Automation Reference Document

Department: Application Development

Product/Process: Test Automation framework development and Implementation

Prepared By:

Document Owner(s)	Project/Organization Role
Sanjay Kumar, Bangalore	Test Automation framework

Version Control:

Version	Date	Author	Change Description
1.0	1/2/2017	Sanjay Kumar	Initial Draft

Table of Contents

Project Description

Testing occupies a significant portion of the software development lifecycle in terms of both cost and time. Every feature has to be tested, not only when it is introduced for the first time through a release, but every time a subsequent release carries it with or without enhancements to it. Consequently, regression test cycles are long, and tests within the cycles repeat every month or quarter depending on the release frequency, and often require modification. Manual testing in such a scenario becomes prohibitively expensive, and software vendors are increasingly looking at test automation to reduce the recurring cost and time.

Project Purpose

The Automation Framework is a set of guidelines like coding standards, test-data handling, object repository treatment etc., which when followed during automation scripting produce beneficial outcomes like increase code re-usability, higher portability, reduced script maintenance cost etc. Importantly these are just guidelines and not rules; they are not mandatory and you can still script without following the guidelines. But we will miss out on the advantages of having a Framework.

Business Case

Automation framework to extend the help for testers by writing scripts less code and to reduce boilerplate code while writing for test suites.

Assumptions

Already Installed/Included the following packages:

- Nunit 3.0 Adapter
- Nunit framework
- Selenium 2.53
- Selenium Support
- IE, Firefox, Chrome exe file will be available in DriverServices folder.

Constraints

- Support only for Web application.
- Testing in Distributed Environment has been considered for future enhancement.
- Parallel browser testing has been considered for future enhancement.
- Reports has been implemented using open source Extent reports. Customizing is not possible in reports. We have to customize reports in razor views based on requirement.
- Test Data Driven has been implemented based on Excel test construction. It is tightly coupled with the code behind and any changes in excel will lead to code break in automation framework.

High level requirements

Key Features

- Robust, flexible and extensible framework and support test automation on diverse sets of web applications across domains
- Enables users to perform functional, acceptance and compatibility testing for most web applications.
- Faster execution with Web driver
- Automated HTML report generation and emailing of the same to all stake holders.
- Detailed test execution results with consolidated summary and error snapshots
- Manages multiple execution VM environments to run the tests against a vast combination of browsers.
- Support for sequential and concurrent execution on various browsers
- Supports UI automation with Open Source tools like Sikuli or Test Link.
- Reduced time-to-market

Key benefits

- Built on open source tools / libraries / frameworks to reduce overall costs for customers
- Reduces test automation development phase by over 50% reduction
- High productivity and Low maintenance cost
- Reduces dependency on technically skilled resources
- Avoid redundancy on test execution
- Increases test coverage to enhance the quality and reliability of the end product
- Enables quick updates and shorter learning curve due Selenium user community and other open source communities.

Technology details

- Open source tools / frameworks / add-ons / and utilities which include: Selenium Core, Web driver, C#, Nunit,
- Browser support includes: Firefox, Internet Explorer, Edge, Safari and Google Chrome
- OS support includes: Windows

Fully automatic test execution is of course the number one requirement for test automation framework. Just executing tests is not enough, it should be robust, flexible and support test automation on diverse sets of web application.
the framework must also be capable to log the execution and analyze test outcome, handle errors and report results.

Framework must be easy to use by test engineers or it is very likely to be abandoned. Framework users must be able to design and edit tests, run them and monitor their status easily without any programming skills. It must be easy and fast to maintain both test data and framework code when the tested system changes or updates are needed otherwise. It should also be easy to add new features to the framework.

Automation Framework

Automation Framework increases automation efficiency by minimizing initial coding effort. It is a script-less Framework used for test automation of web applications that are developed on .Net, Selenium and Nunit. The framework provides a platform to implement data driven and Hybrid – keyword + data driven – framework by spreadsheet template. It can be use in any web application project for automation.

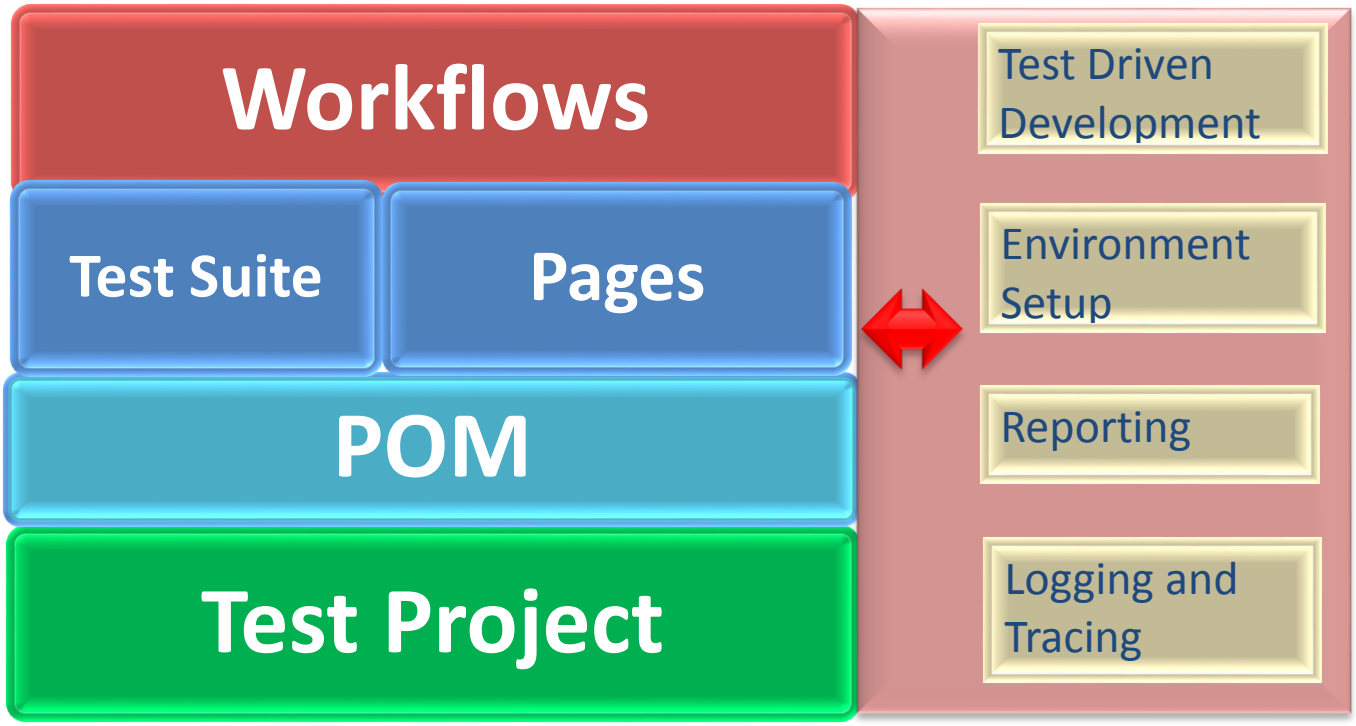
Automation Framework helps enterprises speed up testing using various extension methods and helper classes. The best way to use the framework is to design your test project using Page object model (POM).

The Automation framework provides a comprehensive reporting dashboard for managing tests. We will be using the open source test manager tools like Test Link. However, the framework also supports TDD where we can directly update the status and log reports in test documents and hence we don't require to use any test manager tools.

Framework meets almost all the requirement mentioned above as high level requirement.

- The framework must be able to start execution at a certain time or a new event (e.g. new version of SUT available). It must be possible to start test execution manually with help of command line or Maintain environment setup.
- Framework must be able to start executing tests with a push button and run test cases on its own.
- Framework should handle the errors caused by the system or the test environment not working as expected.
- Verifying the test results as expected outcome.
- Updating the status with fail or pass or other statuses. Besides the status every test case should also get a short but descriptive status message specially for failed test cases.
- To be able to differentiate expected failures from new ones the framework must know what is the expected outcome when test fails.
- Logging should be in detailed. The framework must log what is doing internally to make it easier to debug problems in the framework itself.
- Test reports provide statistical information about the quality of the tested system and they are important for everyone involved in the project including tester, developer and manager etc.

Test Project Design



Test Project

The testing framework should offer point-and-click interface for accessing and interacting with the application components under test—as opposed to presenting line after line of scripting (Script-less automated tests). Testers should be able to visualize each step of the business scenario, view and edit test cases intuitively. This will shorten the learning curve for testers and help QA teams meet deadlines.

The test project contains web application regression test cases with functional test cases and cosmetic test cases. It will be written using page object model (POM). The test project will be written codes related to the project specific test cases. Any common functionalities that could use in other project as well then it will be highly recommended to move in framework.

Page object Model

Page Object Model is an Object repository design pattern in Selenium WebDriver. Creating Selenium test cases can result in an unmaintainable project. One of the reasons is that too many duplicated code is used. Duplicated code could be caused by duplicated functionality and this will result in duplicated usage of locators. The disadvantage of duplicated code is that the project is less maintainable. If some locator will change, you have to walk through the whole test code to adjust locators where necessary. By using the page object model, we can make non-brittle test code and reduce or eliminate duplicate test code. Beside of that it improves the readability and allows us to create interactive documentation.

Last but not least, we can create tests with less keystroke. An implementation of the page object model can be achieved by separating the abstraction of the test object and the test scripts.

To Achieve Page object model in Test Project

Step 1: Create a test project by selecting a library from visual Studio.

Step 2: Create 2 .CS file name as LoginTest and LoginPage under files TestSuites and Pages respectively.

Step 3: Write Selenium related code to interact with Web element in Page i.e. LoginPage.

Step 4: Write Real test suite in Test file i.e. in LoginTest with the help of calling the methods from Pages i.e. LoginPage.

Step 5: Instantiate your driver with the help of Automation framework. We can achieve the same by deriving each test Class from Automation Base Class Hooks and Close your browsers by tear down attribute and calling the function TearDown() from framework. Hope you will be up and running with automation test suite at this steps.

To write Web Element Interaction in pages

```
public class LoginPage
{
    //Create Properties to interact with web elements/Controls by Selectors.
    [FindsBy(How = How.Name, Using = "ctl00$ContentPlaceHolder1$txtLogin")]
    public IWebElement UserName { get; set; }

    [FindsBy(How = How.Name, Using = "ctl00$ContentPlaceHolder1$txtPassword")]
    public IWebElement Password { get; set; }

    [FindsBy(How = How.Name, Using = "ctl00$ContentPlaceHolder1$AsiButton1")]
    public IWebElement LoginBtn { get; set; }
    /// <summary>
    /// Workflows Method to interact with User name, password and button control.
    /// </summary>
    public string LoginToApplication(string[] data)
    {
        string email = string.Empty;
        email = UserName.EnterText(data[0]);
        Password.EnterText(data[1]);
        LoginBtn.Click();
        Console.WriteLine("Login To Application Passed");
        return email;
    }

}
```

So, the above code has two sections. The first one is to create the properties for web element

interaction and the second one is to create the method to interact with the sets of element to achieve set of flow that can be reuse in Any test suite latter.

We have created 3 properties to achieve the workflow of entering the user name, password and clicking the login button. All three properties have been used in LoginToAplication () methods and this method is going to use in Test Suite i.e. LoginTest.cs. LoginToAplication () method can be reuse in many test Suites where ever we required the same workflows. The best part of creating properties is selector Id's which has been written one place and if any changes in Id's we can change there and it will reflect to all test.

To write Real TestSuites in Test

The below code with class name LoginTest is derived from Hooks base class. The Hooks base class has common methods and properties to Create driver instances using Driver, to Close all driver on tear down by TearDown () and to navigate from one url to another by GoTo () method.

The constructor with constraint helps us to invoke which browser to use at the time of testing. it will be taken from environment setup excel file in future. As of now we are using separate resx. File. Also, we are loading the User Credential Data from separate excel. It will be move to Environment setup excel file in future.

We have a page generator which will invoke your page classes and Elements using page factory provided by selenium. So whenever anyone create a new page then it should register on page generator class to invoke the class and create new instances as well invoke all elements in that page class. We call the page generator in constructor to load the web elements.

The real test suits start with attribute name Nunit. The test suits take test data input using testcasesource attribute and execute test cases iteratively for a collection of test data. It has been written in Testdatafactory class which is a part of framework.

Once the data has feed to web elements controls i.e. the LoginToApplication () method through the test data driven then we do assert/verify the expected output with the Actual output and if both match then test gets pass in test explorer or else fails.

Steps to Create Test Suits

- ✓ First of all, it is required to identify tasks that an application has to accomplish.
- ✓ Second, a set of necessary input data has to be created.
- ✓ Third, expected results have to be defined in order one can judge that an application (a requested feature) works correspondingly.
- ✓ Fourth, Executes a test.
- ✓ Finally, compares expected results with actual results, and decides whether the test has been passed successfully.

Find the below code snippet for the same.

```
    /// <summary>
    /// Login Test Suites which inherit Hooks base class which have driver to
    ///instantiate, Teardown, Goto any valid url.
    /// </summary>
    [TestFixture]
    public class LoginTest : Hooks
    {
        //To do list: Code change for browsertype when the test excute in parllel browser
        //default constructor which will invoke browser type and load data for
        //test data driven.
        public LoginTest()
            : base(AutomationSetting.BrowserToRunWith)
        {
            ExcelHelper.PopulateInCollection(Directory.GetCurrentDirectory() +
FilePath.DataDrivenExcelFileLocation);
            GoTo("https://authasidevtm1b.answerssystems.com/");
            page = PageGenerator.ApplicationLoginPage;

        }
        /// <summary>
        /// Login to Application and Test whether the login is succeful for the
        ///positive test data and unsucceful for negative test data.
        /// </summary>
        [Test]
        public void TC02_LoginToApplication()
        {
            //Test Data driven with CutomTestCaseSource Attribute. it brings collection
            //of data from test constructions excel which may be negative or positive.
            List<string[]> tests = new List<string[]>(new
CustomTestCaseSource(typeof(TestDataFactory)).BuildFrom("TC02_LoginToApplication", Path));
            foreach (string[] data in tests)
            {
                if (data.Length != 0)
                {
                    page.LoginToApplication(data);
                }
            }
            Assert.AreEqual("http://fsasidevtm1b.answerssystems.com/Home/",
Hooks.Driver.Url, "login Failed ! Please check Username and Password");
        }
```

So We have covered to write test suites using POM and with framework supports. The real test case could have end to end flow and might be chances to navigate to another page as well where the required controls is not available or your driver might open a new window tab to search the elements new tab and jump back to parent tab. Executing test cases successfully is based on how you call your web controls i.e. How you decorate your page control methods to your test suits. How best you reuse the web element pages method (e.g. LoginToApplicationPage ()) to your test suites.

Workflows is something related to set of web element interaction which gives meaningful test suite story to use directly with test suites. It is a combination of elements which interacts for the purpose of executing a test suite flow. A test suite can have many flow. As example: GoTo () methods from Hooks class directly navigate to another screen, or LoginToApplication () method have set of elements interaction (e.g. fill username and password and click login button) which directly use in a test suits.

The test project has several other responsibilities like logging, tracing, reports generations, Excel metadata to execute the functional test cases, filling test data with test data driven and asserting/verifying the results for the successful test execution. The test project has these features and it can achieve with the help of automation framework.

Concise Reporting

Test report/results is a document which contains summary of test activities. The framework must automatically generate reports of the test run and show the results in an easy-to-read format. The reports should provide specifics about where application failures occurred and what test data was used. Reports must present application screen shots for every step to highlight any discrepancies and provide detailed explanations of each checkpoint pass and failure. Reports must also be easily shared across the entire QA and development teams.

Test Data driven

TestDataSource attribute to read test data from external data sources and executes test based on it. Data sets increases test coverage by performing testing with various inputs and reduce the number of overall test scripts needed to implement all the test cases.

Logging and Tracing

Log4Net has been implemented and it will be use in generating reports. For each check points it should log and give detailed to reports file. Tracing also could be achieved with slight modification in Logger class.

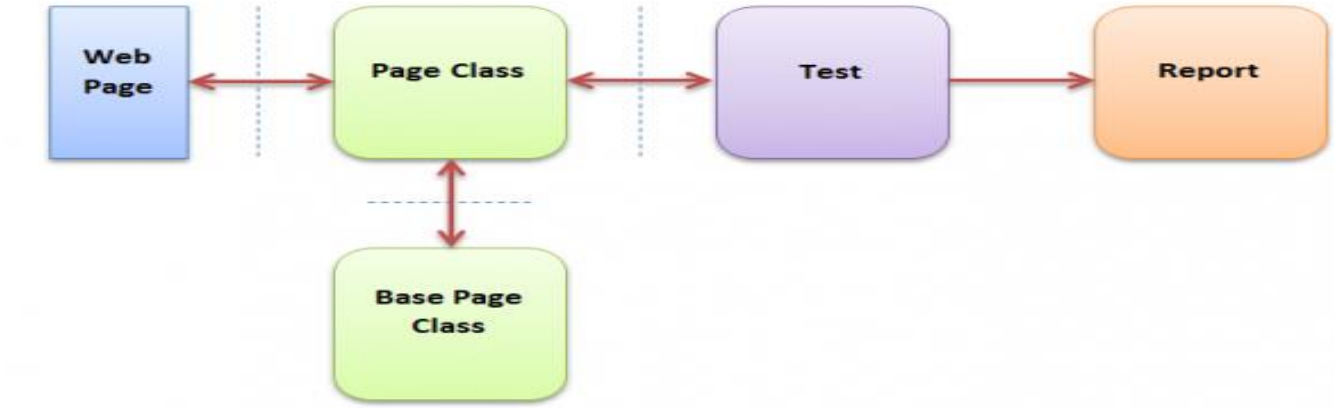
Environment setup

As we discussed in high level requirements that the test suits will run based on Environment setup that may include, User Specific, Server specific, Browser Specific etc. It has not been implemented fully and need to simulate these functionalities in one Environment set up file.

We will more discuss the above features in framework design and implementation section. The most important one is Environment setup.

At this stage, looking at pages and test suits methods, you must have guess the importance of framework.

Test Project Design Summary



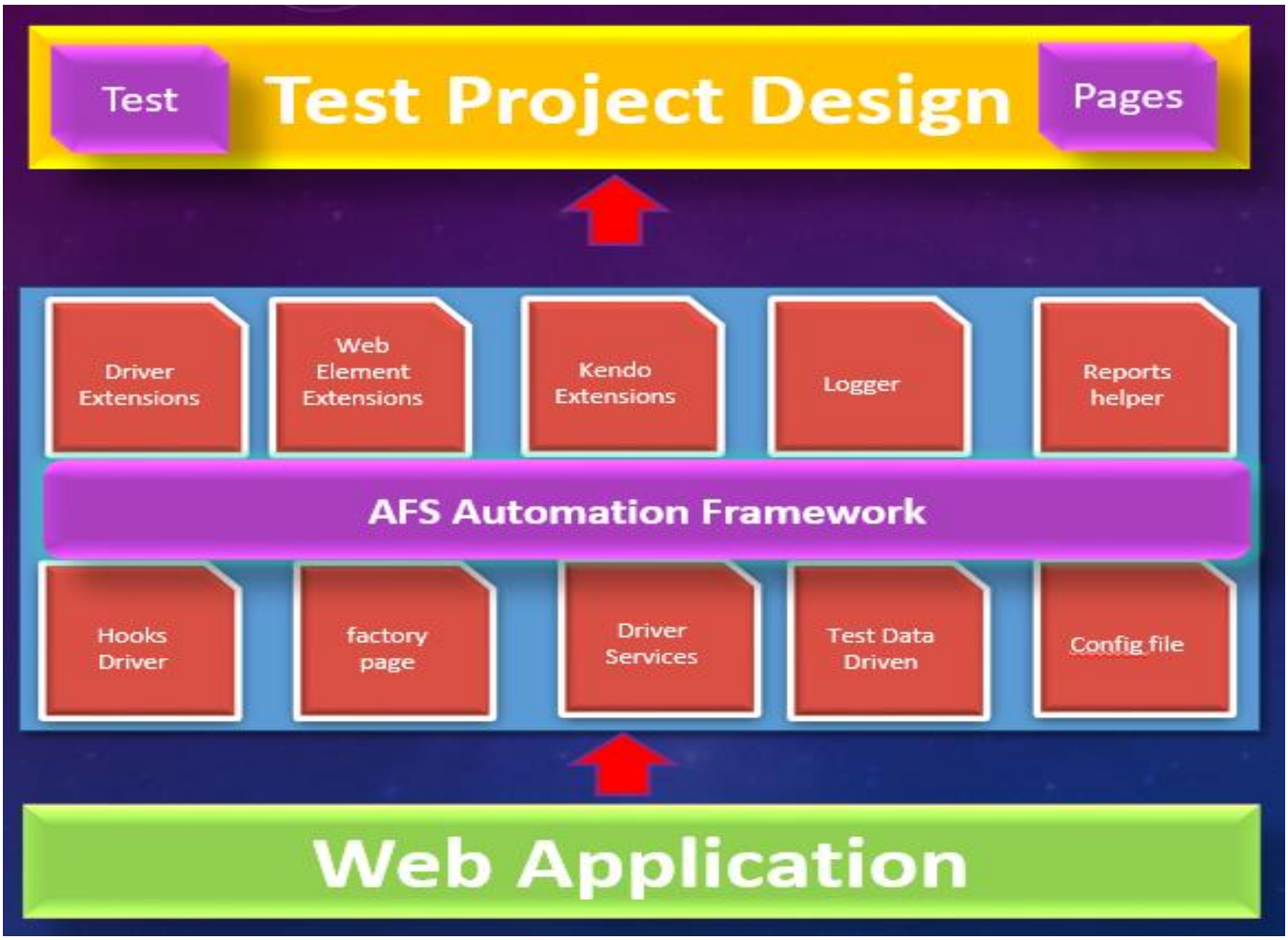
Web Page controls directly communicate with Page class which contains selenium web driver and web elements to interacts with Html controls. Page class will be invoked through page generator base class which internally calling from framework base page and using selenium factory design pattern to invoke pages. The methods from page class will be called directly to test class to frame a test suite. One test suite may have set of methods from page class and use those page method to another test suite to achieve reusability's. The test suite can utilize the framework features such as logger class to log exception, Reports Html view from ExtentManager class and TDD through TestDataFactory class. These features have been implemented in TestSuite i.e. Test Class.

Benefits of using framework while considering the LoginTest Example

1) Reduction of tests time execution and human resources required
2) Reusability of code, Maximum coverage, Recovery scenario, Low cost maintenance, Minimal manual intervention, Easy Reporting.
3) Complete control over the tests' results ("actual results" vs "expected results")
4) Possibility to quickly change test's preconditions and input data, and re-run the tests dynamically with multiple sets of data
5) Selenium provides SendKeys () which has been extended to EnterText () in framework. SendKeys does not return any value neither clean the textbox control before filling the value. We have case where we need to clear the text before entering value and return the value as entered through text box. So the extension method EnterText () has been made for this purpose. So, there are many controls which has been extended to serve our purpose and it has been there in framework to use based on demands. We will be discussing more about those controls in Automation technical design section.
6) Hooks class has implemented to create a single instance driver to use throughout the test until driver gets tear down. Only the Test project has to inherit from Hooks class which
7) Excel helper class will bring the LoginToApplication () Credential to login to application. Test data driven has been implemented by framework. So one test case can hold the positive and negative scenario at the same time.
8) Hooks class have GoTo () method that will be used to navigate to other url seamlessly. Also, it has TearDown () methods which will close all browser.
9) Many more to list out, you will find some more benefits while writing test suits.

Automation Framework

TECHNICAL DESIGN & IMPLEMENTATION



The process of creating Automation framework requires detailed planning and extreme efforts. To achieve desirable features, one should design framework accurately. Such type of framework can then be used for any kind of application with minimum changes in any type of organization who requires

automation testing. In this Document, we have presented idea of hybrid framework and its components. By using this Framework, instead of writing multiple functions separately, we have abstracted those things to excel sheet by giving keywords and test data as input which will help to test the entire functionality of Application under Test. This type of framework can be used widely in any type of web application for automation testing.

As above framework diagram, framework consist several reusable components to serve the purpose, hence it is called as Hybrid framework. Any Test Project will not directly interact with Web application. The framework will be a middle layer between test project and web application. So, the test project should get the extension methods or helper classes, kendo wrappers controls, or driver instance etc. from framework and then interact with web application.

Hooks Base Class

This below code snippet for creating a single instance Driver based on browser types.

```
    /// <summary>
    /// framework gets life from here.
    /// The Driver instance will be used in any test project as a single instance
    /// until it gets close their driver itself.
    /// </summary>
    public class Hooks : ExtentManager
    {
        private BrowserType _browserType;
        private static readonly IDictionary<string, IWebDriver> Drivers = new Dictionary<string,
IWebDriver>();
        private static IWebDriver driver;

        /// <summary>
        /// Driver instance to be used throught out the test project once
        /// instantiated and untill it disposed it self.
        /// </summary>
        public static IWebDriver Driver
        {
            get
            {
                if (driver == null)
                    throw new NullReferenceException("The WebDriver browser instance was not
initialized. You should first call the method InitBrowser.");
                //**give this code once environment set up made for central wait***
                //driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(20));
                return driver;
            }
            private set
            {
                driver = value;
            }
        }

       }
```

The Driver will be invoked into constructor and initiate the invoked browser and the Driver properties will be set to the correct browser instance.

```
        /// <summary>
        /// first constructor to be invoked and the tester choice browser will be initiated.
        /// </summary>
        /// <param name="browser">browser name</param>
        public Hooks(String browser)
        {
            ChooseDriverInstance(_browserType);
        }
        /// <summary>
        /// based on browser, create the instance of browser which is an executable file.
        /// </summary>
        /// <param name="browserType"></param>
        private void ChooseDriverInstance(BrowserType browserType)
        {
            switch (browserType)
            {
                case BrowserType.Firefox:
                    if (driver == null)
                    {
                        driver = new FirefoxDriver();
                        Driver = driver;
                        driver.Manage().Window.Maximize();
                        Drivers.Add("Firefox", Driver);
                    }
                    break;

                case BrowserType.IE:
                    if (driver == null)
                    {
                        driver = new InternetExplorerDriver(Directory.GetCurrentDirectory() +
FrameworkSettings.DriverServicesPath);
                        Driver = driver;
                        driver.Manage().Window.Maximize();
                        Drivers.Add("IE", Driver);
                    }
                    break;

                case BrowserType.Chrome:
                    if (driver == null)
                    {
                        driver = new ChromeDriver(Directory.GetCurrentDirectory() +
FrameworkSettings.DriverServicesPath);
                        Driver = driver;
                        driver.Manage().Window.Maximize();
                        Drivers.Add("Chrome", Driver);
                    }
                    break;
            }
        }
```

The Hooks class also contains TearDown () and GoTo () methods. These methods will be responsible to close browser and Navigate to any valid url respectively. The below code snippets has been written for the same.

```
        public static void GoTo(string url)
        {
            foreach (var key in Drivers.Keys)
            {
                Drivers[key].Url = url;
            }
        }

        public void TearDown()
        {
            CloseAllDrivers();
        }

        private static void CloseAllDrivers()
        {
            foreach (var key in Drivers.Keys)
            {
                Drivers[key].Close();
                Drivers[key].Quit();
            }
        }
```

Hooks class can be extended with common driver related functionalities e.g. driver to open the new browser tab, perform some action and return back to parent tab. So, such functionalities can be written in Hooks class.

Factory page

To invoked all page class and related web elements.

```
/// <summary>
/// To intitantiate any class Instances using factory pattern.
/// return GetPage<YourPageToIntanciate>()>
/// </summary>
 public abstract class BasePage
 {
     //Page generator using genrics and constraint to avoid constructor invocation
     //every time to create instance for each page class.
     public static T GetPage<T>() where T : new()
     {
         var page = new T();
         PageFactory.InitElements(Hooks.Driver, page);
         return page;
     }
 }
```

To use the above Base page factory pattern in any test project: Create a Page Generator class in test project, inherit from base page and Invoked all class pages as below code snippet.

```
/// <summary>
/// PageGenerator class will be there in test project. It inherits from BasPage.
/// GetPage genric method to create instance for
/// each page class with the help of BasePage class which comes from framework.
/// </summary>
public abstract class PageGenerator :BasePage
{
    public static LoginPage ApplicationLoginPage
    {
        get { return GetPage<LoginPage>(); }
    }

}
```

Web Driver Extension Methods

We have listed only few extension methods here.

ScriptExecute (): Selenium have ExecuteScript method with **IJavaScriptExecutor**. This extension method will use to execute any JavaScript valid code but the extension has been made for void return type Where as **ScriptQuery ()** has been made to accept a generic data type that has return value.

```
        public static void ScriptExecute(this IWebDriver driver, string script,
                                                                    params object[] args)
        {
            ((IJavaScriptExecutor)driver).ExecuteScript(script, args);
        }

        public static T ScriptQuery<T>(this IWebDriver driver, string script,
                                                                        params object[] args)
        {
            return (T)((IJavaScriptExecutor)driver).ExecuteScript(script, args);
        }
```

Example: ScriptQuery ()

```
        //To return boolean value for model popup has been closed or not on screen.
        public static bool IsModelPopUpClosed()
        {
            var jsTobeExecuted = Hooks.Driver.ScriptQuery<bool>(string.Format("return
                                 $('#window').parent().is(':visible');"));
            return jsTobeExecuted;
        }
```

Driver Wait

There are different sets of extension methods which may require based on demand. Driver extension methods contains implicit wait, Explicit wait, JQuery Wait, Ajax Wait etc. We achieved driver wait by WebDriverWait (). Similarly we have achieved element wait in Element Extension methods.

```
        // Wait for a specific element to wait untill its gets loaded.
        public static void WaitForElementToLoad(this IWebDriver driver, string attribute,
                                    int? timeoutInSeconds = null)
        {
            TimeSpan timeout = new TimeSpan(0, 0, 100);

            if (timeoutInSeconds.HasValue)
            {
                int timeoutInSecs = timeoutInSeconds.Value;
                timeout = new TimeSpan(0, 0, timeoutInSecs);
            }
            WebDriverWait wait = new WebDriverWait(driver, timeout);
            wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.Id(attribute)));
        }
        //if there is a Juqey load delay such as Dropdown list load.
        public static void WaitForJQuery(this IWebDriver driver, int timeoutInSeconds = 60)
        {
            var js = driver as IJavaScriptExecutor;
            var wait = new WebDriverWait(driver, TimeSpan.FromSeconds(timeoutInSeconds));
            if (js != null)
            {
                try
                {
                    wait.Until(d => (bool)js.ExecuteScript("return jQuery.active == 0"));
                }
            }
        }

       //if there is a need of implicitly wait for element to load for a fix time span.
        public static void ImplicitWait(this IWebDriver driver, int waitSeconds)
        {
            driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(waitSeconds));
        }
        // Wait for Ajax load to complete.
        public static void WaitForAjax(this IWebDriver webDriver, double timeout)
        {
          new WebDriverWait(webDriver, TimeSpan.FromSeconds(timeout)).Until(
             driver =>
             {      var javaScriptExecutor = driver as IJavaScriptExecutor;
                    return javaScriptExecutor != null
                    && (bool)javaScriptExecutor.ExecuteScript("return jQuery.active == 0");
             });
        }
```

Action Driver

It is use to handle mouse events such as Move the mouse hover certain elements, Drag and drop etc. All these actions cannot be directly done using WebDriver reference variable. So in order to perform the mouse auctioning we need to make use of Action class.

Note: Any method that are used from actions class, it has to be followed by perform method i.e.; perform the previous action.

```
public static void ActionDrivers(this IWebDriver driver, string attribute)
{
    IWebElement element = driver.FindElement(By.Id(attribute));
    Actions actions = new Actions(driver);
    actions.MoveToElement(element).Click().Perform();
}
```

Switch to New Window Tab

it will allow to switch between windows tabs.

```
public static void SwitchToWindow(this IWebDriver driver, Expression<Func<IWebDriver,
                                    bool>> predicateExp)
{
    var predicate = predicateExp.Compile();
    foreach (var handle in driver.WindowHandles)
    {
        driver.SwitchTo().Window(handle);
        if (predicate(driver))
        {
            return;
        }
    }

    throw new ArgumentException(string.Format("Unable to find window with condition:
                                            '{0}'", predicateExp.Body));
}
```

Page Titles of Window Tab

Determines whether [is page title] equals [the specified page title].

```
        public static bool IsPageTitle(this IWebDriver webDriver, string pageTitle,
                                       double timeout)
        {
            var wait = new WebDriverWait(webDriver, TimeSpan.FromSeconds(timeout));
            try
            {
                wait.Until(d => d.Title.ToLower(CultureInfo.CurrentCulture) ==
                                         pageTitle.ToLower(CultureInfo.CurrentCulture));
            }
            catch (WebDriverTimeoutException){ return false;}
            return true;
        }
```

IsElementPresent

Method to return bool for finding specific element by locator is available or not.

```
       public static bool IsElementPresent(this IWebDriver webDriver, ElementLocator
                                       locator, double customTimeout)
        {
            try
            {
                webDriver.GetElement(locator, customTimeout, e => e.Displayed);
                return true;
            }
            catch (NoSuchElementException)
            {
                return false;
            }
            catch (WebDriverTimeoutException)
            {
                return false;
            }
        }
```

JavaScript Alert

We can customize java script alert with below code snippet.

```
public static JavaScriptAlert JavaScriptAlert(this IWebDriver webDriver)
{
    return new JavaScriptAlert(webDriver);
}
```

Web Element Extensions: We have listed few extension methods

EnterText: SendKeys will not return the text value that you enter in text controls. Also, it will clear the text box before entering the text value. Hence, to achieve these 2 feature and simulate in 1 method we can use EnterText ().

```
public static string EnterText(this IWebElement element, string value)
{
    //In Selenium 2, text fields are not cleared unless explicitly cleared.
    if (!String.IsNullOrEmpty(element.GetText()))
    {
        element.Clear();
    }
    element.SendKeys(value);
    return element.GetText();
}
```

IsCheckboxOrRadiobtnSelectedByDefault:

This extension method is common for both check box or radio button. It is use to check whether the check box or radio button has been selected value/position by default on page load. The index position starts from 0. It will return Boolean value as true if selected position will match with default position in UI. The parameter default Position means, if there are three radio button in a group such as Male, Female, and Both. Also, Default selection should be Male on page load then your defaultPostion is 0 in current scenario and hence the parameter value of defaultPosition should pass as 0.

```
public static bool IsCheckboxOrRadiobtnSelectedByDefault(this IList<IWebElement>
                                                    elementList, int defaultPosition)
        {
            // Create a boolean variable which will hold the value (True/False)
            bool bValue = false;
            // This statement will return True, in case of the default Radio/Checkbox
            //button is selected
            bValue = elementList.ElementAt(defaultPosition).Selected;
            return bValue;
        }
```

GetTextFromDDL

Get the selected text value from Drop down list.

```
        public static string GetTextFromDDL(this IWebElement element)
        {
            return new SelectElement(element).AllSelectedOptions.SingleOrDefault().Text;
        }
```

SelectCheckboxOrRadioBtn

Select the radio button or checkbox in a group with postionTobeClicked. The index of positionTobeClicked starts from 0.

```
        public static void SelectCheckboxOrRadioBtn(this IList<IWebElement> elementList, int
                                                    postionTobeClicked)
        {
            // Create a boolean variable which will hold the value (True/False)
            bool bValue = false;

            // This statement will return True, in case of the postionTobeClicked Radio/Checkbox
           // button is already selected
            bValue = elementList.ElementAt(postionTobeClicked).Selected;

            // This will check that if the bValue is True means if the postionTobeClicked radio
            // button is selected
            if (bValue == true)
            {
                //don't select since it has been already selected.
            }
            else
            {
                // If the postionTobeClicked radio button is not selected by default, then
                //postionTobeClicked will be selected
                elementList.ElementAt(postionTobeClicked).Click();
            }
        }
```

JavaScriptClick

Click on element using java script. it will be hold good if your element is not visible at the time of element interactions.

```
        public static void JavaScriptClick(this IWebElement webElement)
        {
            IWrapsDriver wrappedElement = webElement as IWrapsDriver;
            if (wrappedElement == null)
                throw new ArgumentException("element", "Element must wrap a web driver");

            IWebDriver driver = wrappedElement.WrappedDriver;
            IJavaScriptExecutor javascript = driver as IJavaScriptExecutor;

            if (javascript == null)
            {
                throw new ArgumentException("Element must wrap a web driver that supports
                                            javascript execution");
            }

            javascript.ExecuteScript("arguments[0].click();", webElement);
        }
```

Note: Refers automation code base to know more about Web Element extensions methods.

Kendo Wrappers Class

Kendo wrappers has been implemented to reduce the tester works to find the parent-child selector Ids. We have written kendo wrappers with the help of Selenium IJavaScriptExecutor. It has ExcuteScript () method which execute any JavaScript code snippet.

Note: We have implemented kendo wrappers for the many controls. Few are listed below.

1) **KendoGrid**
2) **KendoDropDownList**
3) **KendoDatePicker**
4) **KendoTreeView**
5) **KendoMultiSelectDropDownList**
6) **KendoPanel**
7) **KendoSelect**
8) **KendoTab**
9) **KendoComboBox**
10) **GridFilter**

We will describe few wrappers class from above list.

KendoDropDownList

Kendo Widget is a base class for all kendo related controls.

```
public class KendoDropDownList : KendoWidget
{
  public KendoGrid(IWebElement gridName)
       : base(gridName)
   {
       this.gridId = gridName.GetAttribute("id");
       this.driver = (IJavaScriptExecutor)driver;
       this.kendoGrid= string.Format(CultureInfo.InvariantCulture, "var grid =
       $('#{0}').data('kendoGrid');", this.gridId);
   }
   /// Selects item from the Kendo Down List by text and triggers change event.
   public KendoDropDownList Select(string valueOfDataTextField)
   {
       return Select(GetDataTextFieldName(), valueOfDataTextField);
   }
   /// Get Data text field name.
   private string GetDataTextFieldName()
   {
       string dataRoleElement = ScriptQuery<string>(string.Format(" return
       $('#{0}').data('kendoDropDownList').options.dataTextField", this.ddlId));
       return dataRoleElement;
   }

 }


Ex: How to use the KendoDropdownList in Test project. We have to write always in pages
with help of framework.
   [FindsBy(How = How.Id, Using = "contractiddropdown")]
    public IWebElement Contractsddl { get; set; }

    public void SelectContractIdDDLSearch(string ContractID)
    {
        var kendoDDL = new KendoDropDownList(Contractsddl);
        kendoDDL.Select(ContractID);
    }
```

KendoGrid: We can do set of operation and few has been mentioned in code snippet.

```
    public KendoGrid(IWebElement gridName)
            : base(gridName)
        {
            IWebDriver driver = Hooks.Driver;
            this.gridId = gridName.GetAttribute("id");
            this.driver = (IJavaScriptExecutor)driver;
            this.kendoGridForHref = string.Format(CultureInfo.InvariantCulture,
"$('#{0}').data('kendoGrid')", this.gridId);
            this.kendoGrid= string.Format(CultureInfo.InvariantCulture, "var grid =
$('#{0}').data('kendoGrid');", this.gridId);
        }

        public void RemoveFilters()
        {
            string jsToBeExecuted = this.kendoGrid;
            jsToBeExecuted = string.Concat(jsToBeExecuted, "grid.dataSource.filter([]);");
            this.driver.ExecuteScript(jsToBeExecuted);
        }

        public int TotalNumberRows()
        {
            string jsToBeExecuted = this.kendoGrid;
            jsToBeExecuted = string.Concat(jsToBeExecuted, "grid.dataSource.total();");
            var jsResult = this.driver.ExecuteScript(jsToBeExecuted);
            return int.Parse(jsResult.ToString());
        }

        public void Reload()
        {
            string jsToBeExecuted = this.kendoGrid;
            jsToBeExecuted = string.Concat(jsToBeExecuted, "grid.dataSource.read();");
            this.driver.ExecuteScript(jsToBeExecuted);
        }

        public int GetPageSize()
        {
            string jsToBeExecuted = this.kendoGrid;
            jsToBeExecuted = string.Concat(jsToBeExecuted, "return
grid.dataSource.pageSize();");
            var currentResponse = this.driver.ExecuteScript(jsToBeExecuted);
            int pageSize = int.Parse(currentResponse.ToString());
            return pageSize;
        }

        public void ChangePageSize(int newSize)
        {
            string jsToBeExecuted = this.kendoGrid;
            jsToBeExecuted = string.Concat(jsToBeExecuted, "grid.dataSource.pageSize(", newSize,
");");
            this.driver.ExecuteScript(jsToBeExecuted);
        }
```

How to use the Kendo Grid wrappers in test Project?

Select with Gird Id/Name Selector and do all grid related operation using this selector. So you don't need to use different selector for different grid operation, instead you need only Grid Id as selector to do any sets of operation in grid.

```
        [FindsBy(How = How.Id, Using = "operatorsgrid")]
        public IWebElement OperatorsGridId { get; set; }
        public void ChangeGridOperatorPageSize()
        {
            var kendoGrid = new KendoGrid(OperatorsGridId);
            kendoGrid.ChangePageSize(10);
        }

        public void ReloadOperatorGridPage()
        {
            var kendoGrid = new KendoGrid(OperatorsGridId);
            kendoGrid.Reload();
        }
```

Similarly, we can use any Kendo wrappers class and get rid of using complex selector id's. Also, it encourages tester for reusability's of existing page properties and methods.

Test Data Driven

It will be prepared latter by Nitika who has worked on it.

Logging

we have used Log4Net helper class. It has been documented on the below url. However, currently we are directly logging (without log4Net) the exception in Excel file log column if in case thrown exception. The logging contains only exception message or statckTrace. However, if we need more information to log in excel then we should be using the logger helper class that has been implemented in Framework with the help from log4Net. In current requirement, we have to log the details into test construction file and specific to LogDetails column. It may be quite challenging to log into existing file with specific column name without overriding the other column in Excel. However, we have achieved the same without logger class.

http://logging.apache.org/log4net/

https://www.codeproject.com/articles/140911/log-net-tutorial

Html View Reports using Extent Reports Open source

It has been documented on the below links.

http://extentreports.relevantcodes.com/net/