**Which 3 locators exist in every HTML or WEB element**

XPATH

Selector

TAGNAME

**How to implement implicit wait**

1. After 1s time from page load, Page is not loaded from scratch when asynchronous calls are made to generate the request and get the response so in this case finding element, we impose implicit wait on element
2. **driver.manage().timeouts.implicitlyWait(10L,TimeUnit.SECONDS);**
3. Here time, means wait for maximum 10 seconds, if element is not presented beyond 10 seconds then it will flash timeout error.
4. It only works with presence of elements
5. It is tied up with driver, and applicable where driver.findElement() method is executed.
6. Isdisplayed Function

a) checks visibility on page on UI, but false for hidden elements (not visible on page)

b) if no element (specify wrong id or invalid xpath), then error. (driver.findelement)

**implement explicit wait along with visibility of element located**

let's say particualar elemenet will be take max 10 seconds to appear on DOM of a page and visible on UI (visible means element must have height and width greater than 0) after async request.

String errtext = **null**;

WebDriverWait wait = **null**;

WebElement errmsg = **null**;

**try**

{

wait = **new** WebDriverWait(driver,10L);

errmsg = wait.until(ExpectedConditions.*visibilityOfElementLocated*(By.*xpath*("//\*[@id=\\\"view\_container\\\"]/div/div/div[2]/div/div[1]/div/form/content/section/div/content/div[1]/div/div[2]/div[2]/div")));

errtext = errmsg.getText();

}**catch**(Exception ex)

{

errtext="error not found";

}

**implement explicit wait along with presence of element located**

String errtext = **null**;

WebDriverWait wait = **null**;

WebElement errmsg = **null**;

**try**

{

wait = **new** WebDriverWait(driver,10L);

errmsg = wait.until(ExpectedConditions.*presenceOfElementLocated*(By.*xpath*("//\*[@id=\\\"view\_container\\\"]/div/div/div[2]/div/div[1]/div/form/content/section/div/content/div[1]/div/div[2]/div[2]/div")));

//errmsg = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//\*[@id=\\\"view\_container\\\"]/div/div/div[2]/div/div[1]/div/form/content/section/div/content/div[1]/div/div[2]/div[2]/div")));

errtext = errmsg.getText();

}**catch**(Exception ex)

{

errtext="error not found";

}

# difference b/w [VisibilityOfElementLocated Vs presenceOfElementLocated](https://stackoverflow.com/questions/38038920/visibilityofelementlocated-vs-presenceofelementlocated)

You can use both [presenceOfElementLocated](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html" \l "presenceOfElementLocated-org.openqa.selenium.By-) or [visibilityOfElementLocated](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html" \l "visibilityOfElementLocated-org.openqa.selenium.By-) to get the value.

But for the performance perspective, I would guess that [presenceOfElementLocated](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html" \l "presenceOfElementLocated-org.openqa.selenium.By-) will be slightly faster because it's just check that an **element is present on the DOM of a page**. This does not necessarily mean that the element is visible. while the [visibilityOfElementLocated](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html" \l "visibilityOfElementLocated-org.openqa.selenium.By-) has to check that an **element is present on the DOM of a page and visible**. Visibility means that the element is not only displayed but also has a height and width that is greater than 0.

you can consider the following point to choose appropriate method depending on your use case.

* use [presenceOfElementLocated](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html" \l "presenceOfElementLocated-org.openqa.selenium.By-) when you don't care whether if element visible or not, you just need to know if it's on the page.
* use [visibilityOfElementLocated](https://seleniumhq.github.io/selenium/docs/api/java/org/openqa/selenium/support/ui/ExpectedConditions.html" \l "visibilityOfElementLocated-org.openqa.selenium.By-) when you need to find element which should be also visible.

**Fluent wait**

In real scenario, some element takes 10 seconds to load or 20 seconds or 30 seconds to load. either setup explicit wait with max 30 seconds that actually check the presence after every second or have some of the way to setup max wait 30 seconds but check out the presence after every 5 second, means 6 times hit should be made, in addition to that, we want to ignore the exception i.e. NoSuchElementException and want to give our own message.

Wait<WebDriver> wait = **new** FluentWait<WebDriver>(driver)

.~~withTimeout~~(30, TimeUnit.***SECONDS***)

.~~pollingEvery~~(5, TimeUnit.***SECONDS***)

.withMessage("element is not found in 30 seconds")

.ignoring(NoSuchElementException.**class**);

WebElement errmsg1 = wait.until(**new** Function<WebDriver, WebElement>() {

**public** WebElement apply(WebDriver driver) {

**return** driver.findElement(By.*xpath*("//\*[@id=\"view\_container\"]/div/div/div[2]/div/div[1]/div/form/content/section/div/content/div[1]/div/div[2]/div[2]/div"));

}

});

**What are the various way to open web page.**

1. -first way is = driver.get(specify url)
2. -second way is = driver.navigate().to(specify url)
3. -the difference in between two is, that **navigate() method gives us features like back, forward, refresh which are not available in get method.**

**Deal with drop down**

1) **when drop down contains english text and english values.**

*driver.findElement(By.xpath()).sendKeys(<text>);*

2) **when drop down contains information in multi-langual format**

*WebElement dd = driver.findElement(By.xpath());*

*Select s = new Select(dd);*

*s.selectByVisibleText("<text>"); // or // s.selectByValue("<value>");*

3) **select all values from drop down**

*use method "getOptions" belongs to "Select" class or find the elements by using locator "tagname" and pass argument as "option"*

4) **select particular set of drop down, mean to say belongs to particular block**

*examine the drop down and get xpath by just tweak at last, by doing so u will get block.*

**for each loop**

*for(WebElement link: links)*

*{*

*link.getText(); //around sopl statement*

*}*