Accessing to Remote Cloud host :-

* GridClient gclient = new GridClient(
username, Password, ProjectName, url, Port));

* Client client = gclient.LockDeviceForExecution
("deviceName", "sessionName", 10, 50000);

Connecting to local HOST :-

* Initialize client :-

Client client = new Client("localhost", Port);

Common Use :-

client.launch("<This is application url>", true, true);

1st true :- Specifies whether to instrument application.
* Instrumentation allows Selenium to interact with application, enabling features like identification & interaction.

2nd true :- Specifies whether to stop application if it is already running before launching it again.

username = your seetest cloud username

Password = seetest cloud Password

Project = The project name in seetest cloud.

host = The Cloudhost url (https://www.cloud.seetest.io)

Port = The portnumber (usually 443 for HTTPS)

devicename => The name or ID of device you want to lock

sessionName => A name for session.

timeout => The maximum time to wait for device to be available. in seconds.

releasetimeout => The time after which device will be released automatically (in ms).

* localhost ip address ("localhost" or "127.0.0.1")
* Port (The portNo on seetest is running) (default 8889).

Visit Before Page :-

Page No.: YOUVA
Date:

Mobile Device setup :-

Mobile

* Enabling developer option in Android
  go to settings ⇒ developer option ⇒ click on
  ☑ ① Stay awake
  ☑ ② USB debugging

⇒ Connect to mobile device to laptop.
  cmd adb devices
  ⇒ Once some o/p not coming developer option problem

Tool Download & Seetest TABS / Views :-

http://experitest.com/download/seetest automation

Connecting Mobile device :
  in Seetest
Application Manager :- It is a tool to application of Test & integrate Seetest & web object using seetest update feature file.

Recording Test : goto Test tab click on record
  Choose device & application name.

Script editing :
  ↓ in record Test we are checking.

⇒ * wait for Element Vanish :-

boolean b = client.waitForElementToVanish("Name", xpath, 0, 5000);
  ↳ return false, that particular time reached element is display mean.

zone = Native
element = xPath

index = 0;
Property = "text"

M T W T F S S
Page No.:
Date:
YOUVA

Methods in SeeTest :-

* client.click("NATIVE", xPath, 0, 1);
(native, web, Text, Coordinate → Element representation; index; Click)

checks for presence of an Element.

* boolean b = client.isElementFound("NATIVE", xPath, 0);
(index)

Verify the presence of an Element & throws an exception if not found.

client.verifyElementFound("NATIVE", xPath, 0);
↳ index

⇒ Get value of an element's Property.

* String text = client.elementGetProperty("NATIVE", xPath, 0, "text");
(Property)

* sends text to a textbox without clicking on it.

client.elementSendText("NATIVE", xPath, 0, "admin");
(Text to Send)

* get value of element count.

⇒ client.getElementCount("NATIVE", xPath);

* This method throws an exception if desired element is found.

client.verifyElementNotFound("NATIVE", xPath, 0);
(index)

* Wait for Element :-

boolean b = client.waitForElement("NATIVE", xPath, 0, 10000);
(timeout)

↓
No thrown exception
return true or false

* click :-

Zone :- Default zone is selected for identifying objects in 'default' object repository.

Text zone is specified, Selenium Automation Searches for the text specified for element (2nd argument) in the UI & Click on it.

* isElementFound() :- ⇒ Doesn't throw exception, false return
* verifyElementFound() :- Throws exception when not found.
* elementGetProperty() :- Throws exception
* elementSendText() ⇒ "
* verifyElementNotFound() ⇒ "
* waitForElement() ⇒ return true or false } NOT THR exception
* waitForElementToVanish() ⇒ return boolean
* getElementCount() ⇒ Throws Exception

drag :-

X = 0 ⇒ drag vertically.

Y = 0 ⇒ drag horizontally

Zone: "NATIVE"

M	T	W	T	F	S	S

Page No.: | Date: | YOUVA

Advanced Commands :-

* **click on object at specified offset :-**

client.click(zone, xpath, 0, 1, 300, 500);

(0 → index, 1 → No. of clicks, 300 → x coordinate, 500 → y coordinate; click locations)

* **click on specified coordinates :-**

* client.clickCoordinate (300, 500, 1);

(300, 500 → x, y; 1 → No. of clicks)

* **click inside an element :-**

client.clickIn("NATIVE", xpath, 0, "inside", "Text", xpath, 0, 0, 0, 1);

(xpath → parent; 0 → index; "inside" → type of element to be clicked inside parent element; "Text" → identified by the help of; xpath → child; searching starts at ... width)

"Text", "NATIVE", "ID", "XPATH", "web"

inside, left, right, up, down

(zone, xpath, index, direction, clickelementZone, childxpath, index, width of search, height of search, clickCount);

(zone → Type of element; clickelementZone → check type of element inside parent)

* **Drag :-**

client.drag ("Native", "text=Login" (or) xpath, 0, 0, 100);

(xpath → element representation; 0 → index; 0 → x offset drag; 100 → y offset drag.)

* **Element List Select :-** select an element from list.

client.elementListSelect ("id=abc", "text=USA", 0, true);

(locator for list, locator of element to select, index, whether to click or not)

* ElementSetProperty :-

* client.elementSetProperty("NATIVE", "text=LOGIN", 0, "id", "loginButton");

(zone, xPath, index, propertyName, value to set that prop

ElementSwipe :- Performs a swipe on an element.

client.elementSwipe("NATIVE", "text=LOGIN", 0, "RIGHT", 300, 300);

(Native, xPath, index, direction, swipe offset, swipe time

* VerifyIn :- element

verify presence of an element inside an anoth

⇒ client.verifyIn(zone, "text=LOGIN", 0, "inside", zone, "LOGIN", index = 0);

GetCoordinateColor :-

Gets RGB colour value at coordinate x,y cordi

· client.getCoordinateColor(300, 500);

DragDrop :-

x, y Cordina

click.dragDrop("NATIVE", "text=LOGIN", 0, 100, 200);

LongClick :-

client.longClick("NATIVE", "text=LOGIN", 0, 2000);

Duration of click in millisec..

M T W T F S S

Page No.: YOUVA

Date:

Methods :-

* client.releaseClient();

⇒ sendText for element ⇒ client.elementSendText(, ,);

⇒ " " " " &keyboard " ⇒ " .sendText(text);

⇒ Device action :-

client.deviceAction (key);

⇒ Swipe while element Not found in screen :-

* client.SwipeWhileNotFound(direction, 500, 2000, zone, ele, 0, 1000, 5, false);

("Down", pixels, time, zone, &path, @index, wait 1 sec, 5 times, false)

↓ Not click on element found.

⇒ Swipe down 500 pixels over 2 secs

⇒ Search name element with text = "LOGIN"

⇒ wait for 1 sec after each swipe to check if element is found.

⇒ perform upto 5 swipes.

⇒ Not click on the element if it is found.

move swipe by pixels pixels for 2 sec.

⇒ client.swipe("direction", 600, 2000);

offset → difference b/w pixels, how far the swipe will move on screen.

→ swipe down inside dropDown to find element.

⇒ client.elementSwipeWhileNotFound("NATIVE", object, direction, 300, 2000, "Text", element, 0, 1000, 15, true);

↓ ↓ isSwipes

1 min wait element is None or Check.

⇒ to get innerHTML of specific element.

client.element.getProperty(zone, xpath, 0, "innerHTML");

⇒ To get Text ⇒

client.element.getText(zone, xpath, i);

↓

Particular index Text will be found.

⇒ selected element value :-

* client.elementGetProperty(zone, xPath, 0, "selected");

⇒ element is checked or not :

client.elementGetProperty(zone, xPath, 0, "checked");

⇒ getCurrentBrowserTabid :-

* String c = client.getCurrentBrowserTabId();
List<String> h = h.getBrowserTabList();
for (String abc : h) {
client.switchToBrowserTab(abc);

⇒ To count the rows in table :-

client.getElementCount("NATIVE", xPath

Android phone

M T W T F S S
Page No.:
Date: YOUVA

* Device related Commands :-

* Seetest automation commands that are related to actions performed on mobile device itself.

① sendText(" "); ⇒ ({ })

⇒ {HOME} ⇒ pressing home button
⇒ {BKSP} ⇒ press back space
⇒ {PORTRAIT} ⇒ change orientation to Portrait Mode
⇒ {LANDSCAPE} ⇒ change " to landscape "
⇒ {ENTER} ⇒ simulate operation of Enter button
⇒ {WAKE} ⇒ Wake screen
⇒ {UNLOCK} ⇒ unlock device
⇒ {CLOSEKEYBOARD} ⇒ close opened keyboard

⇒ ② Swipe Performs device level not an object
There is no Name, xPath.

③ Flick. (Swipe more fast)
client.flick("DOWN", 400);

④ CloseKeyBoard ⇒ client.closeKeyboard();
e.

⑤ Reboot ⇒ client.reboot(1000);
(0) ⇒ instant reboot.

⑥ GetVisualDump ⇒ returns xml, html into a string.
⑦ pinch ⑧ shake ⇒ both are operations
⑨ Run ⇒ to run adb commands
⑩ P2CX ⇒ returns the given 100 % ⇒ maximum width device in pixels.
⑪ P2CY ⇒ 100% ⇒ maximum height of device in pixels.

M T W T F S S
Page No.:
Date:
YOUVA

Launching WorkJam Application :-

ios, cloudhost :- cloud host-device url

* GridClient grid = new Grid Client (accesskey, url of);

client cl = grid.lockDeviceForExecution("___", "@os='ios' and category='PHONE'", 10, 50000);

cl.setReporter("xml", "___", "Quickstart");

ios, localhost :-

Client cl = new Client("localhost", port, true);
cl.setDevice(devicename);
cl.setReporter("XML", " ", "workjam")
client.launch("com.workjam.___.uat", true, false).

⇒ access key for Authentication.

lockDevice(1st arg * deviceName ⇒ A descriptive name for session.

2nd arg ⇒ * query ⇒ To find the device

3rd arg ⇒ * time out · 4th arg ⇒ release Timeout.

args meth. Lock IOS Device for execution with specified Query.

⇒ SetReporter ⇒ sets the Reporter for test execution.

1st arg ⇒ XML ⇒ Type of Reporter
directory ⇒ The directory to save the report
testName ⇒ The name of the Test.

For Android :- changes @os='android' for cloudhost

localhost ⇒ client.launch("url changes", true, false)

→ Both are called

Gridclient, client :-

* Both used Context of Mobile automation.
* scope: client for local device interaction, while Grid Client is for cloud based interaction.
* Device Management: Grid client includes for reserving & Managing devices in Cloud environment which client does not.

Purpose: The Client class is used to interact directly with a mobile device (or) emulator. It provide methods (or) actions on device, such as launching apps, interacting with UI elements, & Capturing Screenshots.

Purpose: Gridclient :- It used to interact with a cloud based device grid. It allows reserve devices from the cloud, execute tests on them & release them after use.

NATIVE :- → we mentioned means that elements are Native elements.
Native applications built specifically for (android/ios).
* To Access all features (CAM, GPS) : high Performance

WEB :- It accessed through web browser built using Technologies like HTML, CSS
* Easier to update & maintain, No need installation like Native

Default :- It vary based on Context, it generally refer to Standard option provided by system or application. Ex Mobile default software.
like : default browser for Android chrome

Text :- It refers to text based elements or content on app or website
* includes label, Buttons, & other Textual Contents.