**Appium Testing for an iOS App Using Java & Maven**

Appium allows you to automate iOS applications using Java and Maven. Below are the **step-by-step instructions** to set up Appium for iOS testing in a Java Maven project.

**1. Prerequisites**

**System Requirements:**

✅ macOS (Appium requires macOS for iOS automation)  
✅ Xcode (Latest stable version)  
✅ Java (JDK 11 or later)  
✅ Maven (Installed & configured)  
✅ Appium Server (Installed via npm)  
✅ WebDriverAgent (for iOS automation)  
✅ iOS Simulator or Real Device  
✅ IDE (IntelliJ IDEA / Eclipse)

**2. Install Dependencies**

**Install Appium**

bash

CopyEdit

npm install -g appium

**Install Appium Doctor (For Dependency Checks)**

bash

CopyEdit

npm install -g appium-doctor

appium-doctor

✅ Ensure all dependencies (Xcode, Java, Carthage, etc.) are correctly installed.

**Install iOS WebDriverAgent**

bash

CopyEdit

brew install carthage

Then, run:

bash

CopyEdit

cd /usr/local/lib/node\_modules/appium/node\_modules/appium-webdriveragent

mkdir -p Resources/WebDriverAgent.bundle

./Scripts/bootstrap.sh -d

✅ This sets up WebDriverAgent, which Appium uses for iOS automation.

**3. Create a Java Maven Project**

1. Open IntelliJ IDEA / Eclipse.
2. Create a **Maven Project**.
3. Add the following **dependencies** to pom.xml:

xml

CopyEdit

<dependencies>

<!-- Selenium WebDriver -->

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>4.6.0</version>

</dependency>

<!-- Appium Java Client -->

<dependency>

<groupId>io.appium</groupId>

<artifactId>java-client</artifactId>

<version>8.3.0</version>

</dependency>

<!-- TestNG for Test Execution -->

<dependency>

<groupId>org.testng</groupId>

<artifactId>testng</artifactId>

<version>7.4.0</version>

<scope>test</scope>

</dependency>

</dependencies>

**4. Start Appium Server**

Run the Appium server before executing tests:

bash

CopyEdit

appium

or, with logs:

bash

CopyEdit

appium --relaxed-security --log appium.log

**5. Write Appium Test for iOS**

**Set Up BaseTest.java**

Create a BaseTest.java file to handle Appium’s setup and teardown.

java

CopyEdit

import io.appium.java\_client.ios.IOSDriver;

import io.appium.java\_client.remote.MobileCapabilityType;

import org.openqa.selenium.remote.DesiredCapabilities;

import java.net.MalformedURLException;

import java.net.URL;

import java.time.Duration;

public class BaseTest {

protected static IOSDriver driver;

public static void setup() throws MalformedURLException {

DesiredCapabilities caps = new DesiredCapabilities();

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

caps.setCapability(MobileCapabilityType.PLATFORM\_VERSION, "17.0"); // Update your iOS version

caps.setCapability(MobileCapabilityType.DEVICE\_NAME, "iPhone 14 Pro"); // Update device name

caps.setCapability(MobileCapabilityType.AUTOMATION\_NAME, "XCUITest");

caps.setCapability(MobileCapabilityType.APP, "/Users/yourusername/path-to-your-app.app"); // Update your .app file path

caps.setCapability("noReset", true);

driver = new IOSDriver(new URL("http://127.0.0.1:4723/"), caps);

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

}

public static void tearDown() {

if (driver != null) {

driver.quit();

}

}

}

**Create LoginTest.java**

A test that interacts with a login screen.

java

CopyEdit

import org.openqa.selenium.By;

import org.testng.Assert;

import org.testng.annotations.AfterClass;

import org.testng.annotations.BeforeClass;

import org.testng.annotations.Test;

public class LoginTest extends BaseTest {

@BeforeClass

public void setUp() throws Exception {

setup();

}

@Test

public void testLogin() {

// Locate and interact with login elements

driver.findElement(By.xpath("//XCUIElementTypeTextField")).sendKeys("testuser");

driver.findElement(By.xpath("//XCUIElementTypeSecureTextField")).sendKeys("password123");

driver.findElement(By.xpath("//XCUIElementTypeButton[@name='Login']")).click();

// Validate login success

String successMessage = driver.findElement(By.xpath("//XCUIElementTypeStaticText[@name='Welcome']")).getText();

Assert.assertEquals(successMessage, "Welcome");

}

@AfterClass

public void tearDown() {

tearDown();

}

}

**6. Run The Test**

Run tests using Maven:

bash

CopyEdit

mvn test

or using TestNG:

bash

CopyEdit

mvn clean test -Dtest=LoginTest

**7. Debugging and Logs**

* View Appium logs in appium.log
* Use Appium Inspector to locate elements visually

**8. Enhancements**

✅ Add **Page Object Model (POM)** for better structure  
✅ Run tests on **Real iOS Device** with proper provisioning  
✅ Integrate with **CI/CD (Jenkins, GitHub Actions)**