Selenium

Selenium is a versatile software testing framework used to automate web browsers, enabling developers and testers to automate website testing across different browsers and platforms. It supports various programming languages and integrates with testing frameworks, offering flexibility and extensibility

Advantages of Selenium

1. Open Source and Free to Use

- Selenium is an open-source tool, meaning its source code is available for anyone to use, modify, and distribute.
- This makes it a cost-effective solution for organizations of all sizes, as it eliminates licensing fees.

2. Cross-Browser and Cross-Platform Compatibility testing

- Selenium supports testing on a wide range of browsers, including Chrome, Firefox, Safari, Edge, and more.
- It also works across different operating systems, such as Windows, Linux, macOS, and more.

3. Language Support

- Selenium provides bindings for various programming languages, including Java, Python, C#, Ruby, PHP, and more.
- This allows developers to choose the language they're most comfortable with for writing test scripts.

4. Community Support and Resources:

- Selenium has a large and active community of users and developers.
- This provides access to a wealth of resources, including documentation, tutorials, and support forums.

Disadvantages of Selenium

1. Focus on Web Application Testing:

- Selenium primarily focuses on testing web applications, meaning it's not suitable for testing desktop or mobile applications directly.
- For testing mobile applications, one would need to use tools like Appium, according to BrowserStack.

2. Limited Built-in Features:

- Reporting and Debugging: Selenium lacks comprehensive built-in reporting and debugging capabilities. It doesn't automatically generate detailed reports with screenshots or logs, and you might need to integrate third-party tools for these features.
- Image Comparison: Selenium does not have built-in image comparison functionality.
- Automatic Waiting: Selenium doesn't automatically handle waiting for elements to load or become available, requiring explicit code for waiting times.

3.Selenium does not support dynamically changing objects

Components of Selenium

Selenium having 4 components

1, Selenium Integrated Development Environment (IDE)

- This is a browser extension for Chrome and Firefox that allows users to record and playback interactions with web pages. It's a great tool for quickly creating simple test cases and getting familiar with Selenium.

2, Selenium Remote Control (RC)

- While no longer the primary focus, Selenium RC was the original Selenium tool that enabled remote browser automation. It used a server to communicate with browsers and was a predecessor to Selenium WebDriver. Kind of record and playback tool and have a feature of changing test data

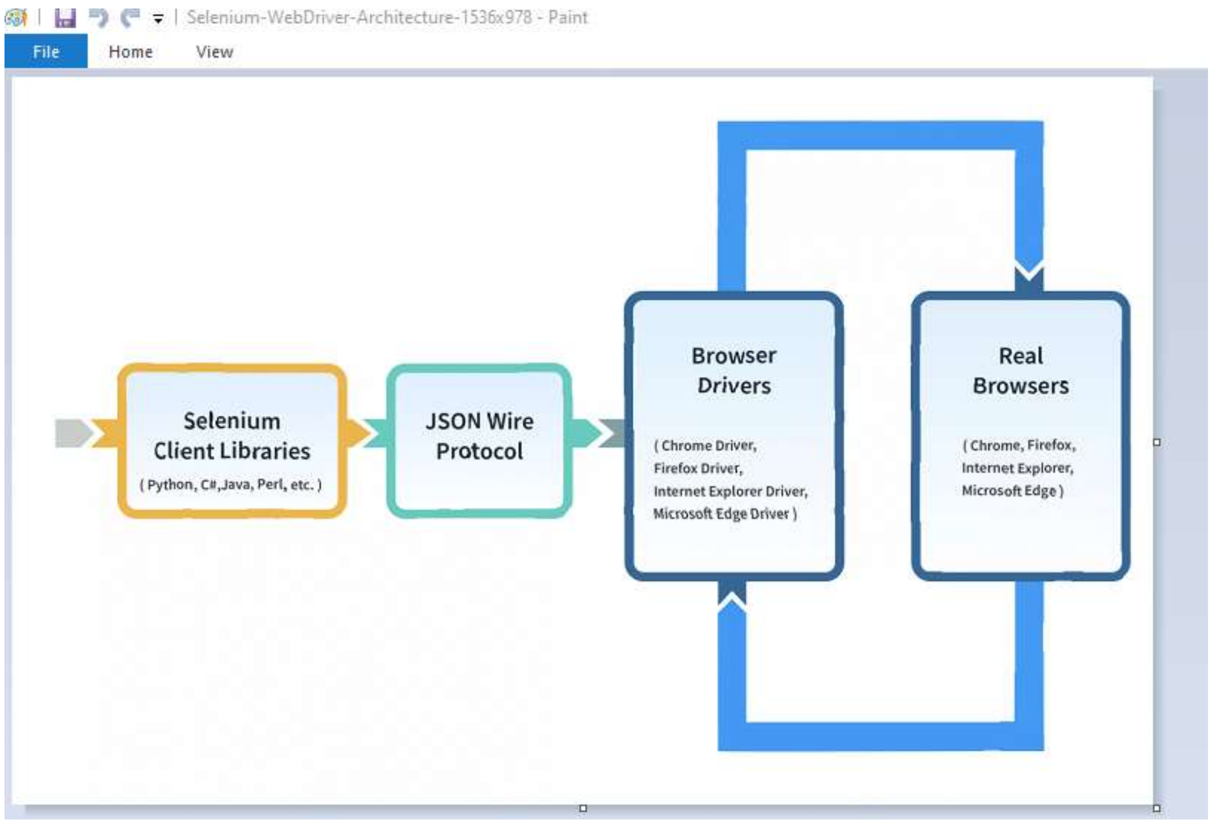
3, Selenium Web Driver

This is the core component of Selenium, providing a programming interface for directly controlling web browsers. It allows developers to write automated tests in various programming languages, such as Java, Python, and JavaScript.

4, Selenium Grid

Additional tool used for parallel execution for connection establishment in cloud. Selenium Grid is a tool that enables parallel test execution across multiple browsers and machines. This can significantly speed up the testing process and provide broader coverage.

Architecture Of Selenium


Selenium-WebDriver-Architecture-1536x978 - Paint
File
Home
View
Selenium Client Libraries
(Python, C#,Java, Perl, etc.)
JSON Wire Protocol
Browser Drivers
(Chrome Driver, Firefox Driver, Internet Explorer Driver, Microsoft Edge Driver)
Real Browsers
(Chrome, Firefox, Internet Explorer, Microsoft Edge)

Selenium Client Library: This component provides language-specific bindings or APIs (e.g., Java, Python, Ruby) that allow users to write test scripts and interact with the WebDriver.

WebDriver W3C Protocol: WebDriver is a protocol that provides a standard way for web browsers to communicate with an automation script. In Selenium 4, it focuses on W3C WebDriver Protocol, for better consistency and compatibility across different browsers.

Browser Drivers: These are executable files that establish a communication channel between the WebDriver and the actual web browsers such as Chrome, Firefox, Safari, etc. Each browser requires its specific driver (e.g., ChromeDriver, GeckoDriver, etc.) to enable WebDriver to control and automate browser actions.

Real Browsers: These are web browsers like Google Chrome, Mozilla Firefox, Microsoft Edge, etc., where the actual testing and automation take place. The WebDriver interacts with these browsers through their respective browser drivers to perform actions like clicking elements, filling forms, navigating pages, and validating content.

Initializing the browser

Create an object of chrome driver for open the chrome browser.

Code: driver=new ChromeDriver(); driver.get("Link");

Screen Maximize code:

To maximizing the opened chrome browser Code:

Driver.manage().window().maximize();

Browser closing command

Two commands are usually using in selenium

1. Closing

2. Quit

In Selenium WebDriver, driver.close() closes the currently focused browser window, and driver.quit() closes all browser windows opened by the driver and ends the WebDriver session. If the currently focused window is the only one open, driver.close() also quits the driver.

driver.close();

driver.quit();

Browser Commands

1.To get title To get the title of the current page.

Code: String title=driver.getTitle(); And print the title.

2.To get URL To get the URL of the current page.

Code: String url=driver.getCurrentUrl(); And print the URL.

3.Window handle To get windows handleid.

Code: String handleid=driver.getWindowHandle(); And print the handleid.

4. To get source code To get source code of the current page.

Code: String source=driver.getPageSource(); it print the source code of the page

Navigation Commands

The navigation commands are used to navigating actions of to, Back, Forward and refresh.

1. To
   Navigate to a specific URL.
   driver.navigate().to("Link");
2. Back
   Navigate to go back previous page.
   driver.navigate().back();
3. Forward
   Navigate to go forward last open page.
   Code: driver.navigate().forward();
4. Refresh
   To refresh the current page.
   Code: driver.navigate().refresh();