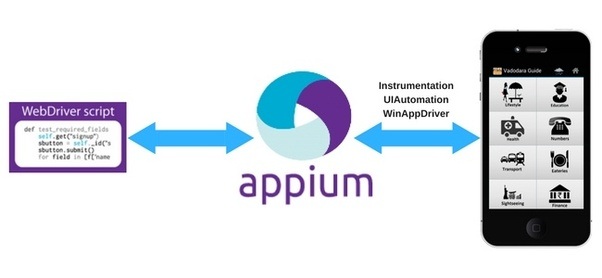
BÁO CÁO NHÓM 01

Version 1.0



**BẢNG GHI NHẬN THAY ĐỔI TÀI LIỆU**

|  |  |  |  |
| --- | --- | --- | --- |
| **Ngày** | **Phiên bản** | **Mô tả** | **Người thay đổi** |
| 25/03/2018 | 1.0 | Viết báo cáo nội dung mục [1,4] | 1642017 – Phạm Thị Xuân Hiền |
| 01/04/2018 | Bổ sung nội dung mục [6] |
| 13/04/2018 | 1.1 | Bồ sung phần [2,3,5] |

MỤC LỤC

[1. Kiểm thử tự động 3](#_Toc511510045)

[1.1 Kiểm thử tự động là gì? 3](#_Toc511510046)

[1.2 Kiểm thử tự động khi nào? 3](#_Toc511510047)

[1.3 Kiểm thử tự động sử dụng ở đâu? 3](#_Toc511510048)

[1.4 Tại sao phải kiểm thử tự động? 3](#_Toc511510049)

[1.5 Khó khăn khi kiểm thử tự động? 3](#_Toc511510050)

[1.6 Các bước tiến hành kiểm thử tự động? 4](#_Toc511510051)

[2. Mobile Testing 4](#_Toc511510052)

[2.1 Tại sao chúng ta cần mobile testing 4](#_Toc511510053)

[2.2 Thách thức gì khi test trên mobile testing 6](#_Toc511510054)

[2.3 Loại mobile testing 9](#_Toc511510055)

[2.4 Các dạng ứng dụng trên mobile 9](#_Toc511510056)

[3. Tìm hiểu các công cụ kiểm thử tự động trên mobile 10](#_Toc511510057)

[3.1 Appium 10](#_Toc511510058)

[3.2 Robotium 10](#_Toc511510059)

[3.3 MonkeyRunner 10](#_Toc511510060)

[3.4 UI Automator 10](#_Toc511510061)

[3.5 Selendroid 11](#_Toc511510062)

[3.6 MonkeyTalk 11](#_Toc511510063)

[3.7 Testdroid 11](#_Toc511510064)

[3.8 Calabash 11](#_Toc511510065)

[3.9 Frank 11](#_Toc511510066)

[3.10 SeeTest 11](#_Toc511510067)

[4. Appium 12](#_Toc511510068)

[4.1 Appium là gì? 12](#_Toc511510069)

[4.2 So sánh Appium với công cụ kiểm thử tự động khác 12](#_Toc511510070)

[4.3 Ưu điểm của Appium 12](#_Toc511510071)

[4.4 Nhược điểm của Appium 13](#_Toc511510072)

[4.5 Cách thức hoạt động của Appium 13](#_Toc511510073)

[4.6 Appium Inspector 13](#_Toc511510074)

[4.7 Các phần mềm cài đặt 14](#_Toc511510075)

[4.7.1 Java Development Kit (JDK) 14](#_Toc511510076)

[4.7.2 Android SDK 18](#_Toc511510077)

[4.7.3 Appium Server 23](#_Toc511510078)

[4.7.4 Appium Client 26](#_Toc511510079)

[4.7.5 Selenium 27](#_Toc511510080)

[4.7.6 Genymotion 28](#_Toc511510081)

[4.8 Video hướng dẫn cài đặt Appium 30](#_Toc511510082)

[5. Kịch bản và demo 31](#_Toc511510083)

[5.1 Kịch bản demo 31](#_Toc511510084)

[5.2 Demo 32](#_Toc511510085)

[5.2.1 Appium hướng dẫn viết Test Script 32](#_Toc511510086)

[5.2.2 Appium Data Driven and TestNG 45](#_Toc511510087)

[5.2.3 Appium Record/ PlayBack 45](#_Toc511510088)

[6. Tài liệu tham khảo 46](#_Toc511510089)

# Kiểm thử tự động

## Kiểm thử tự động là gì?

Là sử dụng công cụ để thực hiện tự động một phần hay toàn bộ quy trình kiểm thử và kiểm tra kết quả mà không cần sự can thiệp của con người.

## Kiểm thử tự động khi nào?

* Các trường hợp kiểm thử lặp đi lặp lại để đảm bảo tính năng của phần mềm/ sản phẩm
* Thực hiện các trường hợp mà kiểm thử thủ công khó thực hiện
* Các trường hợp kiểm thử cần tốn nhiều thời gian.

## Kiểm thử tự động sử dụng ở đâu?

* Kiểm thử tự động sử dụng trong các giai đoạn kiểm thử: Unit Testing (kiểm thử đơn vị), Integration Testing (Kiểm thử tích hợp)
* Kiểm thử tự động sử dụng trong các loại kiểm thử: Smoke Testing (kiểm thử khói), Functional Testing (Kiểm thử chức năng), Regression Testing (Kiểm thử hồi quy)
* Kiểm thử tự động sử dụng trong kỹ thuật kiểm thử: Black Box Testing (Kiểm thử hộp đen)

## Tại sao phải kiểm thử tự động?

* Kiểm thử tự động sử dụng các công cụ có thể ghi lại bộ kiểm tra này và phát lại nó theo yêu cầu.
* Tiết kiệm thời gian kiểm thử
* Tự động hóa không cần can thiệp của con người nên có thể chạy tự động kiểm tra mà không cần giám sát
* Tự động tăng tốc độ thực hiện kiểm tra
* Tự động hóa giúp tăng phạm vi kiểm tra
* Kiểm tra bằng tay có thể trở nên nhàm chán và do đó dễ bị lỗi
* Cải thiện độ chính xác
* Nhanh hơn 70% so với kiểm tra thủ công

## Khó khăn khi kiểm thử tự động?

* Test Scripts là mã nguồn
* Xác nhận (Validation) 1 testcase là pass hay fail
* Một số tình huống cần sự can thiệp của con người
* Mất chi phí cho các công cụ tự động hóa như phí bản quyền, bảo trì, tìm hiểu, giáo dục
* Các công cụ kiểm thử tự động mặc dù rất thuận tiện về nhiều phương diện nhưng thực tế dù như thế nào đi chăng nữa thì nó cũng không phải là một công cụ có thể thay thế hoàn toàn quá trình kiểm thử. Để thực hiện các thiếp lập tự động thì vẫn cần có con người, phải bỏ công sức, tiền bạc và thời gian

## Các bước tiến hành kiểm thử tự động?

|  |  |
| --- | --- |
|  | **Bước 1**: Phân tích khả năng áp dụng kiểm thử tự động  **Bước 2**: Lựa chọn công cụ kiểm thử tự động thích hợp  **Bước 3**: Xây dựng môi trường làm việc  **Bước 4**: Viết kịch bản kiểm thử, thực thi và phân tích kết quả |

# Mobile Testing

## Tại sao chúng ta cần mobile testing

Như chúng ta đã biết thì công nghệ điện thoại di động và các thiết bị thông minh hiện nay là xu hướng và cũng là tương lai của thế giới. Mỗi ngày có hàng triệu application được download từ AppStore và Google Play về các thiết bị cá nhân. Các ứng dụng di động rất phong phú đa dạng đáp ứng đủ các nhu cầu về học tập, công việc, chăm sóc sức khỏe hay giải trí của người dùng.

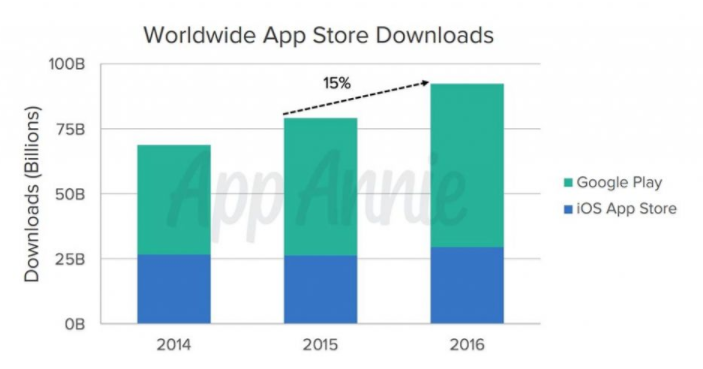
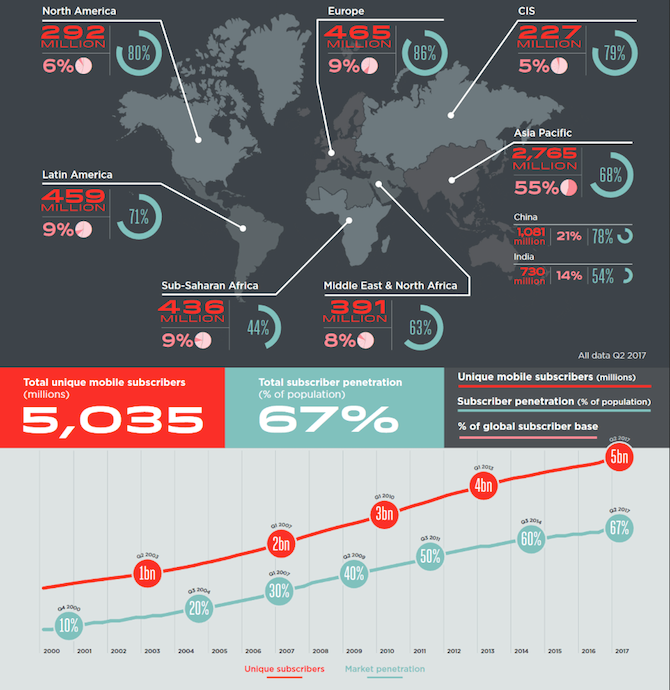




Mốc thời gian phát triển của điện thoại di động:

* Năm 2003: đạt ngưỡng 1 tỷ người dùng di động toàn cầu
* Năm 2007: số người dùng điện thoại di động tăng gấp đôi lên 2 tỷ người
* Năm 2010: tổng số người dùng di động toàn cầu là 3 tỷ người
* Năm 2013: Hơn 4 tỷ người dùng di động

Với xu hướng ngày càng nhiều người sử dụng điện thoại di động, các nhà phát triển ứng dụng di động đã lợi dụng điểm mạnh này cho ra những sản phẩm ứng dụng ngày càng nhiều và số lượng download ứng dụng thường lên đến hàng triệu lượt cho một sản phẩm nào đó.

Nhu cầu của con người mỗi ngày càng tăng họ đòi hỏi các ứng dụng phải có chức năng phức tạp, đa dạng, phong phú hơn => dẫn đến sự cạnh tranh của các nhà phát triển ngày càng cao.

Các nhà phát triển phải làm sao để mà cho ra một ứng dụng nhanh nhất có thể và phải đảm bảo rằng sản phẩm chạy tốt và không lỗi (vì sp bị lỗi không bao giờ được đánh giá cao, thường gây tổn thất về mặt tiền tệ, vấn đề pháp lý và không thể khắc phục thiệt hại về hình ảnh thương hiệu sản phẩm)

* Test Automation For Mobile ra đời (nhờ có nó mà làm cho năng suất của dự án tăng lên nhiều lần không chỉ về chất lượng, chi phí, thời gian mà còn tăng lên cả khả năng delivery (chuyển))

## Thách thức gì khi test trên mobile testing

|  |
| --- |
| 1. **Mobile Device’s**   Mobile có nhiều hãng sản xuất => ra nhiều device thì phải test trên nhiều device khác nhau, không thể bỏ một device nào hết. Hãng sản xuất nhiều, nhiều cái phải test hơn  Test web thì cũng có mấy browser để test thôi. Hệ điều hành cố định không thay đổi nhiều (vài 3 hệ điều hành chính như android, window, ios)  Ví dụ: ứng dụng test trên web thì nhanh nhưng test trên mobile thì phải test nhiều mobile, test nhiều thứ, thời gian test nhiều hơn  C:\Users\SONY\Desktop\tải xuống.jpg |
| 1. **User’s loyalty (Lòng trung thành của người dùng đối với sản phẩm)**   Ví dụ 1: download app chụp hình về, mới chụp xong chỉnh sửa ảnh, mới chỉnh sửa xíu mà thấy pin tụt nhiều và nóng lên. Cảm giác app không an toàn thì remove liền, tải app khác.  Ví dụ 2: download app thanh toán online, mình có cảm giác nó cứ lấy thông tin mình, cảm giác không an tâm, mình cũng bỏ qua, mình không xài app đó nữa   * Lòng chung thành của khách hàng rất là cao   Mỗi người có một đánh giá về chất lượng khác nhau, không ai giống ai, nhưng thường những tính năng mà người ta đeo đuổi miết nghĩa là app mình phụ thuộc hệ điều hành nào, các tương thích các hành vi nó giống hệ điều hành đó.  Ví dụ: android mình kéo nó vào thùng rác mình delete, còn iOS thì mình nhấn vào bấm nút chéo tự delete, nên mình thấy không có nút đó mình không muốn xài  iOS hay dựa vào cái tôi của mỗi người, nên sản phẩm nó bán rất là chạy và cạnh tranh rất là nhiều so với Android  C:\Users\SONY\Desktop\lòng trung thành.jpgC:\Users\SONY\Desktop\how-to-build-customer-loyalty.png |
| 1. **Mobile Platforms (nền tảng di động)**   Mobile có rất là nhiều hệ điều hành, nhưng thực chất mình chỉ test trên hệ điều hành chính Android và iOS. Một hệ điều hành nó thì có rất là nhiều version. Có những app chạy trên version này thì được, chuyển sang version khác thì không chạy được  Ví dụ: HĐH android, thì có version 4.0, 4.1, 4.2, 4.3, 4.4, 5, 6, 8.0, 8.1 => rất là nhiều  HĐH Mac, Linus, Window thì không có nhiều  C:\Users\SONY\Desktop\mobile platform.pngC:\Users\SONY\Desktop\cross-platform-mobile-apps-development-612x252.png |
| 1. **Mobile Device Screen Sizes**   Kích thước thì có nhiều màn hình kích thước khác nhau  Ví dụ:Trên iOS thì có Iphone, iphone bao gồm iphone 4, iphone 5, iphone 6, iphone 6 plus, iphone 7, iphone 7 plus, iphone 8, iphone X => thì mình biết bao nhiêu loại thì mình test  Còn Android, hãng HTC kích cỡ làm khác hãng Samsung làm khác. Mỗi lần bảng build trên Samsung chạy oke, qua HTC nó bể có khi mất hẵn cả button hay nó bị mất luôn làm khách hàng ra không được và cũng không được. Cảm giác rất ức chế  C:\Users\SONY\Desktop\ios size 2.png  Hình ảnh 1: size iOS  C:\Users\SONY\Desktop\size android.PNG  Hình ảnh 2: size android |

## Loại mobile testing

* Functional testing - Kiểm thử chức năng
* Performance testing - Kiểm thử hiệu suất
* Security testing - Kiểm thử bảo mật
* Usability testing - Kiểm thử tính khả dụng
* Compatibility testing - Kiểm thử khả năng tương thích
* Recoverability Testing - Kiểm thử khả năng khôi phục

## Các dạng ứng dụng trên mobile

* + **Native Application**: các ứng dụng này được phát triển trên một nền tảng cụ thể và được cài trên thiết bị (gọi là ứng dụng cơ bản nghĩa là ứng dụng được viết trên một hệ điều hành nào đó)

Ví dụ: Android: ngôn ngữ Java

IOS: ngôn ngữ Objective – C, Swift

Window Phone: ngôn ngữ C# kết hợp với XAML

1 ứng dụng viết ra chạy trên được 1 hệ điều hành thôi. Hay ở chỗ nó chạy rất là nhanh, nó truy cập vào các thiết bị của mình ví dụ như volume, camera… rất là dễ, vì nó viết trên nền tảng của HĐH đó

* + **Web Application**: các ứng dụng được truy cập thông qua trình duyệt của thiết bị (chạy trên browser máy nào cũng chạy được)

Được tạo bởi html và code ở back end (PHP, [C#](https://toidicodedao.com/tag/c-hay-ho/), Java, …)

Yếu điểm: nó sẽ không thể truy cập được vào các thiết bị device như camera, volume… của mình, can thiệp vào hệ điều hành rất là khó (không thể truy cập vào máy mình và không thể can thiệp vào hệ điều hành)

Ví dụ: mình muốn button lớn hơn hoặc nhỏ hơn mà HDH sinh ra đó nhưng mà không thể convert được

Ưu điểm: devide hệ điều hành nào cũng xài được

* + **Hydrid Application**: là loại ứng dụng kết hợp các yếu tố của cả Native App và Web App

Viết 1 lần nhưng có thể build ra và dùng cho nhiều hệ điều hành



# Tìm hiểu các công cụ kiểm thử tự động trên mobile

## **Appium**

Appium là một công cụ kiểm thử tự động mã nguồn mở để kiểm thử các ứng dụng Android và iOS. Developers có thể kiểm thử các ứng dụng mobile, web mobile, ứng dụng hybrid trên phần mềm này. Để chạy thử nghiệm, Appium sử dụng giao diện WebDriver. Nó hỗ trợ C#, Java, Ruby và nhiều ngôn ngữ khác thuộc về thư viện WebDriver. Tester có thể kiểm thử các ứng dụng gốc được viết bằng SDK Android và iOS, ứng dụng web mobile, và ứng dụng hybrid có chứa web view. Là một công cụ nền tảng chéo, nó cho phép các lập trình viên tái sử dụng mã nguồn giữa Android và iOS.

## Robotium

Là một công cụ mã nguồn mở cho phép kiểm tra các ứng dụng Android của tất cả các phiên bản. Nó kiểm tra các ứng dụng Native và Hybrid. Việc kiểm thử của Robotium được viết bằng Java. Sử dụng công cụ này, bạn có thể dễ dàng viết các trường hợp kiểm thử hộp đen tự động mạnh mẽ cho các ứng dụng Android. Nó tự động hóa nhiều hoạt động của Android và tạo các trường hợp kiểm thử vững chắc trong thời gian tối thiểu.

## MonkeyRunner

MonkeyRunner được thiết kế đặc biệt để thử nghiệm các thiết bị và ứng dụng ở cấp khung/chức năng. Công cụ này có các tính năng tuyệt vời như kiểm soát nhiều thiết bị, kiểm thử hồi quy, mở rộng tự động hóa, và thử nghiệm chức năng để kiểm tra các ứng dụng và phần cứng Android. Kiểm thử của MonkeyRunner được viết bằng ngôn ngữ Python. Developer không cần thay đổi mã nguồn để tự động kiểm thử.

## UI Automator

Để thử nghiệm giao diện người dùng của app, UI Automator tạo ra các trường hợp thử nghiệm UI trên Android. Gần đây, nó đã được mở rộng bởi Google. Công cụ này liên tục tương tác với các phần mềm và ứng dụng Android. Nó hoạt động với tất cả các thiết bị hỗ trợ Android từ phiên bản 4.1 trở lên. Trong các phiên bản trước, tester cần phải sử dụng các công cụ khác. Ngoài ra, UI Automator có thể khóa và mở khóa máy tính bảng hoặc smartphone.

## Selendroid

Là một trong những phần mềm kiểm thử tự động hàng đầu, Selendroid kiểm thử UI của Android dựa trên các ứng dụng và web mobile native và hybrid. Các kiểm tra Client API được viết bằng Selendroid 2. Công cụ này hỗ trợ cắm các thiết bị phần cứng. Hơn nữa, nó có khả năng đặc biệt để tương tác với nhiều thiết bị Android cùng một lúc. Selendroid tương thích cao với các giao thức dây JSON.

## MonkeyTalk

MonkeyTalk tự động kiểm tra chức năng của các ứng dụng Android và iOS. Người phi kỹ thuật cũng có thể chạy thử nghiệm trên nền tảng này vì nó không đòi hỏi kiến thức chuyên sâu về lập trình và kịch bản. Các kịch bản của MonkeyTalk khá dễ hiểu và đơn giản. Tester cũng có thể tạo báo cáo XML và HTML bằng công cụ này. Ngoài ra, nó có ảnh chụp màn hình khi thất bại xảy ra. MonkeyTalk hỗ trợ trình giả lập, thiết bị mạng và tethered.

## Testdroid

Đây là chương trình dựa trên đám mây dành cho thử nghiệp ứng dụng di động, giúp các nhà phát triển tiết kiệm chi phí phát triển, loại bỏ chi phí hoạt động không lường trước được và cải thiện thời gian đưa ra thị trường. Đây là một trong những nền tảng tốt nhất để kiểm tra thiết bị iOS và Android có độ phân giải màn hình khác nhau. Testdroid là một công cụ giúp làm giảm nguy cơ bằng các kiểm thử thiết bị nhanh nhẹn. Nó cũng cải thiện đánh giá của người dùng hàng ngày.

## Calabash

Calabash hoạt động hiệu quả với .NET, Ruby, Flex, Java và các ngôn ngữ lập trình khác. Nó kiểm tra các ứng dụng di động native và hybrid. Lập trình viên có thể có các API cho phép các ứng dụng gốc chạy trên các thiết bị cảm ứng. Calabash liên quan đến các thư viện cho phép mã kiểm tra tương tác với các ứng dụng hybrid và native. Nó cũng hỗ trợ framework Cucumber.

## Frank

Frank cho phép chỉ kiểm tra các ứng dụng iOS. Framework kết hợp JSON và Cucumber. Nó chứa một thanh tra ứng dụng “Symbioate” cho phép các nhà phát triển có thông tin chi tiết về ứng dụng đang chạy. Nó phù hợp nhất cho ứng dụng dựa trên web và emulator. Nó có thể tích hợp với CI và chạy thử nghiệm trên các thiết bị và simulator.

## SeeTest

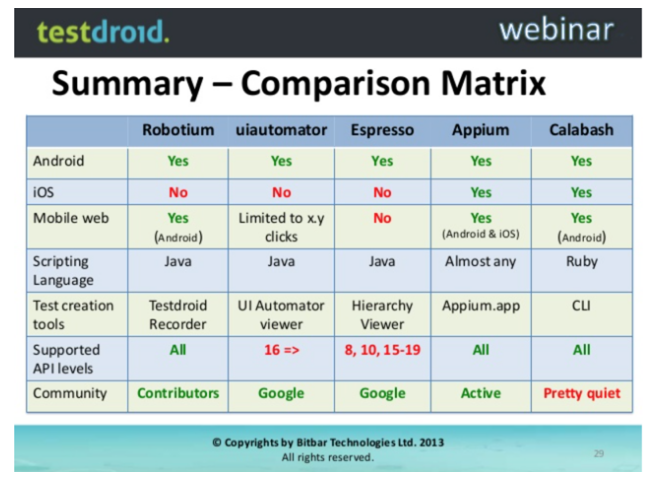
SeeTest Automation là một giải pháp nền tảng. Nó cho phép chạy cùng một kịch bản trên các thiết bị khác nhau. Nó cho phép các nhà phát triển chạy thử nghiệm trên một số thiết bị song song. Là một công cụ kiểm thử tự động mạnh mẽ, nó có khả năng kiểm tra website, mobile app. Nó hỗ trợ iOS, Android, Symbian, BlackBerry, và Window Phone. Các tính năng quan trọng nhất của công cụ này là thử nghiệm điện thoại, pin, và trình duyệt, …

# Appium

## Appium là gì?

* Appium là một công cụ mã nguồn mở được sử dụng để kiểm thử tự động các ứng dụng trên di động (Native App, Mobile Web App và Hydrid App)
* Hỗ trợ cả thiết bị trên nền tảng iOS và Android.
* Appium không phụ thuộc vào hệ điều hành của thiết bị di động, vì vậy nó cho phép chúng ta tự do sử dụng các loại thiết bị di động mà không phải quan tâm hệ điều hành của chúng là gì. Appium làm được điều này bởi vì framework của nó sẽ biên dịch các lệnh Selenium Web Driver thành các lệnh UIAutomator (đối với Android) hay UIAutomation (đối với iOS). Do đó, nó không phụ thuộc vào loại thiết bị hơn là hệ điều hành.
* Đặc biệt, Appium hỗ trợ “đa nền tảng” (Cross Platform) cho phép bạn sử dụng API giống nhau để viết test cho các nền tảng khác nhau (iOS và Android). Điều này khá là tiện lợi khi bạn muốn sử dụng lại các Test Suites của mình.
* Appium hỗ trợ kiểm thử tự động phần mềm trên cả Emulator, Simulator và Devices.
* Bên cạnh đó, Appium hỗ trợ viết Test cho rất nhiều ngôn ngữ lập trình (Java, Ruby, Python, PHP, Javascript…)

## So sánh Appium với công cụ kiểm thử tự động khác



## Ưu điểm của Appium

* Appium là một tool mã nguồn mở được xây dựng và đóng góp tích cực bởi cộng đồng lớn.
* Appium là một ứng dụng đơn giản và nhẹ, nó sử dụng rất ít bộ nhớ trong quá trình test. Kiến trúc đơn giản hoạt động như một proxy giữa thiết bị test và bộ Automation Tool – Kit.
* Test trên các ứng dụng Native không cần có SDK bởi vì Appium được tích hợp với các API tiêu chuẩn dùng trong kiểm thử tự động và chạy được trên nhiều nền tảng để chứng minh cho triết lí tái sử dụng và thích ứng với các tiêu chuẩn hiện có của nó.
* Test scripts có thể viết bằng nhiều ngôn ngữ tùy theo sở thích tương thích với Selenium WebDriver như Java, C#, Ruby, Javascript with Node.js, PHP, Python hay Perl và có thể sử dụng bất kỳ testing framework nào.
* Các lợi ích khác của Appium là nó mở ra cho các Cross – Platform Mobile Testing có nghĩa là cùng một thử nghiệm sẽ làm việc trên nhiều nền tảng.
* Có thể tự động hóa các Web, Hydrid và Native Mobile Applications.

## Nhược điểm của Appium

* Appium có những giới hạn về mặt kỹ thuật. Ví dụ: trong iOS chỉ có thể chạy 1 phiên bản trên các thiết bị trên mỗi hệ điều hành Mac.OS. Chạy script rất chậm trên platform iOS. Hỗ trợ hành động cử chỉ có giới hạn
* Appium hỗ trợ Android phiên bản từ 4.2 trở lên (tức là Android API level > 17). Các phiên bản thấp hơn, Appium sử dụng một thư viện mã nguồn mở khác là Seledroid thay vì UIAutomator.
* Không hỗ trợ Toast Message

## Cách thức hoạt động của Appium

* Appium sử dụng kiến trúc Client – Server. Khi kết nối được thiếp lập, nó lắng nghe các lệnh và thực thi các lệnh đó trên thiết bị được kết nối và trả về kết quả thông qua HTTP Response. Bởi vì, sử dụng kiến trúc Client – Server nên người dùng có quyền tự viết Test Script bằng bất kỳ ngôn ngữ nào hỗ trợ HTTP Client API.
* Đối với nền tảng iOS, Appium sử dụng thư viên UIAutomation, trong khi đó cho phiên bản mới hơn của nền tảng Android, Appium sử dụng UIAutomator framework. Ngoài ra Appium còn hỗ trợ các phiên bản cũ hơn của Android bằng cách sử dụng Selendroid. Còn đối với windows, nó sử dụng WinAppDriver của Microsoft.
* Các thư viên hỗ trợ của ngôn ngữ lập trình phía Client như C#, Java, Ruby, PHP hay Python hỗ trợ Appium Extensions cho Selenium WebDriver Protocol, vì vậy thay vì WebDriver Client, người dùng có quyền sử dụng các thư viện của ngôn ngữ hỗ trợ.

## Appium Inspector

Appium có một công cụ hỗ trợ rất tiện lợi gọi là Appium Inspector dùng để Record và Playback. Nó hiểu được các thành phần phân cấp. Giống như các công cụ kiểm thử tự động khác, nó có thể ghi lại và thực hiện lại tất cả các hành động của ứng dụng Native và tạo ra một Test Scripts từ đó.

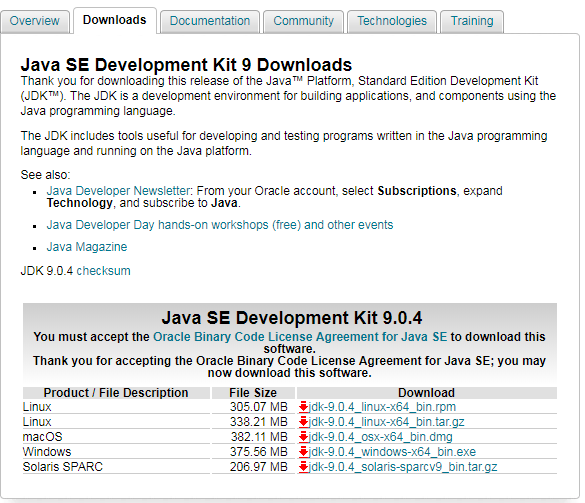
Do đó, bất cứ khi nào người dùng muốn ghi lại những hành động thao tác bằng tay thành một Scripts hay muốn tìm ra giá trị, tên, mô tả hay những thuộc tính khác của thành phần được sử dụng trên lệnh XPath, Appium Inspector sẽ giải quyết được những vấn đề đó.

## Các phần mềm cài đặt

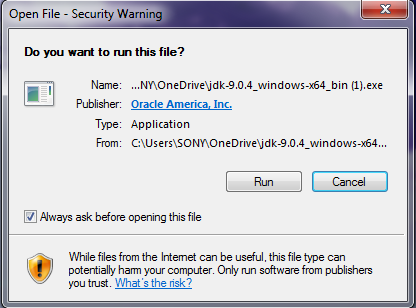
### **Java Development Kit (JDK)**

1. **Link download Java Development Kit (JDK)**

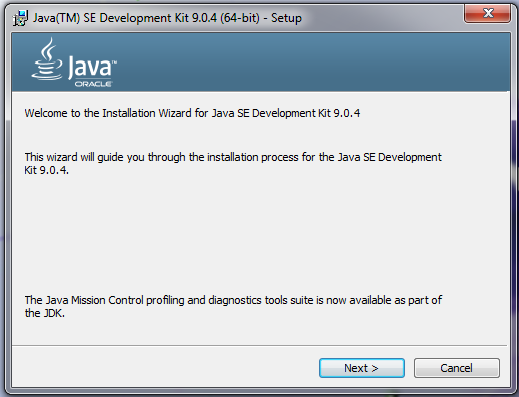
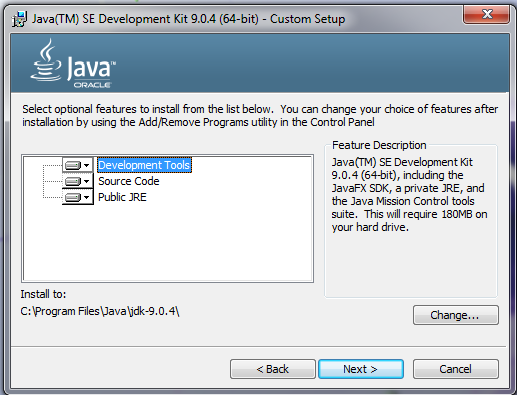
<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

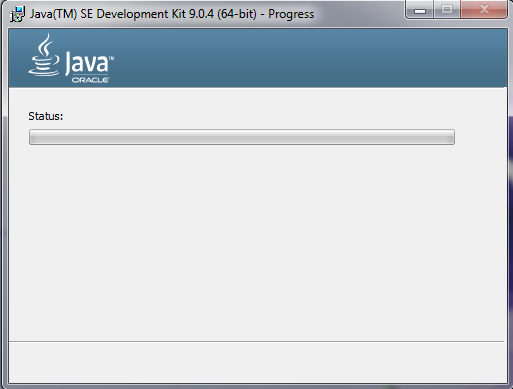


1. **Cài đặt Java Development Kit (JDK)**
2. Chọn “**Run**”



1. Chọn “**Next**”

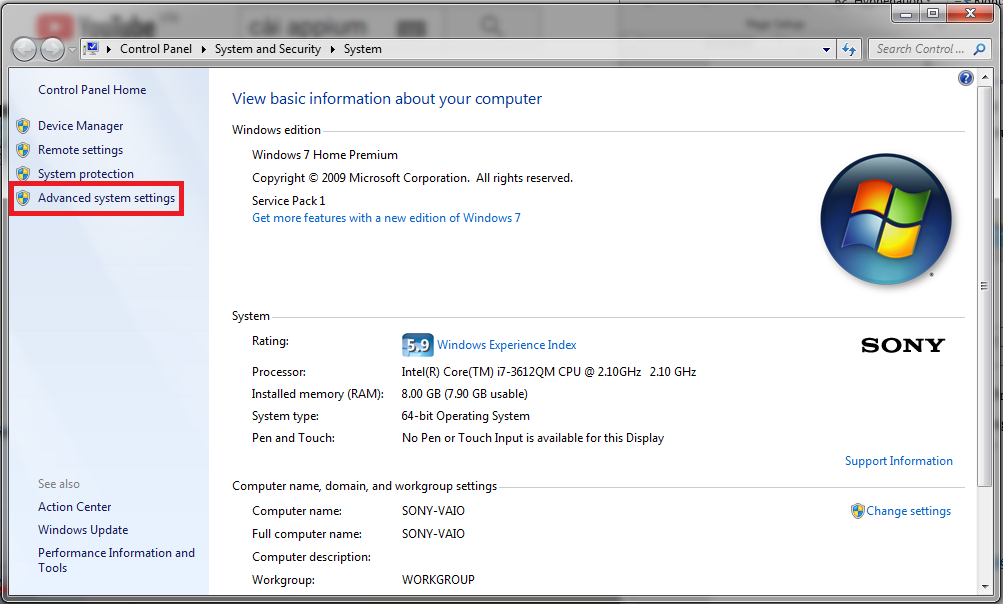


1. Chọn “Close” cài đặt JDK thành công



**Sau khi cài đặt JDK, ta tạo biến môi trường theo các bước**:

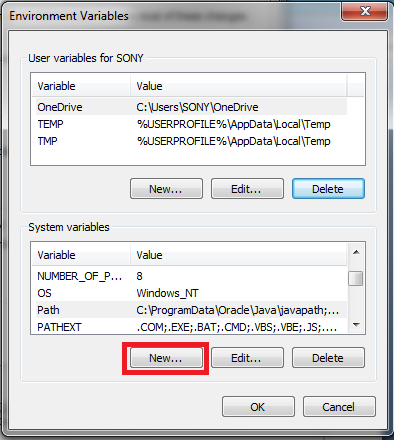
1. Chuột phải My Computer -> Properties
2. Chọn tab “**Advanced system settings**”



1. Chọn “**Enviroment Variables**”

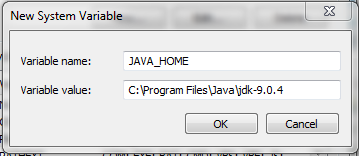


1. Phía dưới mục **System Variables** chọn **New**

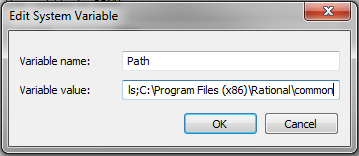


1. Dòng **Variable Name**: JAVA\_HOME

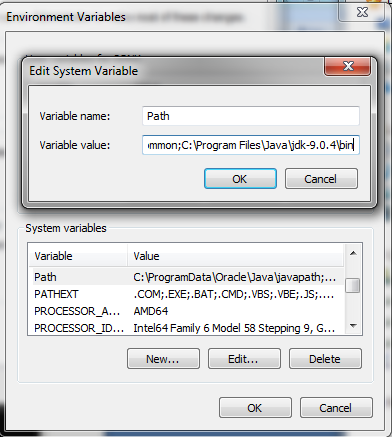
Dòng **Variable Value**: C:\Program Files\Java\jdk-9.0.4 (là đường dẫn đến thư mục jdk mà đã cài khi nãy. Window 64 bit là **Program Files**, 32 bit là **Program Files(x86)**)



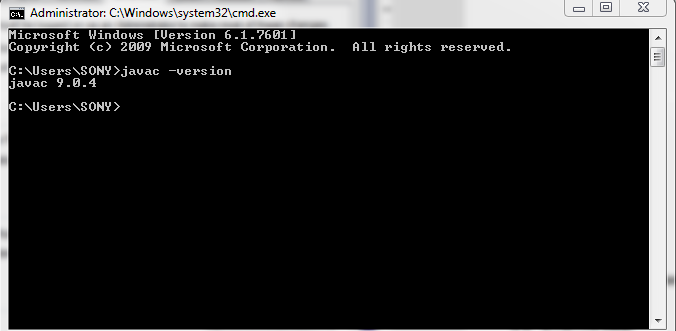
1. Trong phần System Variables này ta chọn **Path**, chọn **Edit**



1. Điền thêm đoạn **“;C:\Program Files\Java\jdk-9.0.4\bin”** hoặc **“;%JAVA\_HOME%\bin”**vào sau cuối

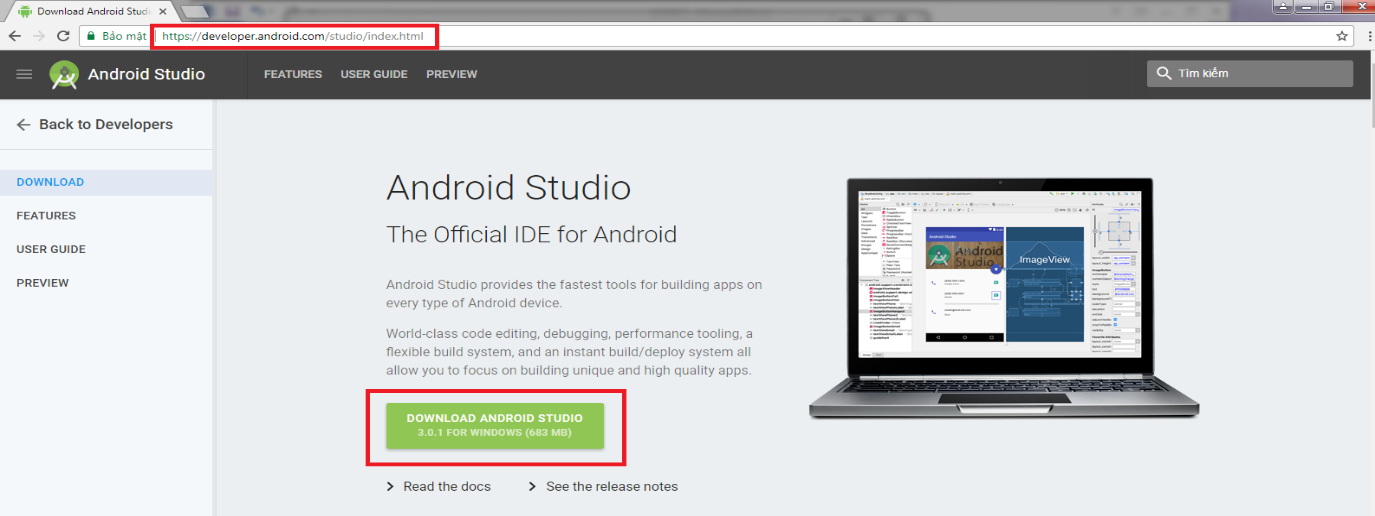


1. Kiểm tra: vào **Start** gõ **cmd** sau đó **Enter**. Cửa sổ cmd hiện ra, ta gõ “**javac -version**”



### **Android SDK**

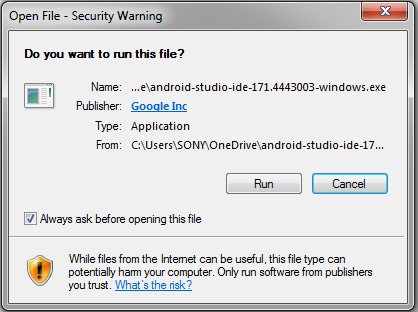
1. Link download Android Studio: <https://developer.android.com/studio/index.html>



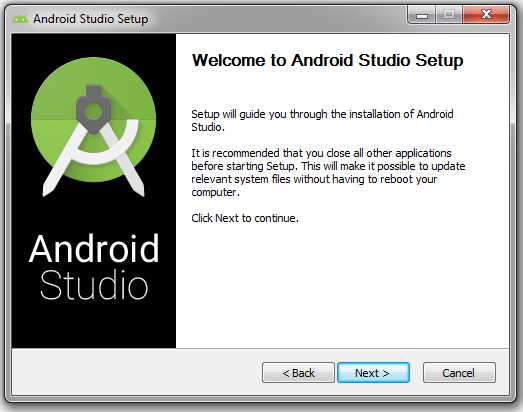
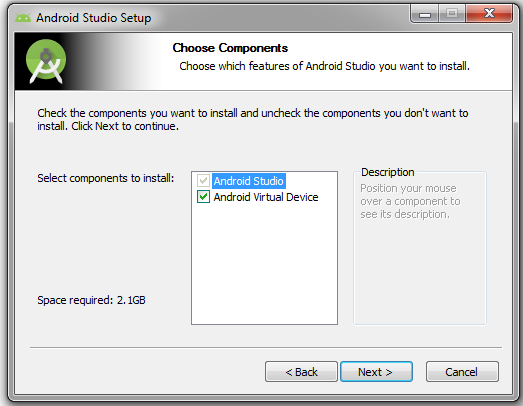
1. Sau khi download xong ta có file.

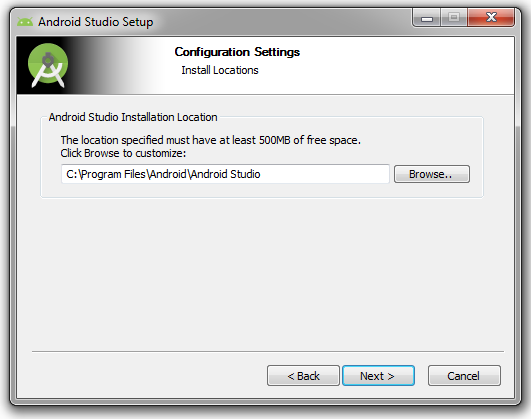


1. Tiến hành cài đặt, chọn “**Run**”

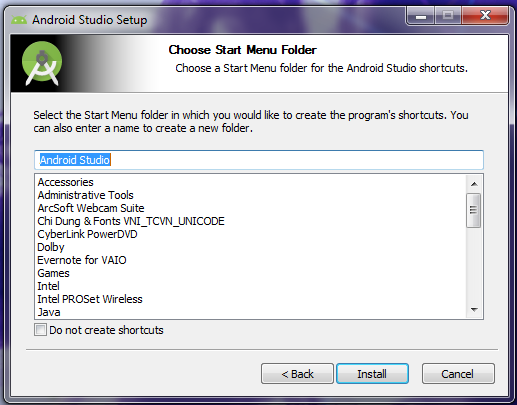
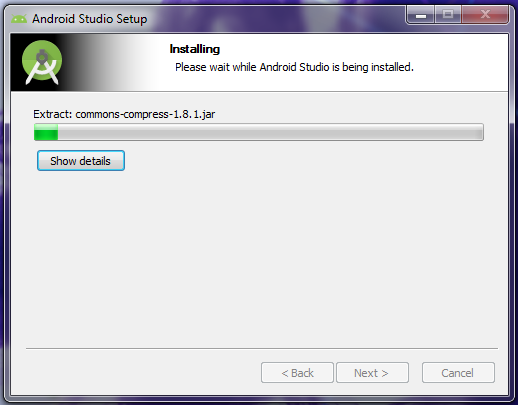


1. Chọn “**Next**”

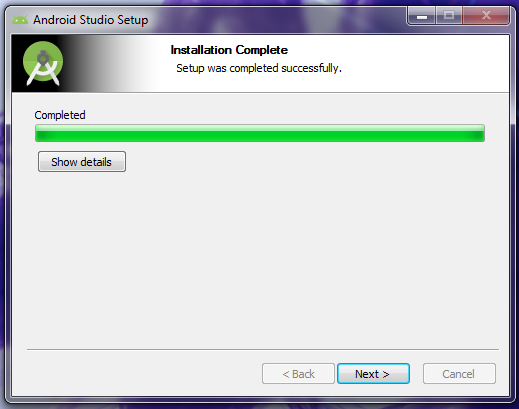
 



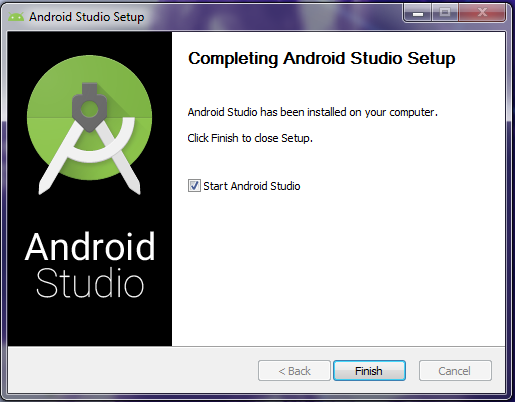
1. Chọn “**Install**”

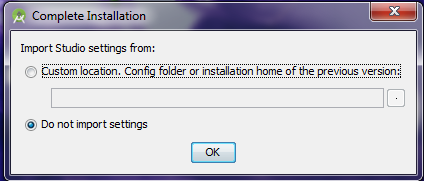
1. Chọn “**Next**”



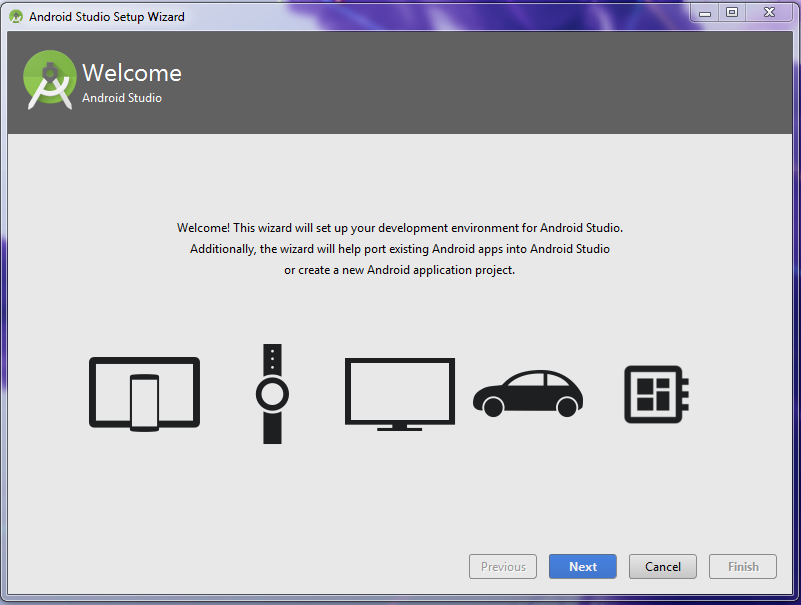
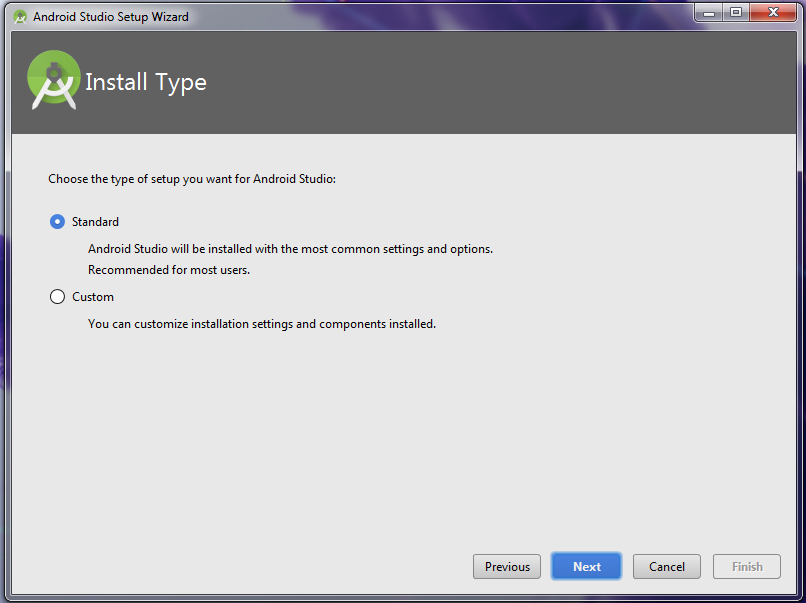
1. Chọn “**Finish**”, kết thúc quá trình cài đặt.

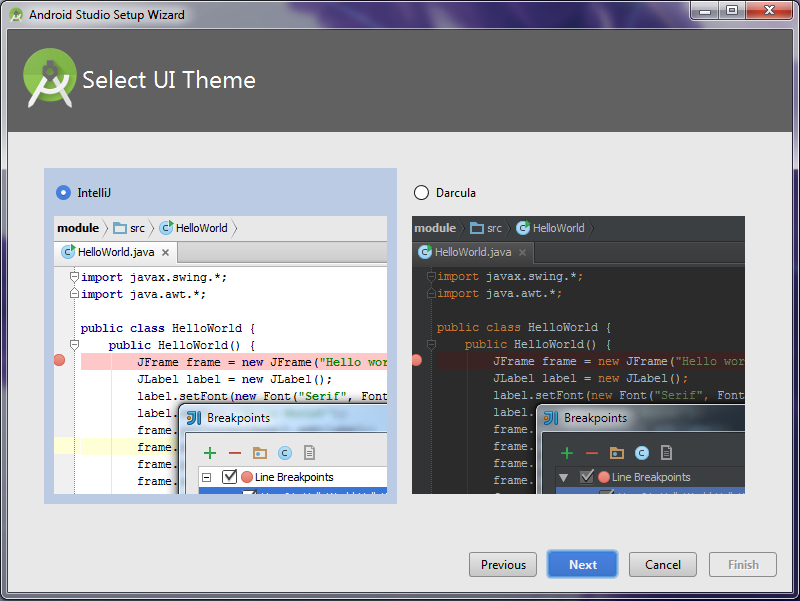


1. Chọn “**OK**”

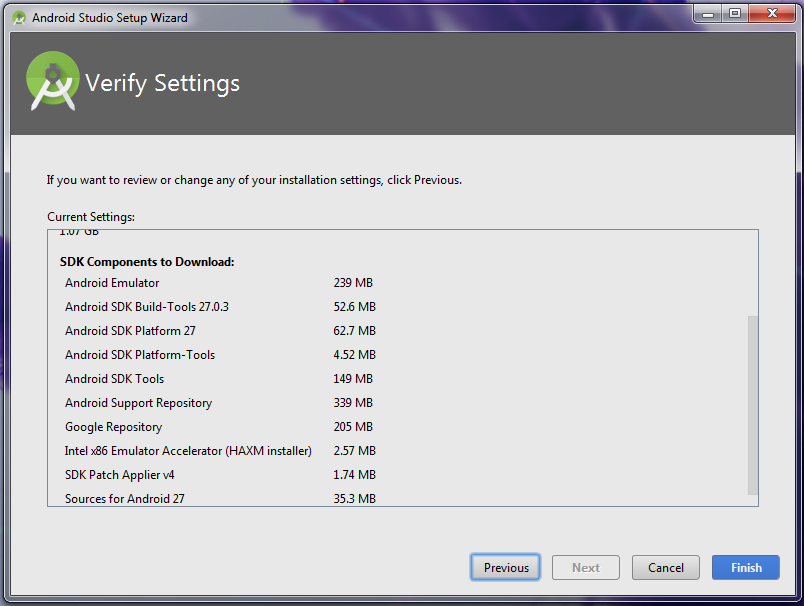
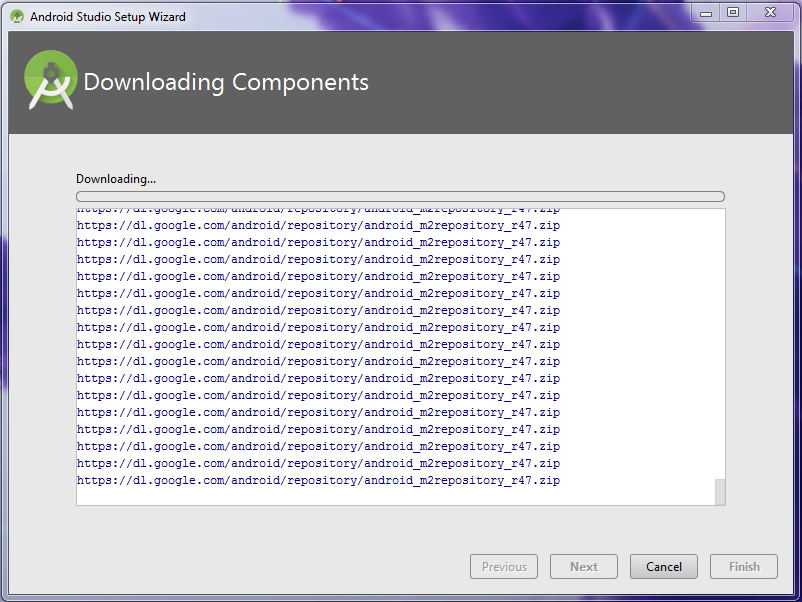


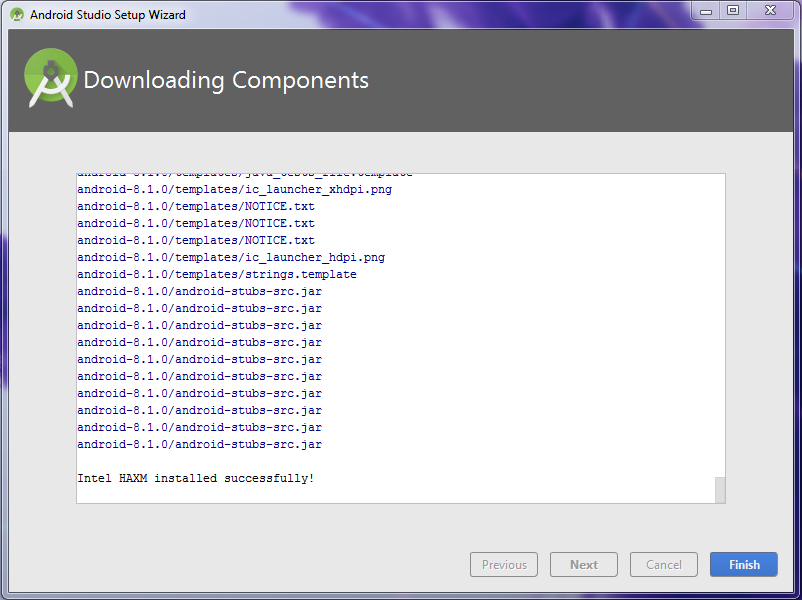
1. Chọn “**Next**”



1. Chọn “**Finish**”

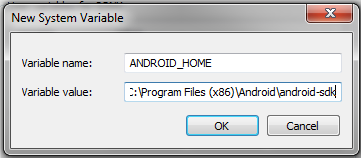
 



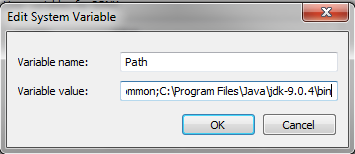
1. Thêm biến môi trường ANDROID\_HOME

Dòng **Variable Name**: ANDROID\_HOME

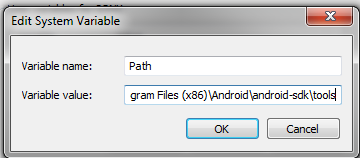
Dòng **Variable Value**: “C:\Program Files (x86) \Android\android-sdk” (là đường dẫn đến thư mục sdk mà đã cài khi nãy)



1. Trong phần System Variables này ta chọn **Path**, chọn **Edit**

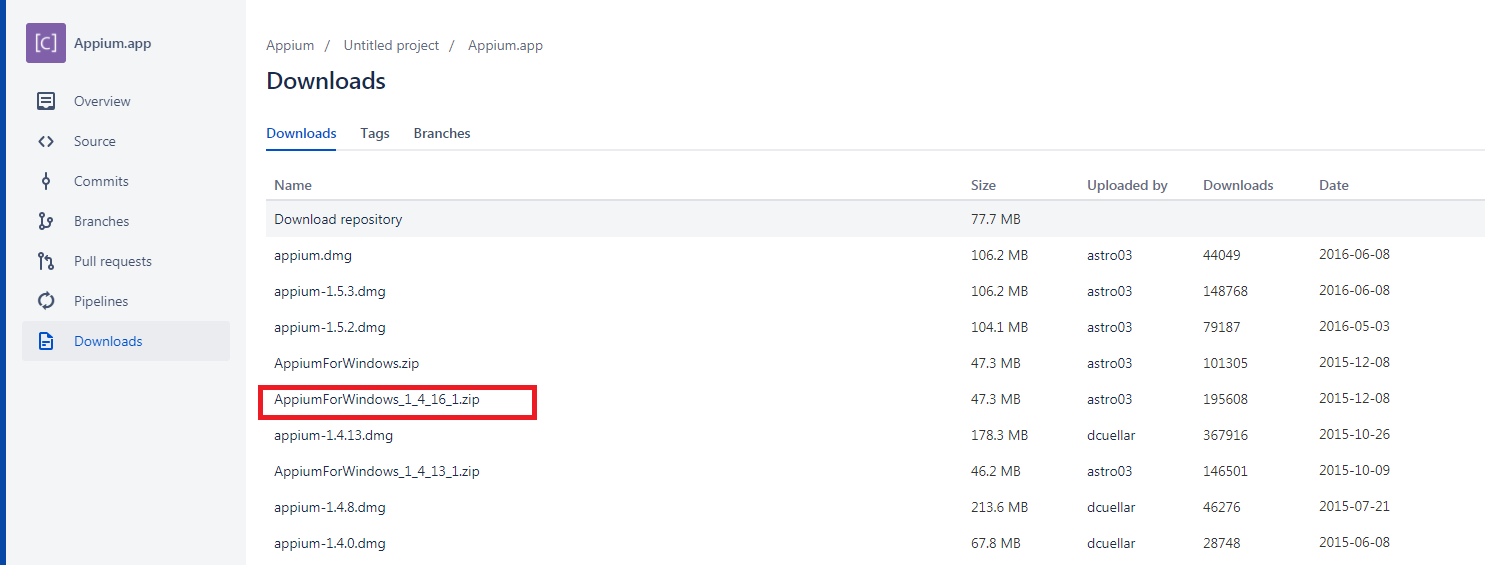


1. Điền thêm đoạn **“C:\Program Files (x86)\Android\android-sdk\platform-tools;C:\Program Files (x86)\Android\android-sdk\tools”** hoặc **“;%ANDROID\_HOME\ platform-tools**%;**;%ANDROID\_HOME\ tools**” vào sau cuối

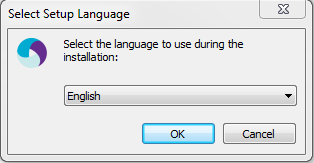


### **Appium Server**

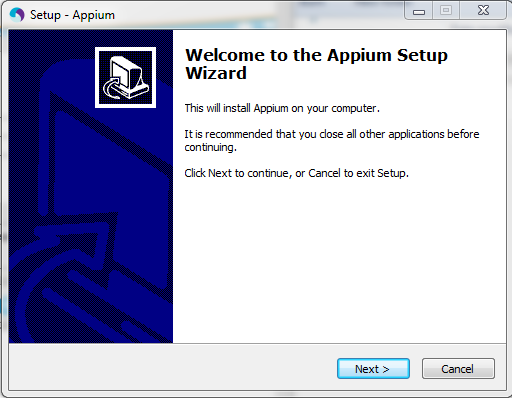
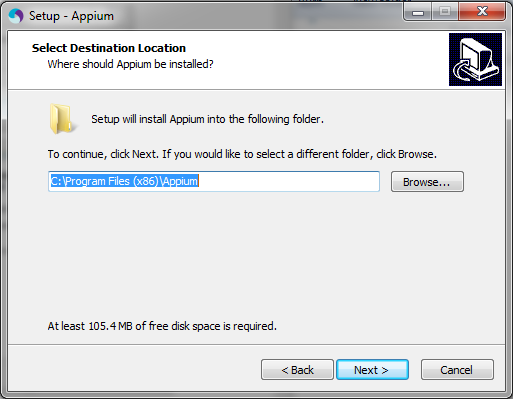
1. Link download Appium: <https://bitbucket.org/appium/appium.app/downloads/>
2. Download “**AppiumForWindows\_1\_4\_16\_1.zip**”

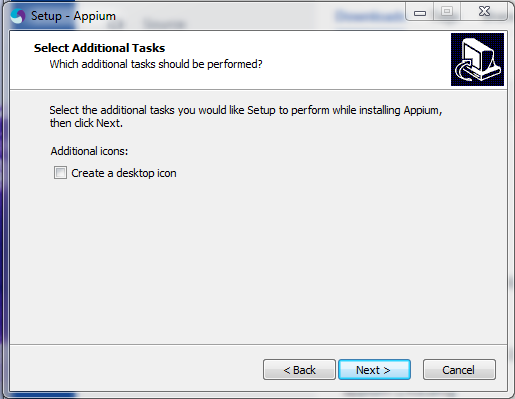


1. Chọn **OK**

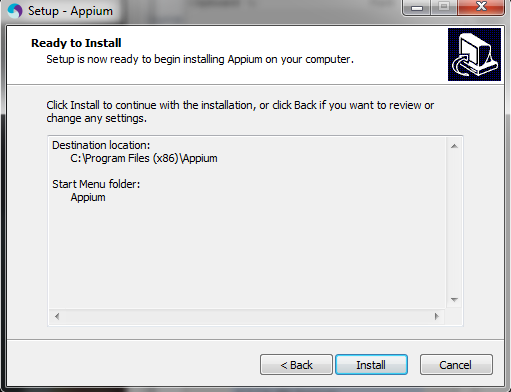
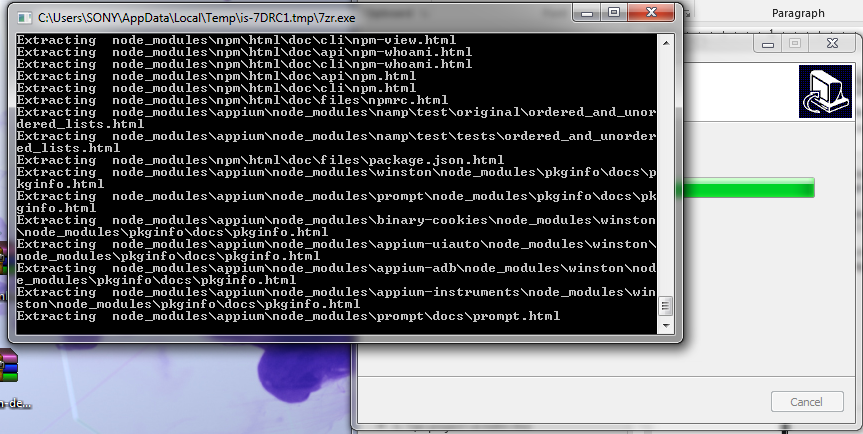


1. Chọn “**Next**”

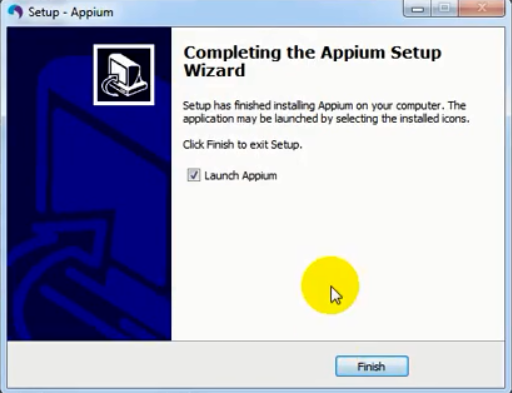
 

1. Chọn “**Install**”

1. Chọn “**Finish**”.



**Cài đặt thành công Appium**



Khởi động hoặc tắt Appium Server

Xóa các log

Thông tin phần mềm

Trình giám sát các bước chạy khi Test

Các thiết lập cho Developer

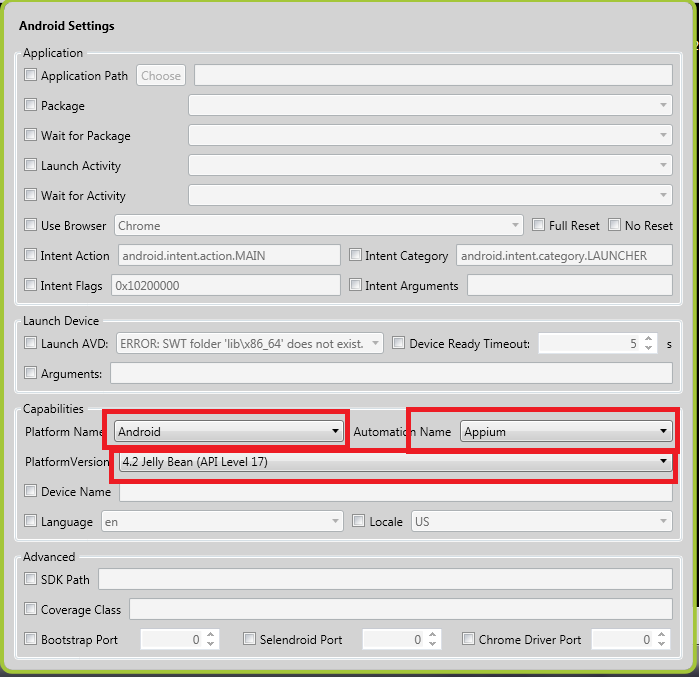
Các thiết lập cho Appium (Server, Port…)

Các tùy chỉnh thông tin cho thiết bị Android

Sau khi cài đặt thành công, mở ứng dụng Appium và cấu hình cho Appium server như sau:

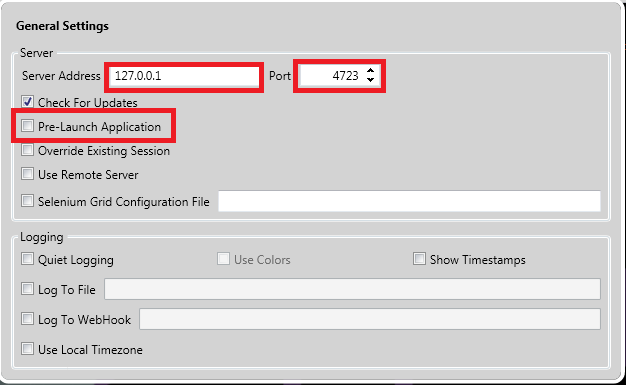
1. Ở tab **“Android Settings”**

* Chọn **Platform Name** = “Android”
* Chọn **Automation Name** = “Appium”
* Chọn **Platform Version** = 4.2 Jelly Bean (API Level 17) (phiên bản hệ điều hành của thiết bị Android được kết nối) hoặc có thể chọn phiên bản khác.



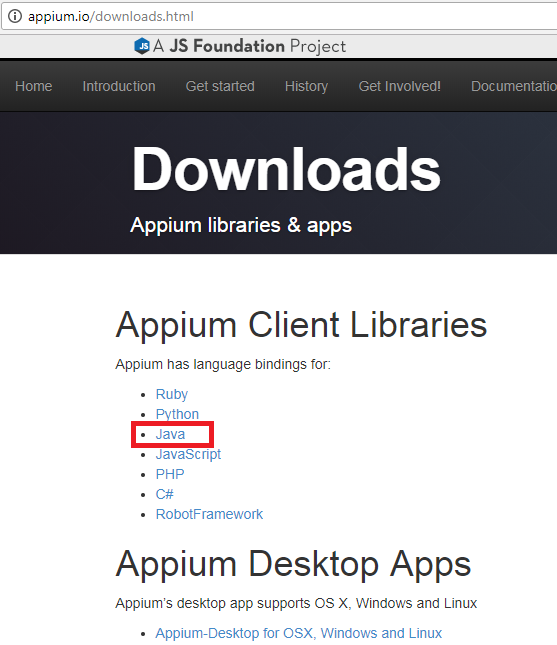
1. Ở tab **“General Settings”**

* Chọn **Server Address =** 127.0.0.1 và **Port** = 4723
* Bỏ chọn ở checkbox “**Pre – Launch Application**” nếu nó được chọn

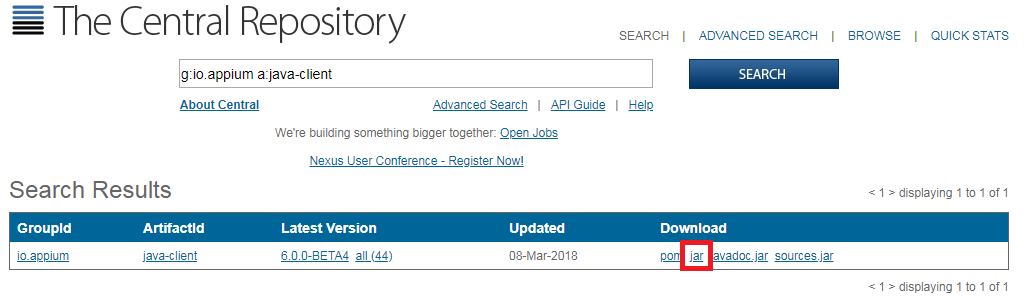


### **Appium Client**

1. Link download: <http://appium.io/downloads.html>
2. Chọn vào Java



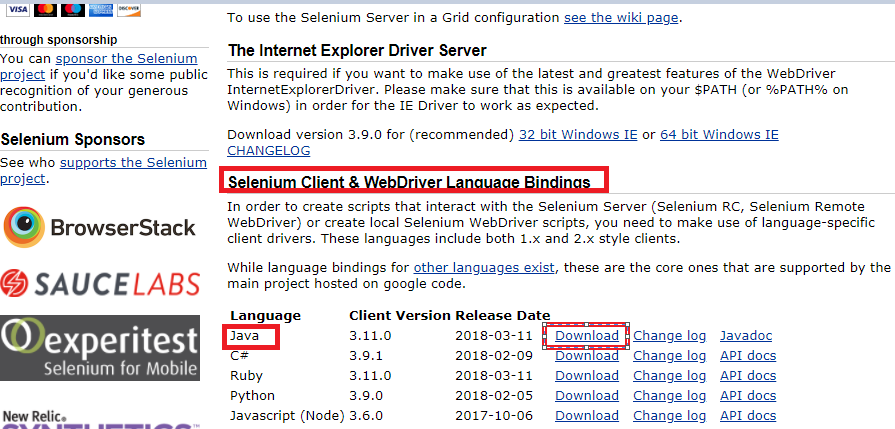
1. Download file.jar



**Lưu ý: File này dùng để add vào project Android khi thực hiện viết Test Script**

### **Selenium**

1. Link download: <https://www.seleniumhq.org/download/>
2. Tìm Selenium Client & WebDriver Language Bindings.



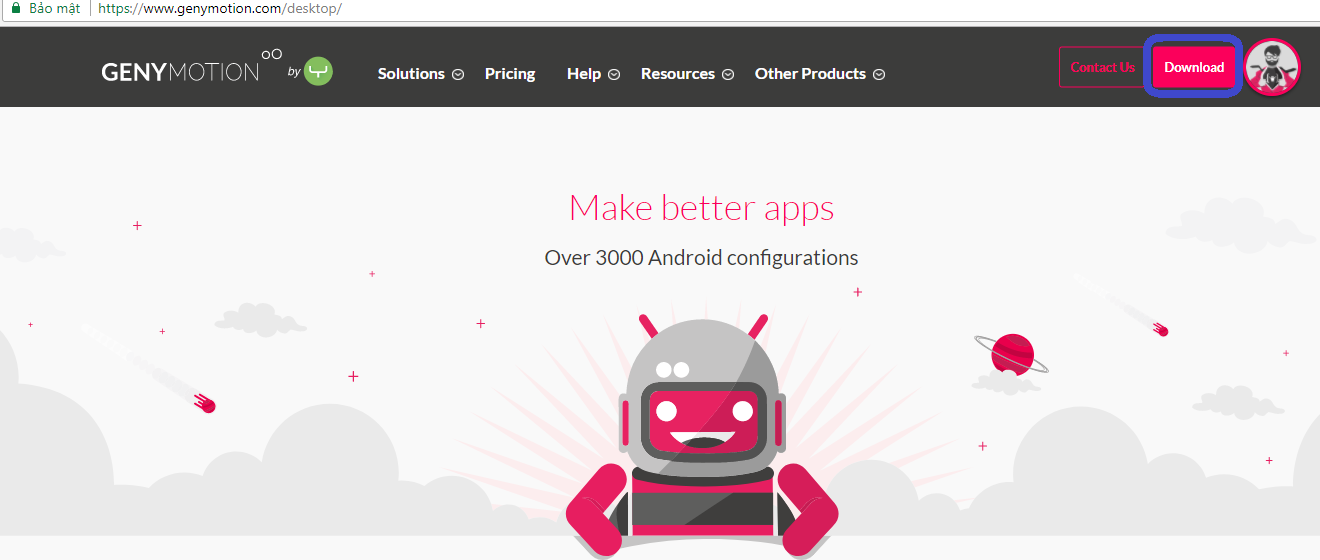
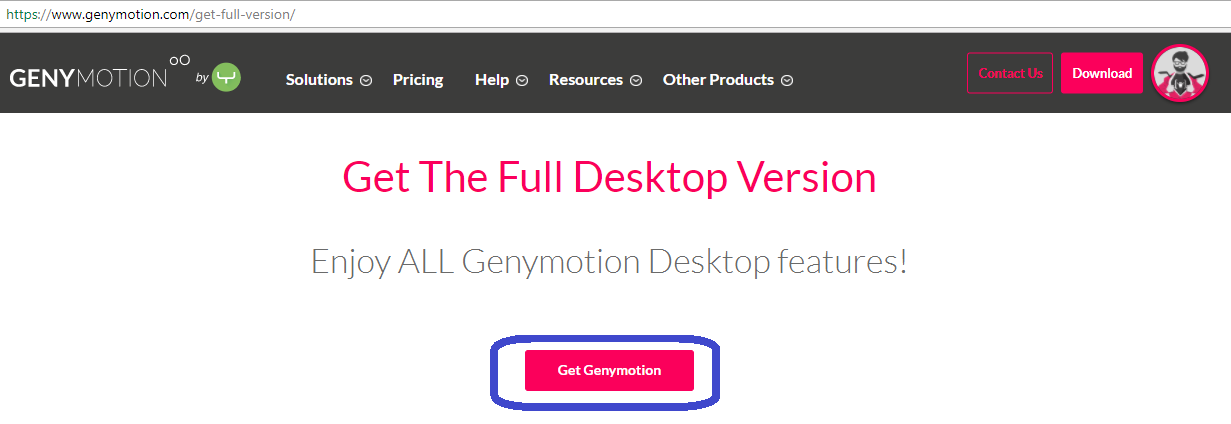
1. Download thư mục .zip và giải nén ra.

**Lưu ý: Các file trong thư mục giải nén là các file.jar dùng để add vào project Android khi thực hiện viết Test Script**

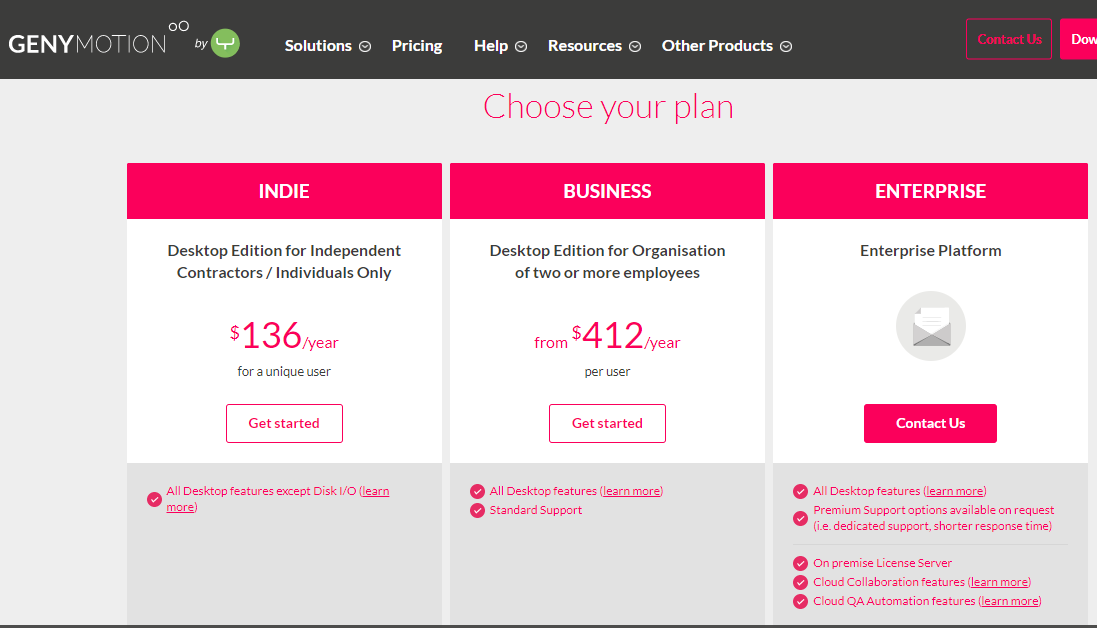
### **Genymotion**

**Lưu ý: Nếu chúng ta sử dụng test trên máy thật thì không cần cài máy ảo Genymotion.**

* **Download**
* Link: <https://www.genymotion.com/account/login/>
* Đăng nhập tài khoản vào Genymotion
* Đăng nhập thành công, ta chọn Download

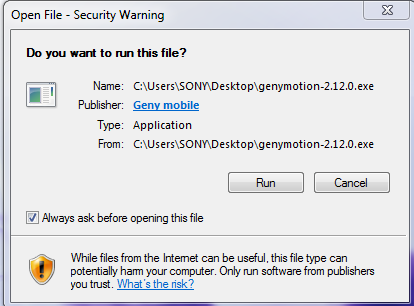
 

* Chọn bảng phủ hợp để download về máy

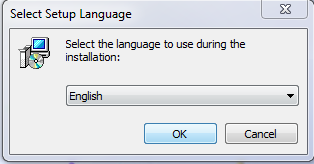


* **Tiến hành cài đặt**

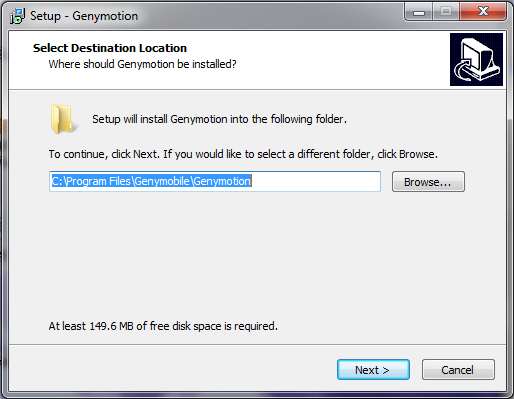
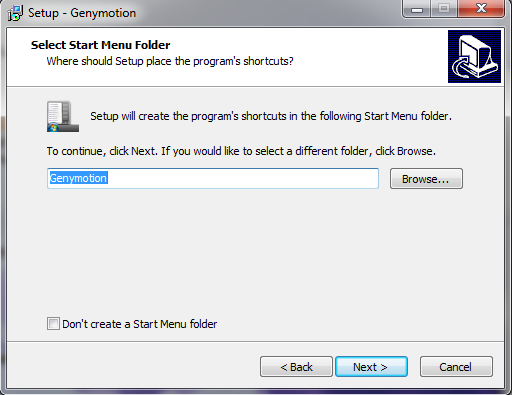
1. Chọn **Run**

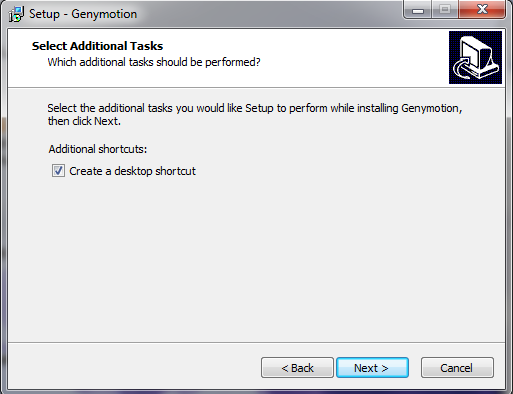


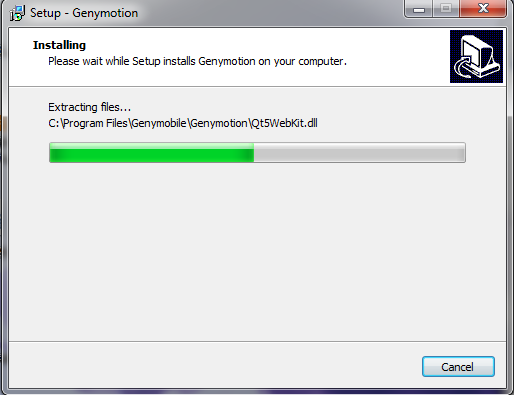
1. Chọn **OK**



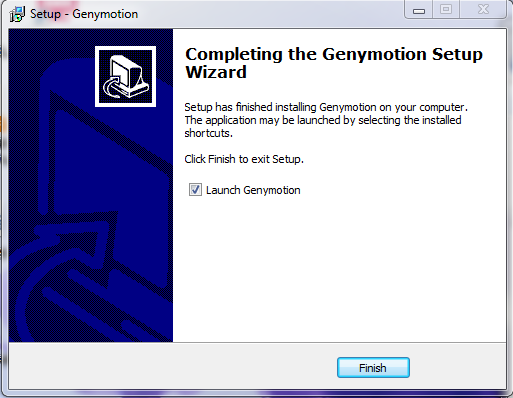
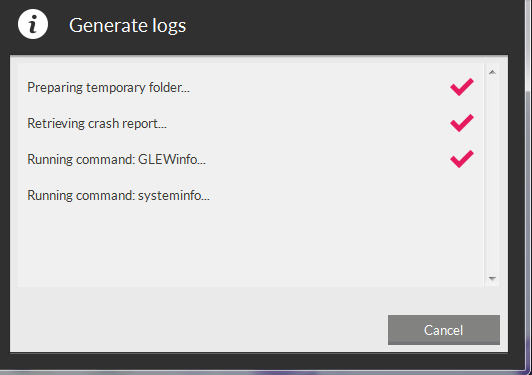
1. Chọn **Next**



1. Chọn **Finish**



## Video hướng dẫn cài đặt Appium

<https://tinyurl.com/y6whteps>

# Kịch bản và demo

## Kịch bản demo

Kịch bản 1: Kiểm tra họ tên rỗng

* Input
* Họ tên:
* Email/ SĐT: 0919606015
* Mật khẩu: hung123456
* Ngày sinh: ngày hiện tại
* Check radio button giới tính: Nam
* Output: Hiển thị thông báo lỗi trên textview cho người dùng biết “Xin vui lòng nhập họ tên”

Kịch bản 2: Kiểm tra email/SĐT rỗng

* Input
* Họ tên: ThienHung
* Email/ SĐT:
* Mật khẩu: hung123456
* Ngày sinh: ngày hiện tại
* Check radio button giới tính: Nam
* Output: Hiển thị thông báo lỗi trên textview cho người dùng biết “Xin vui lòng nhập email”

Kịch bản 3: Email không hợp lệ

* Input
* Họ tên: ThienHung
* Email/ SĐT: abcd
* Mật khẩu: hung123456
* Ngày sinh: ngày hiện tại
* Check radio button giới tính: Nam
* Output: Hiển thị thông báo lỗi trên textview cho người dùng biết “Emai/ số điện thoại không hợp lệ”

Kịch bản 4: Số điện thoại đã đăng ký rồi

* Input
* Họ tên: ThienHung
* Email/ SĐT: 0945356532
* Mật khẩu: hung123456
* Ngày sinh: ngày hiện tại
* Check radio button giới tính: Nam
* Output: Hiển thị thông báo lỗi trên textview cho người dùng biết “Số điện thoại này đã được đăng ký”

Kịch bản 5: Số điện thoại không hợp lệ

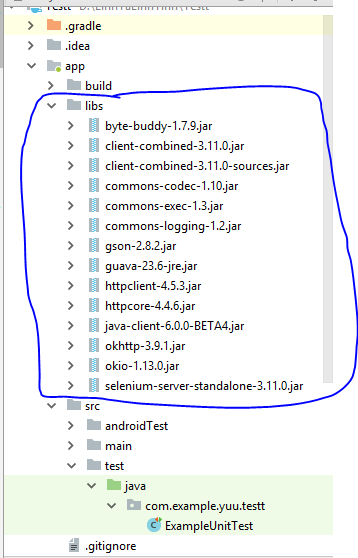
* Input
* Họ tên: ThienHung
* Email/ SĐT: 091960
* Mật khẩu: hung123456
* Ngày sinh: ngày hiện tại
* Check radio button giới tính: Nam
* Output: Hiển thị thông báo lỗi trên textview cho người dùng biết “Email/ SĐT không hợp lệ”

## Demo

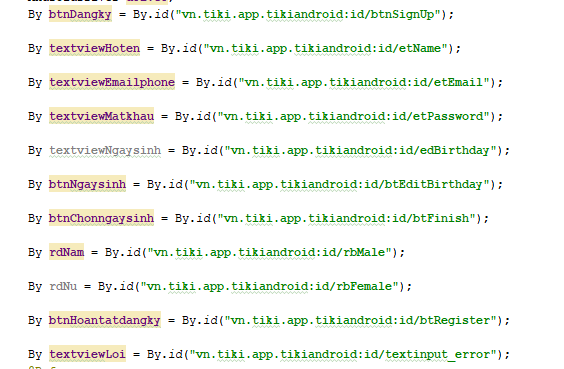
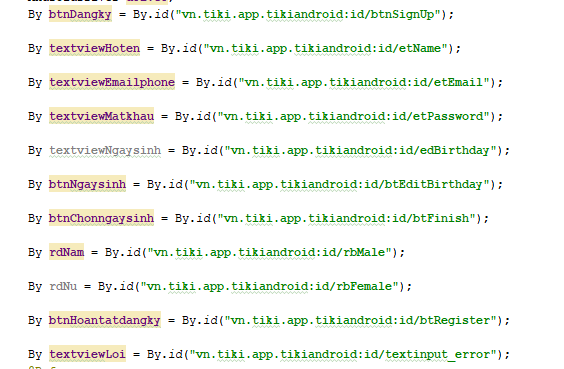
### **Appium hướng dẫn viết Test Script**

Video demo: <https://tinyurl.com/yd2fjwwz>

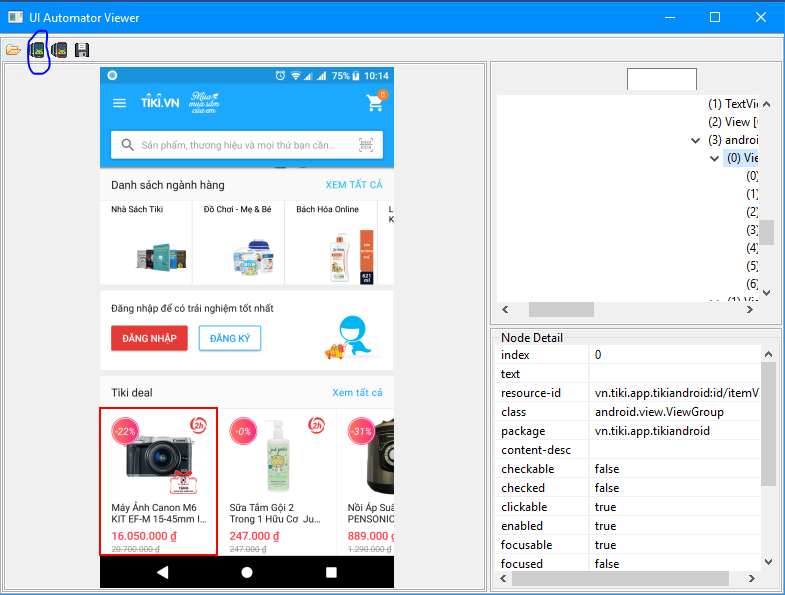
Các bước viết test script:

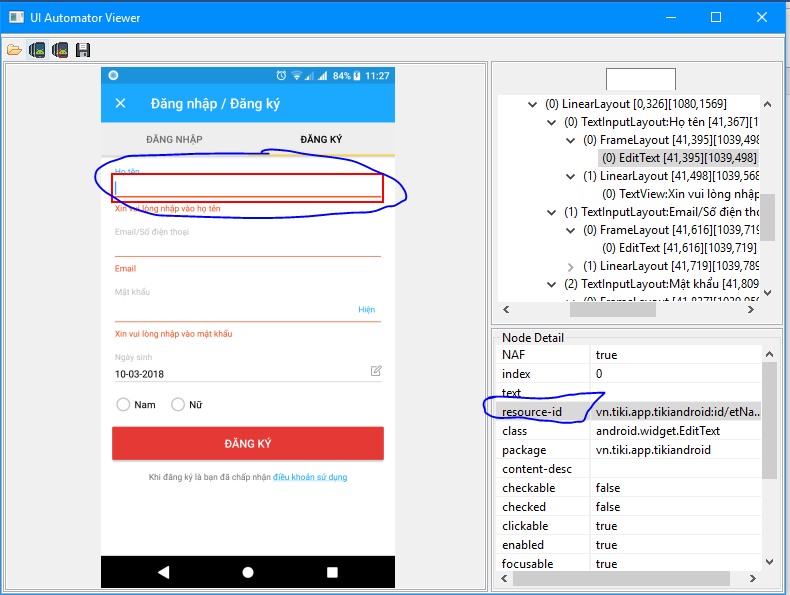
**B1 : Add thự việc vào Android**   


**B2 : Khai báo các control cần thiết để test**

Cách lấy id các control

1. Vào thư mục theo đường dẫn : C:\Users\Yuu\AppData\Local\Android\Sdk\tools\bin
2. Chọn uiautomatorviewer.bat
3. Vào màn hình App cần tìm các id



1. Click vào icon như ảnh dưới
2. Hiển thị màn hình giao diện và ta chọn các con trol cần muốn xem id . 

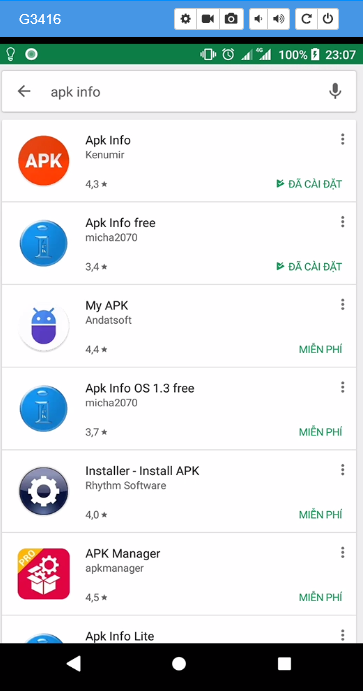
**B3 : Viết hàm kết nối Appium**



Cách tìm AppPackge và AppAvtivity

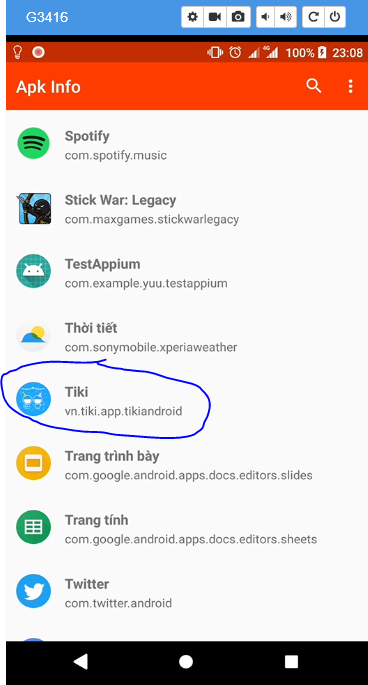
* **Cách 1 : dùng ứng dụng APK info ( Chỉ dùng cho ứng dụng ít Activity )**

B1 : vào Store google gõ tìm APK info

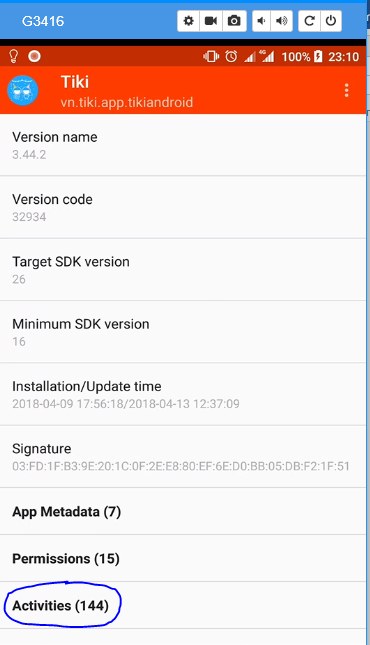


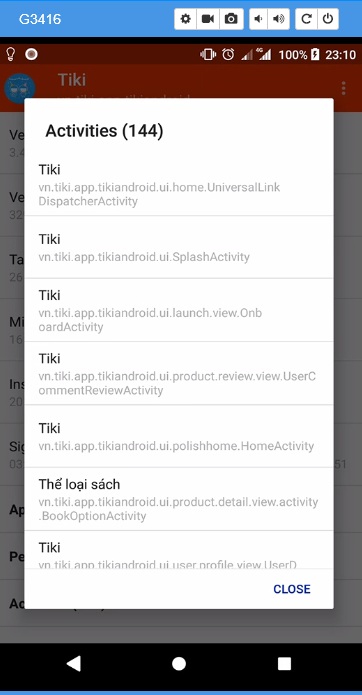
B2 : Cài đặt

B3 : chạy App và chọn Application ta cần tìm Activity



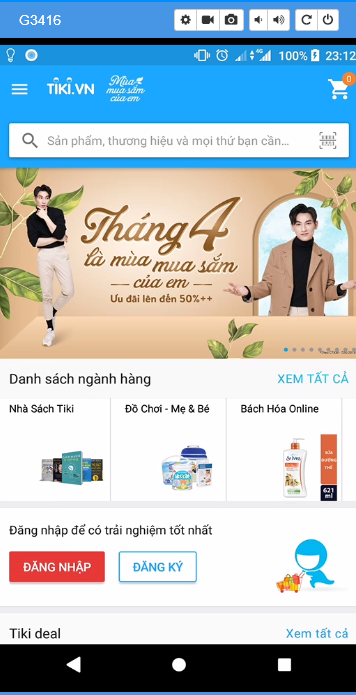
B4 : Chọn mục Activity ứng dụng hiển thị các Activity .



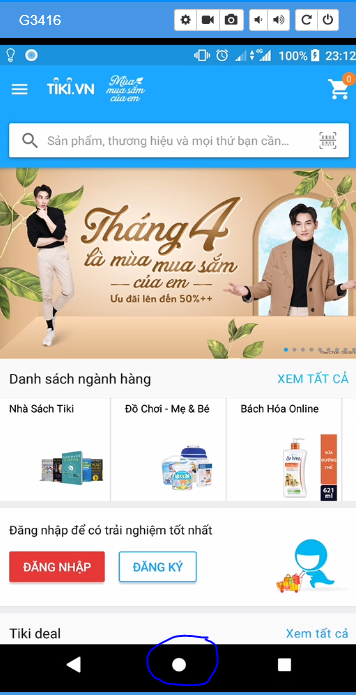


* **Cách 2 : Dùng command line lệnh Adb Logcat**

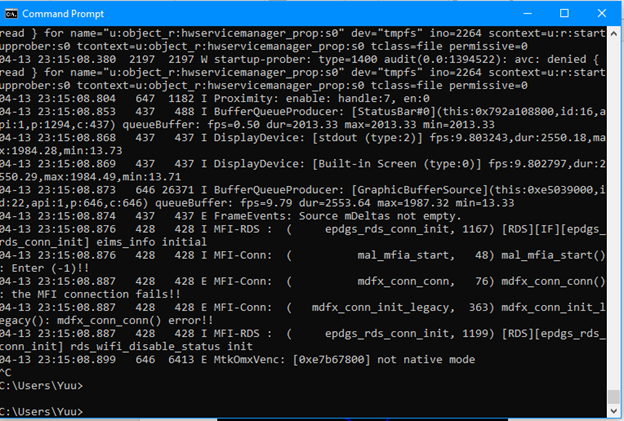
B1 : Mở App cần tìm Activity



B2 : Bấm nó Application nó pause



B3 : Resume lại Application và gõ nhanh lệnh adb logcat trong command line để chừng 2s bấm Ctrl + c 2 lần dừng lai .



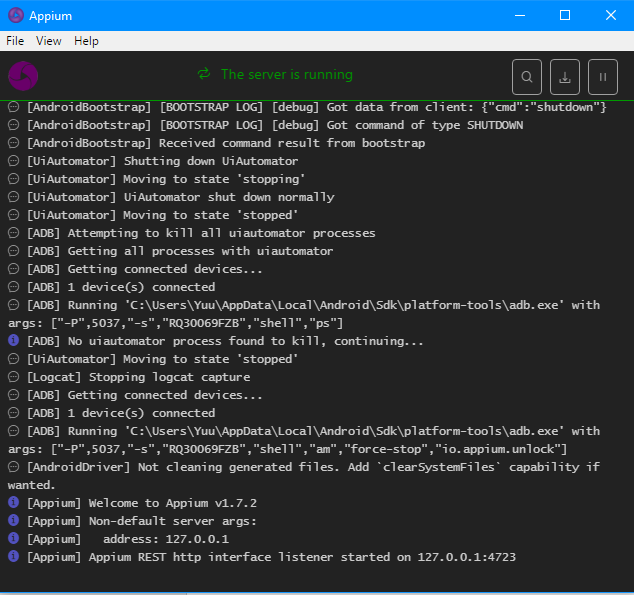
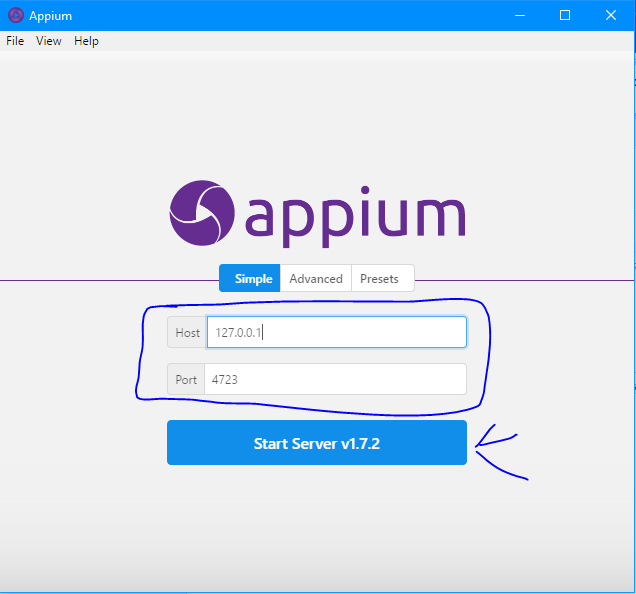
Ta copy đống lệnh này ra ngoài và tìm Activity main trong đó

**B4 : Viết script cho từng trường hợp**

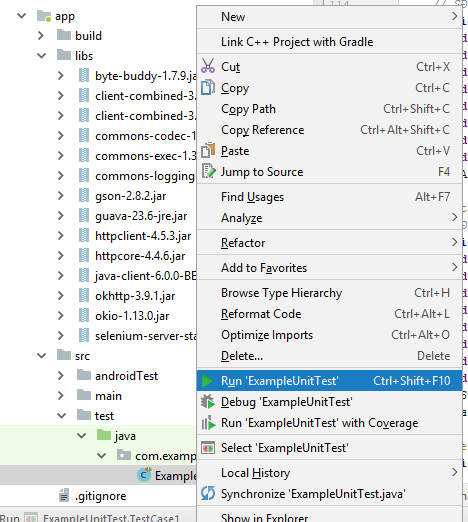


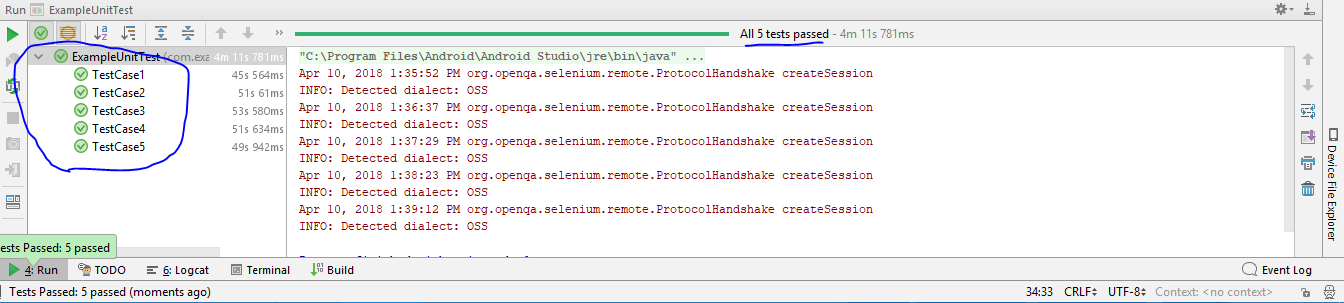


**B5 : Chạy Appium**

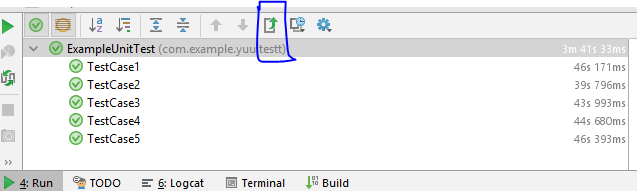


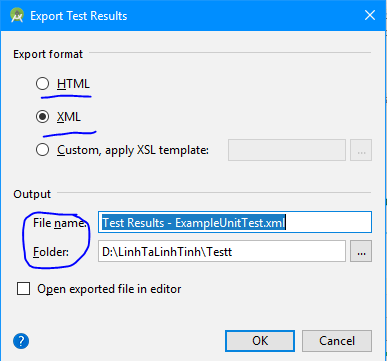
**B6 : Thực thi test script**



**B7 : Ngồi chờ tự chạy các testscript và xem bao nhiêu testcase Pass/Fail** 

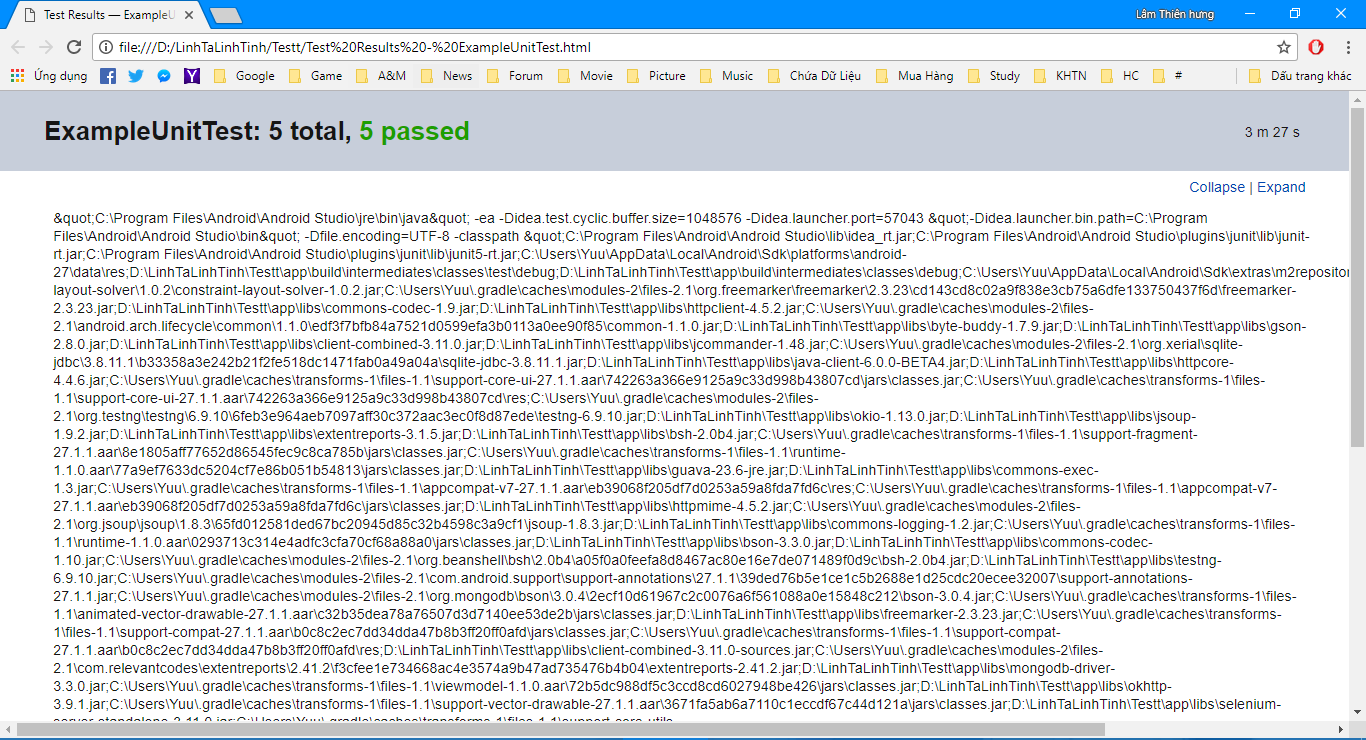
B8 : Xuất file report ta chọn vào icon như hình bên dưới . Chọn định dang cần xuất ( html hoặc xml ) và đường dẫn xuất file -> Bấm ok





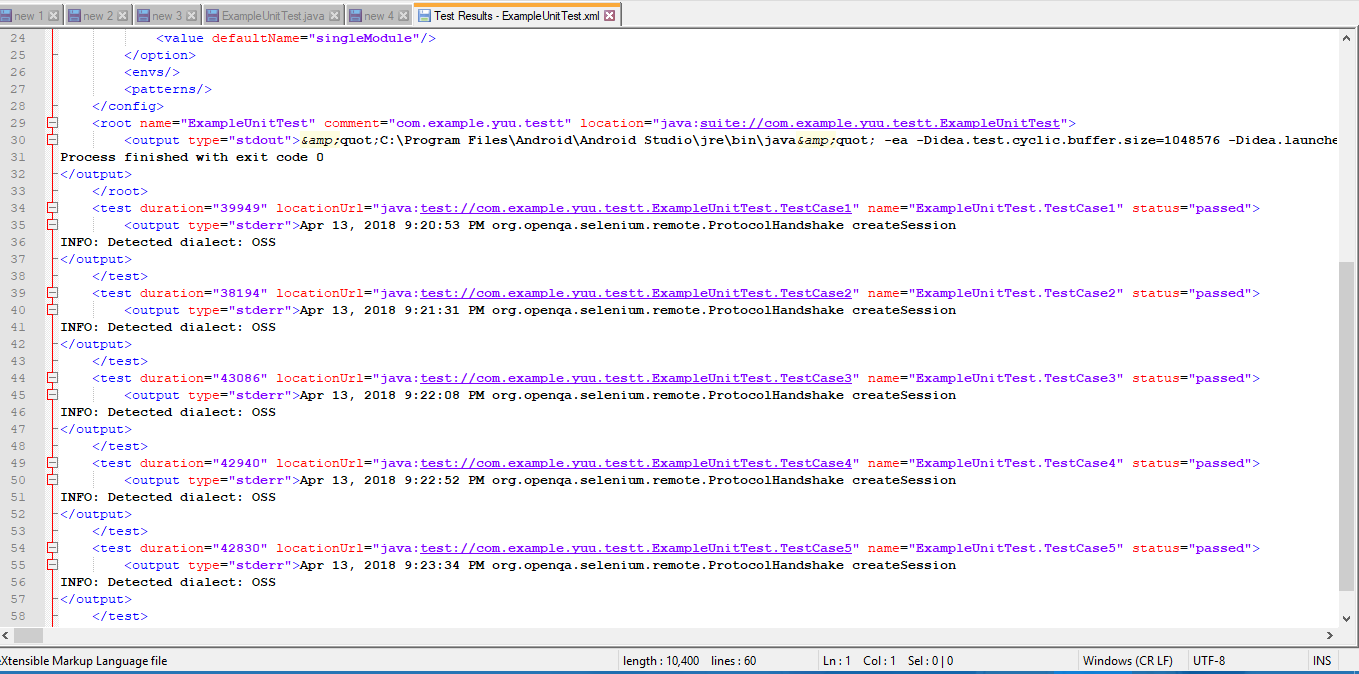
Kết quả

HTML





XML



### **Appium Data Driven and TestNG**

<https://tinyurl.com/y7zovrae>

### **Appium Record/ PlayBack**

<https://tinyurl.com/y7cm7cx7>

# Tài liệu tham khảo

<https://drive.google.com/drive/folders/1QxujHb3TnZiIiXiMzA61ORz-EiySBm2G> (Chương 7 - trong thư mục tài liệu môn học của Thầy Trương Phước Lộc)

<https://viblo.asia/p/tim-hieu-ve-kiem-thu-tu-dong-dich-1l0rvmxDGyqA>

<http://appium.io/>