Selenium WebDriver Implicit Wait, Explicit Wait and Fluent Wait

Press Space for next page →

Agenda

1. PageLoadTimeOut
2. Code Example: PageLoadTimeOut
3. Implicit Wait
4. How Implicit Wait Works?
5. Code Example: Implicit Wait
6. Explicit Wait
7. Why Use Explicit Wait?
8. Scenario of Using Explicit Wait
9. Code Example: Explicit Wait

PageLoadTimeOut

> A timeout value that specifies the maximum amount of time Selenium will wait for a web page to load.

> If the page does not load within the specified time, Selenium will throw an exception.

When to use it?

- When you are unsure how long a web page will take to load.
- When you want to prevent your tests from hanging indefinitely.

```
1   driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(3));
2   driver.get("https://learn-automation.com/selenium-webdriver-tutorial-for-beginners/");
```

Code Example: PageLoadTimeOut

```
class PageLoadTimeOutExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(20));

        driver.get("https://learn-automation.com/selenium-webdriver-tutorial-for-beginners/");
        System.out.println(driver.getTitle());
        driver.quit();
    }
}
```

Implicit Wait

Purpose: Implicit wait is used to instruct the WebDriver to wait for a certain amount of time before throwing a NoSuchElementException. This is useful when elements take time to load or become visible on the web page.

Global Setting: Once set, the implicit wait time is applied to all subsequent WebDriver element lookups. This means every time an element is searched for, WebDriver will wait up to the specified time before failing.

Syntax: `driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));`

How Implicit Wait Works?

Duration: The duration is specified in seconds, and it is set only once. For example, if the implicit wait is set to 10 seconds, WebDriver will wait up to 10 seconds for each element search.

Default Behavior: By default, WebDriver does not have any implicit wait set. If you don't set an implicit wait, WebDriver will immediately throw an exception if an element is not found.

Overriding with Explicit Wait: Implicit wait can be overridden by explicit waits in specific scenarios where a different waiting time is required. Explicit waits allow for more fine-grained control over waiting conditions.

Code Example: Implicit Wait

```
class ImplicitWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(3));

        driver.get("https://www.facebook.com/");
        System.out.println(driver.getTitle());

        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

        driver.findElement(By.xpath("//input[@id='email']")).sendKeys("deb.nexxvali@gmail.com");
        driver.findElement(By.xpath("//input[@id='pass']")).sendKeys("deb.nexxvali@gmail.com");
        driver.findElement(By.xpath("//button[text()='']")).click();

        driver.quit();
    }
}
```

Explicit Wait

Purpose: Explicit wait is used to pause the execution of the script until a certain condition is met or the maximum specified time has elapsed. This is particularly useful when dealing with elements that may take unpredictable amounts of time to appear or become interactable.

Condition-Based Waiting: Unlike implicit wait, explicit wait allows you to wait for specific conditions to be true. Common conditions include the presence of an element, visibility of an element, element to be clickable, etc.

WebDriverWait Class: Explicit wait is implemented using the WebDriverWait class in conjunction with ExpectedConditions.

Syntax::

```
1   WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
2   WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("someId")));
```

Why Use Explicit Wait?

Flexibility and Control: Explicit waits provide greater flexibility and control over the waiting mechanism. You can specify the exact conditions under which the wait should proceed or time out, making tests more robust and reliable.

Timeouts: Explicit wait will wait for the specified amount of time (in seconds) before throwing a TimeoutException if the condition is not met. You can also specify polling intervals to check the condition at regular intervals.

Avoid Overusing Implicit Waits: It's generally a good practice to use explicit waits for elements that have variable load times or require specific conditions to be met. Combining implicit and explicit waits should be done cautiously to avoid conflicts.

Scenario of Using Explicit Wait

Commonly Used Conditions:

- presenceOfElementLocated
- visibilityOfElementLocated
- elementToBeClickable
- textToBePresentInElement
- alertIsPresent

Example Scenarios:

- Waiting for an element to become clickable before attempting a click.
- Waiting for a specific text to appear in an element before proceeding.
- Waiting for an alert to be present and accepting it.

Code Example: Explicit Wait

```
class ExplicitWaitExample {
    public static void main(String[] args) {
        WebDriver driver = new ChromeDriver();
        driver.get("http://seleniumpractise.blogspot.in/2016/08/how-to-use-explicit-wait-in-selenium.html");

        driver.findElement(By.xpath("//button[text()='Click me to start timer']")).click();

        // By default, it  accepts in Seconds
        WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(15));

        // Here we will wait until element is not visible, if element is visible then it will return web element
        // or else it will throw exception
        WebElement element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//p[text()='WebDriver']")));

        boolean status = element.isDisplayed();

        if (status) {
            System.out.println("====== WebDriver is visible======");
        } else {
            System.out.println("====== WebDriver is not visible======");
        }
    }
}
```

Thank you ❤️

qa-june-2024-automation-with-java-slides