Appium with Serenity

Appium is an open-source tool for automating native, mobile web, and hybrid applications on iOS mobile, Android mobile, and Windows desktop platforms. Appium is "cross-platform": it allows you to write tests against multiple platforms (iOS, Android, Windows), using the same API. This enables code reuse between iOS, Android, and Windows testsuites.

Technology stack

**Serenity:** For pageObject class and their predefined function.

**Maven:** as a project management tool.

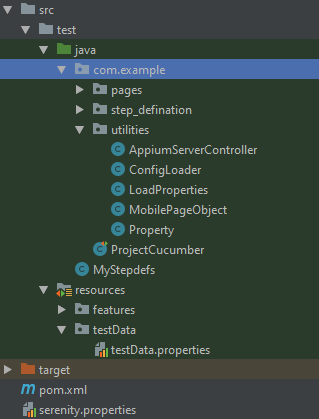
**Java**: Programming language.

**Appium server:** Appium is a server written in Node.js

**Selenium:** It is a portable framework for testing web applications.

**IntelliJ IDEA**: An integrated development environment for developing computer software

Project Structure



1. **Java file(.java)** use to write scripts

**Packages in java folder :-**

1. **steps/step\_defenition :-** This package contain all the step\_defenation which we use in cucumber files(.feature)
2. **pages :-** This package contains all the pageObjects and their related functions of Page Object Model.
3. **utilities :-** This package contains all the classes and functions which are common in all over the project.

* **AppiumServerController:-** Launching and stopping the appium server programmatically.
* **LoadProperties:-** contains functions used to load property files.
* **MobilePageObject**
* **Properties:-** Server Configuration

**2. In Resources folder we store multiple files like :-**

1. **cucumber file**(.feature) in the feature folder.
2. **Property files** (.property) in propertyFiles folder.

**3. serenity.properties file** is used to store various serenity properties(timeout, screenshots etc ) and appium properties(desired capabilities).

Desired Capabilities:

Desired Capabilities are keys and values encoded in a JSON object, sent by Appium clients to the server when a new automation session is requested.

**For Android:**

|  |
| --- |
| {  "platformName": "Android",  "appPackage": "com.example",  "platformVersion": "9",  "deviceName": "Samsung M10",  "udid": "5200c541436b16d3",  "appActivity": "com.example.view.activities.SplashActivity",  "automationName": "uiautomator1",  "autoGrantPermissions": "true"  "app":<path>  } |

**For iOS:**

|  |
| --- |
| {  "automationName": "XCUITest",  "platformName": "iOS",  "platformVersion": "14.5",  "deviceName": "iPhone 12 Pro Max",  "udid": "BAC303D4-F4C9-4D5C-94A0-811864169E6F",  "app": "/Users/<UserName>/Downloads/Appium Framework/app/UIKitCatalog.app"  } |

Requirements

* Java

[Download](https://www.oracle.com/in/java/technologies/javase-downloads.html) and install java and set JAVA\_HOME

* Nodejs
* Brew installer for mac  
  install Brew by run this command :-

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

* Install carthage  
  **brew install carthage** ,it provides u details of simulator and devices of iOS attached with your mac by  
  **Instruments -s devices** command
* AndroidStudio

for using packages tools and emulator

Set ANDROID\_HOME

* Xcode

for using packages tools and simulator

* IntelliJ IDEA
* Appium server and client

For server you need to install appium by **npm install -g appium**

For appium desktop download and install visit [**link**](https://github.com/appium/appium-desktop/releases/tag/v1.21.0)

Use **appium desktop** for inspect element

* Device or Emulator

You can script either on device or on emulator/simulator

* Vysor for device projection

If you use a device instead of an Emulator or simulator you need this to project your device.

How to run

* **mvn clean** is used to clean previous report
* **mvn clean verify** is used to clean previous reports and create a new report while running the scenarios.