## splash 对象属性

function main(splash)

return {hello=”world!”}

end

main()方法的第一个参数是splash，它类似于Selenium中的WebDriver对象，我们可以调用它的一些属性和方法来控制加载过程。

### args

该属性可以获取加载时配置的参数，比如URL，如果为GET请求，它还可以获取GET请求参数；如果为POST请求，它可以获取表单提交的数据。Splash也支持使用第二个参数作为args，例如：

function main(splash, args)

local url = args.url

end

这里第二个参数args相当于splash.args属性，以上代码等价于：

function main(splash)

local url = splash.args.url

end

### js\_enabled

这个属性是Splash的JavaScript执行开关，可以将其配置为true或者false来控制是否执行JavaScript代码，默认为true，例如：这里禁止执行JavaScript代码：

Function main(splash, args)

Splash:go(“https://www.baidu.com”)

Splash.js\_enabled = false

Local title = splash:evaljs(“document.title”)

Return {title = title}

End

运行结果会抛出异常

一般来说，不用设置此属性，默认开启即可

### resource\_timeout

此属性可以设置加载的超时时间，单位是秒。如果设置为0或者nil(类似python中的None)，代表不检测超时。示例如下：

Function main(splash)

Splash.resource\_timeout = 0.1

Assert(splash:go(‘httpw://www.baidu.com))

Return splash:png()

End

这里将超时时间设置为0.1秒。如果在0.1秒之内没有得到响应，就会抛出异常。

此属性适用于网页加载速度较慢的情况下设置，如果超过了某个时间无响应，则直接抛出异常并忽略即可。

### images\_enabled

此属性可以设置图片是否加载，默认是加载的。禁用该属性之后，可以节省网络流量并提高网页加载速度。但是需要注意的是，禁用图片加载可能会影响JavaScript渲染。因为禁用图片之后，它的外层DOM节点的高度会受到影响，进而影响DOM节点的位置。因此，如果JavaScript对图片节点有操作的话，其执行就会受到影响。

另外值得注意的是，Splash使用了缓存。如果一开始就加载出来了网页图片，然后禁用了图片加载，再重新加载页面，之前加载好的图片可能还会显示出来，这时直接重启Splash即可。

禁用图片加载的示例如下：  
function main(splash, args)

Splash.images\_enabled = false

Assert(splash:go(‘https://www.jd.com’))

Return {png = splash:png()}

这样返回的页面截图就不会带有任何图片，加载速度也会快很多。

### plugins\_enabled

此属性可以控制浏览器插件（如Flash插件）是否开启。默认情况下，此属性是false，表示不开。使用如下代码控制其开关和关闭。

Splash.plugins\_enagled = true/false

### scroll\_position

通过设置此属性可，我们可以控制页面上下或者左右滚动。这是一个比较常用的属性，示例如下：

Function main(splash, args)

Assert(splash:go(‘https://www.taobao.com’))

Splash.scroll\_position={y = 40}

Return {png = splash:png()}

End

这样我们可以控制页面向下滚动400像素值。

如果让页面左右滚动，可以传入x参数，代码如下：

Splash.scroll\_position={x=100,y=200}

## Splash 对象的方法

### go()

该方法用来请求某个链接欸，而且它可以模拟GET和POST 请求，同时支持传入请求头，表单等数据，其用法如下：

Ok, reason = splash: go{url, baseurl = nil, headers = nil, http\_method=”GET”,body = nil, formadata = nil}

#### 参数说明：

url:请求的URL  
 baseurl:可选参数，默认为空，表示资源加载相对路径  
 headers：可选参数，默认为空，表示请求头。  
 http\_method:可选参数，默认为GET,同时支持POST

body:可选参数，默认为空，发POST请求时的表单数据，使用的Content-type为application/json  
 formdata:可选参数，默认为空，POST的时候的表单数据，使用的Content-type为application/x-www-form-urlencoded

该方法返回结果是ok和原因reason的组合，如果ok为空，代表网页加载出现了错误，此时reason变量中包含了错误的原因，否则证明页面加载成功。示例如下：

Function main(splash, args)

Local ok,reason = splash:go{<http://httpbin.org/post>,http\_method=”POST”,body=”name= Germey”}

If ok then

Return splash:html()

End

End

我们模拟了一个POST请求，并传入了POST的表单数据，如果成功，则返回页面的源代码。

### 2、wait()

此方法可以控制页面的等待时间，使用方法如下：

Ok, reason = splash:wait{time,cancel\_on\_redirect=false, cancel\_on\_error = true}

#### 参数说明：

Time:等待的秒数

Cancel\_on\_redirect: 可选参数，默认为false，表示如果发生了重定向就停止等待，并返回重定向结果

Cancel\_on\_error:可选参数，默认为false，表示如果发生了加载错误，就停止等待

返回结果同样是ok和原因reason的组合

示例如下：  
function main(splash)

Splash:go(‘https://www.taobao.com’)

Splash:wait(2)

Return {html=splash:html()}

End

实现访问淘宝并等待2秒，随后返回页面源代码的功能

### 3、jsfunc()

此方法可以直接调用JavaScript定义的方法，但是所调用的方法需要用双中括号包围，相当于实现了JavaScript方法到Lua脚本的转换。示例如下：  
function main(splash,args)

Local get\_div\_count = splash:jsfunc([[

Function(){

Var body = document.body;

Var divs = body.getElementsByTagName(‘div’);

Return divs.length

}

]])

Splash:go(‘https://www.baidu.com’)

Return (“There are %s DIVS”):format(get\_div\_count())

End

首先，我们声明了一个JavaScript定义的方法，然后再界面加载成功后调用了此方法计算出了页面中div节点的个数

#### 关于：

JavaScript到Lua脚本的更多转换细节，可以参考文档：  
 https://splash.readthedocs.io/en/stable/scripting-ref.html#splash-jsfunc

### 4、evaljs()

此方法可以执行JavaScript代码并返回最后一条JavaScript语句的返回结果，使用方法如下：  
result = splash: evaljs(js)

比如，可以用下面的代码来获取页面标题：

Local title = splash:evaljs(‘doucment.title’)

### 5、runjs()

此方法可以执行JavaScript代码，它与evaljs()的功能类似，但是更偏向于执行于某些动作或者声明某些方法。例如：

Function main(splash, args)

Splash:go(‘https://www.baidu.com’)

Splash:runjs(‘foo = function()’ {return ‘bar’})

Return result

End

我们用runjs()先声明一个JavaScript定义的方法，然后通过evaljs()来调用得到的结果

结果：

bar

### 6、autoload()

此方法可以设置每个页面访问时自动加载的对象，使用方法如下：

Ok, reason = splash:autoload{source\_or\_url, source=nil, url =nil}

#### 参数：

Source\_or\_url: JavaScript代码或者JavaScript库链接

Source：JavaScript代码

url: JavaScript 库链接

但是此方法只负责加载JavaScript代码或者库，不执行任何操作。如果要执行操作，可以调用evaljs()或者runjs()方法。示例如下：  
function main(splash, args)

Splash: autoload([[

Function get\_document\_title(){

Return document.title;

}

]])

Splash:go(“https://www.baidu.com”)

Return splash: evaljs(“get\_document\_title”)

End

运行结果：

百度一下，你就知道

另外，我们也可以使用autoload()方法加载某些方法库，如jQuery，示例如下：

Function main(splash, args)

Assert(splash: autoload(‘https://code.jquery.com/jquery-2.1.3.min.js’))

Assert(splash:go(‘https://www.taobao.com’))

Local version = splash: evaljs(‘$.fn.jquery’)

Return ‘JQuery version: ’ ..version

End

运行结果：

JQuery version:2.1.3

### 7、call\_later()

此方法可以通过设置定时任务和延时时间来实现任务延时执行，并且可以在执行前通过cancel()方法重新执行定时任务。示例如下：

Function main(splash , args)

Local snapshots= {}

Local timer = splash: call\_later(function()

Snapshots[“a”]= splash:png()

Splash:wait(1.0)

Snapshots[“b”] = splash:png() //截图

End ,0.2)

Splash:go(“https://www.taobao.com”)

Splash:wait(3.0)

Retrun snapshots

End

可以发现，第一次截图网页还没有加载出来，截图为空，第二次网页便加载出来了

### 8、http\_get()

此方法可以模拟发送HTTP的GET请求，使用方法如下：

Response = splash: http\_get{url, headers=nil, follow\_redirects = true}

#### 参数说明：

url:请求URL

headers：可选参数，默认为空，请求头

follow\_redirects:可选参数，表示是否启动自动重定向，默认为true

示例如下：

Function main(splash, args)

Local treat = require(“treat”)

Local response = splash: http\_get(‘http://httpbin.org/get’)

Return {

Html = treat.as\_string(response.body)

url = response,url,

status = response.status

}

End

### 9、http\_post()

和http\_get()方法类似，此方法用来模拟发送POST请求，不过多了一个参数body，使用方法如下：  
response = splash:http\_post{url, headers=nil, follow\_redirects = true, body =nil}

#### 参数说明：

url:请求的url

headers: 可选参数，默认为空，请求头

follow\_redirects: 可选参数，表示是否启动自动重定向，默认为true

body:可选参数，即表单数据，默认为空

示例：

Function main(splash, args)

Local treat = require(“treat”)

Local json = require(“json”)

Local response = splash:http\_post{‘http://httpbin.org/post’,

Body = json.encode({name = “Germey”}),

Headers={[“conteng-type”]=”application/json”}

}

Return {

Html = treat.as\_string(response.body),

url = response.url,

status= response.status

}

End

### 10、set\_content()

此方法用来设置页面的内容，示例如下：

Function main(splash)

Assert(splash:set\_content(“<html><body><h1>hello</h1></body></html>”))

Return splash:png()

End

### 11、html()

此方法用来获取网页的源代码，示例如下：

Function main(splash, args)

Splash:go(‘https://httpbin.org/get’)

Return splash: html()

End

### 12、png()

此方法用于获取PNG格式的网页截图

Function main(splash, args)

Splash:go(‘https://www.taobao.com’)

Return splash:png()

End

### 13、jpeg()

此方法用于获取JPEG格式的网页截图

Function main(splash, args)

Splash:go(‘https://www.taobao.com’)

Return splash:jpeg()

End

### 14、har()

此方法用于获取页面加载过程描述，示例：

Function main(splash,args)

Splash:go(‘https://taobao.com’)

Return splash:har()

End

### 15、url()

此方法可以获取当前正在访问的URL，示例如下：

Function main(splash, args)

Splash:go(‘https://www.baidu.com’)

Return splash:url()

End

### 16、get\_cookies()

此方法可以获取当前页面的Cookies，示例如下：

Function main(splash,args)

Splash:go(‘https://www.baidu.com’)

Return splash:get\_cookies()

End

### 17、add\_cookies()

此方法可以为当前页面添加Cookie，用法如下：

Cookies = splash:add\_cookies{name, value, path =nil , domain =nil, expires = nil ,httpOnly = nil, secure = nil}

该方法的各个参数代表Cookie的属性。

示例：

Function main(splash)

Splash:add\_cookie{“sessionid“,”237465ghgfsg”,”/”, domain = “http://example.com”}

Return splash:html()

End

### 18、clear\_cookies()

此方法可以清除所有的Cookies，示例：

Function main(splash)

Splash:go(‘https://www.taobao.com’)

Splash:clear\_cookies()

Return splash:get\_cookies()

End

### 19、get\_viewport\_size()

此方法可以获取当前浏览器页面额大小，即宽高，示例：

Function main(splash)

Splash:go(‘https://www.taobao.com’)

Return splash:get\_viewport\_size()

End

### 20、set\_viewport\_size()

此方法可以设置当前浏览器页面的大小，即宽高，

Splash: set\_viewport\_size(width, height)

示例：

Function main(splash)

Splash:set\_viewport\_size(400,700)

Assert(splash:go(‘https://cuiqingcai.com’))

Return splash:png()

End

### 21、Set\_viewport\_full()

此方法可以设置浏览器全屏显示：

Function main(splash)

Splash: set\_viewport\_full()

Assert(splash:go(‘https://cuiqingcai.com’))

Return splash:png()

End

### 22、set\_user\_agent()

此方法可以设置浏览器的User-Agent，示例：

Function main(splash)

Splash: set\_user\_agent(‘Splash’)

Splash:go(‘http://httpbin.org/get’)

Return splash:html()

End

### 23、set\_custom\_hearders()

此方法可以设置请求头，示例如下：

Function main(splash)

Splash:set\_custom\_headers({

[‘User\_Agent’] = “Splash”,

[‘Site’] = “Splash”,

})

Splash:go(‘http://httpbin.org/get’)

Return splash:html()

End

### 24、select()

该方法可以选中符合条件的第一个节点，如果有多个节点符合条件，则只会返回一个，其参数是CSS选择器，示例如下：  
function main(splash)

Splash:go(“httpss://www.baidu.com/”)

Input = splash:select(“#kw”)

Input:send\_text(“Splash”)

Splash:wait(3)

Return splash:png()

End

使用send\_text()方法发填写文本

### 25、select()\_all()

此方法可以选中所有扶额和条件的节点，其参数是CSS选择器，示例：

Function main(splash)

Local treat = require(‘treat’)

Assert(splash: go(‘http://quotes.toscrape.com/’))

Assert(splash:wait(0.5))

Local texts = splash:select\_all(‘.quote . text’)

Local results = []

For index, text in ipairs(text) do

Results[index] = text.node.innerHTML

End

Return treats.as\_array(results)

End

我们通过CSS选择器选中了节点的正文内容，随后遍历了所有节点，将其中的文本获取下来。

### 26、mouse\_click()

此方法可以模拟鼠标点击操作，传入的参数为坐标值x和y。此外，也可以直接选中某个节点，然后调用此方法。示例：

Function main(splash)

Splash:go(‘https://www.baidu.com’)

Input = splash:select(‘#kw’)

Input:send\_text(‘Splash’)

Submit = splash:select(‘#su’)

Submit:wait(3)

Return splash:png()

End

这里我们先选中页面的输入框，输入了文本，然后选中“提交“按钮，调用了mouse\_click()

方法提交查询，然后页面等待三秒，返回截图。