Appium

iOS 9.3 and above: Apple's XCUITest

iOS 9.3 and lower: Apple's UIAutomation

Android 4.3+: Google's UiAutomator/UiAutomator2

Windows: Microsoft's WinAppDriver

Client/Server Architecture

Appium is at its heart a webserver that exposes a REST API. It receives connections from a client, listens for commands, executes those commands on a mobile device, and responds with an HTTP response

Session

Clients initiate a session with a server in ways specific to each library, but they all end up sending a POST /session request to the server, with a JSON object called the 'desired capabilities' object

Desired Capabilities

Desired capabilities are a set of keys and values (i.e., a map or hash) sent to the Appium server to tell the server what kind of automation session we're interested in starting up

Appium 2.0 Core New Features

Drivers’ Independence – ability to install and work with de-coupled web drivers based on your application platform (iOS, Android, Windows OS, Flutter, etc.)

- Usage e.g.: “appium driver install xcuitest”

Plugin Ecosystem – Ability to modify the Appium framework by adding new features via plugins and through integrations with other technologies.

- Usage e.g., “appium plugin install images”, then activate the plugin with your server via:

- “appium --use-plugins=images”

W3C Compatibility – Similarly to Selenium 4 being fully W3C compliant, with Appium 2.0 you should also ensure you’re sending W3C standard capabilities within your test code (see below e.g.).

- “ {platformName” :”iOS”,

“appium:deviceName”: “iPhone 12”,

“appium.automationName”: “XCUITest”,

}

The current Appium Desktop tool is still available but is now being released in two separate packages:

Appium desktop (Server) - https://github.com/appium/appium-desktop

Appium element inspector - https://github.com/appium/appium-inspector

Appium Server

Appium is a server written in Node.js.

$ npm install -g appium

$ appium

Appium Clients

There are client libraries (in Java, Ruby, Python, PHP, JavaScript, and C#) which support Appium's extensions to the WebDriver protocol.

Verifying the Installation

npm install -g appium-doctor

appium-doctor

UIAutomation Driver for iOS < 9.3

XCUITest setup: iOS 9.3 or higher, Appium 1.6

XCUITest is mediated by the WebDriverAgent server. leverages Apple's XCUITest libraries under the hood

"automationName": "XCUITest", "platformName": "iOS"

“browserName” capability to “Safari”

“app” ipa path

“bundleId” already installed app

“udid” device udid

Real Device Setup:

Log paths:

$HOME/Library/Logs/CoreSimulator/\*

$HOME/Library/Developer/Xcode/DerivedData/\*

Automatic config:

{

"xcodeOrgId": "<Team ID>",

"xcodeSigningId": "iPhone Developer"

}

OR

“xcodeConfigFile” capability

In .xcconfig

DEVELOPMENT\_TEAM = <Team ID>

CODE\_SIGN\_IDENTITY = iPhone Developer

Manual Config:

Find Appium path

$ which appium

/path/where/installed/bin/Appium

Go to webdriveragent folder

/path/where/installed/lib/node\_modules/appium/node\_modules/appium-webdriveragent

mkdir -p Resources/WebDriverAgent.bundle

Open WebDriverAgent.xcodeproj in xcode and "Automatically manage signing" for WebDriverAgentLib and WebDriverAgentRunner targets

Change app id if required

(Please make sure the provisioning profile has iOS Distribution type)

(Only for Appium versions below 1.15.0) Appium handles installing the application to the device, using **ideviceinstaller** (which is installed as part of **libimobiledevice**), but it is sometimes easier to pre-install your app using Xcode to ensure there are no problems

Make sure the following settings are enabled on your device:

Settings -> Developer -> Enable UI Automation

Settings -> Safari -> Advanced -> Web Inspector and Remote Automation for web apps

Make sure the keyboard preference in the device under test is Apple official one and the input language is set to English to send texts to XCUIElementTypeSecureTextField

Install idb for better handling of various iOS Simulator operations, such as: biometrics, geolocation setting and window focusing

UiAutomator: older

UiAutomator2: Android SDK Build Tools version is 24, Java 8

Need to define JAVA\_HOME and ANDROID\_HOME

Developer mode should be ON

Starting Appium:

appium -p 8081

driver.getStatus() //get server’s status

Execute mobile commands:

driver.executeScript("mobile: scroll", ImmutableMap.of("direction", "down"));

UIAutomator2 Commands

<https://github.com/appium/appium-uiautomator2-driver#platform-specific-extensions>

Espresso commands

<https://github.com/appium/appium-espresso-driver#platform-specific-extensions>

XCUITest commands

<https://github.com/appium/appium-xcuitest-driver#platform-specific-extensions>

Create driver session:

DesiredCapabilities caps = new DesiredCapabilities();

caps.setCapability(MobileCapabilityType.PLATFORM\_NAME, "Android");

caps.setCapability(MobileCapabilityType.PLATFORM\_VERSION, "12");

caps.setCapability(MobileCapabilityType.UDID, "1dbb7a42");

caps.setCapability("appium:appPackage", "com.google.android.calculator");

caps.setCapability("appium:appActivity", "com.android.calculator2.Calculator");

caps.setCapability("appium:automationName", "UiAutomator2");

URL url = new URL("http://127.0.0.1:4723/wd/hub");

AppiumDriver driver = new AppiumDriver(url, desiredCapabilities);

String sessionId = driver.getSessionId().toString();

End Session:

driver.quit();

Get session capabilities:

Map<String, Object> caps = driver.getSessionDetails();

Navigate Back: (Web context only)

driver.back();

Take ScreenShot:

File scrFile = ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(srcFile, DestFile);

Get the current application hierarchy XML (app) or page source (web)

String pageSource = driver.getPageSource();

driver.manage().timeouts().pageLoadTimeout(30, TimeUnit.SECONDS);

Set the amount of time the driver should wait when searching for elements

driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS); // Duration.ofSeconds(30) for appium2

Set the amount of time, in milliseconds, that asynchronous scripts executed by execute async are permitted to run before they are aborted (Web context only)

driver.manage().timeouts().setScriptTimeout(30, TimeUnit.SECONDS);

The current browser orientation {LANDSCAPE|PORTRAIT}

ScreenOrientation orientation = driver.getOrientation();

Change Orientation

driver.rotate(ScreenOrientation.LANDSCAPE);

Get geo location

Location location = driver.location();

Set location

driver.setLocation(new Location(49, 123, 10)); // Must be a driver that implements LocationContext lat,long,altitude

Get available log types as a list of strings

Set<String> logTypes = driver.manage().logs().getAvailableLogTypes();

Get the log for a given log type. Log buffer is reset after each request

LogEntries logEntries = driver.manage().logs().get("driver");

Appium provides Appium Event Timing to track when events happen. This custom event feature allow users to store a custom event as the feature.

CustomEvent evt = new CustomEvent();

evt.setEventName("funEvent");

evt.setVendor("appium");

driver.logEvent(evt);

Get events stored in appium server

driver.getEvents();

Update the current driver setting on the device, They do not apply to controlling the app or device under test.

driver.setSetting(Setting.WAIT\_FOR\_IDLE\_TIMEOUT, 5000);

driver.setSetting(Setting.SERVER\_PORT, 8082);

Retrieve the current settings on the device

Map<String, Object> settings = driver.getSettings();

Run a WebdriverIO script against the current session, allowing execution of many commands in one Appium request.

String script = "const el = await driver.$('~foo');\n"

+ "await el.click();"

driver.executeDriverScript(script, new ScriptOptions().withTimeout(200));

Activities

String package = driver.getCurrentPackage();

String activity = driver.currentActivity();

driver.startActivity(new Activity("com.example", "ActivityName"));

App

driver.installApp("/Users/johndoe/path/to/app.apk");

driver.isAppInstalled("com.example.AppName");

driver.launchApp();

driver.runAppInBackground(Duration.ofSeconds(10));

driver.closeApp();

driver.resetApp();

driver.removeApp("com.example.AppName"); //appID bundleId

driver.activateApp(packageName);

((AndroidDriver) driver).activateApp(packageName); //Appium2

driver.terminateApp('com.apple.Preferences');

driver.queryAppState('com.apple.Preferences');

// 0 is not installed. 1 is not running. 2 is running in background or suspended. 3 is running in background. 4 is running in foreground

Map<String, String> appStrings = driver.getAppStringMap("en", "/path/to/file"); //Get app strings

driver.endTestCoverage("Intent", "/path"); //Intent to broadcast, Path to .ec file

Clipboard

driver.getClipboard(ClipboardContentType.PLAINTEXT); // get plaintext

driver.getClipboardText();

// base64Content is Base64-encoded content

driver.setClipboard("label", ClipboardContentType.PLAINTEXT, base64Content); //Clipboard data label for Android.

driver.setClipboardText("happy testing")

(For iOS 13+ real devices) Apple security preferences require the WebDriverAgentRunner application to be in foreground in order to be able to receive the system clipboard content

1. Open Springboard, @driver.activate\_app 'com.apple.springboard'. 2. Find the WDA icon with @driver.find\_element :accessibility\_id, 'WebDriverAgentRunner-Runner' (It should be visible on the screen) and click it. 3. Call the get clipboard API. 4. Open the application under test with the Activate App.

Emulator

driver.setPowerAC(PowerACState.OFF); //connected to plug

driver.setPowerCapacity(100);

Files

driver.pushFile("/data/local/tmp/foo.bar", new File("/Users/johndoe/files/foo.bar")); //Place a file onto the device in a particular place

byte[] fileBase64 = driver.pullFile("/path/to/device/foo.bar");

byte[] folder = driver.pullFolder("/path/to/device/foo.bar"); //return zip archive

static File file = new File(FILEPATH);

OutputStream os = new FileOutputStream(file);

os.write(bytes);

os.close();

Interactions

driver.shake();

driver.lockDevice();//not XCUITest

driver.unlockDevice(); //not iOS

boolean isLocked = driver.isDeviceLocked();//not iOS

driver.rotate(new DeviceRotation(xOffsetCenter, yOffsetCenter, radius));

Keys

https://developer.android.com/reference/android/view/KeyEvent.html

driver.pressKeyCode(AndroidKeyCode.SPACE, AndroidKeyMetastate.META\_SHIFT\_ON); //not iOS

driver.longPressKeyCode(AndroidKeyCode.HOME); //not iOS

driver.hideKeyboard();

boolean isKeyboardShown = driver.isKeyboardShown();

Network

driver.toggleAirplaneMode(); //not iOS

driver.toggleData();

driver.toggleWifi();

driver.toggleLocationServices();

driver.sendSMS("555-123-4567", "Hey lol");

driver.makeGsmCall("5551234567", GsmCallActions.CALL);

driver.setGsmSignalStrength(GsmSignalStrength.GOOD); //emu only

driver.setGsmVoice(GsmVoiceState.HOME); //emu only

driver.setNetworkSpeed(NetworkSpeed.LTE);

Performance data //android only

List<String> performanceTypes = driver.getSupportedPerformanceDataTypes();

List<List<Object>> performanceData = driver.getPerformanceData("my.app.package", "cpuinfo", 5);

Screen recording

driver.startRecordingScreen();

driver.startRecordingScreen(new BaseStartScreenRecordingOptions(....));

driver.stopRecordingScreen();

driver.stopRecordingScreen(new BaseStopScreenRecordingOptions(....));

Touch ID //ios only

allowTouchIdEnroll desired capability must be set to true

driver.toggleTouchIDEnrollment(true);

driver.performTouchID(false); // Simulates a failed touch

driver.performTouchID(true); // Simulates a passing touch

System

driver.openNotifications();

//Retrieve visibility and bounds information of the status and navigation bars

Map<String, String> systemBars = driver.getSystemBars();

String time = driver.getDeviceTime();

driver.getDeviceDensity();

driver.fingerPrint(1); //emu only

Element

MobileElement elementOne = (MobileElement) driver.findElementByAccessibilityId("SomeAccessibilityID");

List<MobileElement> elementsOne = (List<MobileElement>) driver.findElementsByAccessibilityId("SomeAccessibilityID");

**Accessibility ID** Read a unique identifier for a UI element. For XCUITest it is the element's accessibility-id attribute. For Android it is the element's content-desc attribute.

**Class name** For IOS it is the full name of the XCUI element and begins with XCUIElementType. For Android it is the full name of the UIAutomator2 class (e.g.: android.widget.TextView)

**ID** Native element identifier. resource-id for android; name for iOS.

**Name** Name of element

**XPath** Search the app XML source using xpath (not recommended, has performance issues)

**Image** Locate an element by matching it with a base 64 encoded image file

**Android UiAutomator** (UiAutomator2 only) Use the UI Automator API, in particular the UiSelector class to locate elements. In Appium you send the Java code, as a string, to the server, which executes it in the application’s environment, returning the element or elements.

**Android View Tag** (Espresso only) Locate an element by its view tag

**Android Data Matcher** (Espresso only) Locate an element using Espresso DataMatcher

**IOS UIAutomation** When automating an iOS application, Apple’s Instruments framework can be used to find elements

Actions

element.click();

element.sendKeys("Hello world!");

element.clear();

element.getText();

element.getTagName();

element.getAttribute("content-desc");

boolean isSelected = element.isSelected();

boolean isEnabled = element.isEnabled();

boolean isDisplayed = element.isDisplayed();

Point location = element.getLocation();

Dimension elementSize = element.getSize();

Rectangle rect = element.getRect();

String cssProperty = element.getCssValue("style"); //web elements

Context

String context = driver.getContext(); // return NATIVE\_APP, WEBVIEW\_id/pkg name

Set<String> contextNames = driver.getContextHandles();

driver.context(contextNames.toArray()[1]);

Actions

//mouse action

Actions action = new Actions(driver);

action.moveTo(element, 10, 10);

action.click();

action.doubleClick();

action.clickAndHold();

action.release();

action.perform();

//Touch action

TouchActions action = new TouchActions(driver);

action.singleTap(element);

action.doubleTap(element);

action.down(10, 10);

action.moveTo(50, 50);

action.up(20, 20);

action.longPress(element);

action.scroll(element, 10, 100);

action.flick(element, xOffset, yOffset, speed);

action.perform();

//Multi action

TouchActions actionOne = new TouchAction();

actionOne.press(10, 10);

actionOne.moveTo(10, 100);

actionOne.release();

TouchActions actionTwo = new TouchAction();

actionTwo.press(20, 20);

actionTwo.moveTo(20, 200);

actionTwo.release();

MultiTouchAction action = new MultiTouchAction();

action.add(actionOne);

action.add(actionTwo);

action.perform();

//Appium2 W3C actions

WebElement dragMe = (MobileElement) driver.findElementsByAccessibilityId("SomeAccessibilityID");

WebElement target = (MobileElement) driver.findElementsByAccessibilityId("SomeOtherAccessibilityID");

Point source = dragMe.getCenter();

Point target = driver.findElementByAccessibilityId("dropzone").getCenter();

PointerInput finger = new PointerInput(PointerInput.Kind.TOUCH, "finger");

Sequence dragNDrop = new Sequence(finger, 1);

dragNDrop.addAction(finger.createPointerMove(Duration.ofMillis(0),PointerInput.Origin.viewport(), source.x, source.y));

dragNDrop.addAction(finger.createPointerDown(PointerInput.MouseButton.LEFT.asArg()));

dragNDrop.addAction(finger.createPointerMove(Duration.ofMillis(700), PointerInput.Origin.viewport(),target.x, target.y));

dragNDrop.addAction(finger.createPointerUp(PointerInput.MouseButton.LEFT.asArg()));

driver.perform(Arrays.asList(dragNDrop));

Web

//Window

Set<String> windowHandles = driver.getWindowHandles();

String windowHandle = driver.getWindowHandle();

driver.switchTo().window("windowHandle ");

String title = driver.getTitle();

Dimension windowSize = driver.manage().window().getSize();

driver.manage().window().setSize(new Dimension(10, 10));

Point windowPosition = driver.manage().window().getPosition();

driver.manage().window().setPosition(new Dimension(10, 10));

driver.manage().window().maximize();

driver.close() //close window

//navigation

driver.get("http://appium.io/");

String url = driver.getCurrentUrl();

driver.back();

driver.forward();

driver.refresh();

//Storage

Set<Cookie> allcookies = driver.manage().getCookies();

driver.manage().addCookie(new Cookie("foo", "bar"));

driver.manage().deleteCookieNamed("cookie\_name");

driver.manage().deleteAllCookies();

//frame

driver.switchTo().frame(3);

driver.switchTo().parentFrame();

//Javascript

((JavascriptExecutor) driver).executeAsyncScript("window.setTimeout(arguments[arguments.length - 1], 500);");

((JavascriptExecutor) driver).executeScript("window.setTimeout(arguments[arguments.length - 1], 500);");

//Running Tests