# Poco使用手册

## Poco简介

Poco是一个强大的类库，并易于构建自己的应用程序。

Poco帮助你创建跨平台的应用程序（一次编写，多平台编译和运行）。

模块化和可扩展，可应用于嵌入式到企业级程序。

提供全面，易懂的编程接口。

使用C++语言，快速且高效。

Poco崇尚简易。

在设计，代码风格和文档上保持一致。

Poco强调代码质量，包括可读性，综合性，一致性，编码风格和可测试性。

Poco使得C++编程更加容易。

## 安装

pip install pocoui

（截止到2019年4月，poco的最新版本为1.0.76）

## 调用

### Unity

from poco.drivers.unity3d import UnityPoco

poco = UnityPoco()

# for unity editor on windows

# poco = UnityPoco(('localhost', 5001), unity\_editor=True)

ui = poco('...')

ui.click()

### Android native APP

from poco.drivers.android.uiautomation import AndroidUiautomationPoco

poco = AndroidUiautomationPoco(use\_airtest\_input=True, screenshot\_each\_action=False)

### Cocos-lua

from poco.drivers.std import StdPoco

poco = StdPoco()

### Cocos-js

from poco.drivers.cocosjs import CocosJsPoco

poco = CocosJsPoco()

### iOS

from poco.drivers.ios import iosPoco

poco = iosPoco()

### Std-broker

from poco.drivers.std import StdPoco

from poco.utils.device import VirtualDevice

poco = StdPoco(15004, VirtualDevice('localhost'))

## Poco类

**classPoco(agent, \*\*options)**

是poco.acceleration.PocoAccelerationMixin的子类

### 参数：

#### action\_interval:

time interval to wait for the action (such as touch or swipe) completion performed on device and for the UI to become still (stable). Default value is 0.8s.

等待在设备上执行的动作(如触摸或滑动)完成和UI变得静止(稳定)的时间间隔。默认值为0.8s。

#### poll\_interval:

the minimum time needed between each poll events (such as waiting for UI element to appear on the screen). Polling is done periodically.

每个轮询事件之间所需的最短时间(例如等待UI元素出现在屏幕上)。轮询是定期进行的。

#### pre\_action\_wait\_for\_appearance:

time interval to wait before the action (such as click or swipe) is performed. If the target UI element does not appear on the screen after this time interval, the [PocoNoSuchNodeException](https://poco-chinese.readthedocs.io/zh_CN/latest/source/poco.exceptions.html" \l "poco.exceptions.PocoNoSuchNodeException" \o "poco.exceptions.PocoNoSuchNodeException) is raised

在执行操作(例如单击或滑动)之前等待的时间间隔。如果目标UI元素在此时间间隔后不出现在屏幕上，则会引发PocoNoSuchNodeException的异常。

#### touch\_down\_duration:

Touch down step duration of the click operation last for. If this argument is provided, this value will set to self.agent.input module. Note that not all implementation of poco support this parameter. If not support, you may see a warning.

按下持续的单击操作的步骤持续时间。如果提供了此参数，则此值将设置到self.agent.input模块。注意，并非所有poco实现都支持此参数。如果不支持，您可能会看到警告。

### 方法

#### \_\_init\_\_

初始化方法，定义Poco类的各项参数。

**def** \_\_init\_\_(self, agent, \*\*options):

super(Poco, self).\_\_init\_\_()

self.\_agent = agent

*# options*

self.\_pre\_action\_wait\_for\_appearance = options.get('pre\_action\_wait\_for\_appearance', 6)

self.\_post\_action\_interval = options.get('action\_interval', 0.8)

self.\_poll\_interval = options.get('poll\_interval', 1.44)

**if** 'touch\_down\_duration' **in** options:

touch\_down\_duration = options['touch\_down\_duration']

**try**:

touch\_down\_duration = float(touch\_down\_duration)

**except** ValueError:

**raise** ValueError('Option `touch\_down\_duration` should be <float>. Got *{}*'

.format(repr(touch\_down\_duration)))

self.\_agent.input.setTouchDownDuration(touch\_down\_duration)

self.\_pre\_action\_callbacks = [self.\_\_class\_\_.on\_pre\_action]

self.\_post\_action\_callbacks = [self.\_\_class\_\_.on\_post\_action]

self.\_agent.on\_bind\_driver(self)

#### \_\_call\_\_

调用方法，通过查询表达式选择UI元素。

这是内部方法，了解即可，实际上用不到。

**def** \_\_call\_\_(self, name=**None**, \*\*kw):

*"""*

*Call Poco instance to select the UI element by query expression. Query expression can contain specific name*

*and/or other attributes. Invisible UI elements will be skipped even if "visible=False" argument is set.*

*Selection process is not executed instantly, the query expression is stored in the UI proxy and the selection is*

*executed only then when the UI element(s) info is required (such get the point coordinates where to click,*

*and/or retrieve the specific attribute value).*

*Examples:*

*This example shows selecting a Button named 'close'::*

*poco = Poco(...)*

*close\_btn = poco('close', type='Button')*

*Args:*

*name (:obj:`str`): name of the UI element to be selected*

*Keyword Args:*

*xx: arbitrary key value pair that stands for selecting the UI matching the value of ``UI.xx``*

*xxMatches (:obj:`str`): arbitrary key value pair that stands for selecting the UI matching the regular*

*expression pattern ``UI.xx``*

*In keyword args, you can only use `xx` or `xxMatches` at the same time. Using both with the same attribute does*

*not make sense. Besides, `xx` should not start with ``\_`` (underscore) as attributes start with ``\_`` are*

*private attributes that used by sdk implementation.*

*::*

*# select the UI element(s) which text attribute matches the pattern '^close.\*$'*

*poco = Poco(...)*

*arb\_close\_btn = poco(textMatches='^close.\*$')*

*Returns:*

*:py:class:`UIObjectProxy <poco.proxy.UIObjectProxy>`: UI proxy object representing the UI element matches*

*the given query expression.*

*"""*

**if** **not** name **and** len(kw) == 0:

warnings.warn("Wildcard selector may cause performance trouble. Please give at least one condition to "

"shrink range of results")

**return** UIObjectProxy(self, name, \*\*kw)

#### wait\_for\_any

等待参数里的任意一个控件出现，出现则返回该控件，超时则抛出PocoTargetTimeout异常。超时时间默认为120秒。

**def** wait\_for\_any(self, objects, timeout=120):

*"""*

*Wait until any of given UI proxies show up before timeout and return the first appeared UI proxy.*

*All UI proxies will be polled periodically. See options :py:class:`poll\_interval <poco.pocofw.Poco>` in*

*``Poco``'s initialization for more details.*

*Args:*

*objects (Iterable<:py:class:`UIObjectProxy <poco.proxy.UIObjectProxy>`>): iterable object of the given UI*

*proxies*

*timeout (:obj:`float`): timeout in seconds, default is 120s*

*Returns:*

*:py:class:`UIObjectProxy <poco.proxy.UIObjectProxy>`: the first appeared UI proxy*

*Raises:*

*PocoTargetTimeout: when none of UI proxies appeared before timeout*

*"""*

start = time.time()

**while** **True**:

**for** obj **in** objects:

**if** obj.exists():

**return** obj

**if** time.time() - start > timeout:

**raise** PocoTargetTimeout('any to appear', objects)

self.sleep\_for\_polling\_interval()

#### wait\_for\_all

等待参数里的任意一个控件出现，出现则返回该控件，超时则抛出PocoTargetTimeout异常。超时时间默认为120秒

**def** wait\_for\_all(self, objects, timeout=120):

*"""*

*Wait until all of given UI proxies show up before timeout.*

*All UI proxies will be polled periodically. See option :py:class:`poll\_interval <poco.pocofw.Poco>` in*

*``Poco``'s initialization for more details.*

*Args:*

*objects (Iterable<:py:class:`UIObjectProxy <poco.proxy.UIObjectProxy>`>): iterable object of the given UI*

*proxies*

*timeout (:obj:`float`): timeout in seconds, default is 120s*

*Raises:*

*PocoTargetTimeout: when not all of UI proxies appeared before timeout*

*"""*

start = time.time()

**while** **True**:

all\_exist = **True**

**for** obj **in** objects:

**if** **not** obj.exists():

all\_exist = **False**

**break**

**if** all\_exist:

**return**

**if** time.time() - start > timeout:

**raise** PocoTargetTimeout('all to appear', objects)

self.sleep\_for\_polling\_interval()

#### freeze

生成一个当前poco控件树的快照。这个新的poco实例是来自当前poco实例(Self)的一个副本。新poco实例的层次结构是固定的和不可变的。从该冻结的poco实例中调用“dump”函数时，速度会非常快。该实例实际上是一个**FrozenPoco**类的实例。

**def** freeze(this):

*"""*

*Snapshot current \*\*hierarchy\*\* and cache it into a new poco instance. This new poco instance is a copy from*

*current poco instance (``self``). The hierarchy of the new poco instance is fixed and immutable. It will be*

*super fast when calling ``dump`` function from frozen poco. See the example below.*

*Examples:*

*::*

*poco = Poco(...)*

*frozen\_poco = poco.freeze()*

*hierarchy\_dict = frozen\_poco.agent.hierarchy.dump() # will return the already cached hierarchy data*

*Returns:*

*:py:class:`Poco <poco.pocofw.Poco>`: new poco instance copy from current poco instance (``self``)*

*"""*

**class** **FrozenPoco** (Poco):

**def** \_\_init\_\_(self, \*\*kwargs):

hierarchy\_dict = this.agent.hierarchy.dump()

hierarchy = create\_immutable\_hierarchy(hierarchy\_dict)

agent\_ = PocoAgent(hierarchy, this.agent.input, this.agent.screen)

kwargs['action\_interval'] = 0.01

kwargs['pre\_action\_wait\_for\_appearance'] = 0

super(FrozenPoco, self).\_\_init\_\_(agent\_, \*\*kwargs)

self.this = this

**def** \_\_enter\_\_(self):

**return** self

**def** \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):

**pass**

**def** \_\_getattr\_\_(self, item):

**return** getattr(self.this, item)

**return** FrozenPoco()

#### wait\_stable

休眠固定的秒数，以等待UI变得静止(稳定)。不需要手动调用此方法。它会在需要时自动调用。

**def** wait\_stable(self):

*"""*

*Sleep for fixed number of seconds in order to wait for the UI to become still (stable).*

*There is no need to call this method manually. It's automatically invoked when required.*

*"""*

time.sleep(self.\_post\_action\_interval)

#### sleep\_for\_polling\_interval

每次轮询事件后，休眠固定秒数。不需要手动调用此方法。它会在需要时自动调用。

**def** sleep\_for\_polling\_interval(self):

*"""*

*Sleep for fixed number of seconds after each poll event.*

*There is no need to call this method manually. It's automatically invoked when required.*

*"""*

time.sleep(self.\_poll\_interval)

#### agent

访问poco代理实例的只读属性。有关更多细节，请参见：py:class:’poco.agent.PocoAgent’。

**@property**

**def** agent(self):

*"""*

*Readonly property to access poco agent instance. See :py:class:`poco.agent.PocoAgent` for more details.*

*Returns:*

*:py:class:`poco.agent.PocoAgent`: poco agent instance*

*"""*

**return** self.\_agent

#### click

给定坐标下对目标设备执行点击(触摸、敲击等) 操作。

坐标(x,y)要么是个2元列表，要么是个2元元组。x和y的坐标值必须在0~1之间，以表示屏幕的百分比。例如，坐标`[0.5，0.5]`代表屏幕的中央，坐标`[0，0]`代表`左上角’。

有关归一化坐标系的更多细节，见“归一化坐标系系统”。

尝试点击绝对坐标时，需要事先获得屏幕的分辨率，然后手动计算其百分比。

例：试图在一个1920\*1080的屏幕上点击绝对坐标为(100，100)的点，需要写成poco.click([100.0 / 1920, 100.0 / 1080])的形式。

当点击范围在屏幕外，（即x和y<0或>1时）抛出*InvalidOperationException*异常。

**def** click(self, pos):

*"""*

*Perform click (touch, tap, etc.) action on target device at given coordinates.*

*The coordinates (x, y) are either a 2-list or 2-tuple. The coordinates values for x and y must be in the*

*interval between 0 ~ 1 to represent the percentage of the screen. For example, the coordinates ``[0.5, 0.5]``*

*represent the `center` of the screen and the coordinates ``[0, 0]`` represent the `top left corner`.*

*See ``CoordinateSystem`` for more details about coordinate system.*

*Examples:*

*Click the point of ``(100, 100)`` of screen which resolution is ``(1920, 1080)``::*

*poco.click([100.0 / 1920, 100.0 / 1080])*

*Args:*

*pos (:obj:`list(float, float)` / :obj:`tuple(float, float)`): coordinates (x, y) in range of 0 to 1*

*Raises:*

*InvalidOperationException: when clicked outside of the screen*

*"""*

**if** **not** (0 <= pos[0] <= 1) **or** **not** (0 <= pos[1] <= 1):

**raise** InvalidOperationException('Click position out of screen. pos=*{}*'.format(repr(pos)))

ret = self.agent.input.click(pos[0], pos[1])

self.wait\_stable()

**return** ret

#### rclick

个人理解应该是右键点击。功能暂未实现。

调用时直接抛NotImplementedError的异常。

**def** rclick(self, pos):

**raise** NotImplementedError

#### double\_click

个人理解应该是双击。功能暂未实现。

调用时直接抛NotImplementedError的异常。

**def** double\_click(self, pos):

**raise** NotImplementedError