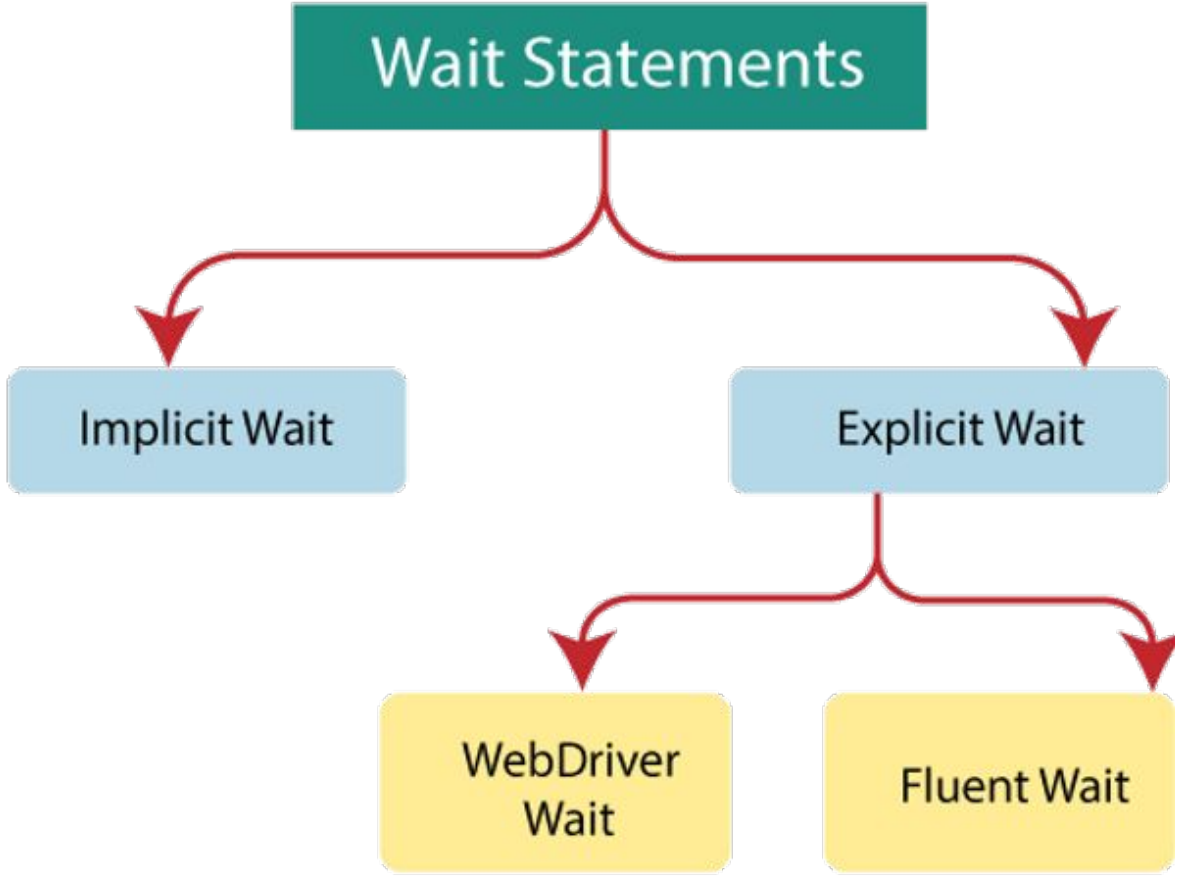
SDJ INTERNATIONAL COLLEGE

Selenium Python

Unit 5

– By Tushar Sir



– By Tushar Sir

Implicit, Explicit and Fluent Wait

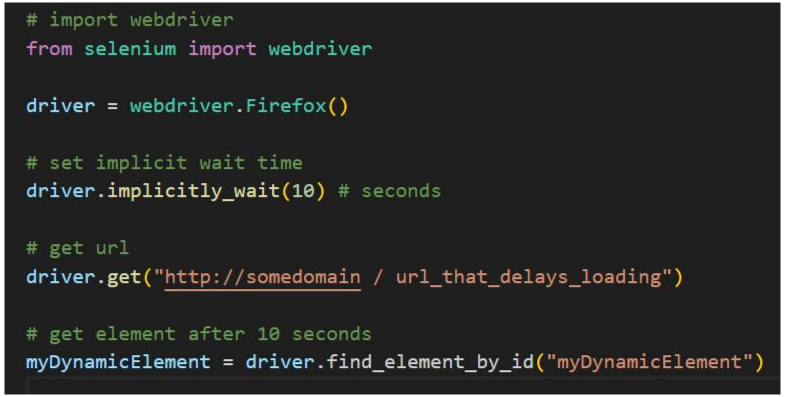
- Selenium Python is one of the great tools for testing automation. These days most of the web apps are using AJAX techniques.
- When a page is loaded by the browser, the elements within that page may load at different time intervals.
- This makes locating elements difficult: if an element is not yet present in the DOM, a locate function will raise an ElementNotVisibleException exception. Using waits, we can solve this issue.
- Waiting provides some slack between actions performed – mostly locating an element or any other operation with the element. Selenium Webdriver provides two types of waits – implicit & explicit.

– By Tushar Sir

Implicit, Explicit and Fluent Wait

Implicit Waits :

- An implicit wait tells WebDriver to poll the DOM for a certain amount of time when trying to find any element (or elements) not immediately available. The default setting is 0. Once set, the implicit wait is set for the life of the WebDriver object.



```python
# import webdriver
from selenium import webdriver

driver = webdriver.Firefox()

# set implicit wait time
driver.implicitly_wait(10) # seconds

# get url
driver.get("http://somedomain / url_that_delays_loading")

# get element after 10 seconds
myDynamicElement = driver.find_element_by_id("myDynamicElement")
```

- This waits up to 10 seconds before throwing a TimeoutException unless it finds the element to return within 10 seconds.

– By Tushar Sir

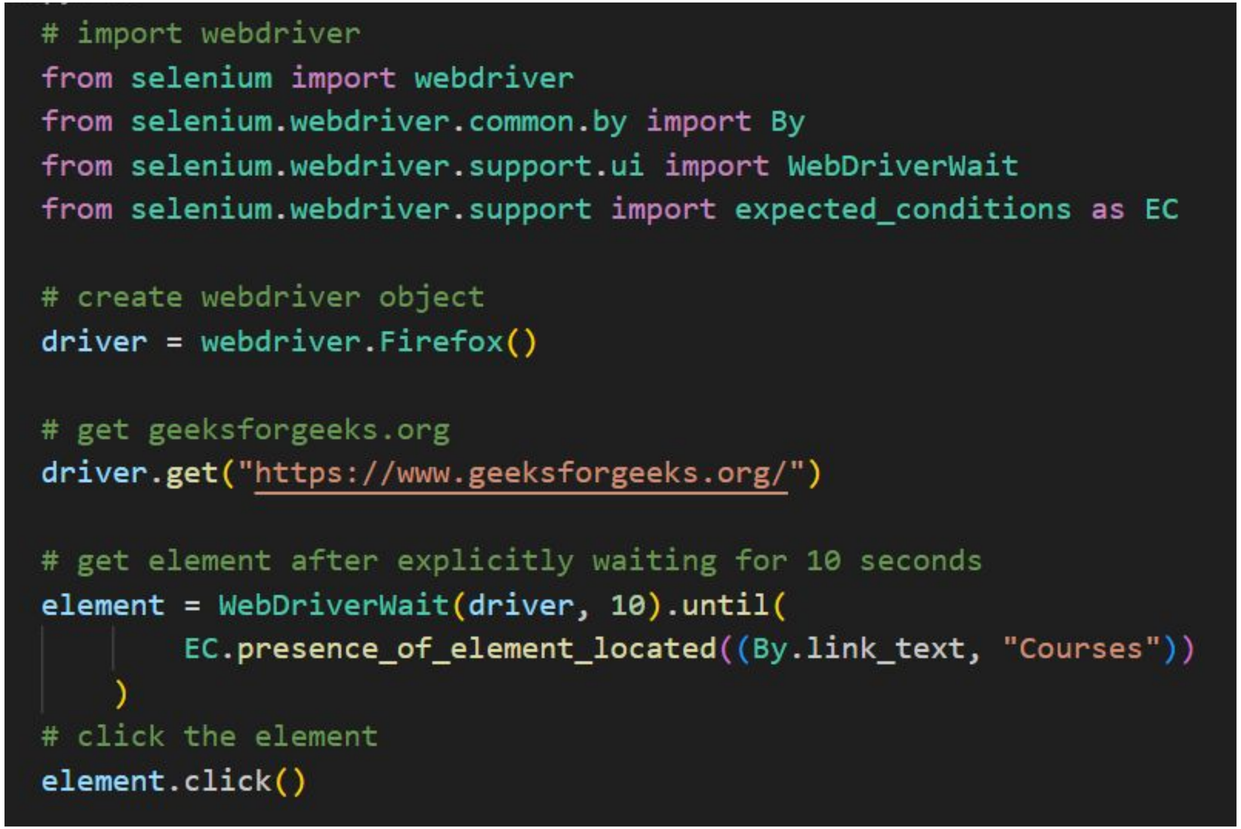
Implicit, Explicit and Fluent Wait

Explicit Waits :

- An explicit wait is a code you define to wait for a certain condition to occur before proceeding further in the code.
- The extreme case of this is time.sleep(), which sets the condition to an exact time period to wait.
- There are some convenience methods provided that help you write code that will wait only as long as required.
- Explicit waits are achieved by using webdriverWait class in combination with expected_conditions.
- This waits up to 10 seconds before throwing a TimeoutException unless it finds the element to return within 10 seconds. WebDriverWait by default calls the ExpectedCondition every 500 milliseconds until it returns successfully.

– By Tushar Sir

Implicit, Explicit and Fluent Wait



```python
# import webdriver
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

# create webdriver object
driver = webdriver.Firefox()

# get geeksforgeeks.org
driver.get("https://www.geeksforgeeks.org/")

# get element after explicitly waiting for 10 seconds
element = WebDriverWait(driver, 10).until(
        EC.presence_of_element_located((By.link_text, "Courses"))
    )
# click the element
element.click()
```

– By Tushar Sir

Implicit, Explicit and Fluent Wait

Expected Conditions –

- There are some common conditions that are frequently of use when automating web browsers. For example, presence_of_element_located, title_is, ad so on. one can check entire methods from here – Convenience Methods. Some of them are –
    - title_is
    - title_contains
    - presence_of_element_located
    - visibility_of_element_located
    - visibility_of
    - presence_of_all_elements_located
    - element_located_to_be_selected
    - element_selection_state_to_be
    - element_located_selection_state_to_be
    - Alert_is_present

– By Tushar Sir

Implicit, Explicit and Fluent Wait

Fluent Wait :

- Fluent Wait in Selenium marks the maximum amount of time for Selenium WebDriver to wait for a certain condition (web element) becomes visible. It also defines how frequently WebDriver will check if the condition appears before throwing the **"ElementNotVisibleException".**
- To put it simply, Fluent Wait looks for a web element repeatedly at regular intervals until timeout happens or until the object is found.
- Fluent Wait commands are most useful when interacting with web elements that can take longer durations to load. This is something that often occurs in Ajax applications.

– By Tushar Sir

Implicit, Explicit and Fluent Wait

Here are some commonly used methods and commands for Fluent Wait in Selenium:

- **WebDriverWait(driver, timeout):** The given WebDriver instance and timeout value are used to create a new instance of `WebDriverWait` through this command.
- **until(expected_condition):** This method is used to define the condition to wait for. It waits until the provided expected condition is met or until the timeout expires. Some commonly used expected conditions are:
  - **`visibility_of_element_located(locator)`:** Waits until the element located by the given locator becomes visible.
  - **`presence_of_element_located(locator)`:** Waits until the element located by the given locator is present in the DOM.
  - **`element_to_be_clickable(locator)`:** Waits until the element located by the given locator is visible and enabled, allowing it to be clicked.
  - **`text_to_be_present_in_element(locator, text)`:** Waits until the element located by the given locator contains the specified text.

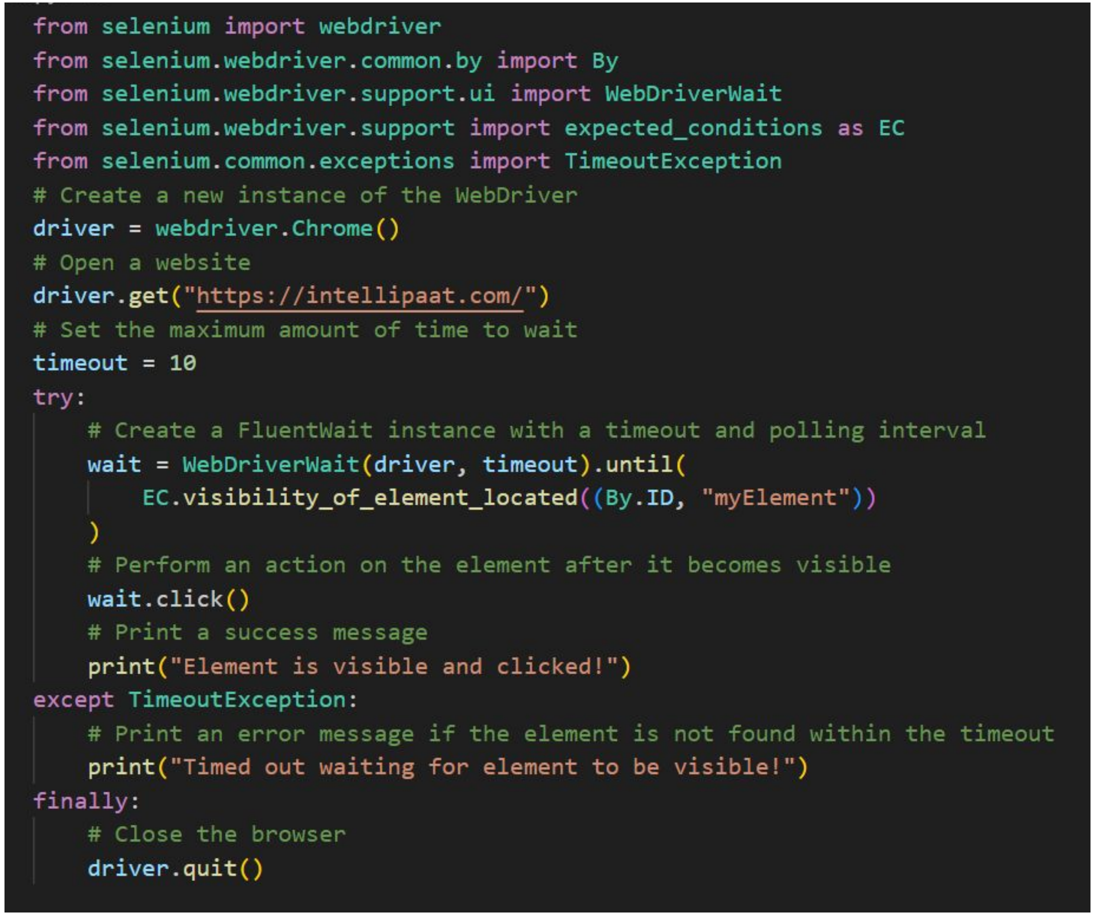
– By Tushar Sir

Implicit, Explicit and Fluent Wait

- **polling(interval):** This method sets the polling interval in seconds. It specifies the frequency at which Fluent Wait will check for the expected condition to be true.
- **ignore(exception_type):** This method specifies the exception(s) to be ignored during polling. It allows the wait to continue even if the specified exception(s) occur.
- **until_not(expected_condition):** This method waits until the provided expected condition is not met or until the timeout expires. It is useful when waiting for an element to disappear or become invisible.
- **timeout_exception():** This method throws a `TimeoutException` if the expected condition is not met within the specified timeout period.

– By Tushar Sir

Implicit, Explicit and Fluent Wait



```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import TimeoutException
# Create a new instance of the WebDriver
driver = webdriver.Chrome()
# Open a website
driver.get("https://intellipaat.com/")
# Set the maximum amount of time to wait
timeout = 10
try:
    # Create a FluentWait instance with a timeout and polling interval
    wait = WebDriverWait(driver, timeout).until(
        EC.visibility_of_element_located((By.ID, "myElement"))
    )
    # Perform an action on the element after it becomes visible
    wait.click()
    # Print a success message
    print("Element is visible and clicked!")
except TimeoutException:
    # Print an error message if the element is not found within the timeout
    print("Timed out waiting for element to be visible!")
finally:
    # Close the browser
    driver.quit()
```

– By Tushar Sir

Implicit, Explicit and Fluent Wait

Implicit Wait in Selenium	Explicit Wait in Selenium
Applies to all elements in a test script.	Applies only to specific elements as intended by the user.
No need to specify "ExpectedConditions" on the element to be located	Must always specify "ExpectedConditions" on the element to be located
Most effective when used in a test case in which the elements are located with the time frame specified in implicit wait	Most effective when used when the elements are taking a long time to load. Also useful for verifying property of the element, such as *visibilityOfElementLocated, elementToBeClickable, elementToBeSelected*

– By Tushar Sir

Implicit, Explicit and Fluent Wait

The driver is asked to wait for a specific amount of time for the element to be available on the DOM of the page.	The driver is asked to wait till a certain condition is satisfied.
It is a global wait and applied to all elements on the webpage.	It is not a global wait and applied to a particular scenario.
It is simple and easy to implement.	It is more complex in implementation compared to implicit wait.
It affects the execution speed since each step waits for this wait till it gets the element it is looking for.	It does not affect the execution speed since it is applicable to a particular element of the page.
It does not catch performance issues in the application.	It can catch performance issues in the application.

– By Tushar Sir

Implicit, Explicit and Fluent Wait

- **polling(interval):** This method sets the polling interval in seconds. It specifies the frequency at which Fluent Wait will check for the expected condition to be true.
- **ignore(exception_type):** This method specifies the exception(s) to be ignored during polling. It allows the wait to continue even if the specified exception(s) occur.
- **until_not(expected_condition):** This method waits until the provided expected condition is not met or until the timeout expires. It is useful when waiting for an element to disappear or become invisible.
- **timeout_exception():** This method throws a `TimeoutException` if the expected condition is not met within the specified timeout period.

– By Tushar Sir

Apache POI

- Apache POI, a project run by the Apache Software Foundation, and previously a sub-project of the Jakarta Project, provides pure Java libraries for reading and writing files in Microsoft Office formats, such as Word, PowerPoint and Excel.
- Apache POI is an open-source java library to create and manipulate various file formats based on Microsoft Office. Using POI, one should be able to perform create, modify and display/read operations on the following file formats.
- For Example, Java doesn’t provide built-in support for working with excel files, so we need to look for open-source APIs for the job.
- Apache POI provides Java API for manipulating various file formats based on the Office Open XML (OOXML) standard and OLE2 standard from Microsoft. Apache POI releases are available under the Apache License (V2.0).

– By Tushar Sir

Apache POI

Some important features of Apache POI are as follows:

- Apache POI provides stream-based processing, that is suitable for large files and requires less memory.
- Apache POI is able to handle both XLS and XLSX formats of spreadsheets.
- Apache POI contains HSSF implementation for Excel ’97(-2007) file format i.e XLS.
- Apache POI XSSF implementation should be used for Excel 2007 OOXML (.xlsx) file format.
- Apache POI HSSF and XSSF API provide mechanisms to read, write or modify excel spreadsheets.
- Apache POI also provides SXSSF API that is an extension of XSSF to work with very large excel sheets.

– By Tushar Sir

Apache POI

- SXSSF API requires less memory and is suitable when working with very large spreadsheets and heap memory is limited.
- There are two models to choose from – the event model and the user model. The event model requires less memory because the excel file is read in tokens and requires processing. The user model is more object-oriented and easy to use.
- Apache POI provides excellent support for additional excel features such as working with Formulas, creating cell styles by filling colors and borders, fonts, headers and footers, data validations, images, hyperlinks, etc.

– By Tushar Sir

Apache POI

Commonly used components of Apache POI

- **HSSF (Horrible Spreadsheet Format):** It is used to read and write xls format of MS-Excel files.
- **XSSF (XML Spreadsheet Format):** It is used for xlsx file format of MS-Excel.
- **POIFS (Poor Obfuscation Implementation File System):** This component is the basic factor of all other POI elements. It is used to read different files explicitly.
- **HWPF (Horrible Word Processor Format):** It is used to read and write doc extension files of MS-Word.
- **HSLF (Horrible Slide Layout Format):** It is used for read, create, and edit PowerPoint presentations.

– By Tushar Sir

Apache POI - Read and Write Data from Excel File

- Learn to read excel, write excel, evaluate formula cells and apply custom formatting to the generated excel files using Apache POI library with examples.
- If we are building software for the HR or Finance domain, there is usually a requirement for generating excel reports across management levels. Apart from reports, we can also expect some input data for the applications coming in the form of excel sheets and the application is expected to support this requirement.
- Apache POI is a well-trusted library among many other open-source libraries to handle such usecases involving excel files. Please note that, in addition, we can read and write MS Word and MS PowerPoint files also using the Apache POI library.

https://howtodoinjava.com/java/library/readingwriting-excel-files-in-java-poi-tutorial/

– By Tushar Sir

Database Testing using MySQL

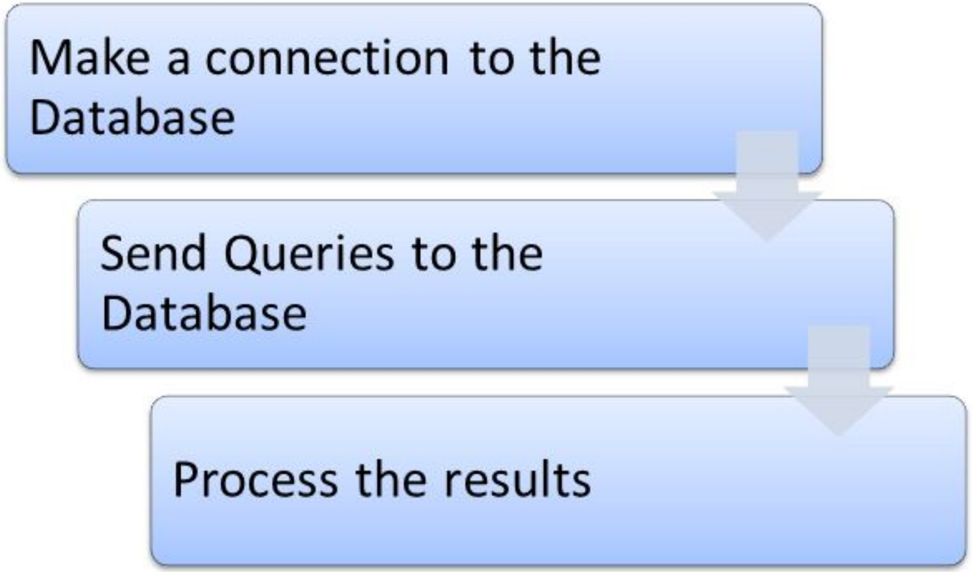
- Selenium Webdriver is limited to Testing your applications using Browser. To use Selenium Webdriver for Database Verification you need to use the JDBC (“Java Database Connectivity”).
- JDBC (Java Database Connectivity) is a SQL level API that allows you to execute SQL statements. It is responsible for the connectivity between the Java Programming language and a wide range of databases. The JDBC API provides the following classes and interfaces
- Driver Manager
- Driver
- Connection
- Statement
- ResultSet
- SQLException

– By Tushar Sir

Database Testing using MySQL

How to Connect Database in Selenium

- In order to test your Database using Selenium, you need to observe the following 3 steps


Make a connection to the Database
Send Queries to the Database
Process the results

- https://guru99.com/database-testing-using-selenium-step-by-step-guide.html

– By Tushar Sir

Database Testing Using DB2

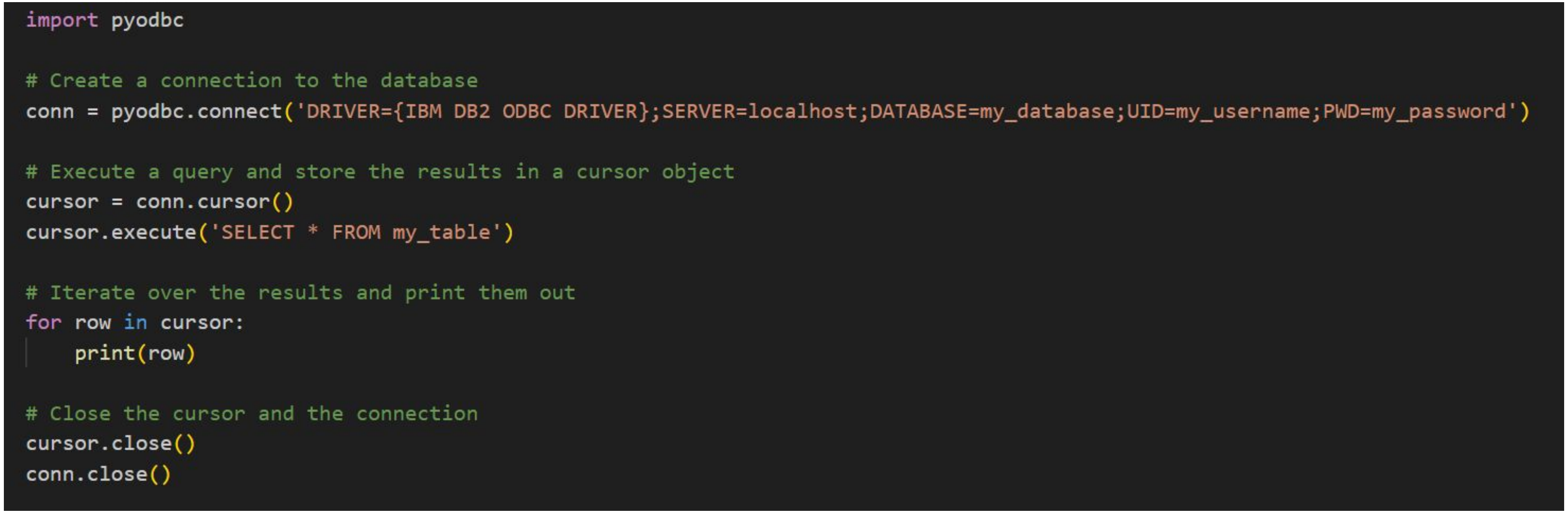
Here are the steps on how to perform database testing using DB2 in Selenium Python:

- Create a database in command prompt and insert the tables.
- Establish a connection to the database using JDBC.
- Execute the MySQL queries and process records present in the database.
- Integrate TestNG with JDBC to perform Database Testing.

– By Tushar Sir

Database Testing Using DB2

Here is an example of how to connect to a DB2 database using JDBC in Python:



```
import pyodbc

# Create a connection to the database
conn = pyodbc.connect('DRIVER={IBM DB2 ODBC DRIVER};SERVER=localhost;DATABASE=my_database;UID=my_username;PWD=my_password')

# Execute a query and store the results in a cursor object
cursor = conn.cursor()
cursor.execute('SELECT * FROM my_table')

# Iterate over the results and print them out
for row in cursor:
    print(row)

# Close the cursor and the connection
cursor.close()
conn.close()
```

You can then use this connection to execute other queries against the database, such as inserting, updating, and deleting data.

– By Tushar Sir

Ajax Call handling

- AJAX is a technique that allows a web page to retrieve small chunks of data from a web server in an asynchronous manner without having to reload the entire page every time.
- Using AJAX, you can create dynamic, fast, and highly interactive web pages based on asynchronous calls exchanged in the background.
- The use of this technique has grown substantially in recent years, some practical applications are as follows:
  - Chat and instant messaging
  - Autocomplete and autosuggest features
  - Instant login systems

– By Tushar Sir

Ajax Call handling

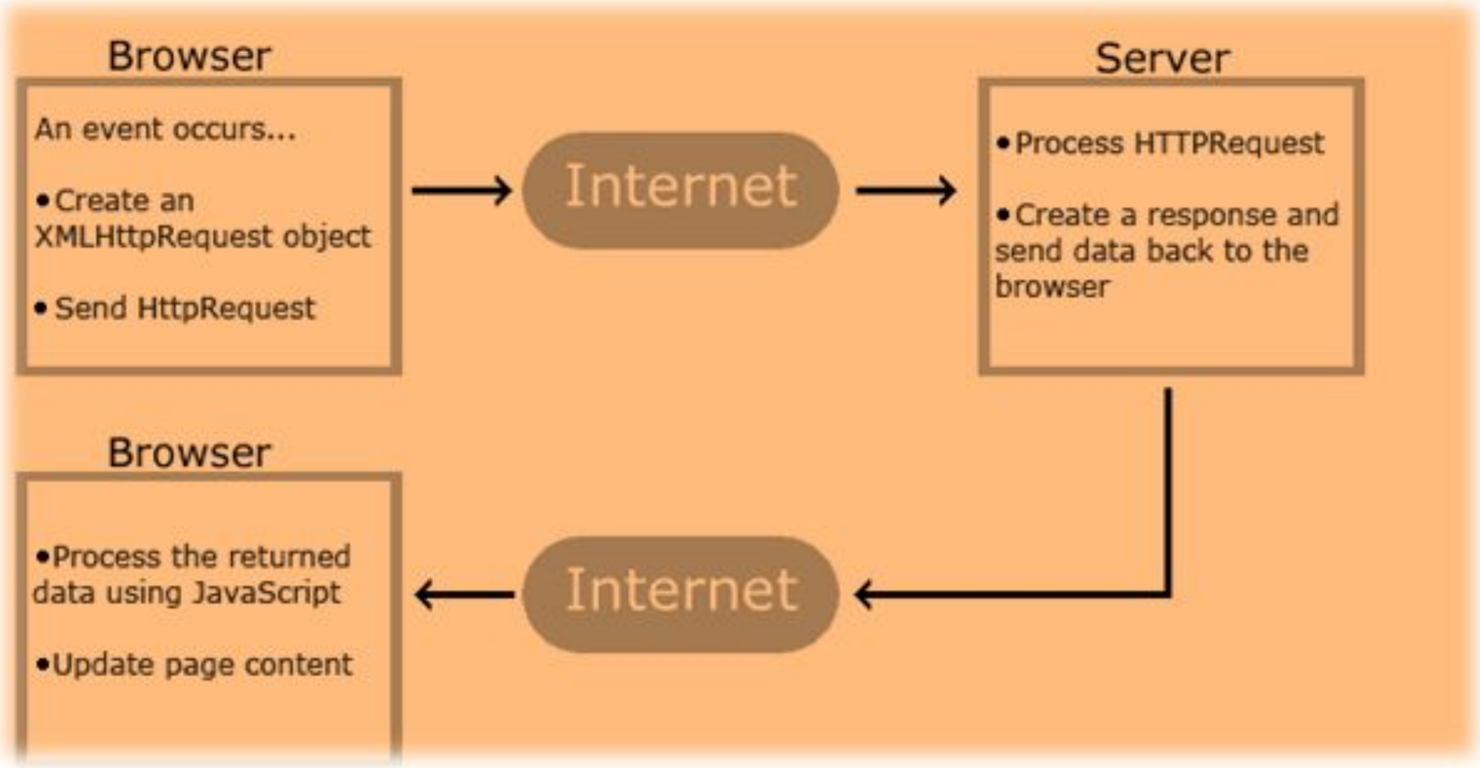
How Ajax Works?

- For example, when you click on submit button, JavaScript will make a request to the server, interpret the result and update the current screen without reloading the webpage.
- An Ajax call is an asynchronous request initiated by the browser that does not directly result in a page transition. It means, if you fire an Ajax request, the user can still work on the application while the request is waiting for a response.
- AJAX sends HTTP requests from the client to server and then process the server's response, without reloading the entire page. So when you make an AJAX call, you are not pretty sure about the time taken by the server to send you a response.

From a tester's point of view, if you are checking the content or the element to be displayed , you need to wait till you get the response. During AJAX call the data is stored in XML format and retrieved from the server.

– By Tushar Sir

Ajax Call handling


Browser
An event occurs...
•Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
•Process HTTPRequest
•Create a response and send data back to the browser
Internet
Browser
•Process the returned data using JavaScript
•Update page content

– By Tushar Sir

Ajax Call handling

- AJAX allows the Web page to retrieve small amounts of data from the server without reloading the entire page.
- To test Ajax application, different wait methods should be applied
  - ThreadSleep
  - Implicit Wait
  - Explicit Wait
  - WebdriverWait
  - Fluent Wait
- Creating automated test request may be difficult for testing tools as such AJAX application often use different encoding or serialization technique to submit POST data.

– By Tushar Sir

Ajax Call handling

How to Handle Ajax Calls in Selenium Webdriver?

- The biggest challenge in handling Ajax call is knowing the loading time for the web page.
- Since the loading of the web page will last only for a fraction of seconds, it is difficult for the tester to test such application through automation tool.
- For that, Selenium Webdriver has to use the wait method on this Ajax Call.
- So by executing this wait command, selenium will suspend the execution of current Test Case and wait for the expected or new value.
- When the new value or field appears, the suspended test cases will get executed by Selenium Webdriver.

– By Tushar Sir

Ajax Call handling

- Following are the wait methods that Selenium Webdriver can use

Thread.Sleep()

- Thread.Sleep () is not a wise choice as it suspends the current thread for the specified amount of time.
- In AJAX, you can never be sure about the exact wait time. So, your test will fail if the element won't show up within the wait time. Moreover, it increases the overhead because calling Thread.sleep(t) makes the current thread to be moved from the running queue to the waiting queue.
- After the time 't' reached, the current thread will move from the waiting queue to the ready queue, and then it takes some time to be picked by the CPU and be running.

Implicit Wait()

- This method tells webdriver to wait if the element is not available immediately, but this wait will be in place for the entire time the browser is open. So any search for the elements on the page could take the time the implicit wait is set for.

– By Tushar Sir

Ajax Call handling

Explicit Wait()

- Explicit wait is used to freeze the test execution till the time a particular condition is met or maximum time lapses.

WebdriverWait

- It can be used for any conditions. This can be achieved with WebDriverWait in combination with ExpectedCondition
- The best way to wait for an element dynamically is checking for the condition every second and continuing to the next command in the script as soon as the condition is met.
- But the problem with all these waits is, you have to mention the time out unit. What if the element is still not present within the time? So there is one more wait called Fluent wait.

– By Tushar Sir

Ajax Call handling

Fluent Wait

- This is an implementation of the Wait interface having its timeout and polling interval. Each FluentWait instance determines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition.

Challenges in Handling Ajax Call in Selenium Webdriver

- Using “pause” command for handling Ajax call is not completely reliable. Long pause time makes the test unacceptably slow and increases the Testing time. Instead, “waitforcondition” will be more helpful in testing Ajax applications.
- It is difficult to assess the risk associated with particular Ajax applications
- Given full freedom to developers to modify Ajax application makes the testing process challenging
- Creating automated test request may be difficult for testing tools as such AJAX application often use different encoding or serialization technique to submit POST data.

– By Tushar Sir

Listeners in Selenium

- In Selenium, listeners are classes that track events that happen during the execution of Selenium tests. They can be used to perform actions or log information when specific events occur.
- Listeners can listen to events such as:
  - Entering data
  - A button click
  - Navigation to a page
  - An exception
  - A test case starting or ending
  - A test step passing or failing

https://testsigma.com/blog/listeners-in-selenium/

– By Tushar Sir

Listeners in Selenium

- Listeners in Selenium are classes that implement specific interfaces in order to track events that occur during the execution of Selenium tests.
- You can use them to perform additional actions or log information when specific events happen, such as a test case starting or ending, or a test step passing or failing.
- Listeners are defined with the help of an interface in Selenium. They can customize system behavior by performing actions on events as defined in code.
- There are two types of Selenium listeners: **WebDriver Listeners, TestNG Listeners.**
- We use TestNG listeners to perform operations before and after a test method is executed. On the other hand, we use WebDriver listeners to listen to WebDriver events like navigating to a new URL and clicking an element.
- It’s worth noting that there are several built-in listeners which are provided by Selenium. In addition to this, developers can also create their own custom listeners by implementing the interfaces or classes and registering them to listen to the events.

– By Tushar Sir

Listeners in Selenium

Here are a few examples of how listeners can be used in Selenium:

- **Screenshot on failure:** You can use a listener to take a screenshot of the browser window when a test case fails. This can be useful for debugging and identifying the cause of the failure.
- **Logging:** You can also use them to log information about the test execution, such as the start and end time of each test case, and the test case status (pass/fail). This can be useful for generating detailed test execution reports.
- **Email notification:** They can be used to send an email notification when a test execution completes. This can be useful for alerting the development team of the test execution results.
- **Performance Metrics:** A listener can also be used to measure and record the execution time of each test method. It can be used to log test-wise or suite-wise performance metrics, which will be helpful in identifying performance bottlenecks.

– By Tushar Sir

JavaScript handling in Selenium

- Selenium webdriver can execute Javascript. After loading a page, you can execute any javascript you want. A webdriver must be installed for selenium to work.
- One of the features of Selenium is the ability to execute JavaScript. This can be useful for a variety of tasks, such as scrolling down a page, clicking on a hidden element, or getting the text of a element.
- To execute JavaScript in Selenium, you can use the **execute_script()** method. This method takes a JavaScript string as an argument and returns the result of the evaluation.
- For example, the following code will scroll down a page by 100 pixels:

```
driver.execute_script("window.scrollTo(0, 100)")
```

– By Tushar Sir

JavaScript handling in Selenium

- You can also use the execute_script() method to click on a hidden element. For example, the following code will click on an element with the id "my_hidden_element":

  driver.execute_script("document.getElementById('my_hidden_element').click()")

- The execute_script() method can be used to get the text of an element. For example, the following code will get the text of an element with the class "my_class":

  text = driver.execute_script("return document.getElementsByClassName('my_class')[0].textContent")

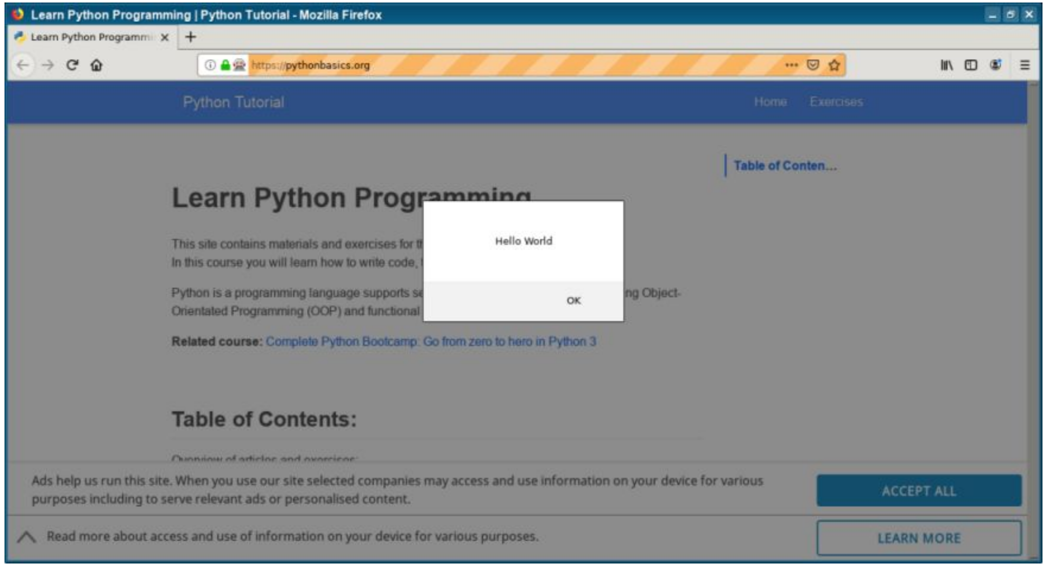
- The execute_script() method is a powerful tool that can be used to automate a variety of tasks in Selenium.

– By Tushar Sir

JavaScript handling in Selenium

Example: -

```
from selenium import webdriver
driver=webdriver.Firefox()
driver.implicitly_wait(3)
driver.get("https://pythonbasics.org")
js = 'alert("Hello World")'
driver.execute_script(js)
```



Learn Python Programming | Python Tutorial - Mozilla Firefox

Learn Python Programmi

https://pythonbasics.org

Python Tutorial Home Exercises

Table of Conten...

Learn Python Programming

This site contains materials and exercises for t
In this course you will learn how to write code,

Python is a programming language supports se ng Object-
Orientated Programming (OOP) and functional

Hello World

OK

Related course: Complete Python Bootcamp: Go from zero to hero in Python 3

Table of Contents:

Ads help us run this site. When you use our site selected companies may access and use information on your device for various purposes including to serve relevant ads or personalised content.

ACCEPT ALL

Read more about access and use of information on your device for various purposes.

LEARN MORE

https://pythonbasics.org/selenium-execute-javascript/#:~:text=Selenium%20webdriver%20can%20execute%20Javascript,js%20is%20your%20javascript%20code.

– By Tushar Sir

Thank You

- **By Tushar Sir**