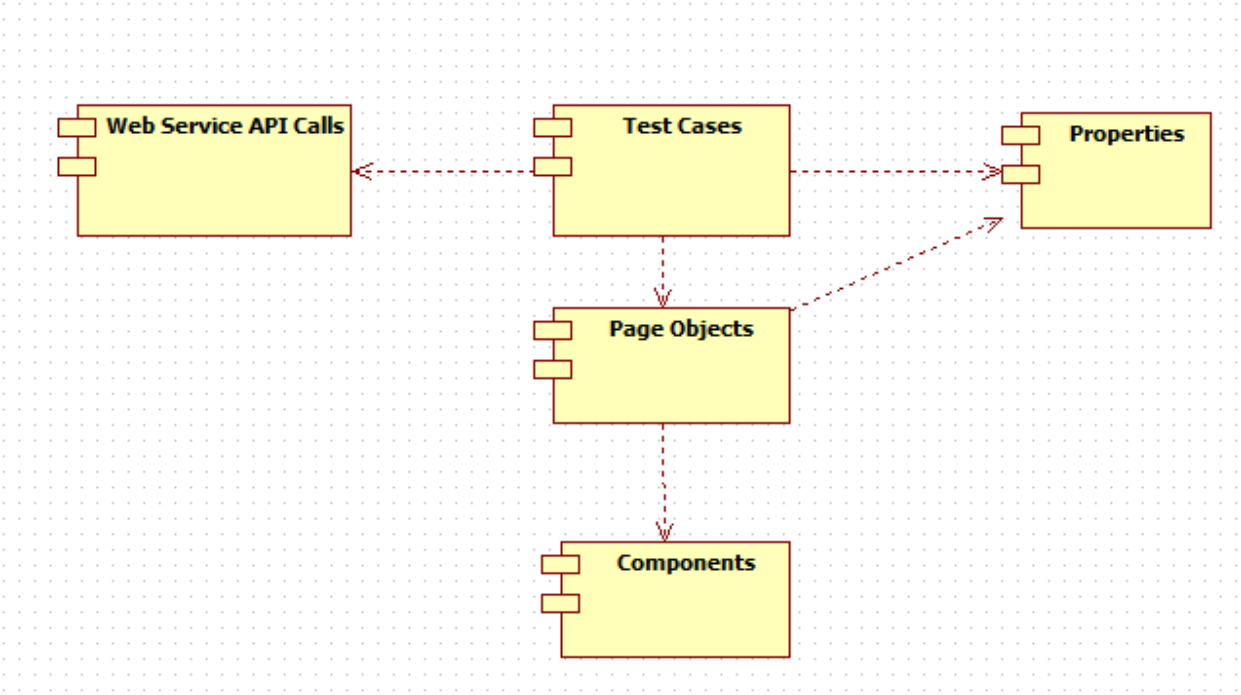
Medium Tests || How to add a new test case?

Overview of Framework



Goals

1. Test case should be data independent so use API call to create the dependent data before test case runs
2. Test case should be pojo type and easy to understand by any developer, introduce the PageObject and Component concept to hide all selenium complexity
    1. In the Test case we should set the pojo property of Page Object, which will synchronize with Selenium web element when sync method is called, this will allow us to re-use the pojo in multiple test case

API calls:

API calls are used to create the pre-requisite objects that are needed in console for test to run.

e.g. For "create an IO with mandatory fields" test case, an Advertiser is needed in console. These dependent test data would be created in testSetUp method of Test class.

Properties:

Test data used in test cases are kept in the properties files. i.e. To create an Advertiser, we need Advertiser name, Primary URL, etc.

Test Case:

Test method would represent the actual use case.

There are primarily three types of test methods:

- Workflow [Create/Edit/Delete object] - This kind of test case would check if create/edit happens correctly in console.
- Validation [Negative use case] - This kind of test case would check that invalid values are handled correctly by console.
- UI checks [UI verifications] - This kind of test cases would check specific fields/values are available in console.

Page Objects:

The Page Object pattern represents the screens of your web page as a series of objects. Each page on UI would be represented by Page Objects class. This class would contain web page elements and its interactions.

All page object class would be suffixed with PO text.

More detailed page: Page Objects

Components:

Any project has a lot of components which are re-used in entire application like SmartTables, AutoSearch. We have created classes which provides all the interactions with such components. i.e. Search in SmartTable.

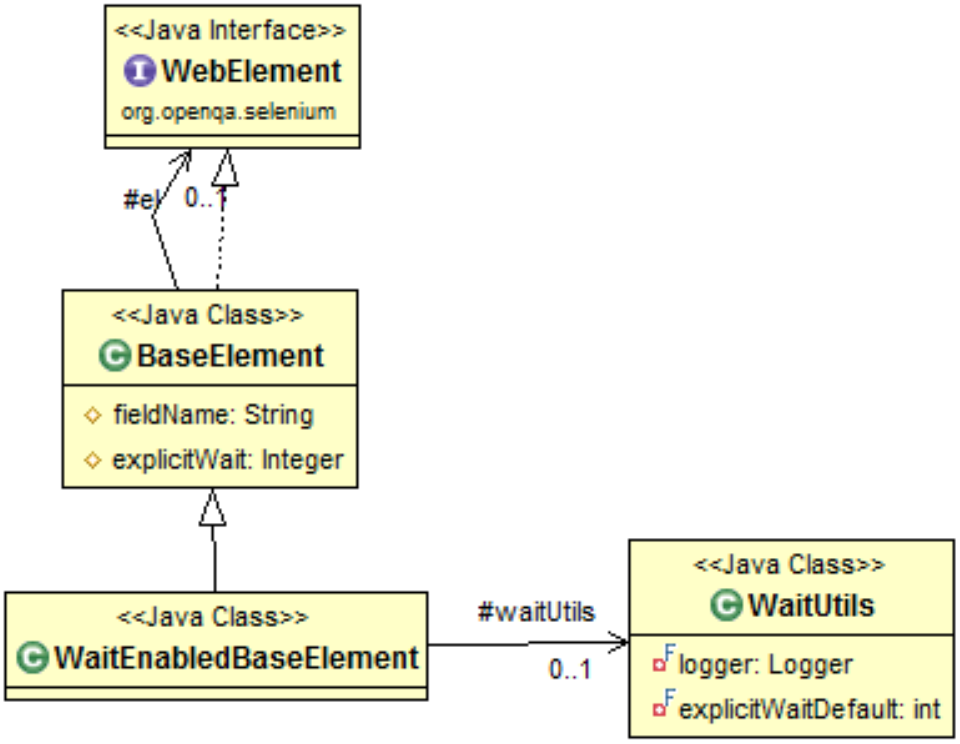
Handle Timeout

One of the standard problem with selenium framework is time out, framwork need to wait for element to be visible, so putting the timeout before all needed element is error pron process

This framework has introduce Two component to handle the timeout

1. WaitEnabledBaseElement - Creating the webElement of this Type will automatically attach the default timeout.
2. 'SeleniumTimeout Annation - If default timeout is not enough then we can override the default Timeout by SeleniumTimeoutAnnotation

WaitEnabledBaseElement



Goals

1. Do not put time out code on every element, it is error prone, what if someone accidentally remove the time out code while bug fix/maintainence activity
2. Create WebEnabledWebElement instead of WebElement for actionable element (like button)
3. With this approach you don't have to put Timeout before each actionable element

Create SeleniumTimeout

There are the case, where WebEnabledWebElement timeout is not enough we need explicit timeout, to handle this create SeleniumTimeout annotation which should be able to accept the timeout as an argument

```
{code}
@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

public @interface SeleniumTimeout {
int explicitWait() default 90;
}
{code}
```

Customize Page Factory

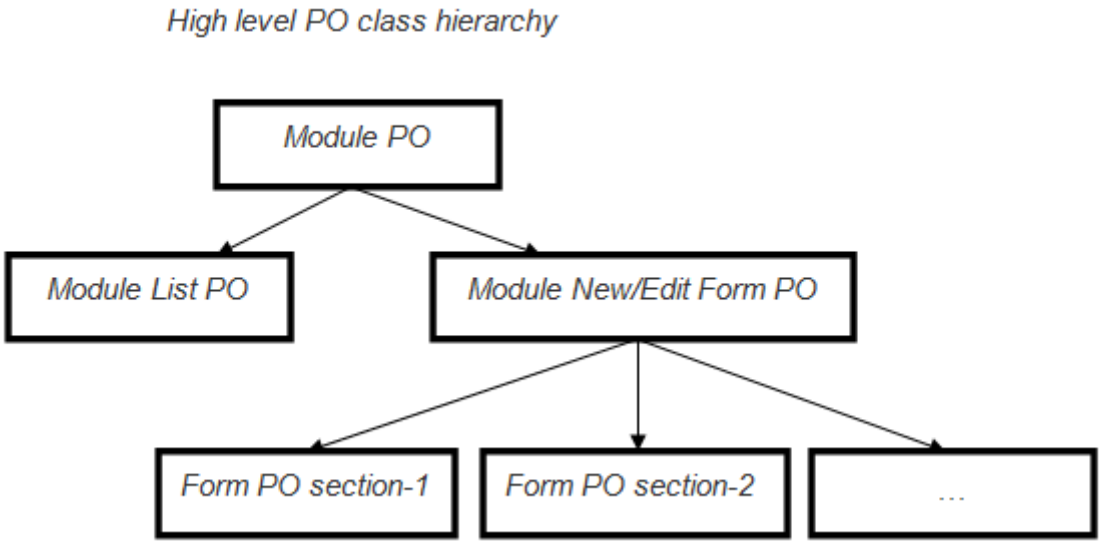
Customize the Page factory to instiate the wrapper WebEnabledWebElement and support SeleniumTimeout annotation

Refer Code :

```
	private static <T> T instantiateObject(Object el, Class<T> pageClassToProxy,
Field field) {

			final Constructor<T> constructor =
pageClassToProxy.getConstructor(WebElement.class);

			T obj = constructor.newInstance(el);

			((BaseElement)obj).setElementName(field.getName());

			if(field.isAnnotationPresent(SeleniumTimeout.class)){

				SeleniumTimeout annotation = (SeleniumTimeout)
field.getAnnotation(SeleniumTimeout.class);

				Integer wait = annotation.explicitWait();

				if(wait != null) {

					((BaseElement)obj).setExplicitWait(wait);

				}

			}
```

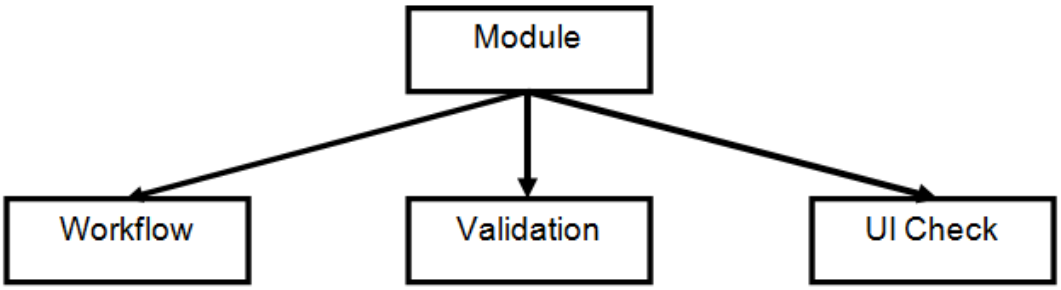
```
        return obj;


    }
```

Page Object



2.3 Step 3: Write New Test Method

High level hierarchy tree of how test methods are divided amongst test classes



In this case we want to save the Insertion Order having Frequency Cap value, So this test method falls under the **Workflow** category.