

Prerequisites

1. Copy the TouchActionsApi.java file into the Utils package present in the src/main/java
2. Create a LoginTest.java in the (default package) present in the src/test/java
3. Provide the Capabilities in the appium inspector and start the session after appium server and android emulator is started.

Code in LoginTest.java:

```
import org.testng.annotations.BeforeMethod;
import config.Capability;
import io.appium.java_client.android.AndroidDriver;
public class LoginTest{
    AndroidDriver driver;
    TouchActionsApi action;
    @BeforeClass
    public void setUp() throws Exception{
        Capability.startEmulator();
        Capability.startAppium().start();
        Thread.sleep(2000);
        driver = Capability.capabilities();
        action = new TouchActionsApi(driver);

        driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));
    }
    @Test(priority = 1)
    public void allowPermission(){
```

```

        WebElement popup =
driver.findElement(By.id("com.android..."));
        if(popup.isDisplayed()){
            action.tap(popup);
        }
    }
    @Test(priority = 2)
    public void selectLanguage(){
        WebElement lang =
driver.findElement(By.xpath("//android.widget. . . ."));
        action.tap(lang);
    }
    @Test(priority = 3)
    public void login(){
        driver.findElement(By.id("com.forbinary. . .
"))).click();
        driver.findElement(By.id("com.forbinary. . .
"))).sendKeys("India");

        action.tap(driver.findElement(By.xpath("//androi
d.widget.TextView[@text = 'India (IN)']"))););
        driver.findElement(By.id("com.forbinary. . .
")));
        action.tap(driver.findElement(By.id("com.forbina
ry. . . ."))));
    }
    @AfterClass
    public void tearDownMethod() throws Exception{
        driver.quit();
    }
}

```

Inside the Capability.java:

```
Make -> dc.setCapability("noReset", false);
```

New Topic

Create a Folder named mobileTest in the desktop. Open the Command Prompt by typing cmd in the path of folder created.

Type commands in commandPrompt:

```
> npm init -y
```

Creates the necessary folders and files

```
> npm install @wdio/cli
```

```
> npx wdio config
```

Click on yes

Choose E2E Testing - of Web or Mobile Applications

Questions: Where is your automation backend located?

Choose - On my local machine

Questions: Which environment you would like to automate?

Choose - native, hybrid, and mobile . . .

Select - UiAutomator2

Questions: Which framework do you want to use?

Choose - Mocha(<https://mochajs.org/>)

Questions: Do you want to use Typescript to write tests?

Choose - n

Questions: Do you want WebdriverIO to autogenerate some test files?

Choose - n

Choose - Spec

Questions: Do you want to add a plugin to your test setup?

Choose - Wait for

Questions: Would you like to include Visual Testing?

Choose - yes

Questions: Do you want to add service to your test setup?

Choose - Appium

Questions: Do you want me to run `npm install`

Choose - y

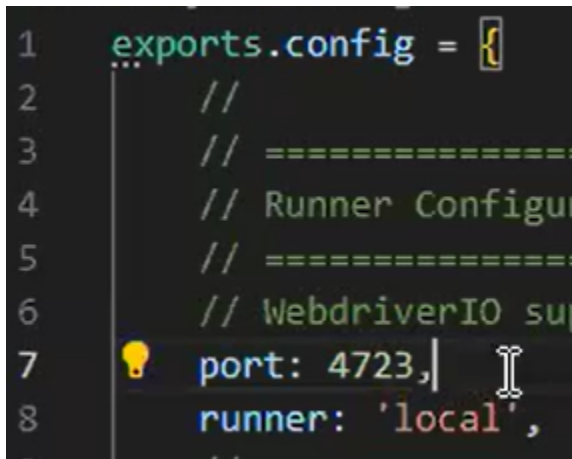
Choose - no for appium setup using appium-installer

To open the Visual Studio code

> code .

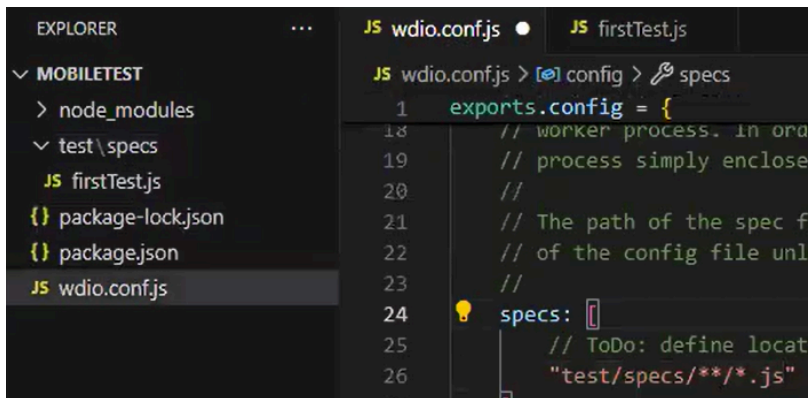
In the wdio.conf.js:

Add > port:4723

A screenshot of a code editor showing the configuration for wdio.conf.js. The file is a JavaScript object with several properties. The 'port' property is set to 4723, and the 'runner' property is set to 'local'. The code is as follows:

```
1  exports.config = {  
2    //  
3    // =====  
4    // Runner Configur  
5    // =====  
6    // WebdriverIO sup  
7    port: 4723,  
8    runner: 'local',
```

Add > spec files

A screenshot of the Visual Studio Code interface. The Explorer sidebar on the left shows a project structure with a folder named 'MOBILETEST' containing 'node_modules', 'test/specs', and 'firstTest.js'. The main editor area shows the 'wdio.conf.js' file. The 'specs' property is being edited, and the value 'test/specs/**/*.js' is entered. The code is as follows:

```
1  exports.config = {  
18    // worker process. In orde  
19    // process simply enclose  
20    //  
21    // The path of the spec fi  
22    // of the config file unle  
23    //  
24    specs: [  
25      // ToDo: define locati  
26      "test/specs/**/*.js"
```

From the appium inspector copy the Capabilities and Paste:

```

54 capabilities: [{
55     // capabilities for local Appium web tests on an Android Emulator
56     "platformName": "Android",
57     "platformVersion": "14",
58     "udid": "emulator-5554",
59     "automationName": "UiAutomator2",
60     "appPackage": "io.appium.android.apis",
61     "appActivity": "io.appium.android.apis.ApiDemos" - API Demos
62 }],

```

```

54 capabilities: [{
55     // capabilities for local Appium web tests on an Android Emulator
56     platformName: "Android",
57     "appium:platformVersion": "14",
58     "appium:udid": "emulator-5554",
59     "appium:automationName": "UiAutomator2",
60     "appium:appPackage": "io.appium.android.apis",
61     "appium:appActivity": "io.appium.android.apis.ApiDemos"
62 }],

```

Run the emulator and start the appium server, Check adb devices.

In the firstTest.js file create a describe and it block.

In the terminal space of visual studio:

```
> npx wdio
```