Web Performance Engineering - Finding Best Data Structure For Automatically Collected HTML Locators

Mr. Prabhat Padhy

M. S. (BITS Pilani)
VP - Technology
Apmosys Technologies Pvt. Ltd.
Navi Mumbai, Maharashtra

Mrs. Pratima Nayak

B. E. Computer Engineering
Chief Technical Officer
Apmosys Technologies Pvt. Ltd.
Navi Mumbai, Maharashtra

Mr. Siddhesh Vaidya

M. E. Information Technology
Automation Test Engineer
Apmosys Technologies Pvt. Ltd.
Navi Mumbai, Maharashtra

Abstract: **The research basically aims to identify:**

I. **Finding the best Data Structure.**
II. **Automatic detection of locators.**

This in turns leads to track (navigate) for the URL and its content. Locators are defined as an address that identifies a web element uniquely within the webpage. It is a command that tells Selenium IDE which GUI elements like – Text Box, Buttons, Check Boxes etc., it needs to operate on. Finding correct GUI elements is a prerequisite for creating an automation script but, accurate identification of GUI elements is much more difficult than it sounds. Sometimes, you might even end up working with incorrect GUI elements or no elements at all! Hence, using the right locator ensures that the tests are faster, more reliable or has lower maintenance over releases.

Keywords— Locators, Selenium IDE

I. INTRODUCTION

Locators are said as an address (path towards intended subject) that identifies a web element uniquely (unsimilar to any other) within the webpage. It is a command that instructs Selenium IDE which GUI elements like – Text Box, Buttons, Check Boxes etc, it needs to work on. Searching appropriate GUI elements is a prior understanding or knowledge needed for creating an automation script but, extremely accurate identification of GUI elements is very much more difficult than it sounds. Sometimes, you might even end up working with wrong GUI elements or no elements at all! Therefore, utilizing the correct locator ensures that the tests are speeder, more reliable or has lower maintenance over releases.

If you are very sure enough to use unique IDs and Classes, then you are usually ready. But there will be some moments when choosing the right locator will become a nightmare because of the complexity of finding the web elements in the webpage.

Selenium is still the most sought-after framework when it comes to web automation testing. Though frameworks like Cypress, Puppeteer, etc., are playing the catch-up game, Selenium still rules the automation testing framework charts. As per my experience, every QA who wants to dabble with Selenium should have a sensible understanding of locators in Selenium WebDriver.

Playwright is an freely available resource, cross-browser automation framework that provides end to end testing without the need for any third-party tools. Selenium locators can be treated as the building block of any type of Selenium automation test script.
The reason is so simple behind this is that – locators in Selenium WebDriver helps you in performing the requisite interactions with the elements in the Document Object Model. Selecting the best locators in Selenium WebDriver is one of the hint to ensure that the tests are less brittle – a factor that leads to the unsuccess of Selenium automation tests.[4] [5]

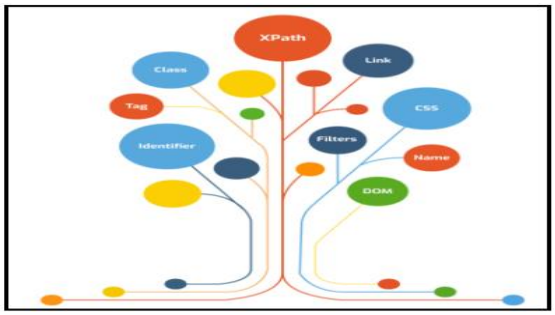

Fig 1: Types of Locators

II. KNOW ABOUT DATA STRUCTURE

A data structure is a specialized format for organizing, processing, retrieving, and storing data. There are various basic and most advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures makes it easy for users to access and work with the data they need in appropriate ways. Most importantly, data structures frames the organization of information so that machines and humans can better understand it.

In computer science and computer programming, a data structure may be chosen or designed to store data for the purpose of using it with various algorithms. In some cases, the algorithm's basic operations are tightly coupled to the data structure's design. Each data structure contains information about the data values, relationships between the data and -- in some cases -- functions that can be applied to the data.[2][3]

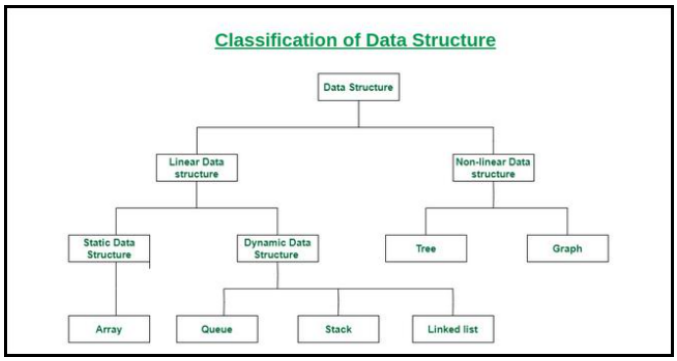

Fig 2: Classification of Data Structure

III. NEED OF DATA STRUCTURE

The structure of the data and the synthesis of the algorithm are relative to each other. Data presentation must be easy to understand so the developer, as well as the user, can make an efficient implementation of the operation. Data structures provide an easy way of organizing, retrieving, managing, and storing data. Here is a list of the needs for data.

1. Data structure modification is easy.
2. It requires less time.
3. Save storage memory space.
4. Data representation is easy.
5. Easy access to the large database.

IV. HOW TO LOCATE WEB ELEMENT IN DOM? [7]

As we are aware that, the primary thing to initiate with is to search for the HTML element in DOM (Document Object Model), for which we need to grab the locator. We can perform the below steps to recognize the web element in DOM on a web browser:

1. **Document Object Model** can be accessed in Google Chrome either by pressing **F12** or by **right click** on the web page and then by selecting **Inspect**

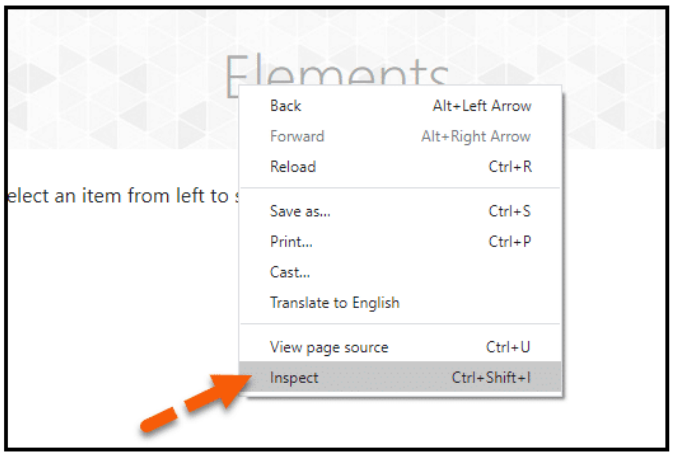

Fig 3: Way to inspect element

2. It will open the Developer Tools console, once we click on the "Inspect option". By default, it will open the "Elements" tab, which represents the complete Document Object Model structure of the web page. Now, if we hover the mouse pointer over the HTML tags in the DOM, it will highlight the corresponding elements it represents on the webpage.

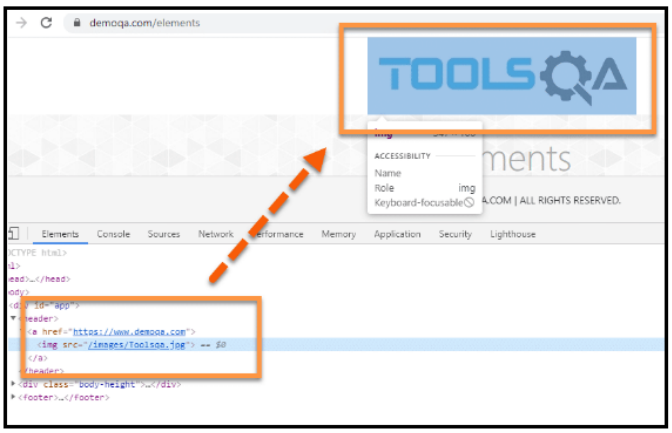

Fig 4: Selection of a element

3. Now, the major concern is, how do we find the web element in the DOM. Click on the "Mouse Icon" arrow and then select the web element on the web page. It will directly highlight the corresponding HTML element in the Document Object Model. Suppose we want to search for the HTML elements corresponding to the banner image (as shown below by marker 3). When we select the mouse point and click on the banner image, it will automatically highlight the corresponding HTML element, as shown by marker 2, in the below given screenshot.

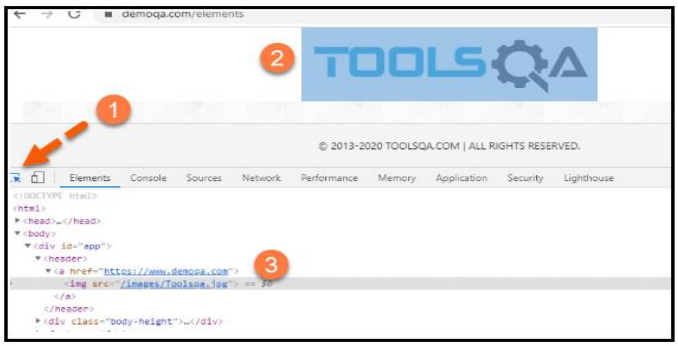

Fig 5: Hovering of element using arrow

V. WHAT LOCATORS ARE SUPPORTED BY SELENIUM?

Selenium supports the following locators:

- **ClassName** – A ClassName operator uses a class attribute to identify an object.
- **cssSelector** – CSS is used to create style rules for webpages and can be used to identify any web element.
- **Id** – Similar to class, we can also identify elements by using the ‘id’ attribute.
- **linkText** – Text used in hyperlinks can also locate element
- **name** – Name attribute can also identify an element

- **partialLinkText** – Part of the text in the link can also identify an element
- **tagName** – We can also use a tag to locate elements
- **xpath** – Xpath is the language used to query the XML document. The same can uniquely identify the web element on any page.

VI. HOW TO LOCATE WEB ELEMENT USING ATTRIBUTE ID? [1]

"ID" as a locator is one of the most regular ways of searching for elements on a web page. An ID for a web element always needs to be unique (should not match with others). The ID is one of the fastest and unique ways of locating web elements and is considered as one of the most trustable methods for determining an element. But with the advancement in technologies and more of the dynamic web pages, "IDs" are generated dynamically and generally not the reliable way to locate a web element, as they change for different users.

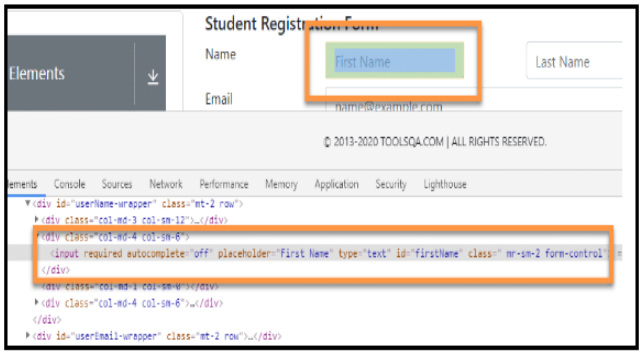

Fig 6: Hovering of element using ID

Method:

<input required="" autocomplete="off" placeholder="First Name" type="text" id="firstName" class=" mr-sm-2 form-control">

As we can see, the HTML tag contains the attribute "**id**" inside the input tag. The **id** here used is the "**firstName***" which we can use to locate this element in the web page. Now, to find the "**First Name**" text box on the web page, we can use the following syntax:

VII. HOW TO LOCATE WEB ELEMENT USING NAME?

Like the id attribute, Selenium also offers users another way to find the element using the name attribute. But like an id, a web page can have multiple elements with the same "name" attribute. So, if we are going to use the name attribute for the identification of a web element, we should always make sure that the name attribute must contain a unique value. Otherwise, it may identify several different elements on the same page, which may have the same name value. So, the bottom line is name attribute value should be also unique.

If more than one elements have the same value of the 'name' attribute, then, Selenium will just select the first values in the page which matches the search criteria. So, we always recommend making sure that the value of the name attribute should be unique when we select it for locating a web element.

Fig 7: Locating element using Name

<input name="gender" required="" type="radio" id="gender-radio-1" class="custom-control-input" value="Male">
We can see the attribute '*name* ' with value as '*gender*'. We can use the following syntax to locate the web element using the "**By**" class.

By.name ("gender");

VIII. How to locate a web element by using the "ClassName" attribute?

ClassName allows SeleniumSelenium to find web elements based on the **class** values of the DOM. For example, if we want to identify or perform any operation on the form element, we can use the class to identify it.

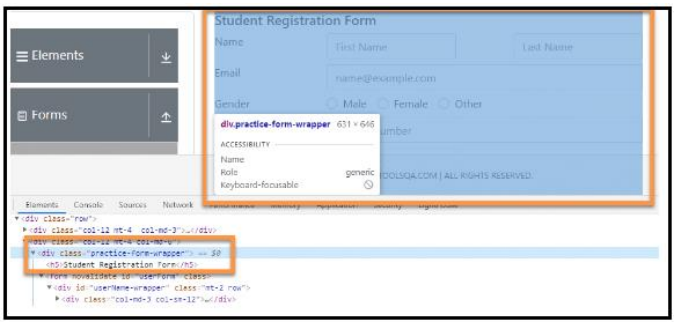

Fig 8: Locating element using ClassName

IX. PROBLEM STATEMENT

The main aim for the application is to automatically fetch the data from the Locators and store data in object repository and verify which of the Data Structure best suited for the particular research from the Data Structures like Tree, Linked List, and Graph.

X. ALGORITM FOR AUTOMATIC DETECTION OF WEB ELEMENT LOCATORS [6]

Step 1: SET DRIVER.GET = URL
Step 2: TRAVERSE URL
FOREACH (WebElementlink: variables)
Step 3: RECEIVE DATA
Step 4: Repeat Step 1 and 2 for receiving data and iterating
Step 5: EXIT

XI. ALGORITHM FOR TRAVERSING & INSERTING DATA USING SINGLY LINKED LIST, TREE & GRAPH

Traversing: We can traverse the entire Linked List using a single pointer variable called ***START***. The ***START*** nodes contains the address of the first node in turns stores the address of succeeding node. Using this technique, the individual nodes of the list will form a chain of nodes. If START = NULL, this means that the Linked List is empty and contains no nodes.

ALGORITHM FOR TRAVERSING A LINKED LIST

Step 1: [INITIALIZE] SET PTR = START
Step 2: Repeat Steps 3 and 4 while PTR != NULL
Step 3: Apply Process to PTR->DATA
Step 4: SET PTR = PTR->NEXT [END OF LOOP]
Step 5: EXIT

ALGORITHM FOR INSERTING DATA IN A LINKED LIST

Step 1: IF AVAIL = NULL, then
Write OVERFLOW
Go to Step 7
[END OF IF]
Step 2: SET New_Node = AVAIL
Step 3: SET AVAIL = AVAIL->NEXT
Step 4: SET New_Node->DATA = VAL
Step 5: SET New_Node->Next = START
Step 6: SET START = New_Node

Step 7: EXIT

ALGORITHM FOR BINARY SEARCH TREE

Step 1: IF TREE->DATA = VAL OR TREE = NULL, then
Return TREE
ELSE
IF VAL < TREE->DATA
Return searchElelement(TREE->LEFT, VAL)
ELSE
Return searchElelement(TREE->RIGHT, VAL)
[END OF IF]
[END OF IF]

Step 2: End

XII. COMPARATIVE STUDY AS PER DATA STRUCTURE

Parameters	Linked List	Binary Search Tree	Graph
Definition	It is a node-based data structure which consists of information and address towards other nodes	It is hierarchical data structure in which nodes are available in parent and child manner	It is a closed loop data structure
Time Complexity	Time complexity of Linked List is O(n)	Time complexity of Tree is O(log n)	Time complexity of Tree is O(V+E)
Space Complexity	Space complexity of Tree is O(n)	Space complexity of Tree is O(n)	Space complexity of Graph is O(V2) O(V^{2})O(V2)
Feasibility	Complex to find and extract the locators	Simple as compared to Linked List	No proper data gets retrieved

Table 1: Comparison of Data Structures as per parameters

XIII.EXPERIMENTAL RESULTS

Sr. No.	URL	Linked List	Tree	Graph
		Time Taken (in msec.)		
1	www.iciciprulife.com	18298	11235	12665
2	www.msbte.org.in	47423	3153	25326
3	www.saraswatbank.com	4593	4236	14238

Table 2: Comparison in terms of Milliseconds

XIV.GRAPHICAL REPRESENTATION OF THE STUDY

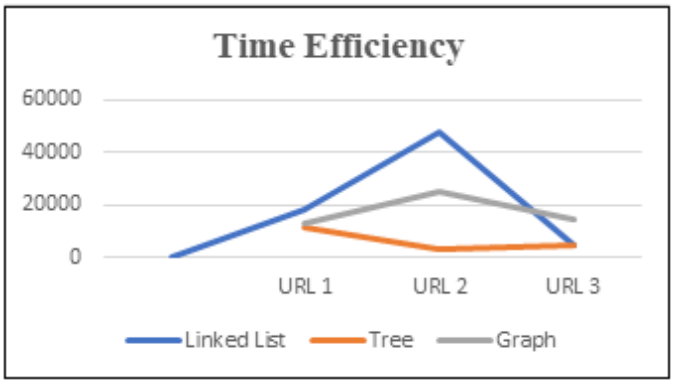

Fig 9: Graphical Representation in terms of efficiency

XV. RESULTS OF IMPLEMENTATION

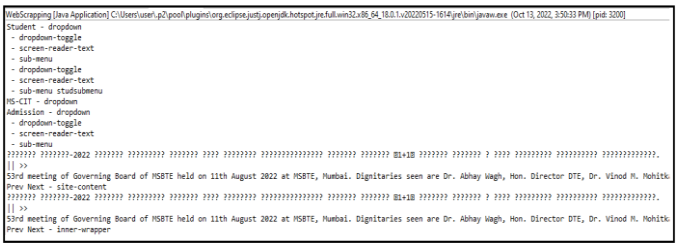

Fig 10: Web elements value fetched from the supplied URL

Fig 11: Various attributes getting fetched

```
Time to add data in linked list icici pru is : 18298 msec
Time to add data in linked list msbte  is : 47423 msec
Details fetched successfully
```

Fig 12: Displaying time required to store data in Data Structure

XVI. CONCLUSION

The outcome of the current implementation leads to tell that it follows Tree Data Structure. The results are getting fetched out in a non-linear way i.e. the architecture goes very much in deep to search for all the elements by exploring each and every URL within the link.

XVII. REFERENCES

[1] https://www.browserstack.com/guide/locators-in-selenium
[2]https://www.geeksforgeeks.org/what-is-data-structure-types-classifications-and-applications/
[3]https://www.techtarget.com/searchdatamanagement/definition/data-structure
[4] https://www.edureka.co/blog/locators-in-selenium/
[5] https://en.wikipedia.org/wiki/Selenium_(software)
[6] Data Structures Using C, Reema Thareja, Second Edition
[7] https://www.toolsqa.com/selenium-webdriver/selenium-locators/

XVIII. GLOSSARY

1. URL 1: www.iciciprulife.com
2. URL 2: www.msbte.org.in
3. URL 3: www.saraswatbank.com