NO

DATE 23/12/18

Q. What is synchronisation & how to synchronize web driver?

→ The process of matching speed of automation tool with speed of application is called as 'synchronisation'.

We can synchronise web driver in two ways →

1) By using implicit wait
2) By using explicit wait

1) Implit wait -

- It will make web driver to wait till the specified element is identified within given timeout.
- If the element is not getting identified in a given time interval, then it will throw an exception ("No such element exception")

```
driver.manage().timeouts().implicitlyWait(30, TimeUnit.Seconds)
```

Ref of Webdriver interface	Ref of options interface	Ref of options interface	Abstract method of Timeouts	Predefined class in java	Constant defined in time unit class

2) Explicit wait -

This will make webdriver wait till the specified duration or till the expected condition is giving result as pass or fail in the given time interval.

- In explicit wait, we have 3 different types of waits
  i) Thread.sleep (long args) (2000) → This is not recommended to use frequently.
  ii) Webdriverwait
  iii) Fluentwait

Webdriverwait →

- Webdriverwait is also known as smart wait as we can specify the expected condition for required elements. But it can be applied only for specified element.
- Explicit wait gives better options than implicity wait as it will wait for dynamically loaded ajax elements.
- To use webdriverwait, we need to create an instance of webdriver wait class by passing webdriver reference & the time unit as an argument to its constructor.
- With the help of webdriver wait reference, call 'until' method which is an overloaded method, the argument should be expected conditions. Those conditions can be

NO

DATE

NO

DATE

```
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.elementsToBeClickable
                (By.cssSelector(".logout")));
```

1) alertIsPresent()
2) elementSelectionStateToBe()
3) elementToBeClickable()
4) elementToBeSelected()
5) frameToBeAvailableAndSwitchToIt()
6) invisibilityOfTheElementLocated()
7) ———— " ———— WithText()
8) presenceOfAllElementsLocatedBy()
9) ——— " ——— ElementsLocated()
10) textToBePresentInElement()
11) ——— " ——— Located()
12) ——— " ——— Value()
13) titleIs()
14) titleContains()
15) visibilityOf()
16) ——— " ——— AllElements()
17) ——— " ——— LocatedBy()
18) VisibilityOfElementLocated()

NO

DATE

Retriving Value from the Application -

- While doing validation, we compare expected result & actual result & then we will report the status.
- Acutual result will be taken from the appln during runtime, in order to do this we use folloling imp method -
  - i) getTitle()
  - ii) getCurrentUrL()
  - iii) getAttribute() →

This method will help us to retrive the web element attribute value based on attribute name

eg.

```
WebElement userNameField = driver.findElement
                (By.id("username"));
String placeHolder = userNameField.getAttribute
                ("placeholder");
S.o.p("placeHolder is: "+placeHolder);
```

OR

```
S.o.p("Placeholder is: "+driver.findElement
                (By.id("username")).getAttribute("
                Placeholder"));
```

NO

DATE

4) getText()

This method is used to retrive the inner text of following html objects -

<a> inner text

 ———

 ———

 ———

<td ——— </td>

e.g. WebElement btn = driver.FindElement(By.id ("loginButton");

S.o.p ('Btn name is" + btn.getText());

html code → < button, type = "Submit" id = "loginButton"> sign in </ button>

OR

S.D.P ("Btn name is:" + driver.findElement (By.id ("loginButton")).getText());

5) getCssValue()

This method will help us to retrive css attribute value based on attribute name.

This method is useful when we need to verify a button color, text color OR find style against css property.

e.g.

WebElement btn = driver.findElement(By.id ("loginButton")

s.o.p ("btn color:" + btn.getcssvalue("Background"))

OR

s.o.p. ("btn color:" + driver.findElement (By.id ("loginButton")).getCssValue("background"));

Functions	ReturnType	Result
isDisplayed()	boolean	T → displayed F → Not displayed
isEnabled()	boolean	T → enable F → disable
isSelected()	boolean	T → selected F → Not selected

NO

DATE

```
Public class Functional Testing
{
   Static WebDriver driver;
   p.s.v.m (String [] args)
{
/*Set path for chrome driver executable*/
system.setProperty ("webdriver.chrome.driver;
"E:\\selenium softwares\\chromedriver....");
   driver = new ChromeDriver ();
 driver.geturl ("https://demo.actitime.com/");

 /**implicit wait */
driver.manage().timeouts().implicitlywait
        (30, timeUnit.second);
test UserNameField();
test checkbox();
test LoginBtn();
  }

 public static void testloginBtn()
{
  WebElement  btn = driver.findElement (By.id("loginBtn"));
s.o.p ("Display:"+ Btn.isDisplayed ();
s.o.p  ("Enable:" + Btn.isEnable();
s.o.p ("btn name is "+Btn.getText().equals("Login"));
s.o.p ("btn Colo"+ btn.getCssValue ("Background");
```

NO

DATE

```
public static void testcheckBox () {
webelement checkBox = driver.findElement (By.id
("keep logged in checkbox");
s.o.p ("Display:"+ checkBox.isDisplayed ());
—— ,, "Enable+ + checkBox.isEnabled ());
—— ,, ("default - " selection + checkBox.isSelected ());
checkBox.click ();
s.o.p ("After selection:"+ checkbox.isSelected ());
checkBox.Click ();
s.o.p ("After deselection:" + checkbox.isSelected());
}

 public static void testUserNameField()
{
"WebElement UserNameField = driver.findElement
  (By.id ("username"));
s.o.p ("Display"+ userNamefield.isdisplayed());
s.o.p ("Enabled"+ userNamefield.isEnabled());
String PlaceHolder = userNamefield.getAttribute (
"placelolder"));
s.o.p ("PlaceHolder is " + placeholder);
}
}
```