pyand Documentation

Release 1.0.0

ardevd

Mar 08, 2018

Contents:

Release 1.0.0

Date Mar 08, 2018

pyand is a simple Python library for Python 2.7 that allows you to easily work with adb and fastboot connected Android devices. Usage is pretty simple and should be intuitive for anyone familiar with Python as well as adb and fastboot.

Although this documentation describes the most important methods and classes it is not a complete API reference.

CHAPTER 1

Installation

Installation is simple. Pyand is a simple Python module that enables developers to interact with ADB and Fastboot. There are currently two recommended ways of installing pyand.

1.1 Dependencies

- Python 2 (2.6 or later)
- ADB and Fastboot available on the system

1.2 easy_install

If you have easy_install for Python-2.7 installed, you can use it to install pyand pretty easily.:

```
$ git clone https://github.com/ardevd/pyand
$ sudo easy_install-2.7 pyand
```

1.3 AUR PKGBUILD

There is also an officially supported PKGBUILD available. You can grab the PKGBUILD from the GitHub repo.

CHAPTER 2

Getting Started

2.1 Simple Example

pyand exposes two classes. ADB and Fastboot. Usage is simple and to verify that everything is set up correctly we can import ADB and Fastboot and check for connected devices:

```
>>> from pyand import ADB, Fastboot
>>> adb = ADB()
>>> adb.get_devices()
{0: '15901aabbccdd124', 1: 'abc1951124de1241'}
>>> adb.set_target_by_id(1)
'[+] Target device set: abc1951124de1241'
>>> adb.get_model()
'Nexus_5'
>>> adb.set_system_rw()
'remount succeeded'
>>> adb.reboot(2)
>>> fb = Fastboot()
>>> fb.get_devices()
{0: 'abc1951124de1241'}
```

To start we have to import the ADB and Fastboot classes to be able to access the associated methods. As long as the ADB and Fastboot binaries are in your $PATH the above example should work. Otherwise you will have to specify the path to binary when instantiating the object.:

```
>>> adb = ADB('~/android-sdk/platform-tools/adb')
```

Furthermore, notice how we set a device target. Since you can have several devices connected it's important for pyand to know which device you want to interact with.

CHAPTER 3

Basics

This section covers the most basic concepts in pyand

3.1 Devices

You can easily use adb to get a list of the currently connected devices.:

```
>> from pyand import ADB
>> adb = ADB()
>> adb.get_devices()
{0: '12601aabbccdd124', 1: 'abc1551124de1241'}
```

Notice how `get_devices()` returns a dictionary with an index and the device identifier.

3.2 Setting a target device

In order to interact with a device you need to tell pyand which device you want to interact with. You do this by specifying either the device index from the dictionary returned from `get_devices()` or by the device id.:

```
>> adb.get_devices()
{0: '12601aabbccdd124', 1: 'abc1551124de1241'}
>> adb.set_target_by_id(1)
[+] Target device set: abc1551124de1241
>> adb.set_target_by_name('abc1551124de1241')
[+] Target device set: abc1551124de1241
```

3.3 Getting device state

Note how that dictionary returned by `get_devices()` does not indicate the state of the device(s). You can use `get_state()` for that. Remember to set a device target first.:

```
>> adb.get_state()
unauthorized
```

CHAPTER 4

Indices and tables

- search