POORNIMA

Experiment - 8

Perform various Commands

Types of Selenium Commands

Selenium Commands are basically classified in 3 Categories.

① Actions

② Accessors

Actions are the Selenium Commands that generally Manipulate the state of the applications. Execution of Actions generates events like click this link, select that option, type this box, etc. if an action fails, or has a bug the execution of current test is stopped.

★ Open(url) → it launches the desired URL in the specific browser and it accepts both relative and absolute URLs'

★ Types(locator, value) → It sets the value of an input field, similar to user typing actions.

★ Store(expression, variableName) ⇒ This Command

Specifies the name of a variable in which the result is to be stored and expression is the value to store.

2) Accessors :- Accessors are the Selenium commands that examine the state of the application and store the result in variables. They are also used to automatically generate Assertions.

StoreTitle (Variable Name) → This command gets the title of the current Page.

StoreText (locator, variable ~~Vari~~ Name) → The Command gets the text of an element.

Store Body Text (Variable Name) → This Command gets the entire text of the Page.

Store All Button (VariableName) → This Command gets the entire text of the Page.

StoreAllFeilds (Variable Name) → it retrieve the ID's of all input field on the Page.

Store All Links (Variable Name) → it returns the IDs of all ~~the~~ links on the page.

POORNIMA

~~A~~ Test Case Manually:
we will generate Test script for search a Test operation on any publically available Search engine google.

- Launch firefox browser
- Click on the selenium icon present on the top right corner ~~at~~ on your browser.
- ~~q~~ It will launch the default interface of Selenium IDE.
- Enter the project name ~~as~~ as "Manual Test".
- Enter the test case name as "Search Test".

- Click on the Command text box present on the Test script Editor Box.
- Modify the properties of first Command as:
- Command : open
- Target : https://www.google.co.in.

- During execution of the test case, this command will load the google search engine web page on your firefox browser.

- Now, we have to add a Command that will click on the google search engine text box. for this we need a unique identification element of the text box which will ~~keep~~ help the IDE to identify the target location.

→ The Method for ~~the~~ finding a unique identification element involves inspection of HTML Codes.

- open URL : https://www.google.co.in your firefox Browser.
- Right click on the google search text box and select inspect element.
- It will launch a window containing ~~@~~ all the specific Codes involved in the development of the text box.
- Pick the input tag element that contains an ID name for the text box.

- Modify the properties of Second Command as:
- Command : click at
- Target : id = lst-ib

- during execution of the test case, this command will click on the search text box present on the google search engine webpage.

→ Command : type
Target : id = lst-ib
Value : amazon

POORNIMA

During execution of the Test Case, the Command will type the specified text on the google Search Text box.

- Right click on the google search button and Select element inspect.
- It will laugh a window Containing all the specific codes involved in the development of the ~~spary~~ search button.
- Pick the name element that Contains the specified name for the google search button.
- Modify the properties of fourth Command Command or
- Command: Click at
- Target: name = btnG
- During execution of the test Case, the Command will click on the search button present on the google search engine web page.
- Click on the "Run Current Test" button present on the tool bar Menu of the IDE. It will execute all of your Inserted Commands on the browser and gives you an overall summary.
- The log name deploys the overall Summary of the executed test scripts.

POORNIMA

Experiment - 9

* Introduction to JIRA

-> please get JIRA software free Cloud Service in the link by Signup from our email Id.

https://www.atlassian.com/software/jira/free.

Step 1 : Create a scrum project.
once you create and login to an account in jira software, you can select a template from the library select scrum next, you'll be prompted to choose a project ~~of~~ type and Choose team-Managed scrum template once ~~yo~~ you've created your project you will land ~~on~~ on the empty backlog. The backlog is also known as the product backlog and contains items for the project.

Step 2 : Create user stories or tasks in the backlog Create a few user stories with the quick create option on the backlog, from user perspective. As a {type of user}, I want {goal} so that I {receive benefit}.

POORNIMA

Step 3 : Create a Sprint
Create your first sprint in the backlog so you can start planning the sprint. Add story point estimates to your stories by adding a number in the story point estimate field.

Step 4 — Start the Sprint in Jira
Name the sprint Add a duration of the Sprint and start and end dates once you Start your sprint, you will be taken to the Active sprints tab in the project. This is where your team will work to pick up items from to-do column and move them into in-progress and eventually done. This will be called a board.

Step 5 View and Burndown Chart.
It's a good idea to check the Burndown chart during a ~~o~~ sprint. In Jira Software, the Burndown chart shows the actual and estimated amount of work to be done in a sprint. The Burndown chart is automatically updated by Jira as you complete work items. To view this chart.

POORNIMA

Step 6 — View the sprint report

→ The sprint report includes the ~~BO~~ Burndown Chart, and lists the work completed, work not completed, and any work added after the sprint started.

Step 7. Complete the sprint in Jira

At the end of the sprint, you must complete it.

POORNIMA

Experiment -10

★ Bug Tracking project in Jira.

① Setup a bug Tracking project in Jira
In Jira to create a defect would be to create an issue of the type "Bug."

② Defect ~~to~~ reporting needs the following information recorded for every issue.

★ Defect ID
★ Defect title
★ Defect description (steps to reproduce)
★ Environment information
★ screenshot attachment)
★ Security.
★ Assign ~~to~~ it to someone
★ Status- All the Status in the bug life cycle.

③ Defect Life-Cycle:
All bug life cycle status as in BugZilla (or any other popular bug Tracker) can be accomplished here too.

④ Comments and Collaboration with the Dev team.

POORNIMA

(5) Linking the defect to a requirement to enable traceability.

(6) Defect can be imported from a CSV file.

(7) defects can be exported into word, XML and printable formats.

(8) Comprehensive Issue Reports.

[illegible]