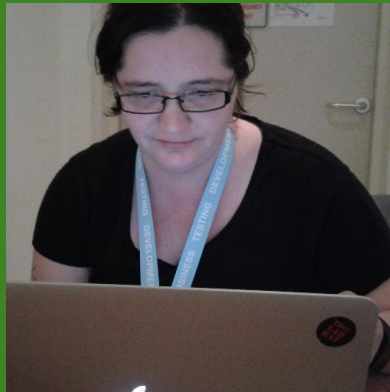




# Building a Mobile Automation Framework Around Appium

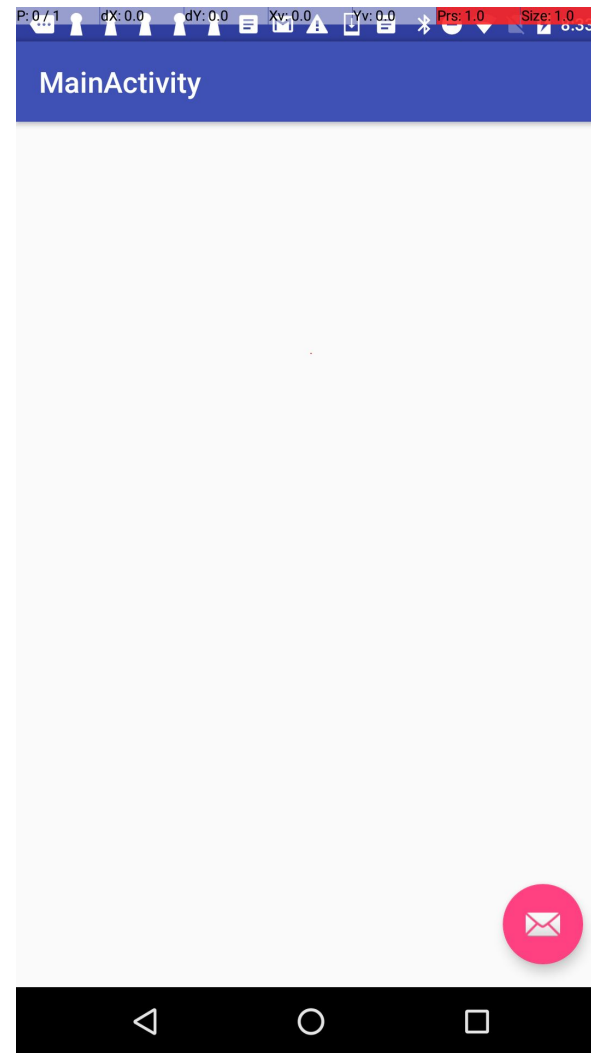
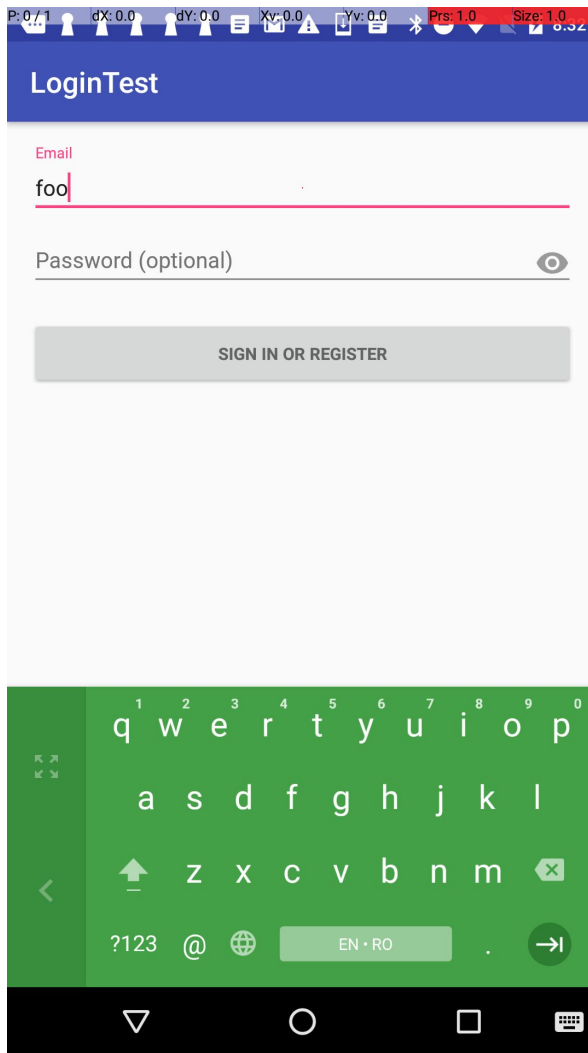
Ru Cindrea - @ru\_altom



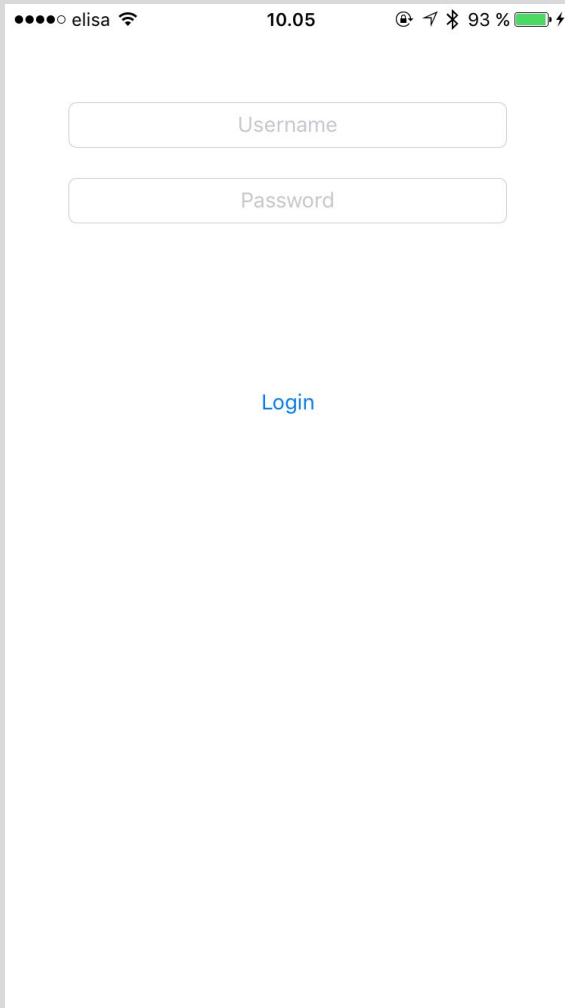
**Who am I?**

**In this demo / talk**

# The test app - Android



# The test app - iOS



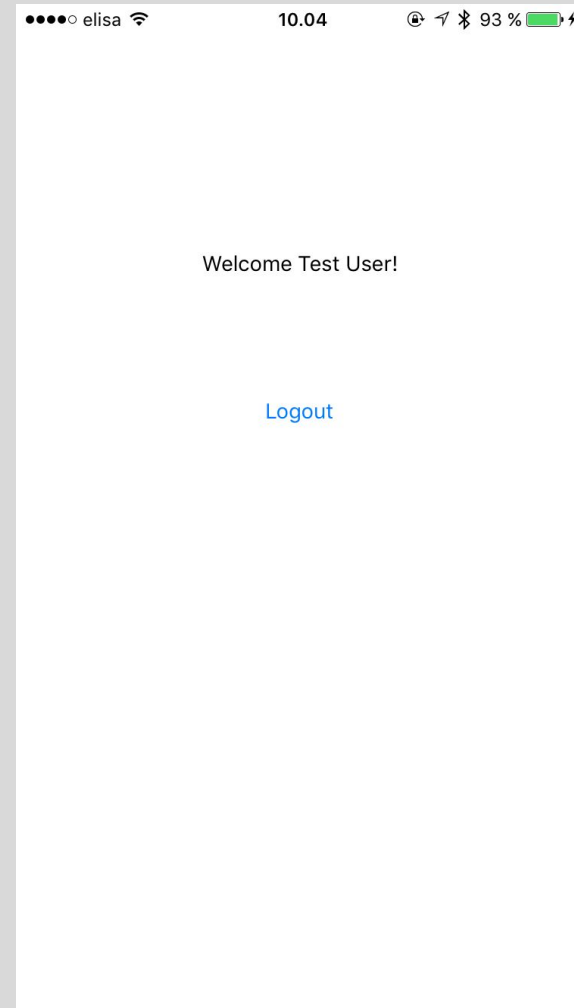
elisa 10.05 93 %

Username

Password

Login

This screenshot shows the login screen of the test app. The status bar at the top displays the carrier 'elisa', the time '10.05', and a battery level of '93 %'. The main content area features two text input fields, one labeled 'Username' and one labeled 'Password'. Below these fields is a blue 'Login' button.



elisa 10.04 93 %

Welcome Test User!

Logout

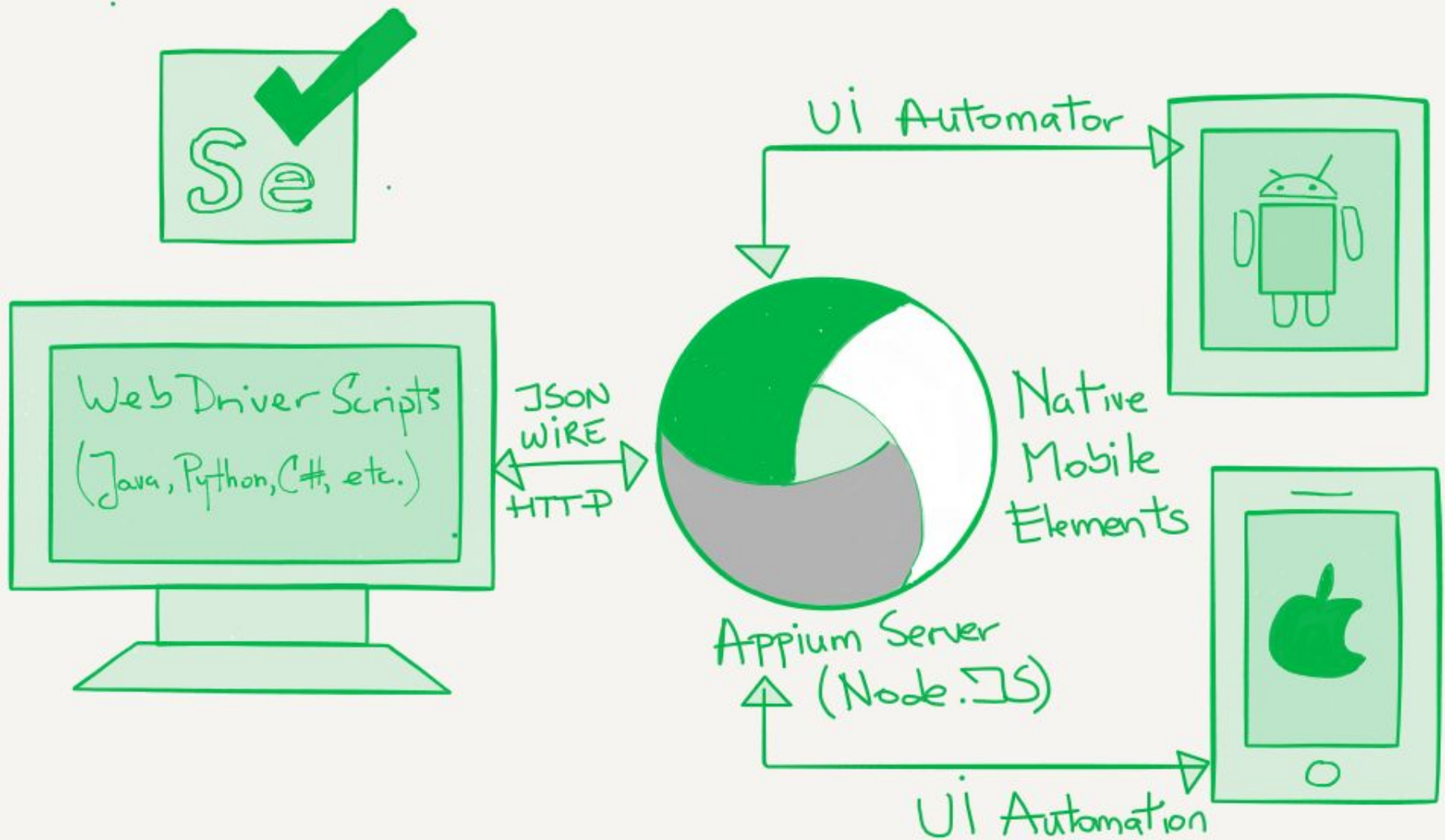
This screenshot shows the welcome screen of the test app. The status bar at the top displays the carrier 'elisa', the time '10.04', and a battery level of '93 %'. The main content area displays the text 'Welcome Test User!' and a blue 'Logout' button.

# Appium for Mobile App Automation

An open source framework for Android and iOS



# Appium



# AndroidTest.java

```
5 public class AndroidTest {
6
7
8     private AppiumDriver<WebElement> driver;
9
10
11     @Before
12     public void setUp() throws Exception {
13         DesiredCapabilities capabilities = new DesiredCapabilities();
14         capabilities.setCapability("deviceName", "Nexus 5X");
15         capabilities.setCapability("app", "executables/app.apk");
16         driver = new AndroidDriver<>(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
17     }
18
19     @After
20     public void tearDown() throws Exception {
21         driver.quit();
22     }
23
24     @Test
25     public void loginTest(){
26         driver.findElement(By.id("email")).sendKeys("foo@example.com");
27         driver.findElement(By.id("password")).sendKeys("hello");
28         driver.findElement(By.id("email_sign_in_button")).click();
29         assertEquals("Main Activity", driver.currentActivity());
30     }
31 }
32
33
34
35
36
37
38
39
40
41
```



# IOSTest.java

```
5 public class IOSTest {
6
7
8     private AppiumDriver<WebElement> driver;
9
10
11     @Before
12     public void setUp() throws Exception {
13         DesiredCapabilities capabilities = new DesiredCapabilities();
14         capabilities.setCapability("deviceName", "iPhpne 6s");
15         capabilities.setCapability("app", "executables/app.ipa");
16         driver = new IOSDriver<>(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
17     }
18
19     @After
20     public void tearDown() throws Exception {
21         driver.quit();
22     }
23
24     @Test
25     public void loginTest(){
26         driver.findElement(By.id("username")).sendKeys("foo@example.com");
27         driver.findElement(By.id("password")).sendKeys("hello");
28         driver.findElement(By.id("loginButton")).click();
29         assertTrue(driver.findElement(By.id("Welcome Test User")).isDisplayed());
30     }
31 }
32
33
34
35
36
37
38
39
40
41
```

# Some common problems

- Maintenance
  - Page Object Model
  - How to add more / better checks?
- Reporting
  - How and what framework to use
  - What to report?
    - Steps taken, screenshots, videos, logs, crashes
- Infrastructure:
  - Device management
  - Appium server management
  - Parallel running




























# Page Object Model - Page Factory

```
@AndroidFindBy(id = "username")  
@iOSFindBy(id = "email")  
private MobileElement email;  
...
```

```
PageFactory.initElements(driver, this);
```

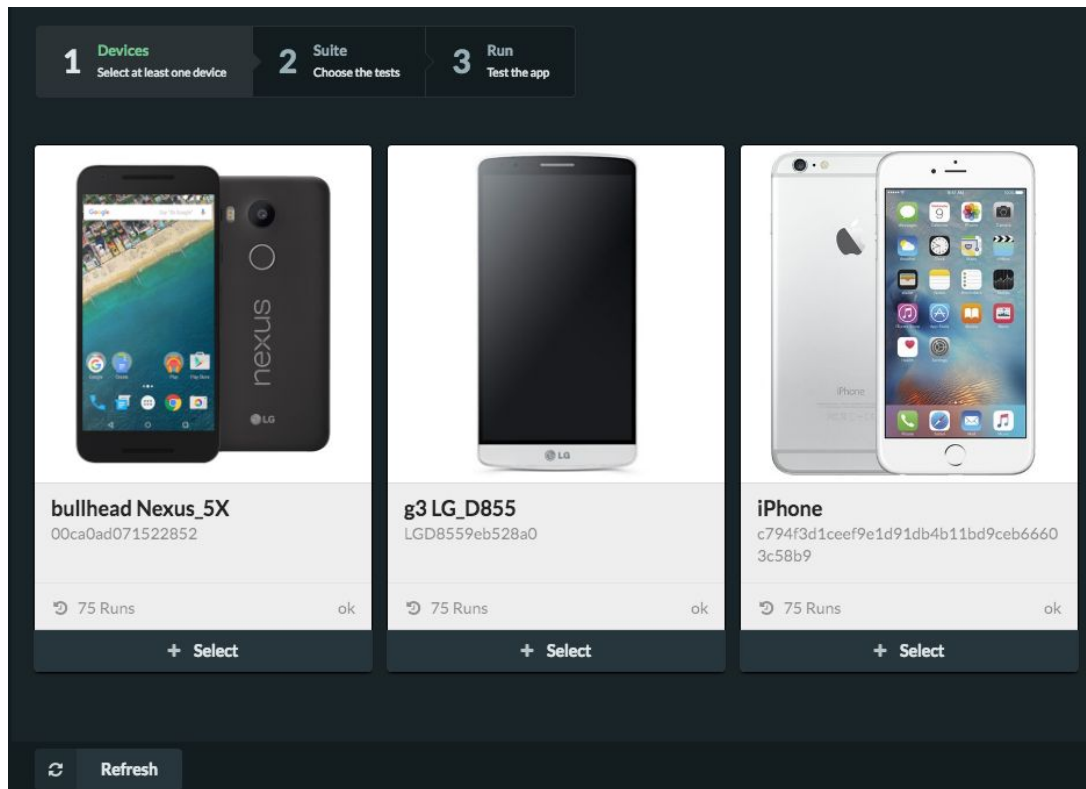
# AltRunner

# alt-runner

- ▼  ro.altom
  - ▼  alrunner
    - ▶  appium
    - ▶  cli
    - ▶  commands
    - ▶  contexts
    - ▶  exceptions
    - ▶  listeners
    - ▶  logs
    -   DeviceBusyIterator
    -   DeviceIdentifier
    -   Platform
    -   ProcessRunner
    -   RunnerListener
    -   **SingleRunner**
    -   SystemProperties
    -   TestSuite
    -   WeaverLoader

- Java module
  - Imported in your test project to run the tests
- Deals with:
  - Reporting + Logs
  - Appium Server Management
  - Device Selection and management

# alt-multi-runner



- NodeJs application
  - Runs locally
- Deals with:
  - Starting the tests
  - Parallel test runs
  - Displaying the results

# **Demo - alt-multi-runner**

# Appium Server Management



# Setup Appium Driver

```
1
2
3
4
5
6
7
8
9
10 public void setupAndroidDriver() throws MalformedURLException {
11     DesiredCapabilities capabilities = new DesiredCapabilities();
12
13     capabilities.setCapability("deviceName", "LG Nexus 5X");
14     capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, MobilePlatform.ANDROID);
15     capabilities.setCapability("app", "executables/app.apk");
16
17     driver = new AndroidDriver<WebElement>(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
18 }
19
20
21
22
23
```

# Setup Appium Driver

```
1
2
3
4
5
6
7
8
9
10 public void setupAndroidDriver() throws MalformedURLException {
11     DesiredCapabilities capabilities = new DesiredCapabilities();
12
13     capabilities.setCapability("deviceName", "LG Nexus 5X");
14     capabilities.setCapability(MobileCapabilityType.PLATFORM_NAME, MobilePlatform.ANDROID);
15     capabilities.setCapability("app", "executables/app.apk");
16
17     driver = new AndroidDriver<WebElement>(new URL("http://127.0.0.1:4723/wd/hub"), capabilities);
18 }
19
20
21
22
23
```

# Runner Class

```
1
2
3 package com.example;
4
5 import ro.altom.alrunner.SingleRunner;
6
7 public class MyRunner extends SingleRunner {
8 }
9
10
11
12
13
14
15 import io.appium.java_client.AppiumDriver;
16 import ro.altom.alrunner.contexts.TestContext;
17
18 public class LoginTest {
19
20     private AppiumDriver driver = (AppiumDriver) TestContext.instance.get().getDriver();
21
22     @Test
23     void loginWithValidCredentials() {
24         ...
25     }
26 }
```

# Device Management

# Device Info

## DeviceInfo

- m DeviceInfo(String, String, Platform, String)
- m DeviceInfo(DeviceInfo)
- m setPropertiesValues(): void
- m getTimeoutProperty(): int
- m getFormattedName(): String
- m isAndroid(): boolean
- m isIOS(): boolean
- m parseAPINumbersFromProperties(String): List<String>
- m isCloud(): boolean
- m toContext(): TestContext
- m getCapabilities(): DesiredCapabilities
- f name: String = ""
- f udid: String = ""
- f model: String = ""
- f platform: Platform = null
- f location: String = ""
- f logPattern: String
- f recording: boolean = false
- f shouldReboot: boolean = false
- f acceptedBatteryLevel: int = 0
- f acceptedAPINumber: List<String> = new ArrayList<>
- f shouldHaveInternet: boolean = false
- f isRecording: boolean
- f status: IStatus
- f instance: ThreadLocal<DeviceInfo> = new ThreadLo

##### Device

```
device.reboot=false
device.record=true
device.waitTimeout=300000
device.healthcheck.internet=false
device.healthcheck.battery=0
device.healthcheck.api=23,22,21,20,19
```

## DeviceQuery

- m getAndroidDevices(): List<DeviceInfo>
- m getIOSDevices(): List<DeviceInfo>
- m get(SingleRunParameters): DeviceInfo
- m getAll(): List<DeviceInfo>

# Reporting

# Allure - An open source test reporting framework

```
@Step("Type username {0}")  
public void typeUsername(String text);
```

```
@Attachment()  
public void takeScreenshot();
```

<http://allure.qatools.ru>

# **Logs - Video, Crash, Memory, CPU**



# Device Commands

## AndroidCommand

- m AndroidCommand(String)
- m getDevices(): List<DeviceInfo>
- m deviceCommand(String[]): ProcessRunner ↑IDeviceC
- m command(String[]): ProcessRunner
- m getAdbPath(): String
- m logCat(): ProcessRunner ↑IDeviceCommand
- m screenRecord(String, String): ProcessRunner ↑IDevice
- m screenRecord(String, String, long): ProcessRunner ↑I
- m batteryParser(String): int
- m getHealthyDevices(): ArrayList<DeviceInfo>
- m verifyAppProcessRunning(String): boolean ↑IDeviceC
- m isAppRunning(String): boolean ↑IDeviceCommand
- m getCpuInfo(String): CpuUsage
- m getMemoryInfo(String): MemoryStats
- m log(): ProcessRunner ↑IDeviceCommand
- f androidHome: String = System.getenv(...)
- f udid: String

## IOSCommand

- m IOSCommand(String)
- m deviceCommand(String[]): ProcessRunner ↑IDeviceC
- m log(): ProcessRunner ↑IDeviceCommand
- m command(String[]): ProcessRunner
- m logCat(): ProcessRunner ↑IDeviceCommand
- m screenRecord(String, String): ProcessRunner ↑IDevice
- m screenRecord(String, String, long): ProcessRunner ↑
- m isAppRunning(String): boolean ↑IDeviceCommand
- m verifyAppProcessRunning(String): boolean ↑IDeviceC
- m getDeviceName(String): String
- m parseCommandOutput(String): List<String>
- m getHealthyDevices(): ArrayList<DeviceInfo>
- m getDevices(): List<DeviceInfo>
- f udid: String = ""

# Device Commands

```
### Reports
reports.appium=true
reports.android.recording=false
reports.android.recordingSubtiles=false
reports.android.logcat=true
reports.android.cpu=false
reports.android.memory=false
reports.android.crashlog=false
reports.android.crashTriggers=WIN DEATH,has died,Force finishing activity,

reports.ios.logcat=true
reports.ios.crashlog=true
reports.ios.crashTriggers=crashed,was killed,Service exited,Terminated
```

# Test Injector

# Demo

**What next?**

what next?

Ideas for the future:

- Extend the runner for Selenium WebDriver
- Add more functionality to the iOS device management and logs
- Open source (some of) it

# Questions?