What is automation?

Automation is a process of making a machine do the work that was previously being done by humans.

Benefits:

- Avoids human error
- Product quality is maintained
- Consistent growth in output
- Can work beyond human capacity
- For example, track renewal train

Even though there are lot of benefits of automation, we should not leave the ground-level skills. Automation can work only upto certain limits, it won't work under unexpected conditions, i.e., automation is useful only in performing repetitive tasks like track laying, renewal, scanning for microscopic cracks in the tracks, etc. It won't work when there is any major mishap on the railway line. Also, automation is susceptible to failure that may lead to a complete halt of all the work.

Automation is the creation and application of technologies to produce and deliver goods and services with minimal human intervention. The implementation of automation technologies, techniques and processes improve the efficiency, reliability, and/or speed of many tasks that were previously performed by humans.

Automation testing

Automation testing refers to the automated testing of a software, in which a developer or a tester creates test scripts for different business processes once with the help of testing tools and programming languages and runs it on the software.

The test script automatically tests the software without human intervention and shows the result. Various test cases are prepared by developers or testers against which various corresponding test cases are mapped, as per the business processes and various functionalities of the software.

Hence, if errors or exceptions occur in the execution of the automation code, then issues are reported, else the software has passed that test.

Usage of automation testing:

1) Performance Testing: Many tools are present, JMeter, LoadRunner, BeesWithGuns, etc.
2) Security Testing: Many tools are present, Invicti, Klocwork, SonarQube, etc.
3) Unit Testing
    a) Many tools are present, JUnit, NUnit, TestNG, PHPUnit, Jasmine, etc.
    b) We use JUnit and TestNG, basically TestNG is an upgraded version of JUnit.
    c) The only thing that JUnit does not support is conditional testing, which is supported by TestNG, i.e., testing of certain functionality in a software is based on satisfaction of certain condition.
4) Functional testing
    a) It is recommended to use manual testing only, except for validations.
    b) Here, Selenium can be used for finding out input validation related issues in the software under test.

Validations:

- Text field
    - Single-line fields
    - Multi-line fields
    - Alphabets
    - Numbers
    - Special characters
    - E-mail address
    - Maximum and minimum length
- Numeric field
    - 0 inputs
    - Positive numbers
    - Negative numbers
    - Floating-point numbers
    - Maximum and minimum length
- File uploads
    - File type
    - File size
- Mandatory and optional fields

Automation testing is a process of automating the testing activities that a tester does manually while testing a software.

These activities can be filling up of lengthy forms, finding out validation issues in the forms, verifying whether the forms have been correctly submitted by checking the success messages as well as record ID's, etc.

Out of many automation tools, we are going to discuss one here in detail, which is called Selenium, that helps the testers in performing the above tasks in an extremely effective manner, i.e., the testers can cover every functionality of the software under test.

What is Selenium?

It is a tool that works on web applications, i.e., the ones that are deployed on web servers and can be accessed through web browsers.

So, basically Selenium helps the testers in automating what they do manually, i.e., filling up of online forms, checking if the forms are correctly submitted, etc. It is especially helpful in repetitive tasks like regression testing or re-testing.

So, through this we come to know that Selenium works only on the front-end of the web application under test. Hence, using Selenium we can perform only Black-box testing.

Features

- Open-source Web UI automation testing suite.
- Originally developed by Jason Huggins in 2004 as an internal tool at Thought Works.

- Platform independent:
    - OS: Windows, Mac, Android, etc.
    - Browser: Chrome, Firefox, Edge, etc.
    - Language: Java, Python, etc.

Components of Selenium Tool Suite

All the functionalities of Selenium are collectively called Selenium Tool Suite. There are a total of four components of Selenium Tool Suite, using which we can automate a web application.

1) Selenium IDE
2) Selenium Grid
3) Selenium RC (Selenium 1)
4) Selenium WebDriver (Selenium 2)

Selenium IDE

There are only two parts in Selenium IDE, record and playback. Here, we record our activities on the web application under test, i.e., form filling and all the navigation activities, save them and play them as and when required.

Selenium IDE is designed to record your interactions with websites to help you generate and maintain site automation, tests, and remove the need to manually step through repetitive takes. Features include:

- Recording and playing back tests on Firefox, Chrome and Edge.
- Organizing tests into suites for easy management.
- Saving and loading scripts, for later playback.

Selenium extension installation steps for Google Chrome. For now, use only Google Chrome for the following exercise. Firefox and Edge can also be used for Selenium IDE.

Exercise:

Navigation: Go to cdac.in-->Products & Services-->High Performance, Grid and Cloud Computing-->More menu

- Automate CDAC website and read the output Selenium code.
- Use the document “Procedure to install Selenium IDE and create recording”.
- Refer the section “Procedure to install Selenium IDE” for installation.
- Refer the section “Steps to record a project in Selenium IDE” for creating a recording.
- Refer the section “Steps to get Selenium code” to get a Java file containing the automation code for the navigations done by you on CDAC website.

Selenium Grid is a smart proxy server that makes it easy to run tests in parallel on multiple machines. This is done by routing commands to remote web browser instances, where one server acts as the hub. This hub routes test commands that are in JSON format to multiple registered Grid nodes.

Selenium RC is an important component in the Selenium test suite. It is a testing framework that enables a QA or a developer to write test cases in any programming language to automate UI tests for web applications against any HTTP website.

Selenium IDE is a good tool for beginners, because you get to know how automation testing works. But there are certain limitations to this.

Limitations:

- So, the code you saw in Selenium IDE exported file, is readable but very difficult to understand and is also not maintainable.
  - For example, if the developer who wrote this code left the company, the newer developer will be facing a next to impossible task of understanding this code.
  - Also, many classes are imported which are never used in the code.
- Not suitable for regression testing, the same recording cannot test the chat feature of WhatsApp after the release of voice call feature.
- Cannot use multiple sets of test data, if we are testing login functionality of an online portal, we cannot use multiple credentials at a time.
- No support for test reports, no support for formal reports on which functionality is working correctly and which is not working correctly.

So, as IT professionals we use Selenium WebDriver.

Selenium WebDriver

Selenium web driver is a programmatic tool to automate a web application.

It provides a full set of methods, fields, constants, sub-classes, etc. to open any web browser, access any URL on that browser, minimize or maximize the window, make it wait for certain event or for a certain amount of time, fill an online form using text fields, file uploads, drop-down menus, etc.

- Interacts with webpage GUI elements exactly the way humans use them.
- Support for multiple programming languages:
  - Java
  - JavaScript
  - C#
  - Python
  - Ruby
- Support for multiple web browsers:
  - Google Chrome
  - Mozilla Firefox
  - Microsoft Edge
  - Opera
  - Safari
  - Internet Explorer
- Built-in method names are easy to understand and implement in your automation testing code.
- The object/reference variable of WebDriver interface in package "org.openqa.selenium" represents the current instance of the browser window in use, along with the URL of the web portal under test.
- Can handle almost any type of GUI elements that are as follows:
  - Single and multi-line text fields

  - Button clicks
  - File uploads
  - Hyperlinks
  - Single and multi-select drop-down menus
  - Many more

Opening a webpage in Selenium

driver.get() method is used to open or navigate to a webpage through the URL parameter value and will wait until the page is fully loaded before returning the execution control to the next part of automation code.

Other useful methods in WebDriver

- String getCurrentUrl()
  - Read the fully qualified URL that is currently open in the browser window that is encapsulated by the current instance of WebDriver.
  - Useful when we are testing a web application that is deployed on multiple URL's or when we need to identify that our system under test is on which webpage in the system.
- String getTitle(): Read and display the title of the currently open webpage.
- void close(): Close the current window of the browser.
- void quit(): Close all the open windows of the browser.
- driver.manage().window().maximize(): Maximizes the window to full screen size.
- driver.manage().timeouts().implicitlyWait(10): Will make the Selenium code to stop the execution and wait for 10 seconds to display the current webpage.

So, only opening the web browser and a website or webpage is not enough. We need to perform more operations like form submission, login, registration, application form filling, etc. host of the tasks we need to do in a web portal.

How to access GUI elements on webpage?

Interface WebElement

A reference variable of this interface represents a GUI element on a webpage, be it text field, button, radio button, etc. All the interactions that need to be done to fill up any online form will be done through the methods declared in this interface.

Example: WebElement we_txtFirstName;

Method "findElement()"

There are many GUI elements present on any webpage, this method will help the tester to traverse to the required GUI element on the webpage.

we_txtFirstName=driver.findElement(By.method(“”));

Class "By"

The findElement() method accesses the GUI element with the help of various Selenium locators. These Selenium locators are defined as static methods in this class.

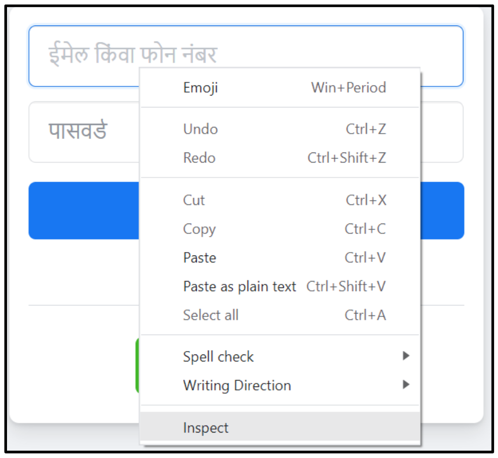
Example: The above <method> of “By” class can be replaced by the following static methods, which are sorted in the order of preference based on their reliability to find any element on a webpage.

- By.id(“String”)
  - The value of attribute with exactly same name is passed as argument, which is already present in the HTML webpage.
- By.name(“String”)
  - The value of attribute with exactly same name is passed as argument, which is already present in the HTML webpage.
- By.className(“String”)
  - The value of attribute "class" is passed as argument, which is already present in the HTML webpage.
- By.cssSelector(“String”)
  - 5 sub-types are present in this, with varying combinations of class, tag, attributes and their values.
- By.linkText(“String”)
  - Full visible text of hyperlink is used as argument.
- By.partialLinkText(“String”)
  - Partially visible text of hyperlink is used as argument.
- By.xpath(“String”)

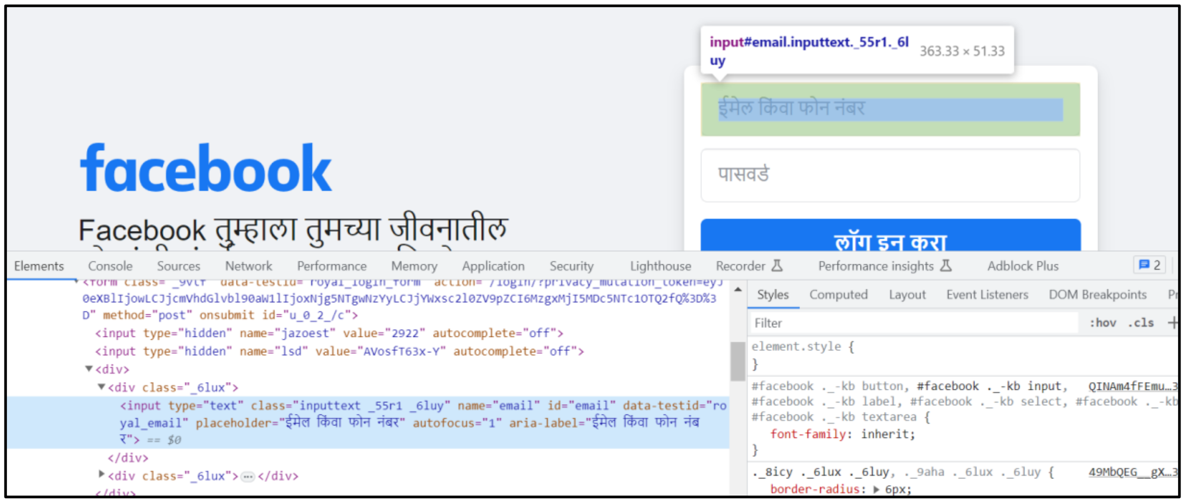
Selenium locators

These are the attributes of the GUI elements in the webpage, written in HTML. The values of these attributes can be passed as arguments to the corresponding methods.

Example 1



Step 1



```
<form class="_9vtf" data-testid="royal_login_form" action="/login/?privacy_mutation_token=eyJ
0eXBlIjowLCJjcmVhdGlvbl90aW1lIjoxNjg5NTgwNzYyLCJjYWxsc2l0ZV9pZCI6MzgxMjI5MDc5NTc1OTQ2fQ%3D%3
D" method="post" onsubmit id="u_0_2_/c">
  <input type="hidden" name="jazoest" value="2922" autocomplete="off">
  <input type="hidden" name="lsd" value="AVosfT63x-Y" autocomplete="off">
 ▼<div>
   ▼<div class="_6lux">
      <input type="text" class="inputtext _55r1 _6luy" name="email" id="email" data-testid="ro
      yal_email" placeholder="ईमेल किंवा फोन नंबर" autofocus="1" aria-label="ईमेल किंवा फोन नंब
      र"> == $0
    </div>
   ▶<div class="_6lux">…</div>
```

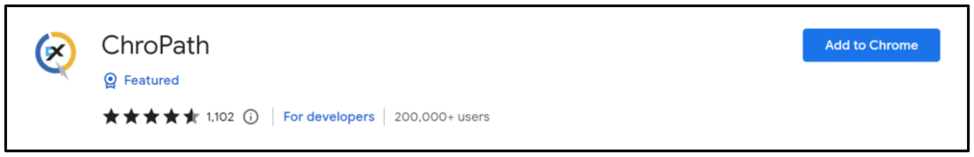
```
element.style {
}
#facebook ._-kb button, #facebook ._-kb input,
#facebook ._-kb label, #facebook ._-kb select, #facebook ._-kb
#facebook ._-kb textarea {
  font-family: inherit;
}
._8icy ._6lux ._6luy, ._9aha ._6lux ._6luy {
  border-radius: ▶ 6px;
```

Step 2

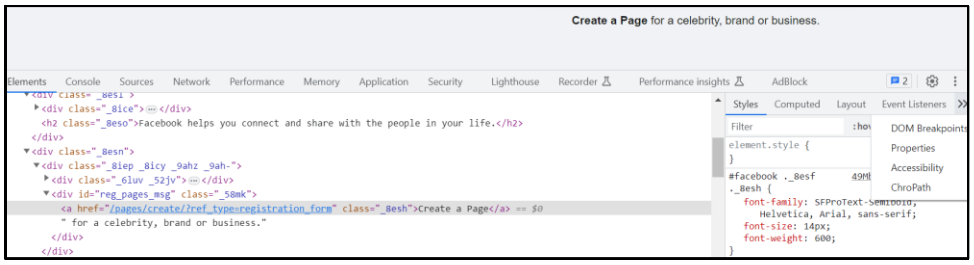
Explanation:

- driver.findElement(By.id(“email”));
- driver.findElement(By.name(“email”));
- CSS (Total 5 types are available, but the most effective is tag and attribute)
    - **driver.findElement(By.cssSelector(“input[data-testid=’royal_email’]”));**
    - driver.findElement(By.className(“inputtext _55r1 _6luy”));
    - driver.findElement(By.cssSelector(“input#email”));
    - driver.findElement(By.cssSelector(“input.inputtext _55r1 _6luy”));

Example 2



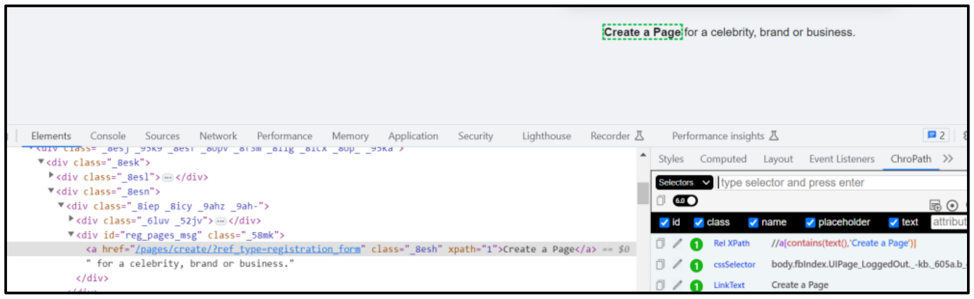
Step 1



```
▼<div class="_8esl">
  ▶<div class="_8ice">…</div>
   <h2 class="_8eso">Facebook helps you connect and share with the people in your life.</h2>
 </div>
▼<div class="_8esn">
  ▼<div class="_8iep _8icy _9ahz _9ah-">
    ▶<div class="_6luv _52jv">…</div>
    ▼<div id="reg_pages_msg" class="_58mk">
       <a href="/pages/create/?ref_type=registration_form" class="_8esh">Create a Page</a> == $0
       " for a celebrity, brand or business."
     </div>
   </div>
```

```
element.style {
}
#facebook ._8esf
._8esh {
  font-family: SFProText-Semibold,
    Helvetica, Arial, sans-serif;
  font-size: 14px;
  font-weight: 600;
}
```

Step 2



Create a Page for a celebrity, brand or business.

```
<div class="_8esk">
  <div class="_8esl">…</div>
  <div class="_8esn">
    <div class="_8iep _8icy _9ahz _9ah-">
      <div class="_6luv _52jv">…</div>
      <div id="reg_pages_msg" class="_58mk">
        <a href="/pages/create/?ref_type=registration_form" class="_8esh" xpath="1">Create a Page</a> == $0
        " for a celebrity, brand or business."
      </div>
```

Rel XPath	//a[contains(text(),'Create a Page')]
cssSelector	body.fbIndex.UIPage_LoggedOut._-kb._605a.b_
LinkText	Create a Page

Step 3

Explanation:

- driver.findElement(By.linkText(“Create a Page”));
- driver.findElement(By.partialLinkText(“Create a”));
- driver.findElement(By.xpath(“//a[contains(text(), ‘Create a Page’)]”));
    - Xpath value starting with “//” is relative xpath, which is quite easy to read and understand.
    - Xpath value starting with “/” is absolute xpath, which is very difficult to track and understand.

So, only accessing and identifying the elements is not enough, we need to perform certain operations like filling information in the text fields, clicking button, radio buttons, checkboxes, etc.

How to use the GUI elements on the webpage?

- WebElement interface click() method will be used to click on the web element found by the preceding findElement method. The click method can be used on the following GUI elements on a webpage.
    - Hyperlink
    - Button
    - Radio button
    - Checkbox
- WebElement interface sendKeys() method will be used to send text and file data from local system to the GUI elements in the online form.
    - Text field
    - File upload
- Select class is for encapsulating drop-down elements and selecting or de-selecting any value in the drop-down menu. It works for single-select and multi-select drop-down menus.
    - deselectAll(): Clear all selected entries.
    - deselectByIndex (int index): Deselect the option at the given index.
    - deselectByValue (String value): Deselect all options that have a value matching the argument.
    - deselectByVisibleText (java.lang.String text): Deselect all options that display text matching the argument.
    - List<WebElement> getAllSelectedOptions()
    - WebElement getFirstSelectedOption()

    - List<WebElement> getOptions()
    - selectByIndex (int index): Select the option at the given index.
    - selectByValue (java.lang.String value): Select all options that have a value matching the argument.
    - selectByVisibleText (java.lang.String text): Select all options that display text matching the argument.
- WebElement interface clear() method will clear the contents of any GUI element of an online form. It can be used on the following web elements.
    - Text input
    - Drop-down menu
    - File upload

Example

- Loads the Google search page
- Enters the search term
- Hits Enter
- Results are displayed

Keys class: It is the class which encapsulates almost all the keys present in the keyboard. It is present in "org.openqa.selenium" package.

Exercise: Automate the Facebook login page, as per the demo given in Selenium Locators section. Use the document "Procedure to create Maven project in Eclipse and add dependencies" for this.

This concludes preliminary Selenium training, for academic level. So, we have only automated the various operations that are required to be done in an application under test.

How to segregate our automation code into test cases/scripts?

This is done by TestNG for us. TestNG is a framework that we can use to divide our complex and large Selenium automation code into different test scripts that will test different modules and different webpages in the system under test. This is done with the help of annotations.

TestNG Annotations: A TestNG Annotation is a piece of code which is inserted inside a program or business logic used to control the flow of execution of test script code.

Primitive annotations:

- @BeforeSuite: The @BeforeSuite annotated method will run before the execution of all the test methods in the suite.
- @BeforeTest: The @BeforeTest annotated method will be executed before the execution of all the test methods of available classes belonging to that folder.
- @BeforeClass: The @BeforeClass annotated method will be executed before the first method of the current class is invoked.
- @BeforeMethod: The @BeforeMethod annotated method will be executed before each test method will run.

- @AfterMethod: The @AfterMethod annotated method will run after the execution of each test method.
- @AfterClass: The @AfterClass annotated method will be invoked after the execution of all the test methods of the current class.
- @AfterTest: The @AfterTest annotated method will be executed after the execution of all the test methods of available classes belonging to that folder.
- @AfterSuite: The @AfterSuite annotated method will run after the execution of all the test methods in the suite.

Features

@Test annotation

- This is the main method of TestNG from where the test execution starts. Any method that is defined under this annotation will be called as one test script mapped with a test case written by a tester, as discussed by my colleague.
- **Priority attribute:** In simple words, which method or which code needs to be executed first can be decided using this attribute of @Test annotation. Values of this attribute can be 0, 1, 2, etc., where “0” will be the highest priority and the method or code marked with 0 priority will be executed first.
- **dataProvider attribute:** This will connect the @DataProvider annotation with @Test annotation, by accepting the name of the data provider as value.

@DataProvider annotation

- Used to provide data from external sources like Excel files, CSV files, various DB’s like Oracle, MySQL, etc. to the test script code and using that test data. This is helpful especially when we need to test the system repetitively with different test data. For example, to test file size and file type validations.
- Hence, the input to data provider will be any data source and the output of data provider will be a 2D object array which will have the data in tabular format.
- Name attribute is mandatory as this will help the test annotation to identify the location from where test data is to be accepted as input.
- Parallel attribute will initiate multiple threads at a time to execute the test scripts using the data from data provider.
- Multi-threaded execution of our program can happen on the discretion of the OS.

Reports

- **testng-results.xml file:** As the name suggests, this is an XML file that will have the execution status of each of the methods or classes under all the TestNG annotations. If there is any exception in the execution of any code, the status will be set as failed, else if the code executed successfully, the status is set as failed. This file can be provided as input to any other test reporting tool or any other tool like Jenkins, so that reports can be displayed in graphical formats or can be sent into the mailbox.
- **emailable-report.html:** This file will have the number of tests that are present in the project, i.e., methods or classes under the annotation “@Test”. Out of the total number of tests, how many

were passed and how many were failed, is also present in the report, along with the names of the method, class and package. It will also provide the time required to execute the code in milliseconds. If data provider is used, details of each parameter are also provided.

Example

- Declare the required variables for WebDriver and GUI elements.
- Open the browser and URL in a new method.
- Enter the Selenium code in a new method.
- Data provider method will be defined separately.

Exercise

- Create a new class LoginTestNG in package testNGprograms.
- Automate the same login process into Facebook.
- Just replace the main() method by @Test annotation.

Lifecycle management

Maven is a project lifecycle management tool, that automates and manages the various phases in the project lifecycle. It binds together all the required input and output files, along with the code files of the project.

It is a basic framework using which we can create our own project structure for all the source code files, supporting files like jars, Excel and CSV files as data stores, reports for formally displaying results of testing, etc.

For example, Maven handles the Web driver executable files by downloading them on the fly, so you don't have to worry about packaging the supporting files along with your project code files when you have to deliver your project to the client. Maven will handle not only this but many more for you.

What is web driver manager?

A maven dependency that will handle the selenium web driver executable files for each web browser. These executable files are the interface between Selenium and each web browser.