**Automation Mobile App Testing (Day9)**

First important thing is you need an Android SDK (Software development kit) <http://developer.android.com/index.html>

**Day4**:

**Android SDK** – software development kit that will help Android developers to develop new mobile apps and be able to deploy them on the mobile. This is created for the Android OS. The applications developed are created as APK’s.

Testing part of SDK:

**Virtual Device:**

* tools – Manage AVDs – you can emulate real device. To Add, select **New** – Add a name. Select the **device type**. Choose **target** such as Android version. **CPU** – select anything such as ARM. **Keyboard**- Hardware keyboard present is important- check it. It helps you type using keyboard on the device. **Skin** can be checked. **Memory options**: RAM 512**; internal storage**: 2000(2GB). Click OK. The device appears in the Android virtual Device manager.
* Select the device and start. It will launch the device.

**Day 5: Testing on the real devices.**

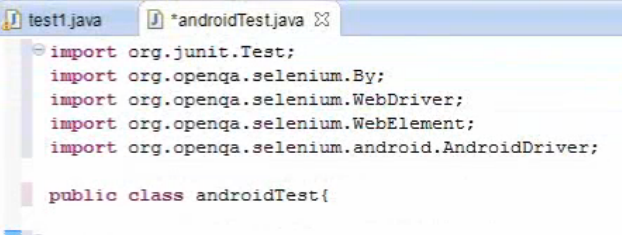
* Dkhasdn

**Day 9: Automation of Mobile Applications.**

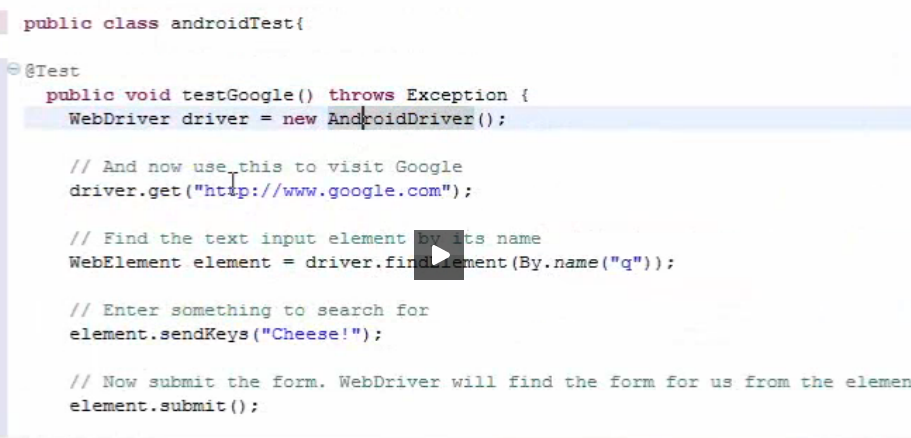
* **SDK manager –** Tools, latest version, previous version of Android and Extras needs to be checked and installed.
* There are 2 types of applications: web based apps and native apps.
* We use Android Device Bridge which is the connection to take the APK files and install on the virtual device.
* **Selenium Android driver :** visit <https://code.google.com/p/selenium/wiki/AndroidDriver> (reference as needed)
* **Download Eclipse. –** eclipse.org. Eclipse is an IDE, a development environment which will let you do coding on lot of technologies, primarily Java. It is also like a notepad on steroids. Extract to a folder.
* **Download latest JRE(java runtime environment) :** you need basic java to run java applications, require JRE or JDK to create executable files of java code and then run it. To run the final applications only java is required.
* **Download Selenium Server Jar file from** [http:**//**seleniumhq.o**rg**](http://seleniumhq.org) **-** selenium lets you write programs that you use to automate. Download 1) selenium IDE. It installs as a add-on on firefox browser. 2)Then install selenium RC web driver(zip) for java client(, copy the zip folder of web driver). 3)download Java binding. 4)Download jUnit jar file. Create a folder called selenium and the selenium server(jar file) from http://junit.org (download latest junit jar files)
* **Download the Android server apk from selenium hq (Android 2.21.0 wiki )from** <https://code.google.com/p/selenium/downloads/detail?name=android-server-2.21.0.apk>Put in the selenium folder. **Install this Android driver APK file** and **ADB install xxxx.apk from platform** tools under cmd prompt.(c/programfiles/android/androidsdk/platformtools – adb (android device bridge) from system to device .
* copy the installed Android driver apk to the same location where adb is
* so connect the device. Open cmd prompt and navigate to that location c:\Program Files<x86>\Android\android-sdk\platformtools
* type the command adb install android.apk ( this is that adroid driver apk file renamed to android) this will install the web driver onto the device.
* On the real device – goto settings apps – webdriver
* Now we need to set up the port forwarding in order to forward traffic from the host machine to the device. In a terminal type : go to cmd cd c:\Program Files<x86>\Android\android-sdk\platformtools ]
* This is giving transfer control protocol telling which port number that this interaction will happen.
* **Cmd is *adb forward tcp:8080 tcp 8080*** this will tell where my test will be executing from and where my device is present. If your device is on a different port in a local network or over the internet, you should be giving that ip address.
* Open Eclipse, start a new project. Add all the related jar files, get the basic web driver code and run the code from eclipse.
* File – new – java project – give project name and finish.
* Go to the project just created and expan to the src folder. Go to src folder, right click on it and say new java class and give a name.
* It will create a basic class. And it displays as public class androidTest {

} (public class and name is the name of the test.)

* Get a basic web driver code from Selenium recording. Take the Java Junit web driver.
* Add the corresponding jar files. Right click on the project, go to properties, java build path – libraries – add external jars – put all the 3 files in the selenium folder.
* Import wherever required to tell eclipse how to recognize



* The following is a sample code



* In the above code, webdriver driver = new AndroidDriver. This means we are creating a new Android driver in this code.
* Driver.get is a command to go to a specific url
* Then I’m finding a web element on the browser with a attribute called name value is q
* Run the code. It should open Google on the device.

**On a real device:**

Install Android WebDriver on Real Device   
  
Connect your device through USB to your machine.   
  
$~/android\_sdk/tools/adb devices   
  
This will give you the serial Id of your connected device. Now install   
the apk on the device:   
  
$./adb -s <serialId> -e install -r android-server.apk   
  
Setup port forwarding:   
  
$~/android\_sdk/adb -s <serialId> forward tcp:8080 tcp:8080   
  
Start the WebDriver Server on the device or emulator   
  
Start the WebDriver application just installed in the device or   
emulator.

RUNNING YOUR TEST APP WITH APPIUM (ANDROID)

First, make sure you have one and only one Android emulator or device connected. If you run adb devices, for example, you should see one device connected. This is the device Appium will use for tests. Of course, to have a device connected, you’ll need to have made an Android AVD (see system setup (Windows, Mac, or Linux) for more information). If the Android SDK tools are on your path, you can simply run:

emulator -avd

And wait for the android emulator to finish launching. Sometimes, for various reasons, adb gets stuck. If it’s not showing any connected devices or otherwise failing, you can restart it by running:

adb kill-server && adb devices

Now, make sure Appium is running:

node .

Then script your WebDriver test, sending in the following desired capabilities:

// java

DesiredCapabilities capabilities = new DesiredCapabilities();

capabilities.setCapability(MobileCapabilityType.PLATFORM\_NAME, "Android");

capabilities.setCapability(MobileCapabilityType.PLATFORM\_VERSION, "4.4");

capabilities.setCapability(MobileCapabilityType.DEVICE\_NAME, "Android Emulator");

capabilities.setCapability(MobileCapabilityType.APP, myApp);

In this set of capabilities, myApp must be either:

A local absolute path to your .apk or a .zip of it

A url of a zip file containing your .apk

A path to one of the sample app relative to the appium install root

Using your WebDriver library of choice, set the remote session to use these capabilities and connect to the server running at port 4723 of localhost (or whatever host and port you specified when you started Appium). You should be all set now!

RUNNING YOUR TEST APP WITH APPIUM (ANDROID DEVICES < 4.2, AND HYBRID TESTS)

Android devices before version 4.2 (API Level 17) do not have Google’s UiAutomator framework installed. This is what Appium uses to perform the automation behaviors on the device. For earlier devices or tests of hybrid (webview-based) apps, Appium comes bundled with another automation backend called Selendroid.

To use Selendroid, all that is required is to slightly change the set of desired capabilities mentioned above, by adding the automationName capability and specifying the Selendroid automation backend. It is usually the case that you also need to use a . before your activity name (e.g., .MainActivity instead of MainActivity for your appActivity capability).

// java

DesiredCapabilities capabilities = new DesiredCapabilities();

capabilities.setCapability(MobileCapabilityType.AUTOMATION\_NAME, "Selendroid");

capabilities.setCapability(MobileCapabilityType.PLATFORM\_NAME, "Android");

capabilities.setCapability(MobileCapabilityType.PLATFORM\_VERSION, "2.3");

capabilities.setCapability(MobileCapabilityType.DEVICE\_NAME, "Android Emulator");

capabilities.setCapability(MobileCapabilityType.APP, myApp);

capabilities.setCapability(MobileCapabilityType.APP\_PACKAGE: "com.mycompany.package");

capabilities.setCapability(MobileCapabilityType.APP\_ACTIVITY: ".MainActivity");

Now Appium will start up a Selendroid test session instead of the default test session. One of the downsides to using Selendroid is that its API differs sometimes significantly with Appium’s. Therefore we recommend you thoroughly read Selendroid’s documentation before writing your scripts for older devices or hybrid apps.