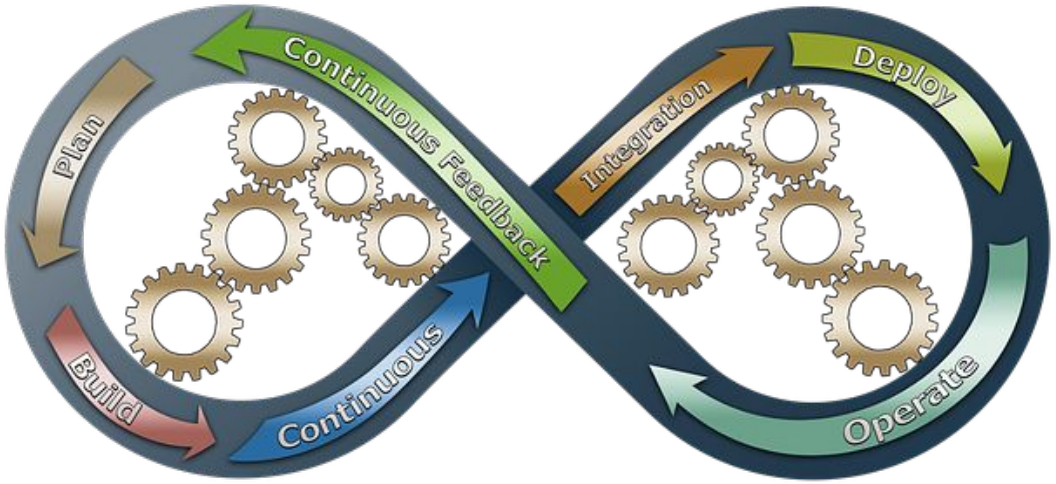


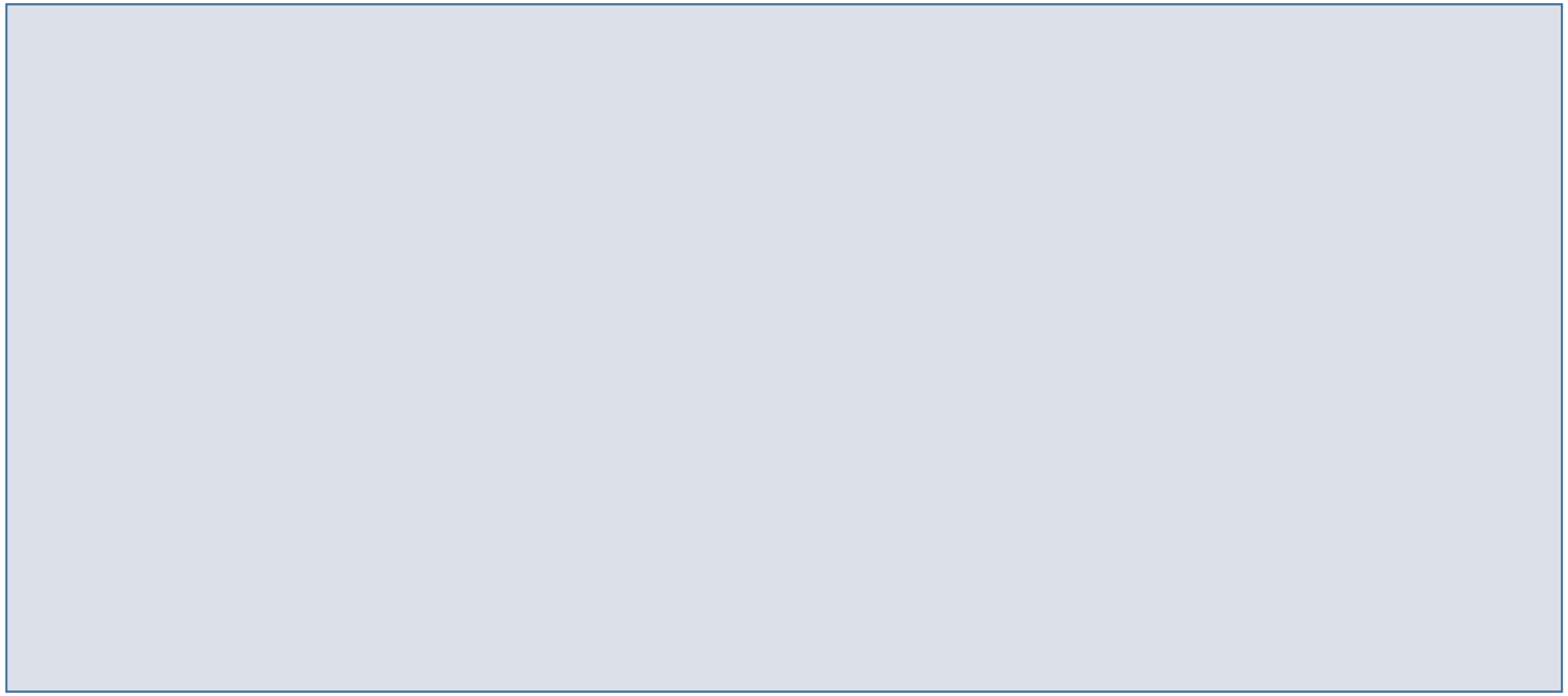
Session 4.7

Synchronization

AN INITIATIVE BY

UNICAL ACADEMY

Introduction



Let’s go!!!

Synchronization

Synchronization can be classified into two categories:

- **Unconditional Synchronization**
- **Conditional Synchronization**

Unconditional Synchronization:

Unconditional synchronization indicates timeout value alone and makes tool to continue further only when specified time is elapsed. One such functions is

```
System.Threading.Thread.Sleep();
```

Conditional Synchronization:

By using conditional synchronization we can specify timeout duration along with some desired conditions to check and then throw an error if nothing happens.

Types of Conditional Synchronization:

1. Implicit Wait
2. Explicit Wait

UNICAL ACADEMY

Implicit Wait:

- Implicit Wait tells the WebDriver to Wait until the stated time before throwing the NoSuchElement/ElementNotVisible exception.
- Waiting time across the test script between each consecutive steps are taken by default. Hence, next testStep will execute only when the specified time is elapsed post executing the previous testStep.

Syntax:

```
driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(30));
```

The Implicit wait takes one argument: TimeSpan (timeToWait)

- FromMilliSeconds
- FromSeconds
- FromMinutes
- FromHours
- FromDays

UNICAL ACADEMY

Explicit Wait:

- Explicit waits are used when page loads dynamically.
- Explicit Wait tells the WebDriver to Wait until the specified condition is met or maximum time elapses before throwing NoSuchElement (or) ElementNotVisible Exceptions.
- Explicit waits are applied for the specified testStep in test script.
- To use Explicit waits, we must first create instance for **"WebDriverWait"** class.

Syntax:

```
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(30));
```

```
wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.Id(element_ID)));
```

The reference variable "wait" informs the WebDriver to wait until the Expected condition to occur (or) Wait for the specified time, whichever shows in first place before throwing an exception.

UNICAL ACADEMY

Page Load Timeout:

- Page load timeout in selenium requests/set the time limit for a page to load, If the page is not loaded within the given time frame selenium throws **TimeOutException**.
- We can set the page load timeout using the pageLoadTimeout method present in Browser classes(FirefoxDriver, ChromeDriver..)

Syntax:

```
driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);
```

- Page load timeout is useful when we perform a performance test, or when we test execution in IE.
- Page Load timeout is applicable only to driver.get() and driver.navigate().to() methods in selenium
- Setting Negative time limit makes the selenium to wait for the page load infinitely

```
// set page load time as infinite (by giving minus value)
driver.manage().timeouts().pageLoadTimeout(-10, TimeUnit.SECONDS);
```

UNICAL ACADEMY

Set Script Load timeout:

- setScriptTimeout sets the time limit for the asynchronous script to finish execution, if the process is not completed before the time limit, **selenium** throws **TimeOutException**
- The setScriptTimeout method affects only **JavaScript code** executed with executeAsyncScript and nothing else. In particular, executeScript is not affected by it.
- The default timeout for setScriptTimeout method is 0 (zero) if we do not set any time executeAsyncScript method may fail because the JavaScript code may take more than zero seconds. So to avoid unnecessary failures, we have to set the setScriptTimeout.
- Run a simple javascript: (Do not set setScriptTimeout) - Now this shall execute without throwing any issue.

```
((JavascriptExecutor) driver).executeScript("alert('I am alert');");
```

- Run a simple Async Script: (Do not set setScriptTimeout) - This shall fail with error - "Timed out waiting for async script result after 0ms."

```
((JavascriptExecutor) driver).executeAsyncScript("document.getElementById('dummy')");
```

- To resolve above issue add below-given setScriptTimeout

```
driver.manage().timeouts().setScriptTimeout(10, TimeUnit.SECONDS);
((JavascriptExecutor) driver).executeAsyncScript("document.getElementById('dummy')");
```

UNICAL ACADEMY

WebDriverWait implementation

WebDriverWait

- It 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.

Syntax:

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

Different Expected Conditions

UNICAL ACADEMY

Expected Conditions:

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()

UNICAL ACADEMY

Implement FluentWait

FluentWait :

- FluentWait can define the maximum amount of time to wait for a specific condition and frequency with which to check the condition before throwing an "*ElementNotVisibleException*" exception.
- It tries to find the web element repeatedly at regular intervals of time until the timeout or till the object gets found.
- We use FluentWait commands mainly when we have web elements which sometimes visible in few seconds and some times take more time than usual.

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(timeout, SECONDS)
.pollingEvery(timeout, SECONDS)
.ignoring(Exception.class);

WebElement foo=wait.until(new Function<WebDriver, WebElement>() {
public WebElement applyy(WebDriver driver) {
return driver.findElement(By.id("foo"));
}
})
```

- FluentWait uses two parameters mainly – *timeout value* and *polling frequency*.

UNICAL ACADEMY

Session Recap