Appium Testing Tutorial

Published On: July 30, 2024

Appium Testing Tutorial

Designed to make the UI automation of many app platforms easier, Appium is an open-source project and ecosystem that includes associated applications. In this Appium testing tutorial, you will understand the concepts of Appium and how to test apps using it.

Download Appium Testing Tutorial PDF

Introduction to Appium Testing

The goal of Appium is to facilitate UI automation across a wide range of platforms (web, desktop, mobile, etc.). These platforms include TV (Roku, tvOS, Android TV, Samsung), PC (macOS, Windows), and mobile (iOS, Android, Tizen)!

Furthermore, it strives to enable automation of code written in many programming languages, such as Python, Java, and JavaScript.

In this Appium tutorial, we cover the following:

- Overview of Appium Testing
- Why is appium testing popular?
- Appium Architecture
- Functionality of Appium on Android and iOS
- Appium Automation Test
- Troubleshooting Common Errors using Appium Automation
- Configure a Local Appium Server on Your System
- Future Aspects of Appium

Appium Interview Questions

Overview of Appium Testing

An amazing open-source project and ecosystem called Appium was created to simplify UI automation for a variety of app platforms.

- Appium can be used to test apps for web browsers like Chrome, Firefox, and Safari, as well as mobile operating systems like iOS, Android, and Tizen.
- It also works with TV platforms like Roku, tvOS, Android TV, and Samsung, as well as desktop systems like Windows and macOS.

Appium testing is used to automate the following:

- Native mobile applications that are produced with iOS, Android, or Windows SDKs.
- Mobile web applications that can be viewed with mobile browsers like Safari, Chrome, or built-in native browser apps for Android smartphones;
- Hybrid mobile apps that are hybrid and wrap the web view with a native wrapper.

With Appium's flexible cross-platform testing framework, testers can create test scripts for a variety of platforms, including Windows, Android, and iOS, all utilizing the same API. This saves QAs time and effort because they may use the same code for iOS and Android.

Download Appium Syllabus PDF

Why is appium testing popular?

Appium has gained enormous traction among developers and testers globally because of its features and advantages, solidifying its position as the preferred choice for dependable and effective app testing.

Cross-Platform Compatibility: Appium mobile testing offers remarkable cross-platform compatibility, enabling testing across a range of platforms, including desktop (MacOS, Windows), TV, web browsers, iOS, Android, Tizen, and more.

Popular Programming Languages Supported: Popular programming languages, including Java, Ruby, Python, and others, are supported by the Appium mobile testing framework. Because they may use their current frameworks and coding knowledge, testers and developers can easily switch to Appium.

Free & Open Source: Appium is an open-source framework that allows users to view and alter the source code as needed, encouraging creativity and guaranteeing openness. Appium is available to businesses of all sizes and financial constraints due to its free nature.

Compatibility with Testing Frameworks: Testers can enjoy a comfortable testing environment because of Appium's smooth integration with a number of well-known testing frameworks, including JUnit, Cucumber, TestNG, and Pytest.

Robust and adaptable: Appium's design facilitates simple customization and extension. Anyone who wishes to create and release Appium drivers for new platforms can always add to Appium

ASP DOTNET Tutorial
Published On: July 31, 2024
ASP DOTNET Tutorial Microsoft created the web framework known as ASP.NET. It is employed in...

Artificial Intelligence Tutorial
Published On: July 30, 2024
Artificial Intelligence Tutorial Artificial intelligence (AI) is significant since it enhances many facets of society...

Appium Testing Tutorial
Published On: July 30, 2024
Appium Testing Tutorial Designed to make the UI automation of many app platforms easier, Appium...

2.0's capabilities.

Native and Web Application Support: Appium's proficiency is in its ability to automate both online and native apps. It has an extensive feature set that allows it to manage a wide range of testing scenarios.

Appium Architecuture

Written on the Node.js framework, Appium is an HTTP server that uses Selenium WebDriver and has a REST (Representational State Transfer) API setup.

It uses a client/server design to function. Appium permits the testing to be conducted using any WebDriver client that is available.

Using the REST API, the following tasks are completed:

- Obtains the connection from the side of the client
- Heeds the instruction
- Carries out the directive on a mobile device
- Gives the client an HTTP response containing the status of the command execution.

Automation is possible inside a session using Appium. In accordance with the client library (Java, JavaScript, PHP, Ruby, Python, and C#), the client starts this session.

With the aid of a JSON object known as the "desired capabilities object," the client ultimately sends a POST or session request to the server.

Subsequently, the server initiates the automation session and replies with a session ID, which is subsequently utilized to transmit additional commands pertinent to the specified session.

The Appium Client, Appium Server, and End Device are the three primary parts of the architecture.

Appium Client

Using language-specific libraries or SDKs, the Appium client enables developers to write Appium-based test automation scripts for mobile applications.

- These client libraries include methods for finding objects, interacting with UI elements, executing gestures, and checking expected actions across a variety of computer languages.
- To configure the testing environment, testers can also provide desired capabilities using the client libraries.

Appium Server

An essential middleman in the framework that makes mobile application automation possible is the Appium server.

- It creates a link between the mobile application and the test script, regardless of whether the application is running in an emulator or on a real device.
- The test script sends orders to the server, which converts them into automation actions tailored to the desired mobile platform via a REST API.
- Through the utilization of Selenium WebDriver's robust functionalities, the Appium server establishes communication with the mobile application, facilitating tasks such as element identification, user interface interaction, gesture emulation, and behavior validation.

The standard interface of the Appium server enables test scripts to be generated in many programming languages and performed effortlessly on various mobile platforms, hence enabling cross-platform mobile testing.

End Device

- An emulator, simulator, or actual device linked to the server where automated tests are run is referred to as the "end device" in the Appium environment.
- These gadgets are essential for conducting testing methods and verifying the effectiveness and operation of mobile applications.

Appium's architecture is based on these essential elements, which enable dependable and effective testing of mobile applications on a range of platforms and gadgets.

Appium Salary

Functionality of Appium on Android and iOS

Appium uses the Mobile JSON Wire/W3C Protocol to function on the Android and iOS platforms. This protocol makes it easier for the Appium client libraries to convert test commands into REST API requests. The Appium server then forwards these inquiries to the associated Android device or emulator.

Android

The important ***bootstrap.jar*** files are used in Appium Android.

- These files allow the device to use automation frameworks such as UI Automator or Selendroid to carry out the commands.
- The test findings are then sent back to the Appium server, which then sends the Appium client an HTTP response with the pertinent status codes.

iOS

Similar to this, Appium uses Apple's UIAutomation API to communicate with iOS devices over the JSON wire protocol.

- The ***WebDriverAgent.app*** files on iOS devices are essential for deciphering the test commands that are sent to them.
- The WebDriverAgent.app efficiently runs requests on iOS devices with the help of XCUITest, making it possible to test and automate iOS applications with ease.

Appium Automation Testing

Follow the concepts below to perform Appium automation testing:

Appium Inspector

One well-liked Appium feature is Appium Inspector, which is used to find UI elements in mobile apps so that automation scripts may be developed for them.

The built-in Appium Inspector allows you to write scripts right from the remote-control user interface. When you start an Appium session, the Inspector automatically attaches and offers a browser-based script development experience.

Appium Server URL

The address or location where the Appium Server is installed is known as the Appium Server URL. Typically, it looks like this: http://:/wd/hub, where is the hostname or IP address of the computer hosting the Appium Server and is the port number that the server is listening on.

Test Script

Clients for the Appium tool are available in Python, Java, Ruby, JavaScript, and more. The test script can be written in any programming language that Appium supports.

Here, Python code serves as an example. Please make sure that Python 3 is installed on your computer before continuing.

Install Appium Client

The official Appium Python client may be found on PyPI under the name “Appium-Python-Client.” It also has the Selenium Python binding.

Step 1: To install the Python Appium client, execute the command below.

pip install Appium-Python-Client

Example

```
import unittest

# Import Appium Python Client

from appium import webdriver

from appium.webdriver.common.appiumby import AppiumBy
```

```
class TestAppium(unittest.TestCase):

def setUp(self):

appium_server_url = "http://<host>:<port>/wd/hub"

device_id = "XXXXXXXXX"

# Desired capabilities to communicate with appium server

capabilities= {

"udid": device_id,

"automationName": "uiautomator2",

"appPackage": "com.android.settings",

"appActivity": "com.android.settings.Settings",

"platformVersion": "14.4",

"platformName": "Android"

}

# Webdriver initialization

self.driver =
webdriver.Remote(command_executor=appium_server_url,
desired_capabilities=capabilities)

def tearDown(self):

if self.driver:

# Terminate the session

self.driver.quit()

def test_find_battery(self):

# locating element

el = self.driver.find_element(by=AppiumBy.XPATH,value='//*
[@text="Battery"]')

# Automation command(click on the element)

el.click()

if __name__ == '__main__':

unittest.main()
```

Step 2: The test script will locate and interact with UI elements, run the settings app, initialize the Appium web driver with specific capabilities,

and terminate the driver.

Note #1

Fantastic! Let's attempt it. Make sure you have an active Appium server operating in a separate terminal session before launching the test.

However, you don't need to bother about the server setup if you are utilizing a device in our cloud. The host to which the device is connected will already be running the Appium server.

Step 3: Use Python 3 to execute the script test.py.

python3 test.py

Make sure you have an active Appium server running before attempting the test.

On the host to which the device is connected, the Appium server will already be operating if it is on our cloud device.

Appium Project Ideas

Troubleshooting Common Errors using Appium Automation

For the Appium automation tool to operate smoothly during test execution, it is essential to troubleshoot frequent issues.

The following common errors and suggestions for fixing them are listed:

Problems with Element Identification

- Check the element locator approach (XPath, ID, class name, etc.) used in your automation script.
- To examine and verify the properties of each element, use Appium Inspector or a comparable tool.
- Verify that the element is inside the viewable viewport, and if necessary, scroll.
- Use suitable waits or synchronization strategies to address timing or dynamic loading problems.

Problems with Timing and Synchronization

Before taking any action, make sure all necessary items are present, visible, or interactable by implementing explicit waits.

- Modify implicit wait timeouts to give elements enough time to load.
- Employ synchronization strategies according to the programming language, such as ***Thread.sleep() or time.sleep()***.

Appium Server and Client Compatibility

- Check if the Appium server and client libraries (such as the Appium Python and Java clients) are compatible.
- If necessary, update or downgrade the client libraries to correspond with the server version.

You may effectively fix typical problems and guarantee a successful Appium automation operation by following these troubleshooting guidelines.

Configure a Local Appium Server on Your System

To guarantee a successful installation, there are multiple stages involved in setting up Appium Server on Windows, Linux, or Mac.

Install Node.js

An overview of each operating system will be found here:

On Windows

Download and install the most recent version of Node.js from the official website (https://nodejs.org).

On Linux

Use the package manager to install Node.js by opening a terminal and entering the following commands:

sudo apt update

sudo apt install nodejs npm

On MacOS

Step 1: To install Homebrew, open the terminal and type the following command:

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Step 2: Use Homebrew to install Node.js by typing the following command into your terminal:

brew install node

Install Appium

To install Appium globally, launch the terminal or command line and type the following command:

npm install -g appium

 # If you are having problems, try the linux/mac command sudo.

sudo npm install -g appium

Install an Appium Driver and Its Dependencies

Installing at least one driver is necessary to use Appium for any kind of useful work. Appium cannot perform any automation functions without a driver. There is a large ecosystem of drivers and plugins available to increase Appium's capabilities.

Utilizing the CLI interface extension for Appium during installation

The management of drivers and plugins can be assigned to Appium through the use of the Extension CLI. You can use CLI commands to tell Appium which extensions to install, update, or remove. Using the CLI, you can install various drivers.

Example

appium driver install uiautomator2

appium driver install xcuitest

appium driver install chromium

These installed packages will be handled by extensions.

$APPIUM_HOME/node_modules/.cache/appium/extensions.yaml.

Install Appium Doctor

To verify that all of the dependencies needed by Appium are installed successfully, Appium Doctor is a useful tool. To install it, use the following command:

npm install -g appium-doctor

 # If you are having problems, try the linux/mac command sudo.

sudo npm install -g appium-doctor

Install JDK

From the Oracle website

(https://www.oracle.com/java/technologies/javase-jdk14-downloads.html),

download and install the Java Development Kit (JDK).

Install Android SDK

From the official website *(https://developer.android.com/studio),* download and install Android Studio. Open Android Studio, then follow the setup process to install the required files, which include the Android SDK.

Configure the SDK and Java environment variables

- Assign the JDK installation path to the JAVA_HOME environment variable. Furthermore, include the "bin" directory of the JDK in the

system's PATH variable.
- The SDK installation path should be the value of the ANDROID_HOME environment variable. Furthermore, include the "tools" and "platform-tools" directories from the SDK in the system's PATH setting.

Verify Installation

Use the terminal or command prompt to run `appium-doctor` to verify that all dependencies and configurations are configured appropriately. Attend to any problems that the Appium Doctor has brought to your attention.

appium-doctor

To launch Appium, issue the following command:

appium

Appium Training

Future Aspects of Appium

Cross-platform mobile testing has advanced dramatically with Appium 2.0, ushering in a revolutionary era.

With revolutionary capabilities to improve mobile app testing, Appium 2.0 places a major emphasis on user experience, cooperation, and adaptability.

Upcoming Features

- Providing the Appium Driver Ecosystem for Developer Empowerment
- Smooth Interaction with the Plugin Environment
- Enhanced User Experience through Better Documentation
- Simplifying for Effectiveness and Significance
- Presenting the Upgraded Features Independent Appium Inspector

New Platform Drivers to Explore

These are a few of the recently released drivers that are compatible with Appium 2.0.

- **Roku**

appium driver install –source=npm @headspinio/appium-roku-driver

- **Tizen TV**

appium driver install –source=npm appium-tizen-tv-driver

- **LG WebOS TV**

appium driver install –source=npm appium-lg-webos-driver

- **Chromium**

appium driver install chromium

New Appium Plugins to Explore

These fresh Appium plugins are compatible with Appium 2.0.

- **AtUnity**

appium plugin install –source=npm appium-altunity-plugin

- **Images**

appium plugin install –source=npm images

- **Execute Driver**

appium plugin install –source=npm execute-driver

Conclusion

Appium gains even more usefulness as a tool for testing mobile apps. We hope this Appium testing tutorial will be useful as you get started. Gain expertise with our **Appium training in Chennai** for a promising career.

Softlogic Academy

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.

Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber