APP指令手册

1.App精灵基本操作

(1).多开模式

多开模板 模板 1 模板 2 模板 3....

示例:

多开模板 device1.txt device2.txt device3.txt .....

注意: 多开 当前目录\Template 模板, 不超过 30 个, 模板名不能有空格 长度不大于 30 机延迟

(2)app模式

app模式

注意:

① .app精灵首行必须使用此指令

(3)服务端口

服务端口 端口号

示例:

app模式

服务端口 4723

注意:

① . 服务端口对应 InitAppSpirit 第二个参数值

②. 多开模式下服务端端口必须间隔 1, 比如第一个端口 4723, 第二端口 4725,第三个端口 4727....

(4)主函数代码格式

```
app模式
_服务端口 4725
<script>
    async function main()
    {
  var appSpirit = await InitAppSpirit("127.0.0.1:60021", 4725,
  "com.android.settings", ".HWSettings", "9.0");
         await appSpirit.pause(5000);
         await appSpirit.saveScreenshot("D:\\app.png");
         var arrPoint;
         if((arrPoint = await FindImage("D:\\app.png", "D:\\1.png",
  "0.9")) != null)
         {
             console.log("x="+arrPoint[0]);//x
             console.log("y="+arrPoint[1]);//y
         }
         else
         {
             console.log("未查找到\n");
         }
     }
 main();
</script>
```

(5) 启动参数

①.配置文件

app精灵 JSData/InitAppSpirit.js

```
  var appSpirit;
  async function
    InitAppSpirit(devicesId
    , serverPort, package,
    activity, version)
    { var opts = {port:
    serverPort,
    capabilities: {
           platformName: "Android",
           platformVersion: version,
           deviceName: "appSpirit",
           udid: devicesId,
           appPackage: package,
           appActivity: activity,
           unicodeKeyboard: "true",
           resetKeyboard: "true",
           noReset: "true",
           newCommandTimeout: "600",
           automationName: "UiAutomator2"
           //automationName: "UiAutomator1"//速度快，但是不稳定
       }
    };
    return (appSpirit = await wdio.remote(opts));}
```

说明：

- appSpirit 全局对象
- InitAppSpirit app 精灵初始化函数。
- 参数一 devicesId 字符串 设备 ID。电脑连接设备，cmd 输入 adb devices 获取参数
- 参数二 serverPort 整型 服务端口。自定义服务端口，多开模式端口必须间隔 1，比如第一个端口 4723，第二端口 4725，第三个端口 4727....
- 参数三 package 字符串app 包名。可通过 获取 App 信息按钮获取
- 参数四 activity 字符串 app活动窗口。可通过 获取App信息 按钮获取
- 参数五 version 字符串 手机系统版本号

②.获取启动参数

```
appSpirit.capabilities;
//通过全局对象获取启动参数
```

(6)全局对象

```
await appSpirit.deleteSession();//关闭 appSpirit 对象
```

2.超时设置

```
await appSpirit.setTimeouts(5000);
//通过此方法实现 设置超时

await appSpirit.setImplicitTimeout(5000);
//设置隐式等待超时， 脚本在搜索元素时应等待的时间，默认隐式等待为0ms

await appSpirit.setAsyncTimeout(5000);
//设置允许由 execute async 执行的异步脚本的等待超时（以毫秒为单位）
```

3.屏幕操作

（1）屏幕方向

```
await appSpirit.getOrientation();
//获取当前设备 横竖屏状态。返回值 字符串 LANDSCAPE(横屏) PORTRAIT(竖屏)

await appSpirit.setOrientation("LANDSCAPE");
//设置 横竖屏， LANDSCAPE(横屏) PORTRAIT(竖屏)
```

（2）屏幕录制

```
await appSpirit.startRecordingScreen();
//开始录制屏幕

await appSpirit.stopRecordingScreen();
//停止录制屏幕；返回值：Base64 编码的字符串。
```

（3）屏幕截图

```
await SaveScreenshot(String pic_path, int x = 0, int y = 0, int width = 0, int height = 0);
//参数1 字符串类型，pic_path 图片路径
//参数2，3，4，5 整型，可选参数，顶点x，y坐标，区域宽和高。若参数为0，则截取全屏
//成功 返回字符串类型，图片的base64
//图片必须以bmp格式保存
```

(4) 摇动手机

```
await appSpirit.shake();
//摇动设备
```

(5) 解锁屏操作

```
await appSpirit.lock();
//锁定设备

await appSpirit.unlock();
//解锁设备

await appSpirit.isLocked();
//判断设备是否被锁定， 如果设备已锁定， 返回 True， 否则为 false

await appSpirit.rotateDevice(100, 100);
//三维旋转设备
```

4. 获取系统状态

```
await appSpirit.getPerformanceDataTypes();
//返回 允许读取的系统状态的信息类型，例如cpu，内存，网络流量和电池

await appSpirit.getPerformanceData('my.app.package', 'cpuinfo', 5);
//返回 字符串数组类型，读取的系统状态信息，例如cpu，内存，网络流量和电池
//参数一，应用程序的软件包名称
//参数二，要读取的系统状态的类型。性能数据类型（cpuinfo | batteryinfo | networkinfo | memoryinfo）
//参数三，尝试读取的次数（可选）
```

5. 电源设置

```
await appSpirit.powerAC('on');
//将电池充电器的状态设置为已连接('on')或未连接('off')（仅模拟器）延迟 1秒-2秒

await appSpirit.powerCapacity(50);
//设置电池百分比（仅模拟器）
```

6.网络操作

```
await appSpirit.toggleAirplaneMode();
//切换飞行模式(部分设备不可用)

await appSpirit.toggleData();
//切换数据服务状态(部分设备不可用)

await appSpirit.toggleWiFi();
//切换WiFi 服务的状态
```

```
await appSpirit.toggleLocationServices();
//切换位置服务的状态

await appSpirit.sendSms('123456789', 'hello spirite');
//发送短信（仅模拟器）

await appSpirit.gsmCall('123456789', 'call');
//拨打GSM 电话（仅模拟器）

await appSpirit.gsmSignal(3);
//设置GSM 信号强大，信号强度在[0，4]范围内

await appSpirit.gsmVoice('home');
//设置GSM 语音状态（仅模拟器）
//参数：GSM 语音状态'unregistered'(未注册), 'home'(家庭), 'roaming'(漫游), 'searching'(搜索), 'denied'(被拒绝), 'off'(关闭),
'on'(打开).

await appSpirit.toggleNetworkSpeed('lte');
//设置网络速度（仅模拟器）
参数：网络类型'full','gsm', 'edge', 'hscsd', 'gprs', 'umts', 'hsdpa', 'lte', 'evdo'
```

7.键盘操作

```
await appSpirit.pressKeyCode(10);
//按Android 设备上的特定键
//参数一，特定按键码值，4 为返回键。其他键值，自行百度搜索

await appSpirit.longPressKeyCode(10);
//长按Android 设备上的特定键

await appSpirit.hideKeyboard();
//隐藏软键盘

await appSpirit.isKeyboardShown();
判断是否显示软键盘，如果显示键盘，返回True
```

8. 栏目相关

```
await appSpirit.openNotifications();
//打开通知栏

await appSpirit.getSystemBars();
//返回状态栏和导航栏的可见性信息
```

```
await appSpirit.getDeviceTime();
//获取系统时间,返回系统时间（字符串）

await appSpirit.getDisplayDensity();
//返回整型，设备pid（像素密度）

await appSpirit.fingerPrint(1);
//指纹识别（仅模拟器）
参数:存储在Android Keystore 系统中的指纹（从1 到10）
```

9.触屏操作、鼠标操作

(1) 鼠标操作(部分设备不可用)

```
moveTo(10, 10);
//将鼠标移动指定元素的偏移位置

click();
//在当前鼠标坐标处单击任意鼠标

doubleClick();
//双击元素

buttonDown();
//当前鼠标坐标处单击并按住鼠标左键

buttonUp();
//释放先前按住的鼠标按钮
```

(2)触屏操作

```
await appSpirit.touchAction({
  action: 'tap',
  x: 30,
  y: 20
});
//触摸xy指定坐标点

await appSpirit.touchDoubleClick(element.elementId);
//在指定元素上触摸双击
//参数，元素id

await appSpirit.touchPerform([
  { action: 'press', options: { x: 100, y: 250 }},
  { action: 'wait',options: {ms: 100}},
  { action: 'moveTo', options: { x: 300, y: 100 }},
  { action: 'release' }
]);
//触摸动作 action行为，options 行为参数
//行为：tap 点击  press 短按  longPress 长按  wait 等待  moveTo滑动
release 释放
//此函数能实现一切 鼠标 和 触屏操作，建议使用。
```

10.地理位置

```
await appSpirit.getGeoLocation();
//获取地理位置,返回值整型 纬度 经度 高度

await appSpirit.setGeoLocation({latitude: "121.21", longitude:
"11.56", altitude: "94.23"});
//设置地理位置,latitude纬度 longitude经度 altitude高度
```

11.系统剪切板

```
await appSpirit.getClipboard();
//获取系统剪贴板的内容 返回值，base64编码的字符串，如果剪贴板为空，则为空
字符串

var data = new Buffer("网页精灵").toString('base64');
await appSpirit.setClipboard(data, 'plaintext');
//设置系统剪贴板的内容，仅设置plaintext纯文本
//参数一 字符串 base64编码
```

12.全局设置

```
await appSpirit.getSettings();
//获取 设定值

await appSpirit.updateSettings({ignoreUnimportantViews: true});
//更改 设定值，例如ignoreUnimportantViews: true    忽略其他不相关元素
```

13.应用App操作

(1)信息操作

```
await appSpirit.startActivity("com.example", "ActivityName");
//通过app包名称和活动名称来启动app 一般用于多个app控制

await appSpirit.getCurrentActivity();
//返回当前app活动窗口的名称

await appSpirit.getCurrentPackage();
//返回当前app包的名称
```

(2) 应用操作

```
await appSpirit.installApp("D: \\test.apk");
//安装app

await appSpirit.isAppInstalled('com.example.AppName');
```

```
//检查设备上是否安装了指定的应用程序，如果已安装，返回true；否则，返回false

await appSpirit.launchApp();
//如果被测应用关闭或在后台运行 ，它将启动被测应用

await appSpirit.background(10);
//将此会话当前运行的应用程序发送到后台,参数时间单位为秒

await appSpirit.closeApp();
//关闭应用

await appSpirit.reset();
//重置此会话的正在运行的应用

await appSpirit.removeApp("com.example.AppName");
//通过包名 卸载应用

await appSpirit.activateApp("com.example.AppName");
//通过包名 激活应用

await appSpirit.terminateApp('com.example.AppName');
//终止设备上指定的应用

await appSpirit.queryAppState('com.example.AppName');
//获取设备上应用状态。返回值，0未安装。1没有运行。2在后台运行或已暂停。3在后台运行。4正在前台运行。

await appSpirit.getStrings("en");
//获取应用程序所有的字符串。
```

(3) 元素操作

① 元素查找

```
await appSpirit.$("~content-desc");
//查找元素，查找元素方法有以下几类：
①、content-desc，UI元素的唯一标识符，该方式需在属性前加 "~" 波浪号content-desc示例，await appSpirit.$("~content-desc");
②、Class name，类名定位，注意app多个相同class name属性元素Class name示例，await appSpirit.$("className");
```

③、UiSelector，使用UI Automator API定位，可设定多个条件定位，**建议使用**UiSelector示例，
var selector ='new UiSelector().text("text").className("className").resourceId("resourceId")';
await appSpirit.$(`android=${selector}`);//text className resourceId可以任意增减组合
④、XPath，XPath 定位，相率较低，但它是万能的定位方式。XPath示例，await appSpirit.$("//android.widget.TextView[@text='网页精灵']");

```
await appSpirit.$$("resource-id");
//查找相同元素属性值数组，查找方式同上
//另外还支持resource-id属性查找数组元素
//查找元素失败元素属性elementId值为undefined，可用作是否查找成功
```

② **元素操作**

```
click();
//点击元素中心点

addValue("网页精灵");
//参数，字符串
//发送字符串到元素

setValue("网页精灵");
//参数，字符串
//设置元素值

clearValue();
//清除元素值

await appSpirit.elementSubmit(formElement.elementId);
//提交表单元素
//参数一， 元素id
```

③ **元素属性**

```
getText();
//返回元素的可见文本

getTagName();
//获取标签名称

getAttribute("content-desc");
//获取指定元素属性的值，如果未设置返回 null

isSelected();
//判断元素是否被选择，返回值，boolean类型

isEnabled();
//判断当前元素是否可用，返回值，boolean类型

isDisplayed();
//判断当前元素是否可见，返回值，boolean类型

getLocation();
//获取元素在页面或屏幕上的顶点坐标，返回x、y，整型

getSize();
//获取元素宽和高，返回width、heigh

await appSpirit.getElementRect(element.elementId);
 //获取元素位置，返回x、y、width、height，矩形位置和大小
//参数1，元素id

getCSSProperty("style");
//获取元素指定的CSS属性值

await appSpirit.getElementLocationInView(element.elementId);
//获取元素在页面或屏幕上的顶点坐标，返回x、y坐标，整型

await appSpirit.getActiveElement();
//获取当前会话的活动元素

isEqual(element);
//判断指定元素与参数一是否引用同一个元素，返回值，boolean类型
//参数一，元素object
```

④大图找小图

```
var arrPoint = await FindImage("d:\\big.png", "d:\\small.png", "0.9");
//大图找小图， 参数1、大图  参数2、 小图  参数3、相似度0.0-1.0
//返回值 数组，arrPoint[0] x坐标，arrPoint[1] y坐标 失败返回null
//3个参数统一为字符串类型，图片必须是png格式
//尽量不要使用中文路径
```

(4)切换App环境

```
await appSpirit.getContext();
//获取当前app是在NATIVE还是WEBVIEW环境
//返回值 当前环境的名称

await appSpirit.getContexts();
//获取app环境集合
//返回所有环境的名称（Array<String>）

await appSpirit.switchContext(contexts[1]);
//切换NATIVE或者WEBVIEW环境
//参数为 要更改环境的名称
```

(5) 应用程序层次结构

```
await appSpirit.getPageSource();
//获取当前的应用程序层次结构XML（应用程序）或页面源码（Web）
//此方法获取应用程序层次结构，便于元素定位
```

14.文件操作

```
var data = new Buffer("Hello World").toString('base64');
await appSpirit.pushFile('/data/local/tmp/file.txt', data);
//设置 设备上文件的内容
//参数一，文件路径
//参数二，base64格式的字符串

await appSpirit.pullFile('/data/local/tmp/file.txt');
//获取设备上文件内容，返回值base64格式的字符串

await appSpirit.pullFolder('/data/local/tmp/');
//获取设备指定文件夹，返回 Base64编码的字符串，文件格式为zip
```

15.web操作

```
await appSpirit.switchToWindow("handle");
//将焦点切换另一个窗口

await appSpirit.closeWindow();
//关闭当前窗口

await appSpirit.getWindowHandle();
//获取当前窗口句柄

await appSpirit.getWindowHandles();
//获取所有窗口句柄列表

await appSpirit.getTitle();
//获取当前页面标题

await appSpirit.getWindowRect();
//获取窗口的位置

await appSpirit.setWindowRect(0, 0, 800, 600);
//设置窗口位置

await appSpirit.getWindowSize();
//获取窗口大小

await appSpirit.setWindowSize(10, 10);
//设置窗口大小

await appSpirit.maximizeWindow();
//最大化窗口

await appSpirit.url("http://www.webspirit.cn");
//打开新的URL

await appSpirit.getUrl();
//获取当前页面url

await appSpirit.back();
//返回上一页
```

```
await appSpirit.forward();
//前一页

await appSpirit.refresh();
//刷新

await appSpirit.getCookies();
//获取cookie

await appSpirit.setCookies([{
   name: 'myCookie',
   value: 'some content'
}]);
//设置Cookie

await appSpirit.deleteCookies("cookie_name");
//删除指定名称的cookie

await appSpirit.deleteCookies();
//删除当前页面所有的cookie

await appSpirit.switchToFrame(3);
//切换指定iframe

await appSpirit.switchToParentFrame();
//切换父iframe

await appSpirit.executeAsync('mobile: shell', {'command':'screencap -p
/sdcard/app.png'});
//执行异步脚本

await appSpirit.execute('mobile: shell', {'command':'screencap -p
/sdcard/app.png'});
//执行脚本   上为，adb命令示例
```

16.执行多条命令

```
const script = `
   var destElement = await driver.$('//android.widget.TextView[@text='移动网络']');
   await destElement.click();
`;
await appSpirit.driverScript(script);
//执行多条命令,script  内部使用  driver调用
//此指令能提高脚本效率
```

17.OCR文字识别

```
//初始化Ocr文字识别功能
await InitOcr(String appId, String apiKey, String secretKey);
//以上三个参数由百度ocr提供

//获取图片文字(中文 数字 英文 常用符号)
await GetImageWords(String image_path);
//参数1 字符串型，图片路径
//成功，返回识别到的内容，失败，返回null

查找文字在图片上的坐标
var arrPoint = await FindImageWords(String imagePath, String words, int index = 0);
//参数1 字符串型，图片路径
//参数2 字符串类型，目标文字串（单个文字查找更精准）
//参数3 查找第n个目标文字的坐标，一般用于区分多个相同目标文字的坐标
//返回值 数组，arrPoint[0] x坐标，arrPoint[1] y坐标 失败返回null
```