# Notes:

Contents

[Notes: 1](#_Toc527104197)

[Login Details: 1](#_Toc527104198)

[How Framework is Constructed 1](#_Toc527104199)

[SearchForSite 2](#_Toc527104200)

[Specflow tips 3](#_Toc527104201)

## Login Details:

Username = [cs@portalqa.com](mailto:cs@portalqa.com)

Password = Testing321

Prod:

Username = [alludin.sawar@gazprom-energy.com](mailto:alludin.sawar@gazprom-energy.com)

Password = Testing321

## How Framework is Constructed

In WebDriverHooks.cs:

Various things are setup, including :

private static readonly Context Ctx = ContextFactory.Instance;

This seems to take the place of the static WebDriver. The Driver seems to be contained within this. So it is accessed by Ctx.Driver

\_chromDriverConfiguration = new ChromeDriverConfiguration(Ctx);

This takes the Ctx that was created above. It will create the driver (driver is added to a list of drivers created earlier in Hooks. I expect multiple drivers can be created for parallel running?)

Hooks does not have a using for Portal.Web.AcceptanceTests namespace, so how can it see DriverConfiguration?

loginPage = new LoginPage(Ctx.Driver);

Creates a new loginPage I assume. But, but have not yet called chromDriverConfiguration = new ChromeDriverConfiguration(Ctx); so how do we have a driver instance to pass in?

I THINK ANY NEW METHODS I WANT TO ADD SHOULD BE PUT INTO THE BASEPOM CLASS. ALL PAGES SEEM TO EXTEND THIS SO ANYTHING I PUT HERE SHOULD BE AVAILABLE.

NEW VERSIONS OF WAITS CAN ALSO GO HERE.

### SearchForSite

protected void SearchForSite(string site, string selectedItem)

{

if (!SitesArePopulated()) return;

ClickSelection(selectedItem);

EnterSearch(site);

}

F12

SitesArePopulated()

F12

public virtual bool SitesArePopulated()

{

Func<IWebElement, bool> condition = element => element.Text == SiteSelectionText;

var webElement = WaitForWithCondition(SiteSelection, condition, 60);

return webElement != null && webElement.Displayed;

}

This defines a function called ‘condition’ that is used in the WaitFoCondition() method

The condition function will look for elements where elemen.Text == SiteSelectionText (the value of which was set earlier)

F12

protected IWebElement WaitForWithCondition(By by, Func<IWebElement, bool> condition, int seconds = 20, bool retry = true)

{

try

{

var wait = new WebDriverWait(Driver, new TimeSpan(0, 0, seconds));

Func<IWebDriver, IWebElement> waitForElement = x =>

{

var elements = WaitFor(by, seconds, false);

var element = elements.FirstOrDefault(e => e != null && **condition**(e) && e.Displayed);

return element;

};

return wait.Until(waitForElement);

}

Here you have a custom wait which looks for elements By the SiteSelection (Xpath) passed in. It then gets the first of the elements found and checks that element (e) is not null is displayed and passes it to the ‘condition’ function and checks that it returns True.

So basically it finds the element by SiteSelection (Xpath), checks actually found an element (not null), the element is displayed AND that its text equals SiteSelectionText.

So, although the element was found by SiteSelection (Xpath) we could see it, it returned element not found because it failed the ‘condition’ function check. This was because it was using the wrong SiteSelectionText to search on.

## Specflow tips

https://stackoverflow.com/questions/17167820/specflow-error-force-regenerate-steps-possible?\_sm\_au\_=iVVNrD7QFk3TRDQs