Difference Between Implicit, Explicit & Fluent Wait

Polling happens in all the waits. In implicit wait, polling happens at every 250 milli seconds, in explicit wait, polling happens at every 500 milli seconds. In fluent wait its customized.

Fluent wait is an explicit wait as we apply it on a particular element, similar to webdriverwait.

FluentWait is Parent class of **WebDriverWait** class.

So, if implicit and explicit both waits are not working for some particular element because it's taking more time to load we can try using fluent wait. Coz it will keep on searching that element for given time.

If implicit is 30 sec and webdriverwait is 30 sec but elements loads after 1 min in dom. So, these two waits may not work. So, using fluent wait can help here for that particular element.

WebDriverWait by default calls the ExpectedCondition every 500 milliseconds(until it returns successfully. And this 500 milli seconds field (FIVE_HUNDRED_MILLIS) is inherited from FluentWait class.

In `FluentWait` you have more options to configure, along with maximum wait time, like polling interval, exceptions to ignore etc.

FluentWait implements **Wait** interface.

```
public class WebDriverWait
extends FluentWait<WebDriver>
```

A specialization of `FluentWait` that uses WebDriver instances.

Implicit Wait

Selenium WebDriver has borrowed the idea of **implicit waits** from **Watir**. This means that we can tell Selenium that we would like it to wait for a certain amount of time before throwing an **exception** that it cannot find the element on the page. We should note that implicit waits will be in place for the entire time the browser is open. This means that any search for elements on the page could take the time the implicit wait is set for.

```
WebDriver driver = new FirefoxDriver();

 driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

driver.get("http://url_that_delays_loading");

WebElement myDynamicElement =
driver.findElement(By.id("myDynamicElement"));
```

Fluent Wait

Each **FluentWait** instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as **NoSuchElementExceptions** when searching for an element on the page.

```
Waiting 30 seconds for an element to be present on the page, checking

  // for its presence once every 5 seconds.

  Wait wait = new FluentWait(driver)

    .withTimeout(30, SECONDS)

    .pollingEvery(5, SECONDS)

    .ignoring(NoSuchElementException.class);

  WebElement foo = wait.until(new Function() {

    public WebElement apply(WebDriver driver) {
```

```
    return driver.findElement(By.id("foo"));
    }
  });
```

Note- If the element is located within this time frame(30 sec) it will perform the operations else it will throw an" **ElementNotVisibleException**"

Explicit Wait

It is more extendible in the means that you can set it up to wait for any condition you might like. Usually, you can use some of the prebuilt **ExpectedConditions** to wait for elements to become clickable, visible, invisible, etc.

```
WebDriverWait wait = new WebDriverWait(driver, 10);

WebElement element = wait.until(ExpectedConditions.elementToBeClickable(By.id("someid")));
```

Difference Between Implicit, Explicit and Fluent Wait

Implicit Wait: During Implicit wait if the Web Driver cannot find it immediately because of its availability, it will keep polling (around 250 milli seconds) the DOM to get the element. If the element is not available within the specified Time an NoSuchElementException will be raised. The default setting is zero. Once we set a time, the Web Driver waits for the period of the WebDriver object instance.
Explicit Wait: There can be instance when a particular element takes more than a minute to load. In that case you definitely not like to set a huge time to Implicit wait, as if you do this your browser will going to wait for the same time for every element.
To avoid that situation you can simply put a separate time on the required element only. By following this your browser implicit wait time would be short for every element and it would be large for specific element.

Fluent Wait: Let's say you have an element which sometime appears in just 1 second and some time it takes minutes to appear. In that case it is better

to use fluent wait, as this will try to find element again and again until it find it or until the final timer runs out.
Solutions: We always get confuse when it comes to using Wait commands, to better understand it we need to remember that there is a difference between several scenarios:
An element not being present at all in the DOM.

An element being present in the DOM but not visible.

An element being present in the DOM but not enabled. (i.e. clickable)

In such situations I'd prefer using explicit wait (`fluentWait` in particular):

```
public WebElement fluentWait(final By locator) {
   Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)
         .withTimeout(30, TimeUnit.SECONDS)
         .pollingEvery(5, TimeUnit.SECONDS)
         .ignoring(NoSuchElementException.class);

   WebElement foo = wait.until(new Function<WebDriver, WebElement>() {
     public WebElement apply(WebDriver driver) {
       return driver.findElement(locator);
     }
   });

   return  foo;
};
```

`fluentWait` function returns your found web element. From the documentation on `fluentWait`: *An implementation of the Wait interface that may have its timeout and polling interval configured on the fly. Each FluentWait instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as NoSuchElementExceptions when searching for an element on the page.*

Explicit wait: Explicit wait is of two types:
1) WebDriverWait

2) FluentWait

both are classes and implements Wait interface.

WebDriverWait is applied on certain element with defined expected condition and time. This wait is only applied to the specified element. This wait can also throw exception when element is not found.

```
WebDriverWait wait = new WebDriverWait (driver, 20);
wait.until(ExpectedConditions.VisibilityofElementLocated(By.xpath(""//button[@value='Save Changes']"")));
```

Fluent wait: Fluent wait is another type of Explicit wait and you can define polling and ignore the exception to continue with script execution in case element is not found.

```
new FluentWait<WebDriver>(driver).withTimeout(30, TimeUnit.SECONDS).pollingevery(10, TimeUnit.SECONDS).ignoring(NoSuchElementException.class);
```

The following are the Expected Conditions that can be used in Explicit Wait

1. alertIsPresent()
2. elementSelectionStateToBe()
3. elementToBeClickable()
4. elementToBeSelected()
5. frameToBeAvaliableAndSwitchToIt()
6. invisibilityOfTheElementLocated()
7. invisibilityOfElementWithText()
8. presenceOfAllElementsLocatedBy()
9. presenceOfElementLocated()
10. textToBePresentInElement()
11. textToBePresentInElementLocated()
12. textToBePresentInElementValue()
13. titleIs()
14. titleContains()
15. visibilityOf()
16. visibilityOfAllElements()
17. visibilityOfAllElementsLocatedBy()
18. visibilityOfElementLocated()