09-03-2017

Handling DISABLED Elements:

If the element is disabled, then we cannot use following methods of WebElement interface:

1) clear();
2) click();
3) sendKeys();
4) submit();

All the remaining methods can be used. If we try to use above mentioned 4 methods, we get InvalidElement StateException.

* Different way of clicking on a button:

```
PSVM(String[] args) {
System.setProperty(" "," ");
WebDriver driver = new FirefoxDriver();
driver.get("gmail");
driver.findElement(By.id("next")).click();
driver.findElement(By.id("next")).sendKeys(Keys.ENTER);
driver.findElement(By.id("next")).submit();
}
```

We can use javascript to handle the element if it is disabled

Ex: Sample:

t1: <input type = "text" id = "t1" value = "admin">

t2: <input type = "text" id = "t2" value = "manager" disabled >

t1: admin

t2: manager

Executing javascript manually:

Press F12 → Click on Console → Type the java script → ENTER.

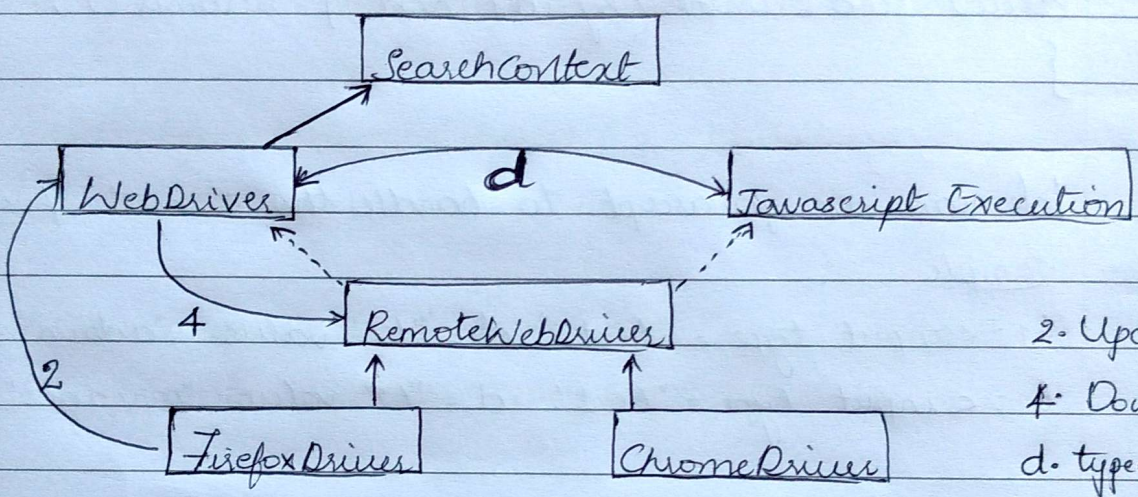
document.getElementById('t1').value = 'qsp'

Executing java script using Selenium (automatic):

To run the javascript in selenium, we use executeScript() method of JavaScriptExecutor class. Generally while writing any script in selenium, we upcast the browser object to WebDriver because of this reason, executeScript() method is hidden. In order to acces hidden methods, either we can use downlasting or typecasting.

Ex: Changing value of disabled text box.

1. System.setProperty(" ", " ");
2. WebDriver driver = new FirefoxDriver();
3. driver.get("file:///D:/page2.html");
4. RemoteWebDriver r = (RemoteWebDriver) driver;
5. r.executeScript("document.getElementById('t2').value='bhanu'");



2. Upcasting
4. Downcasting
d. typecasting

a. System.setProperty(" ", " ");
b. WebDriver driver = new FirefoxDriver();
c. driver.get("file:///D:/page2.html");
d. JavascriptExecutor j = (JavascriptExecutor) driver;
e. j.executeScript("document.getElementById('t2').value='bhanu'");

I.Q How do you enter the value if textbox is disabled?
I.Q How do you clear the value if textbox is disabled?
I.Q How do you click on button if it is disapled?
j.executeScript("document.getElementById('next').click()");
I.Q Write a script to scroll the webpage by using javascript.
for(int i=1; i<=10; i++){
j.executeScript("window.scrollBy(0,500)"); }

10/03/2017.

Encapsulation in Selenium: hiding internal state & requiring all interaction to be performed through an object method is known as encapsulation.

Here data refers to variables, object refers to non static methods

In encapsulation we bind data members (variables) with member functions (methods)

→ Customer.java

```
public class Customer {
    PSVM (String[] args){
    Account a1 = new Account();
    a1.setAmount(100);
    int v= a1.getAmount();
    S.O.P(v); }
}
```

→ Accounts.java

```
public class Account {
        private int amount=0;
    public void setAmount(int newAmt) {
        amount= newAmt ;
    }
    public ~~void~~ int getAmount() {
        return amount;
    }
}
```

In most of the cases, we initialize the variable using the constructor.

Ex → next page.

Account.java

```
public class Account {
    private int amount;                    //declaration
     public Account (int amount){
        this.amount = amount;              //initialization
     }
     public int getAmount (){
      return amount; }                     //utilization.
  }
```

Customer.java

```
  public class Customer{
    P S V M ( String[] args) {
   Account a1 = new Account (1000);
    S.O.P (a1.getAmount()); }
  }
```

Selenium code to click on login button:

```
// using single statement
 driver.findElement(By.id("loginButton")).click();

//using 2 statements
WebElement login = driver.findElement(By.id("loginButton"));
login.click();

// using 3 statements
WebElement login;                          // d
login = driver.findElement(By.id("loginButton"));   // i
login.click();                             // u
```

LoginPage.java

```
public class LoginPage {
    private WebElement login;                    //declaration
    public LoginPage(WebDriver driver) {
    login = driver.findElement(By.id("loginButton"));    //Initialization
    }
    public void clickLogin() {
    login.click(); }                             //utilization
}
```

TestLogin.java

```
public class TestLogin {
    PSVM(String[] args) {
    System.setProperty(" "," ");
    WebDriver driver = new FirefoxDriver();
    driver.get("http://localhost/login.do");
    LoginPage l = new LoginPage(driver);
    l.clickLogin(); }
}
```

Stale Element Reference Exception: Meaning of this exception is reference (address) of the element is stale (old/expired) We get this exception, if we find the element and page is refreshed before performing the action.

Ex:

```
WebDriver driver = new FirefoxDriver();
driver.get("http://localhost/login.do");
WebElement login = driver.findElement(By.id("loginButton"));
driver.navigate.refresh();
login.click();
```

PAGE OBJECT MODEL (POM)

1) Page object model is one of the Java design Pattern.
2) It is used to develop & test webpages.
3) In selenium, we use POM for following reasons -
   a) to handle stale element reference exception
   b) to encapsulate the element.
   c) to develop the script in organised way.

11-03-2017

Developing page object model class:

In page object model class we declare the elements using 'FindBy annotation' (@FindBy). It should be imported from following package - org.openqa.selenium.support.

The syntax of FindBy annotation is -

⇒
```
@FindBy(locator = "value")
  private WebElement elementName;
```

We initialize the element using following statement.

⇒
```
PageFactory.initElements(WebDriver, POMclassObject);
```

Ex:
```
public class LoginPage{
 @FindBy(id = "loginButton")
   private WebElement loginBT;
public LoginPage(WebDriver driver){
   PageFactory.initElements(driver, this);
 }
public void clickLogin(){
   loginBT.click(); }
 }
```

__/__/____

Testlogin.java

```
public class Testlogin{
  Public static void main (String[] args) throws Interrupted
  Exception{
  System.setProperty ("","");
  WebDriver driver = new FirefoxDriver();
  driver.get("http://localhost/login.do");
  LoginPage l= new LoginPage(driver);
  l.clicklogin();
  Thread.sleep(2000);
  l.clicklogin(); }
 }
```

If we do not use init element statement, then we get null pointer exception during runtime.

Developing POM class with multiple elements and without a constructor

Loginpage.java.

```
public class LoginPage {
@FindBy(id="username")
  private WebElement unTB;
@FindBy(id="LoginButton")
  private WebElement loginBT;
@FindBy(name="pwd")
  private WebElement pwTB;
public void Login (String un, String pw) {
  unTB.sendKeys(un);
  pwTB.sendkeys(pw);
 loginBT.click(); } }
```

__/__/____

Testlogin.java (changes to be made in prev ex)

```
LoginPage l= new LoginPage();
PageFactory.initElements(driver, l);
l.login("abc", "xyz");
Thread.Sleep(2000);
l.login("admin", "manager");
```

Handling FindElements in POM class:

Newspage.java

```
public class Newspage {
@FindBy (xpath = "//a")
 private List<WebElement> allLinks;
public NewsPage(WebDriver driver){
 PageFactory.initElements(driver, this); }
public void getAllLinkText(){
 int count = allLinks.size();
 S.O.P(count);
 for(int i=0; i<count; i++){
  WebElement link = allLinks.get(i);
  String text = link.getText();
  S.O.P(text); }
 }
}
```

Testlogin.java

```
public class Testlogin {
 PS VM (String[] args){
System.Setproperty("","");
WebDriver driver = new ChromeDriver();
driver.get("https://news.google.com");
NewsPage n=new NewsPage(driver);
n.getAllLinkText();
 driver.quit(); }}
```

Note: In Selenium - while automating the application, we develop two types of classes
1> POM class
2> Test class

Note: No of POM classes will be same as no of pages present in application.

Note: No of test classes will be same as no of manual test cases selected for automation.

Generally we develop Pom class first & then the test class. We can directly execute only test class; POM class are used only to store elements & its methods.

Hence POM class is also called as element repository class. Somebody also called as page object repository.

* abstraction is a concept in object oriented lang.
* abstract is a keyword in java.

IQ: What is the difference between POM & PageFactory.
→ POM is a java design pattern
→ PageFactory is a selenium class, which is used to implement the pom class.

Note: In POM class we cannot use variable in place of locator value. Below code is invalid

String xp = "//a";
@FindBy (xpath = xp) → we get error
Private list<WebElement> allLinks;

__/__/____

Mock Interview Questions :-

1) Explain Selenium Architecture?
2) Explain Selenium - Java Architecture?
3) What is locator? What are types?
4) Explain xpath.
5) What are the differences b/w findElement & findElements?
6) Explain Implicit & Explicit wait with flow diagram.
7) What are the uses of action class?
8) Write a script to print content of list box in sorted order.
9) Print only the nos present in the table.
10) What are the type of POP-UPs we have? How to handled them?