**FPS Automation Framework Usage**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Description | Author/Editor |
| 0.1 | 13/07/2022 | First Draft | Priya Tayade |
|  |  |  |  |

Table of Contents

[**1.** **What is FPS Automation Framework …………………………………………………………………………………………..**3](#_Toc108429699)

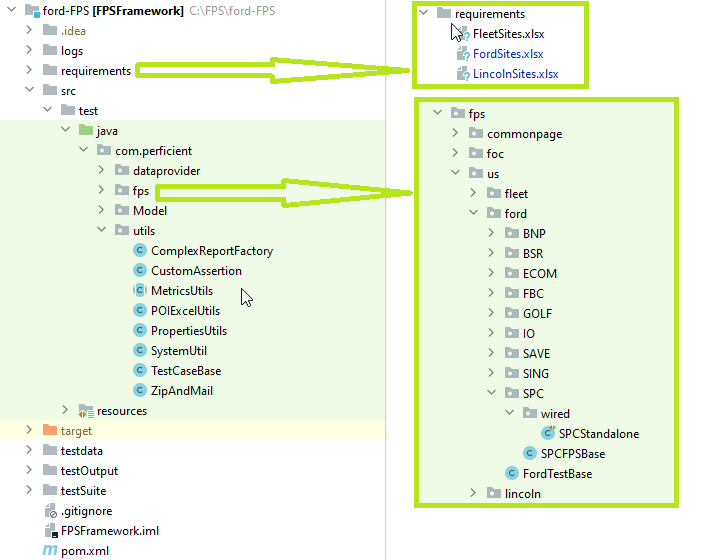
[**2.** **The FPS Automation Framework folder structure** 4](#_Toc108429699)

[**3.** **Working of FPS Framework** 9](#_Toc108429700)

[**4.** **Diagrammatical representation of the Automation Flow** 14](#_Toc108429701)

1. **What is FPS Automation Framework?**
2. FPS automation framework is basically deals with Analytics testing. When user performs any site activity [ex. Click any vehicle CTA] then Set Calls generate in Network tab. This framework can fetch those network calls and assert them with the expected data.
3. This framework is specified in Java language.
4. It is a maven project which makes use of TestNG framework.
5. It includes Selenium scripts to automate site activities.
6. It uses Browser Mob Proxy tool to capture Network calls. It is a tool which is used to capture network traffic, HTTP requests and response in an HAR format. HAR is a json-formatted archive file.
7. It includes methods which are used to extract Set calls [ in the form of JSON data] and parse it as per project need.
8. Scope of FPS testing is for Ford/Lincoln US/CA corners. So, the framework is also designed in same manner. There is different package for every corner.
9. We use IntelliJ IDE to create, debug and execute FPS test scripts.

# **The FPS Automation Framework folder structure**



1. **Pom.xml:**

POM is an acronym for Project Object Model. The pom.xml file contains information of project and configuration information for the maven to build the project such as dependencies, build directory, source directory, test source directory, plugin, goals etc. We don’t need to add external java once dependency added in pom.xml.

1. **logs**:

This folder handles test logs. Whenever test fails, It generate .log file which include all the details such as error, build info, system info and driver info etc.

1. **requirements**:

This folder consists of excel sheets in which all the expected data is stored. This expected data is further being asserted with actual data.

1. **testdata**:

This folder consists of all the property files as per different applications. All the required testdata [Ex: Selenium locators and their values, URLs, Listeners and credentials] are stored at this folder location.

1. **testOutput**:

This folder contains the output files in the form of extent reports which are generated after the execution of testcases.

1. **testSuite:**

This folder is for managing and organizing tests. It consists of application wise XML files. For each application there is one XML file which contains all the tests related to that application. It allows tester to execute group of tests at a time. In FPS, we don’t execute separate test class. We always execute XML file as per application corners. [Ford/Lincoln US/CA]

1. **.gitignore:**

This file belongs to git repository. If you don’t want to push any particular folder/file to remote repository, then you can add it to gitignore file. [Note: target folder is included in this file]

1. **.idea, FPSFramework.iml and External libraries**:

These folders/files are autogenerated by IntelliJ Idea and maven project. There is no as such purpose of it in FPS Framework.

1. **Folder** **– src/test/resource**:

This folder contains log4j.properties file. The log4j. properties file is alog4j configuration file which stores properties in key-value pairs. The log4j properties file contains the entire runtime configuration used by log4j. This file will contain log4j appenders information, log level information and output file names for file appenders.

1. **Model:**

Model package consists of classes which all deals with Nameplate values. [Ford/Lincoln Vehicle for which we need to run test]. It includes all the parameters that we need to validate from the set calls. Parameters such as nameplate, year, brand and trim etc. It includes Setters and getters for the same. [We also called it as Pojo Classes in java]

1. **dataprovider:**

Package dataprovider consist of RequestFactory class. This class has methods which adds HTTP requests (Set calls in FPS case) to a java list. This list is further processed while parsing set calls. HTTP requests are listened by listeners.

1. **FPS:**

This package has 3 subpackages as below:

* 1. ***Commonpage:***

It consists of PageElement classes for all Ford applications. Methods for automating site activities are defined in PageElement classes. It includes methods to automate clicks, enter inputs, submit forms etc.

* 1. ***Foc:***

It is a package which maintains testclasses and baseclasses for ford/Lincoln CA corners. It has again 3 subpackages named as fleet, ford and lincoln. Every package has one baseclass.[ex: focTestBase, LNCTestBase, fleetTestBase etc]. These baseclasses extends testcasebase. It includes methods related to get URLs for that particular site. These 3 subpackages again have separate package for each application. Every application has one baseclass which inherits its parent class. i.e., baseclass of fleet/ford/Lincoln. Every application package has subpackage named as “wired” which maintains testclasses related to those applications. These testclasses inherits the baseclass of that application.

* 1. ***Us:***

It is a package which maintains testclasses and baseclasses for ford/Lincoln US corners. Rest structure flow is as like as foc package.

1. **util**:

This package contains all the Utility classes that will be used in the framework. The classes and its working are listed below.

* 1. ***ComplexReportFactory***:

This class contains ExtentReport generation and its configuration code.

ExtentReport is an HTML reporting library for Java, which can be used with Selenium WebDriver. Once test execution is done, it got generated. ExtentReport gives simple and elaborated execution reports of executed tests.

* 1. ***CustomAssertion***:

This class contains all the assert conditions related to code. These methods are used to check whether the actual data is matching with expected data or not.

* 1. ***MetricsUtils***:

This class contains the methods which deals with Excel sheets. It has methods of loading an Excel sheet, read and write methods of Excel related operations, methods for finding any particular row and their column value and method to replace column values etc.

* 1. ***POIExcelUtils***:

This class contains methods which again deal with Excel sheet. Note: This class is no more in use in FPS framework. [ Note: We are making use of MetricsUtils class]

* 1. ***PropertiesUtils***:

This class contains methods which are used to read and write property files.

* 1. ***SystemUtil***:

This class has methods to load the test data in properties file and driver killer method.

* 1. ***TestCasebase***:

This class contains all the TestNG annotations except *@Test*. It has all the methods which are used for basic setup of framework like setup Browser, teardown & setUpChromeWin32 is used to perform all the operation after and before test execution. Additionally, it includes Browser mob Proxy configuration and methods to extract FPS calls.

* 1. ***ZipAndMail***:

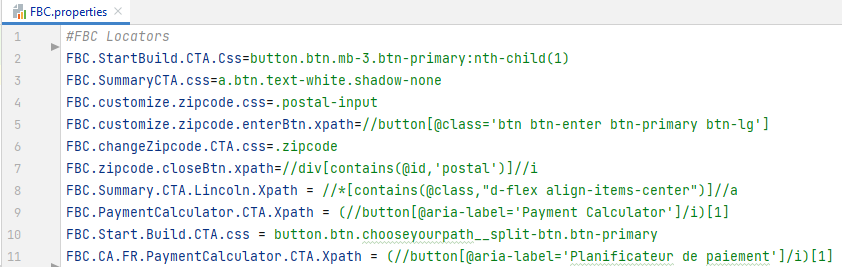
This class contains the methods which deals with mail operation. It has the method which will convert all the reports in zip file and send the mail to the recipients.

# **Working of FPS Framework**

**Note:** To create any class, right click on package in which you want to keep it. Click on “New” then to “Class” option. Give any relevant name as per your requirement.

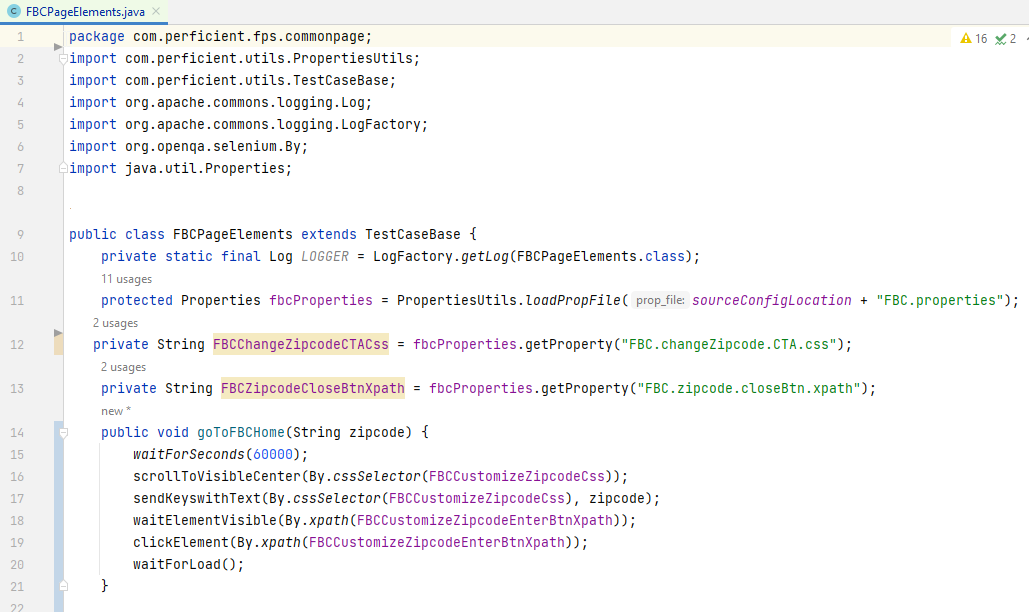
1. **Creating properties file:**

First you need to create property file [ex: FBC.properties] which contains locators data. Your Xpath, CSS selector values must be declared in this file. It looks like below.



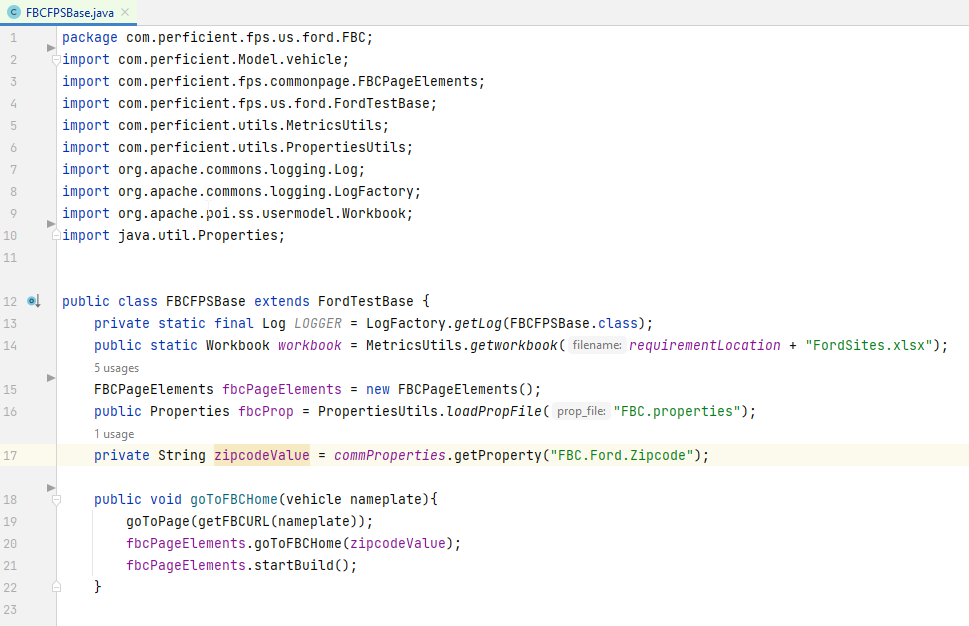
1. **Creating PageElements class:**

There must be one PageElement class for your script. This class defines required methods for automating site activities like clicking any element, putting input, entering zipcode and submitting form etc. It also makes use of properties file data.



1. **Creating Base class:**

Next you need to create base class. As mentioned earlier, every application has one base class which inherits ford/Lincoln/fleet baseclass. This class contains declaration of methods mentioned in PageManager class.



1. **Creating Test class:**

Test class extends the base class that we have created in previous step. It is a class which we actually run as test scripts. This class must contain a method with annotation @Test. @Test is an annotation of TestNG which tells that this method is a “test”, and it should be executed when the user runs the class.



***More detail about Test Class: Please refer above diagram***

**I.** InitialFPS() is a method which gets expected data from excel sheet and store that into Metrics class instance, anotherValue.

**II.** Then we are iterating through vehicle class to get required nameplate value. So that we process our code further with nameplate required.

**III.** TestCaseBase.fpsvar : In testcasebase, one hashmap named as “map” is define. It consists of parameters we need to validate for which site activity. The value, which is needed for the particular test, we are geting this value from map and storing it into TestCaseBase.fpsvar

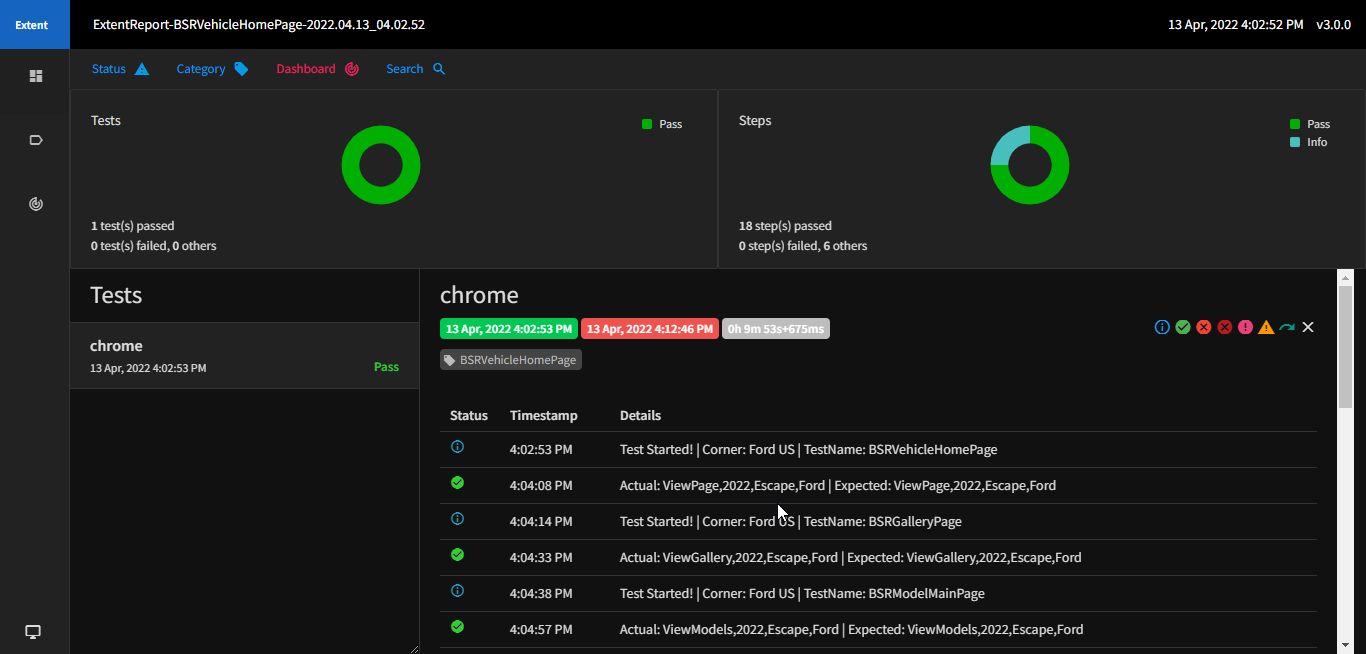
**IV.** In next step, we need to call site activity methods that we need to automate. Once all the above line of code got executed, set call will generated.

**V.** This set call is extracted with the help of **fpssetcall** method. And It will return the actual data to string. We will discuss the working of this method later.

**VI.** Final step is of assertion. It will tell you whether the script got pass or fail.

1. **Report Generation**

After execution of the testcase, Extent report gets generated in Testoutput folder.



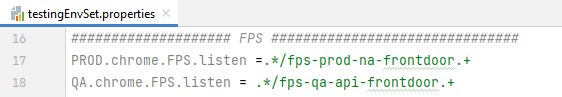
If you want the execution report should be available to all the team members. Then you need to uncomment the method i.e ZipandMethod in Testcasebase class. also add the recipient email ids in mailreport.properties file.



1. **Working of extracting set calls:**

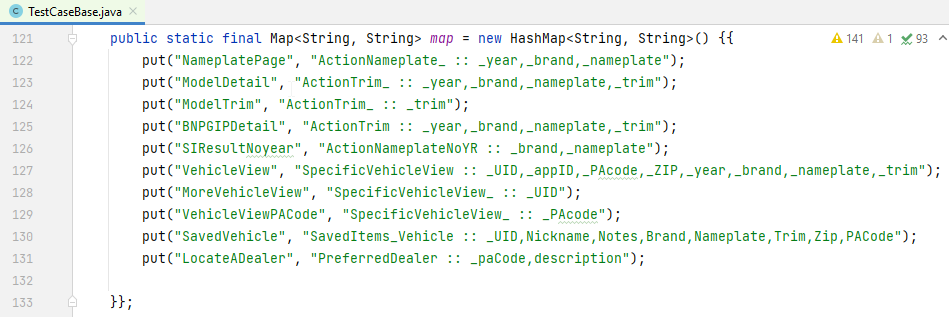
It involves below methods which plays very important role while filtering FPS set calls. All methods are defined in testcasebase class.

***I. addProxyListeners():*** This is a method which listens to a listener. It filters out the HTTP requests and its response as per the listener given, status code and status method(post/get). It also makes use of Browsermob Proxy tool to filter out responses. listeners are mentioned in Rejex pattern as below.



The code in this method gives us the set calls having status code, 200 OK and request method, POST.

**II**. ***fpsSetcall():*** This method is use to filter out the required parameters from captured set call. In fps, parameters to validate are change according to site activity we perform. This site activities are mentioned in FPS tagging plan document. In FPS framework, Parameters are stored as values of keys in hashmap named as “map”.



This method uses GSON to parse json data, which is nothing but request payload. The coding logic to extract parameters is vary as per the requirement. You can refer this method through code directory to get more about it.

Note: There are few more methods defined in testcasebase class to parse parameters of set calls as per the project need only. [Ex: fpsSaveSetCall. fpsEcomSetCall etc]

# **Diagrammatical representation of the FPS Automation Flow**

**Properties file in TestData folder**

**Ford/Lincoln/Fleet TestBase**

**Generates ExtentReport**

**Test Class**

**PageElement Class**

**TestCaseBase Class**

**Application Base Class**

Flow

Extends

Where:

**Excel Data in Requirement** folder