How Does the Object Repository Feature Works

The object repository mechanism is very similar mechanism to Selenium's PageFactory feature.

VeriSoft's framework uses the dynamic proxy mechanism in order to retrieve a fresh copy of the WebElement and to avoid StaleElementException.

Background

When a findElement method is invoked successfully, it means that an object on the DOM is linked to the WebElement. If by any chance this object on the DOM will change - deleted, refreshed, etc. our reference to the object will not be valid. Since we still hold the WebElement object, if we will try to execute any method when the mapping is no longer valid we will get a StaleElementExecption, which means something has happened to the object on the DOM.

Consider the following code:

```
1 WebElement element = driver.findElement(By.id("someId"));
2 driver.manage().refresh();
3 element.click();
```

In line #1 we link an element on the DOM to a WebElement object.

In line #2 we actually destroy all the objects in the DOM and create fresh new objects.

So, in line #3 our reference is no longer valid. We should have performed findElement again in order to maintain a fresh reference to the DOM element.

The issue is even more severe with the page object model.

When we create a new page object instance, a whole list of fields is initialized, and there is no way to assure all of them will still be linked to DOM objects when the time comes to use them.

This is why Selenium introduced the PageFactory mechanism. In order to assure that a WebElement will always be fresh when using it.

How Does a Page Factory Work?

When using a page factory, we annotate the WebElement object and specify the locator which a WebElement should use to locate the object on the DOM.

```
1 @FindBy(xpath = "//input[@name='search']")
2 private WebElement searchBar;
```

In order to populate that WebELement we call the following method in the constructor (for web-based pages):

```
1 PageFactory.initElements(driver, this);
```

`initElements` actually stores a dynamic proxy object in the WebElement. A dynamic proxy object is able to perform additional duties to the WebElement object before and after the execution of the method we set up to do. There is no need to actually call anything, the fact that the WebElement is wrapped with the dynamic proxy will cause an invocation of our choice every time the WebElement is called.

Selenium's dynamic proxy creates a fresh copy, using the `findElement` method with the given locator, therefore making sure the developer does not need to do any work making sure the object is valid.

How Does the ObjectRepositoryWork?

Similar to the PageFactory mechanism and the dynamic proxy mechanism, the object repository feature uses a similar concept. It holds in an annotation the object, and during the construction of the object, a dynamic proxy is inserted into the WebElement object. Every time the WebElement is called, it loops through the available locators from the object repository (sorted by their strengths from stronger locators to

weaker locators) JSON file. Once a match is achieved, the WebElement invocation (the method that was called upon the WebElement object) continues with the fresh copy of the WebElement.