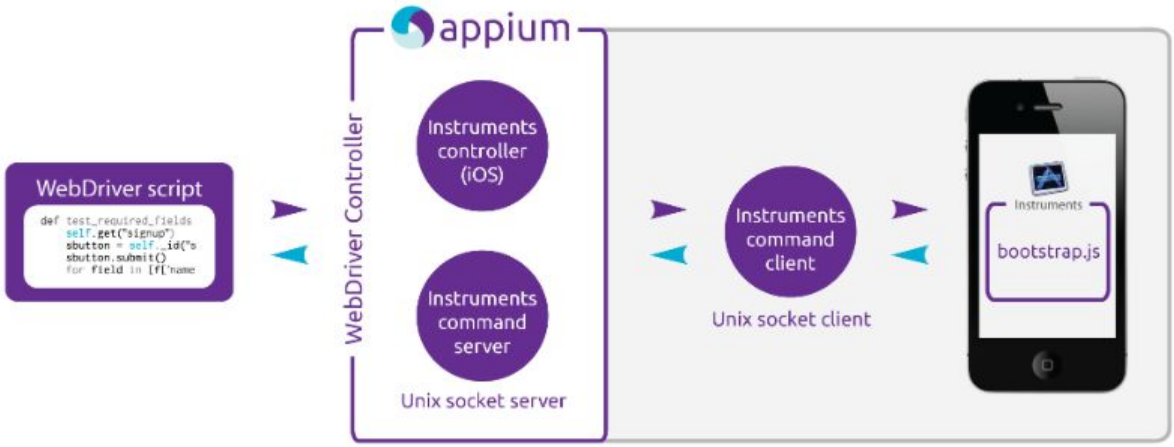
Descripcion

Es un framework de automatizacion mobile para aplicaciones hibridas y nativas en Android e iOS. Funciona como un WebServer (esta construido en una arquitectura Cliente/Servidor) que expone una REST API, el cual la libreria que uno utilice en su codigo se comunicarà mediante Rest Services, realizara las acciones en el dispositivo y luego responderà con un Rest Service.
Lo que lleva a que Appium sea multilenguaje y que se pueda tener una instancia de Appium levantada en una maquina y desde distintas maquinas ejecutarla.



Desired Capabilities

Son las settings de Appium, acà es donde le vamos a setear todos los valores de por ejemplo, el nombre del dispositivo, la version que el dispositivo tiene del S.O, la ruta donde se encuentra la app que queremos instalar, el browser que queremos usar, etc.
Esto lo que hace es decirle a Appium que tiene que buscar un dispositivo que se encuentre levantado (si es virtual) o conectado (si es real) con esas caracteristicas para poder ejecutar el caso de prueba.

Librerias

Existen librerias de appium en distintos lenguajes de programacion.

- Ruby
- Python
- Java
- JavaScript
- C
- PHP
- C#

Instalación

En Windows :

1. Instalar el lenguaje de programacion que queramos utilizar (en este ej es Ruby)
2. Instalar Java : http://download.oracle.com/otn-pub/java/jdk/8u91-b15/jdk-8u91-windows-x64.exe
3. Instalar Node : https://nodejs.org/dist/v4.4.5/node-v4.4.5-x64.msi
4. Instalar Android SDK : https://dl.google.com/android/installer_r24.4.1-windows.exe
5. Instalar Genymotion https://www.genymotion.com/download/ si se quiere, es mas rapido que Android Studio.
6. En Windows ir a “Variables de entorno del sistema”, y en las variables del sistema seleccionar la que diga : “Path” y editarla. Agregarle :
    a. C:\Users\TUUSUARIO\AppData\Local\Android\sdk\tools
    b. C:\Users\TUUSUARIO\AppData\Local\Android\sdk\platform-tools
    c. c:\Ruby22-x64\bin en el caso de que tu lenguaje sea ruby, sino el bin del lenguaje de programación.
    d. C:\Users\TUUSUARIO\AppData\Roaming\npm
7. Agregar tambìen una nueva variable de entorno del sistema con nombre : ANDROID_HOME y valor C:\Users\TUUSUARIO\AppData\Local\Android\sdk
8. Agregar otra variable de entorno del sistema con nombre : JAVA_HOME y valor : C:\Program Files\Java\jdk1.8.0_92
9. Reiniciar el sistema
10. Abrir SDK de Android y actualizar lo que haya que actualizar
11. Abrir una consola y ejecutar : npm install -g appium-doctor
12. Luego ejecutar : appium-doctor
13. Ejecutar en la consola : npm install -g appium
14. Si tenemos algun problema en el paso anterior, ejecutar “Power Shell” de windows como administrador y correr :
    a. npm install -g npm-windows-upgrade
    b. npm-windows-upgrade
    c. npm install -g appium
15. Crear un dispositivo en Android Studio o en GenyMotion

Uso básico en Ruby

A continuaciòn se muestra un ejemplo basico en Ruby para abrir un navegador en un dispositivo virtual, entrar a google, buscar algo y salir.

```
require 'rubygems'
require 'appium_lib'

desired_caps = {
  caps:  {
      platformName:  'Android',
      platformVersion: '5.0',
      deviceName:    'nexus6',
      browserName:   'Browser',
   }
}

appium_driver = Appium::Driver.new(desired_caps)
selenium_driver = appium_driver.start_driver

# Con este metodo se logra que para todos los objetos del tipo Object, no se tenga que llamar
# al driver de Appium para hacer acciones x ejemplo para buscar.
# Ej : @appium_driver.find_element(:id, 'lst-ib')
Appium.promote_appium_methods Object

selenium_driver.get("http://www.google.com/")

text_element = find_element(:id, 'lst-ib')
text_element.click
sleep 5

text_element.send_keys 'Appium and Genymotion using Ruby'
sleep 2

button_element = find_element(:id, 'tsbb')
button_element.click
sleep(5)

driver_quit
puts 'Tests Succeeded!'
```

A continuaciòn se muestra un ejemplo abriendo una aplicacion y navegandola, es una aplicacion basica con un login.

```
require 'rubygems'
require 'appium_lib'
require 'byebug'

APP_PATH = './AUT.apk'

desired_caps = {
  caps:  {
      platformName:  'Android',
      platformVersion: '6.0',
      deviceName:    'nexus6',
      app: APP_PATH
   }
}

puts 'Setting environment to test...'
appium_driver = Appium::Driver.new(desired_caps)
selenium_driver = appium_driver.start_driver
Appium.promote_appium_methods Object
appium_driver.driver.manage.timeouts.implicit_wait = 20

email_element = find_element(:id, "com.example.mkim.aut:id/email")
email_element.send_keys "success@envato.com"
puts 'Email was wrote'

password_element = find_element(:id, "com.example.mkim.aut:id/password")
password_element.send_keys "password"
puts 'Password was wrote'

sign_in_button = find_element(:id, "com.example.mkim.aut:id/email_sign_in_button")
sign_in_button.click
puts 'Sign in button was clicked!'

login_success = find_element(:id, "com.example.mkim.aut:id/login_success")
fail ("Login was failed!") unless login_success.text == "Login Success!"

driver_quit
puts 'Tests Succeeded!'
```

Uso básico en Java :

A continuaciòn se muestra un ejemplo basico en Java para abrir un navegador en un dispositivo virtual, entrar a google, buscar algo y salir.

```
package tests;

import java.net.MalformedURLException;
import java.net.URL;
import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.CapabilityType;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

/**
 * Created by Abstracta on 13/12/2016.
 */

public class WebExample {

    private static WebDriver driver;

    public static void main(String[] args) throws MalformedURLException,
InterruptedException {

        try {
            DesiredCapabilities capabilities = new DesiredCapabilities();
            capabilities.setCapability(CapabilityType.BROWSER_NAME, "Browser");
            capabilities.setCapability("deviceName", "nexus7_4");
            capabilities.setCapability("platformVersion", "4.4.4");
            capabilities.setCapability("platformName", "Android");

            driver = new RemoteWebDriver(new URL("http://127.0.0.1:4723/wd/hub"),
capabilities);
            driver.manage().timeouts().implicitlyWait(80, TimeUnit.SECONDS);

            driver.get("http://www.google.com/");

            WebElement textElement = driver.findElement(By.id("lst-ib"));
            textElement.click();
            Thread.sleep(5000);
```

```
            textElement.sendKeys("Appium and Genymotion using Java");
            Thread.sleep(2000);

            WebElement buttonElement = driver.findElement(By.id("tsbb"));
            buttonElement.click();
            Thread.sleep(5000);
            System.out.println("Test succeded!");
        }
        finally
        {
            driver.quit();
        }
    }
}
```

A continuaciòn se muestra un ejemplo abriendo una aplicacion y navegandola, es una aplicacion basica con un login.

```
package tests;

/**
 * Created by USER on 14-Dec-16.
 */

import java.io.File;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.concurrent.TimeUnit;

import org.junit.Assert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class AppExample {
    private static WebDriver driver;

    public static void main(String[] args) throws MalformedURLException,
InterruptedException {
        try {
            File classpathRoot = new File(System.getProperty("user.dir"));
            File appDir = new File(classpathRoot, "/src/main/resources/");
            File app = new File(appDir, "AUT.apk");
```

```
            DesiredCapabilities capabilities = new DesiredCapabilities();
            capabilities.setCapability("deviceName", "nexus7_4");
            capabilities.setCapability("platformVersion", "4.4.4");
            capabilities.setCapability("platformName", "Android");
            capabilities.setCapability("app", app.getAbsolutePath());

            driver = new RemoteWebDriver(new URL("http://127.0.0.1:4723/wd/hub"),
capabilities);
            driver.manage().timeouts().implicitlyWait(80, TimeUnit.SECONDS);

            System.out.println("Setting environment to test...");
            driver.manage().timeouts().implicitlyWait(60, TimeUnit.SECONDS);

            WebElement emailElement =
driver.findElement(By.id("com.example.mkim.aut:id/email"));
            emailElement.sendKeys("success@envato.com");
            System.out.println("Email was wrote");

            WebElement passwordElement =
driver.findElement(By.id("com.example.mkim.aut:id/password"));
            passwordElement.sendKeys("password");
            System.out.println("Password was wrote");

            WebElement signInButton =
driver.findElement(By.id("com.example.mkim.aut:id/email_sign_in_button"));
            signInButton.click();
            System.out.println("Sign in button was clicked!");

            WebElement loginSuccess =
driver.findElement(By.id("com.example.mkim.aut:id/login_success"));
            loginSuccess.click();
            Assert.assertEquals(loginSuccess.getText(), "Login Success!");
            System.out.println("Test succeded!");
        }
        finally {
            driver.quit();
        }
    }
}
```

Configuración Appium

Para configurar Appium para un dispositivo creado o conectado y que Appium lo reconozca, hacer :

1. Abrir Appium
2. Clickear en Android Settings
3. Si es una prueba en un Browser, seleccionar en use browser, el browser que utilizaremos en el dispositivo.
4. En Launch Device, si es de Android Studio checkear esta opcion y elegir el device de la lista, si estamos usando GenyMotion no funciona bien.
5. En Capabilities seleccionar el Platform Name que es el sistema operativo que tiene el dispositivo, la platformVersion que es la version del S.O, Automation name es Appium, y escribir el device name.
6. Salir de la configuracion y clickear en Play para levantar el device.

Inspector

Appium dispone de una herramienta llamada inspector para navegar la APP y obtener los ID, clases, Xpath, etc. de los elementos de la APP.
Para esto lo que hay que hacer es :

1. Abrir el dispositivo virtual si tenemos, o tener conectado el dispositivo real
2. Abrir Appium
3. Clickear en Android Settings
4. En application path elegir la ruta donde esta la APK
5. Salir de Android Settings e ir a General settings
6. Deshabilitar check for updates
7. Checkear Pre-Launch application
8. Salir de General settings
9. Clickear en la lupa y clickear en refresh

La aplicacion se abrira en el dispositivo y en la vista previa del inspector de Appium, podremos clickear en los elementos y nos dirà el routeId por ejemplo, el cual es el ID del elemento.

PATRONES Y FRAMEWORKS

Appium no controla ni prohibe frameworks o patrones del lenguaje, se puede usar lo que uno quiera. Por ejemplo, se podria armar un proyecto en ruby con :

Ruby + Cucumber + Appium + GenyMotion

En cuanto a los patrones, se pueden (o deben) usar PageObjects de la misma manera que cuando automatizamos cualquier sitio web. Todas las acciones son muy parecidas a como

se definen en Selenium, mas alla de alguna otra que la podemos encontrar en la pagina de Appium o en ejemplos que hay en internet, para deslizarse en la pantalla, etc.