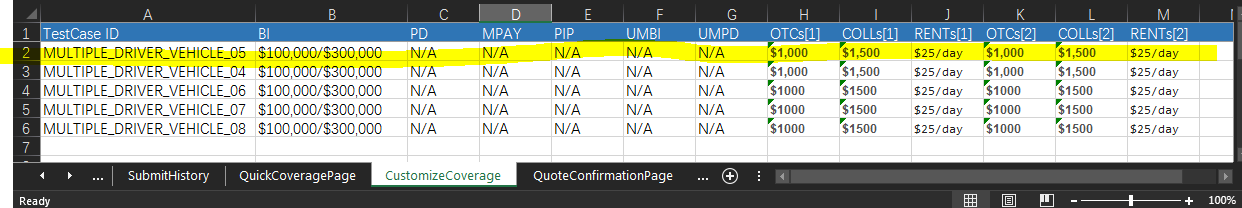
1. Find the SubmitHistory sheet, fill in the test data.
2. Find the QuickCoveragePage sheet, fill in the test data. This flow is not using template, so the test data will be detailed described in QuickCoveragePage sheet.



1. Find the CustomCoverage sheet, fill in the test data.

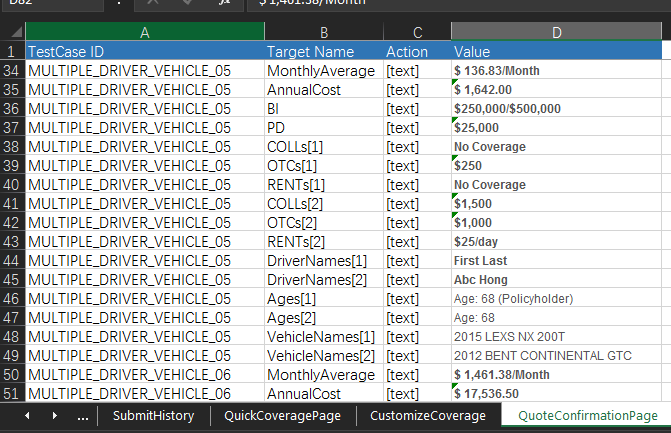
*Note:* For COLLs[1], it stands for the first vehicle’s COLL coverage, COLLs[2] stands for the second vehicle’s COLL coverage.

This is a customized behavior as we can only find one COLL coverage or one RENT coverage after we click corresponding tab. So strictly, COLLs is not a web elements list as there is only one element found each time. But in real website, it looks like there are several COLLs, the only difference is that some of COLLs are hidden. So, we create the behavior to handle them ([1] stands for the tab index, click the specific tab and then update the COLL coverage under this tab), find CustomVehicleCoverageBehavior class in customizedBehavior package for details.



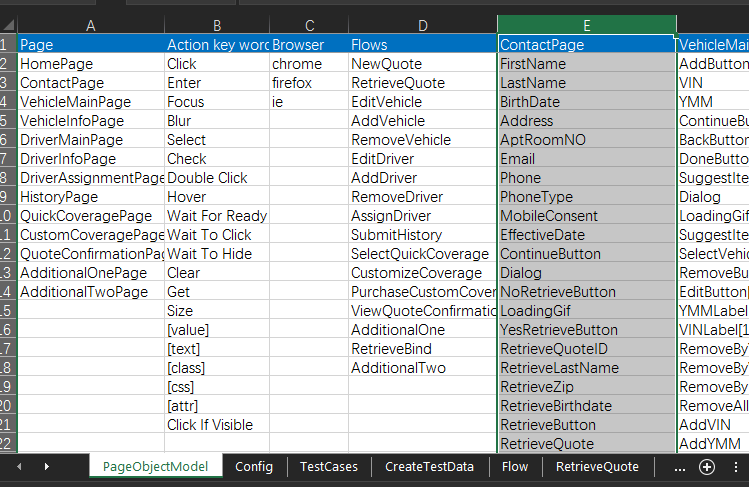
1. Find the QuoteConfirmationPage sheet, fill in the test data.

*Note: [text]* is mentioned above, which is to get the real displayed value of one element on website, and compare it with the given value.

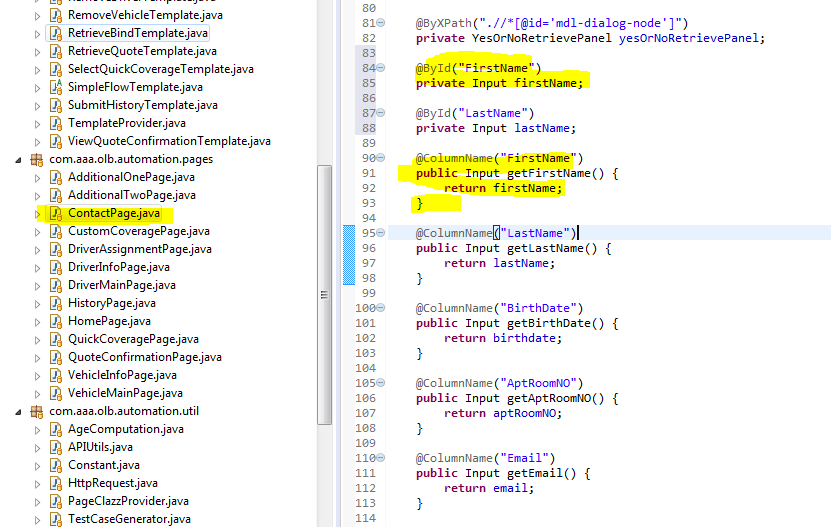


## Create a new template.

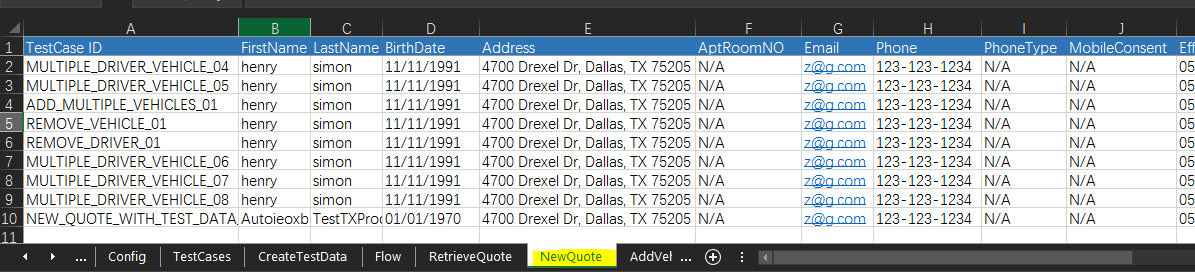
**(skip step 1 and step 2 if no new elements will be appended)**

1. Update page object model sheet if new elements should be appended. 
2. Update specific page Class to handle new created fiedls.

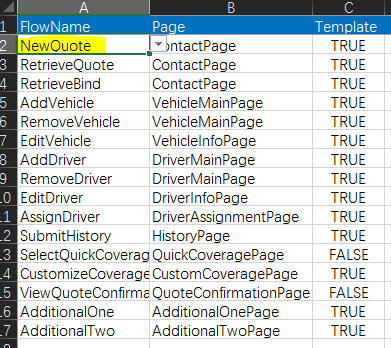
@ColumnName vaule should be consistent with the field name in PageObjectModel sheet.



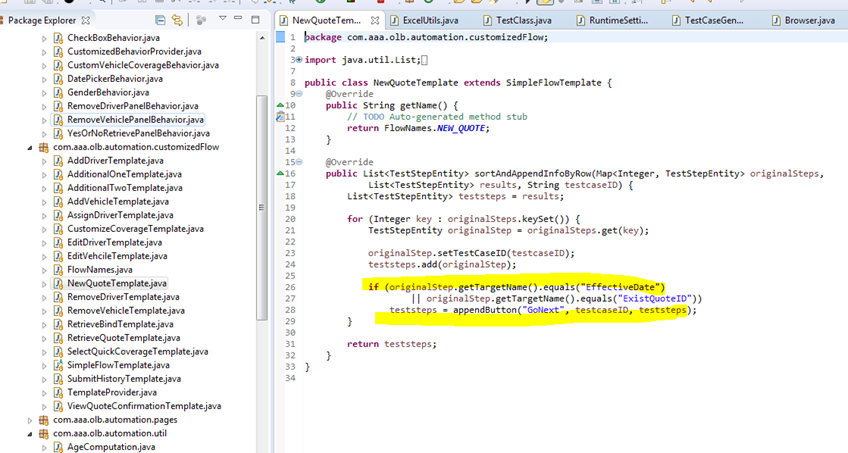
1. Create a new sheet for the template like NewQuote.



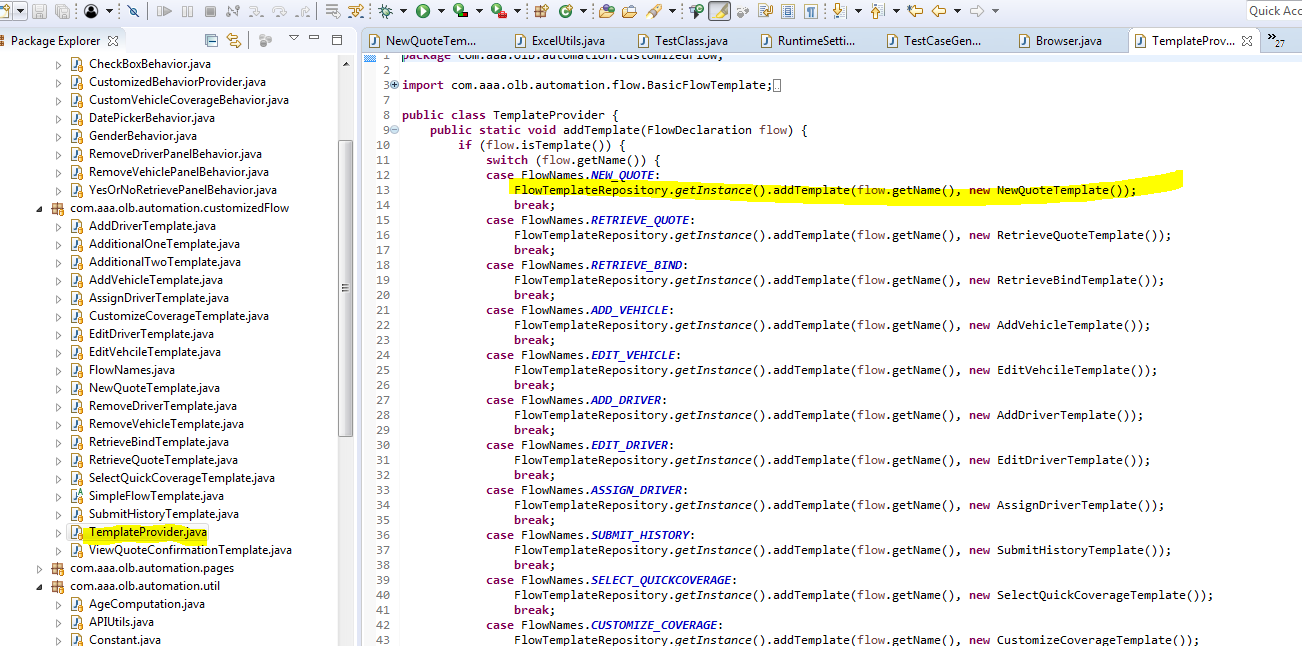
1. Append the template info on *Flow* sheet.



1. Find the customizedFlow package, create the new template class, like NewQuoteTempalte, append some missing but needed behaviors (such as click next buttons, or click recalculate buttons when the coverage is updated), based on the template sheet you created.



1. Append a case to add the new added template to the template repository.

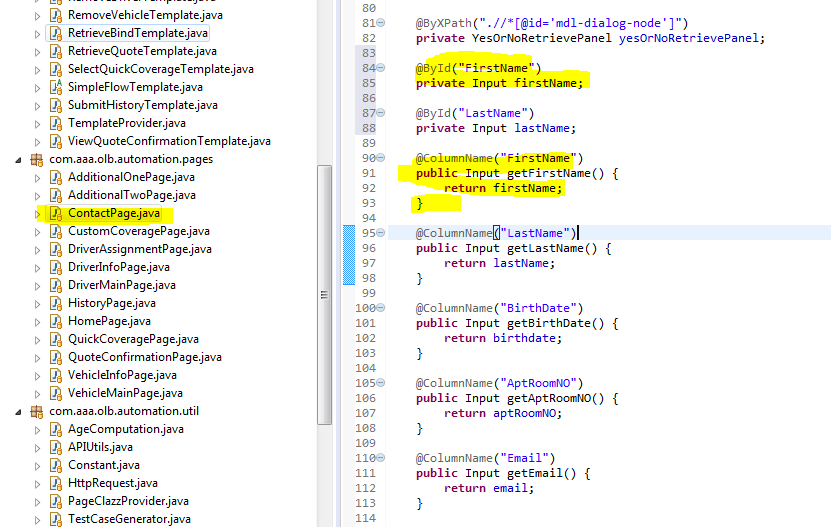


## Short Intro about selenium framework

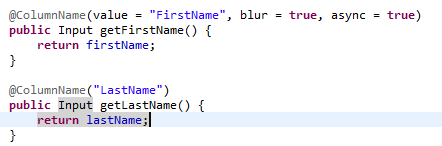
1. Using @ById, @ByClassName, @ByXPath, @ByTag, @ByCss to find the web elements.

The web elements are wrapped into specific Input, Button, Dropdown, etc.

(see more wrapped element types in Controls package of the framework)

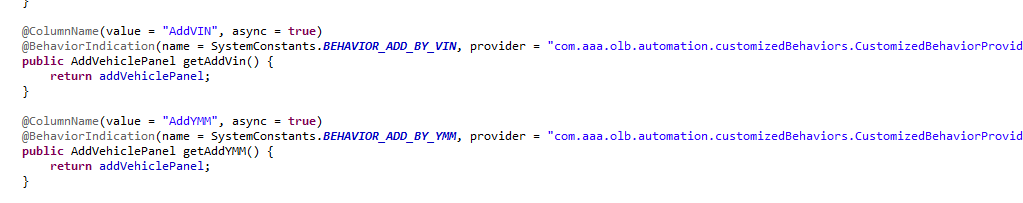


1. Using @ColumnName to map the page elements with the fields from the sheet.
2. Set blur to true if want this field to blur after entering.
3. Set async to true if want this field to wait specific time after entering or clicking.

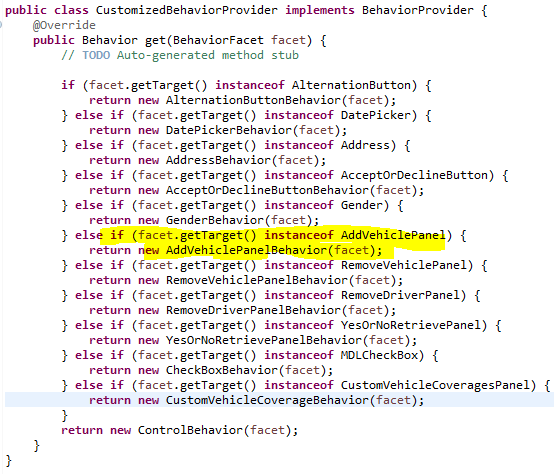


1. Using @BehaviorIndication to customize special behavior of one element or several elements group.

e.g.:



With the given provider in behaviorIndication, find the CustomizedBehaviorProvider, and get the needed AddVehiclePanelBehavior.



Find the right Behavior class, generate different actions depends on different given behavior name in BehaviorIndication.

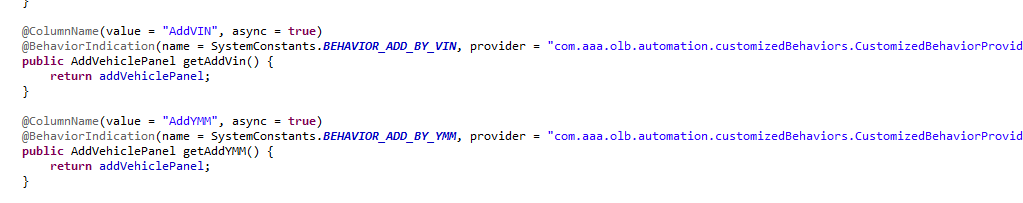


1. Customized component to handle special business.

e.g. AddVehiclePanel

1. Using XPath (or id, tag, etc) to find the needed elements group.

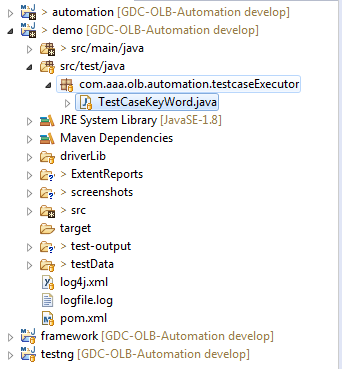




1. Create the component to get every child elements.



1. The entry to run the testcases.



## SendKey(keyname)

Keyname:

***NULL*** ('\uE000'),

***CANCEL*** ('\uE001'), // ^break

***HELP*** ('\uE002'),

***BACK\_SPACE*** ('\uE003'),

***TAB*** ('\uE004'),

***CLEAR*** ('\uE005'),

***RETURN*** ('\uE006'),

***ENTER*** ('\uE007'),

***SHIFT*** ('\uE008'),

***LEFT\_SHIFT*** (Keys.***SHIFT***),

***CONTROL*** ('\uE009'),

***LEFT\_CONTROL*** (Keys.***CONTROL***),

***ALT*** ('\uE00A'),

***LEFT\_ALT*** (Keys.***ALT***),

***PAUSE*** ('\uE00B'),

***ESCAPE*** ('\uE00C'),

***SPACE*** ('\uE00D'),

***PAGE\_UP*** ('\uE00E'),

***PAGE\_DOWN*** ('\uE00F'),

***END*** ('\uE010'),

***HOME*** ('\uE011'),

***LEFT*** ('\uE012'),

***ARROW\_LEFT*** (Keys.***LEFT***),

***UP*** ('\uE013'),

***ARROW\_UP*** (Keys.***UP***),

***RIGHT*** ('\uE014'),

***ARROW\_RIGHT*** (Keys.***RIGHT***),

***DOWN*** ('\uE015'),

***ARROW\_DOWN*** (Keys.***DOWN***),

***INSERT*** ('\uE016'),

***DELETE*** ('\uE017'),

***SEMICOLON*** ('\uE018'),

***EQUALS*** ('\uE019'),

// Number pad keys

***NUMPAD0*** ('\uE01A'),

***NUMPAD1*** ('\uE01B'),

***NUMPAD2*** ('\uE01C'),

***NUMPAD3*** ('\uE01D'),

***NUMPAD4*** ('\uE01E'),

***NUMPAD5*** ('\uE01F'),

***NUMPAD6*** ('\uE020'),

***NUMPAD7*** ('\uE021'),

***NUMPAD8*** ('\uE022'),

***NUMPAD9*** ('\uE023'),

***MULTIPLY*** ('\uE024'),

***ADD*** ('\uE025'),

***SEPARATOR*** ('\uE026'),

***SUBTRACT*** ('\uE027'),

***DECIMAL*** ('\uE028'),

***DIVIDE*** ('\uE029'),

// Function keys

***F1*** ('\uE031'),

***F2*** ('\uE032'),

***F3*** ('\uE033'),

***F4*** ('\uE034'),

***F5*** ('\uE035'),

***F6*** ('\uE036'),

***F7*** ('\uE037'),

***F8*** ('\uE038'),

***F9*** ('\uE039'),

***F10*** ('\uE03A'),

***F11*** ('\uE03B'),

***F12*** ('\uE03C'),

***META*** ('\uE03D'),

***COMMAND*** (Keys.***META***),