FACULDADE ENGENHARIA

Departamento de Informática

Testing with Appium & Python

Appium

- Appium is an open-source automation framework used for testing mobile applications
- It enables the automation of native, hybrid, and mobile web apps on iOS, Android, and Windows platforms

Components

- **Appium Server:** Acts as a bridge between the test scripts and the mobile devices (real or emulated). The server accepts automation commands in various programming languages via the WebDriver protocol and executes them on mobile devices
- **Appium Clients:** Client libraries available in multiple programming languages (e.g., Java, Python, Ruby, C#) that allow testers to write scripts and communicate with the Appium server
- **Appium Inspector:** A graphical user interface tool for viewing the structure of a mobile app's UI elements. It can be used to inspect the app's elements and generate XPath or other selectors for use in scripts

Use Cases

- **Automating Functional Testing of Mobile Apps:** Ensuring the app behaves as expected across different devices and operating systems
- **Cross-Platform Testing:** Running the same set of tests on both iOS and Android platforms with minimal changes
- **Continuous Integration/Continuous Deployment (CI/CD):** Integrating Appium into CI/CD pipelines to perform automated regression testing
- **Mobile Web Testing:** Testing web apps in mobile browsers, providing a web automation experience similar to Selenium but for mobile devices

Step 1: Install Appium and Appium-Python-Client

- Install Appium using npm:

npm install -g appium

- Install the Appium-Python-Client library:

pip install Appium-Python-Client

Step 2: Start the Appium Server

- Start the Appium server by running the following command in the terminal:

appium

- This will start the server and show logs indicating it's ready for connections.

Step 3: Set Up Desired Capabilities (Android)

- Use the following code for Android to set up desired capabilities:

```
desired_caps = {
    'platformName': 'Android',
    'deviceName': 'emulator-5554',
    'app': '/path/to/your/app.apk',
    'automationName': 'UiAutomator2'
}
```

Step 4: Write a Basic Python Script (Android)

- Here's an example Python script to automate an Android app:

```
from appium import webdriver
import time

# Step 1: Set up desired capabilities
desired_caps = {
    'platformName': 'Android',
    'deviceName': 'emulator-5554',
    'app': '/path/to/your/app.apk',
    'automationName': 'UiAutomator2'
}

# Step 2: Initialize the Appium driver
driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)

# Step 3: Find and interact with elements
element = driver.find_element_by_accessibility_id('some_button_id')
element.click()

# Step 4: Quit the driver
driver.quit()
```

Step 5: Write a Basic Python Script (iOS)

- Here's an example Python script to automate an iOS app:

```
from appium import webdriver
import time

# Step 1: Set up desired capabilities
desired_caps = {
    'platformName': 'iOS',
    'platformVersion': '14.0',
    'deviceName': 'iPhone 11',
    'app': '/path/to/your/app.app',
    'automationName': 'XCUITest'
}

# Step 2: Initialize the Appium driver
driver = webdriver.Remote('http://localhost:4723/wd/hub', desired_caps)

# Step 3: Find and interact with elements
element = driver.find_element_by_accessibility_id('some_button_id')
element.click()

# Step 4: Quit the driver
driver.quit()
```

Step 6: Run the Test

- Once your Appium server is running and the script is ready, follow these steps:

1. Make sure the device or emulator is connected.
2. Run the script using the command:

python test_script.py

The script will automate the mobile app, interact with the elements, and close the app.

Exercise

Create a Python script that automates tasks using Appium for a mobile application of your choice.


HOW
WHY
WHERE
WHEN
WHERE
WHY
WHERE
WHO
WHEN
WHAT
WHAT
WHY
WHEN
WHAT
HOW
WHEN
WHO
WHERE
WHY
HOW