

# KikoPlay 脚本开发参考

---

2021.03 By Kikyous, 本文档适用于KikoPlay 0.8及以上版本

## 目录

- [脚本类型](#)
  - [公共部分](#)
  - [弹幕脚本](#)
  - [资料脚本](#)
  - [资源脚本](#)
- [KikoPlay API](#)
- [数据类型](#)
  - [DanmuSource](#)
  - [DanmuComment](#)
  - [AnimeLite](#)
  - [EpInfo](#)
  - [Character](#)
  - [Anime](#)
  - [MatchResult](#)
  - [LibraryMenu](#)
  - [ResourceItem](#)
  - [NetworkReply](#)

## 脚本类型

KikoPlay中有三类脚本：

- 弹幕脚本：位于script/danmu目录下，提供弹幕搜索、下载、发送弹幕等功能
- 资料脚本：位于script/library目录下，提供动画（或者其他类型的条目）搜索、详细信息获取、分集信息获取、标签获取、自动关联等功能
- 资源脚本：位于script/resource目录下，提供资源搜索功能

所有的脚本均为Lua脚本，不同类型的脚本需要提供不同的接口

### 公共部分

每个脚本都应包含一个info类型的table，提供脚本的基本信息，包含这些内容：

```
info = {  
    ["name"] = "Bilibili",           --脚本名称  
    ["id"] = "Kikyous.d.Bilibili",  --脚本id, 不应和其他脚本的id相同  
    ["desc"] = "Bilibili弹幕脚本",  --描述信息  
    ["version"] = "0.1"             --版本信息  
}
```

脚本可以包含设置项，这些项目可以通过KikoPlay “设置”对话框-“脚本”页面-脚本列表的右键菜单-“设置” 进行设置 设置项目包含在脚本的`settings` table中，其中每一项的key为设置项的key，value是一个table，格式如下：

```
settings = {
  ["result_type"] = {
    ["title"]="搜索结果类型", --设置项标题
    ["default"]="2",          --默认值，类型均为字符串
    ["desc"]="条目类型， 2: 动画 3: 三次元", --描述信息
    ["choices"]="2,3"         --可选，如果包含这一项，用户只能从choices中进行选择，
    多个项目用逗号“,”分隔
  }
}
```

KikoPlay在加载脚本后，会将value替换为设置的值，例如上面的示例在加载后，可能会变成：

```
settings = {
  ["result_type"] = "2"
}
```

因此在脚本中，可以直接通过`settings["xxxx"]`获取设置项的值

注意，设置项值的类型都是字符串。如果用户在脚本加载后，从KikoPlay的设置对话框中修改了脚本设置项，KikoPlay会尝试调用脚本的`setoption`函数通知脚本，如果需要对修改进行响应，可以在脚本中添加这个函数：

```
function setoption(key, val)
  -- key为设置项的key，val为修改后的value
end
```

## 弹幕脚本

弹幕脚本需要包含如下函数/table：

- `function search(keyword)`

`keyword`： string，搜索关键字

返回： Array[`DanmuSource`]

完成搜索功能，可选

- `function epinfo(source)`

`source`： `DanmuSource`

返回： Array[`DanmuSource`]

返回source条目包含的分集列表，分集条目也是DanmuSource

- `function danmu(source)`

`source`: DanmuSource

返回: DanmuSource/nil, Array[DanmuComment]

从DanmuSource获取弹幕。如果在获取弹幕后source发生变化，第一个返回值为新的source，否则第一个返回值为nil，第二个值返回弹幕列表

- `supportedURLsRe`

Table，类型为 Array[string]，支持的弹幕URL正则表达式列表，可选

脚本提供的`supportedURLsRe`正则表达式会在“从URL添加弹幕”功能中对用户输入的URL进行过滤，符合条件的URL将被传入脚本，需要脚本提供`urlinfo`函数

例如，bilibili.lua支持的URL正则表达式包括：

```
supportedURLsRe = {  
    "(https?://)?www\\.bilibili\\.com/video/av[0-9]+/?",  
    "(https?://)?www\\.bilibili\\.com/video/BV[\\dA-Za-z]+/?",  
    "av[0-9]+",  
    "BV[\\dA-Za-z]+",  
    "(https?://)?www\\.bilibili\\.com/bangumi/media/md[0-9]+/?"  
}
```

- `sampleSupportedURLs`

Table，类型为 Array[string]，支持的URL示例，可选，会显示在KikoPlay添加弹幕对话框-URL页面-支持的URL类型列表中，主要用于提示用户

- `function urlinfo(url)`

`url`: string

返回: DanmuSource

从url中获取弹幕来源信息

- `function canlaunch(sources)`

`sources`: Array[DanmuSource]

返回: bool

当脚本支持发送弹幕时，需要提供这个函数，检查sources中是否有可发出弹幕的弹幕来源

- `function launch(sources, comment)`

`sources`: Array[DanmuSource]

```
comment: DanmuComment
```

返回：无错误返回`nil`，否则返回错误信息`string`

发送弹幕函数，脚本可以自行选择`sources`中的弹幕来源进行发送，注意，只有`canlaunch`返回`true`才会调用这个函数

## 资料脚本

资料脚本需要至少提供`match`或者同时提供`search`和`getep`函数。

尽管下文使用“动画”一词，但视频类型并不局限于动画。

- `function search(keyword)`

`keyword`: `string`，搜索关键字

```
返回: Array[AnimeLite]
```

完成搜索功能，可选

需要注意的是，除了下面定义的`AnimeLite`结构，还可以增加一项`eps`，类型为`Array[EplInfo]`，包含动画的剧集列表。

- `function getep(anime)`

```
anime: Anime
```

```
返回: Array[EplInfo]
```

获取动画的剧集信息。在调用这个函数时，`anime`的信息可能不全，但至少会包含`name`，`data`这两个字段。

- `function detail(anime)`

```
anime: AnimeLite
```

```
返回: Anime
```

获取动画详细信息

- `function gettags(anime)`

```
anime: Anime
```

```
返回: Array[string], Tag列表
```

KikoPlay支持多级Tag，用“/”分隔，你可以返回类似“动画制作/A1-Pictures”这样的标签

- `function match(path)`

```
path: 文件路径
```

```
返回: MatchResult
```

实现自动关联功能。提供此函数的脚本会被加入到播放列表的“关联”菜单中

- `menus`

Table, 类型为 `Array[LibraryMenu]`

如果资料库条目的scriptId和当前脚本的id相同, 条目的右键菜单中会添加`menus`包含的菜单项, 用户点击后会通过`menuclick`函数通知脚本

- `function menuclick(menuid, anime)`

`menuid`: string, 点击的菜单ID

`anime`: `Anime`, 条目信息

返回: 无

## 资源脚本

资源脚本提供资源搜索功能 (主要是bt类资源)

- `function search(keyword,page)`

`keyword`: string, 搜索关键字

`page`: 页码

返回: `Array[ResourceItem]`

- `function getdetail(item)`

`item`: `ResourceItem`

返回: `ResourceItem`, 包含`magnet`字段的item信息

可选, 如果在搜索中无法确定资源的`magnet`信息, 脚本需要提供`getdetail`函数获取详细信息。

## KikoPlay API

KikoPlay提供的API位于kiko表中, 通过`kiko.xxx`调用

- `httpget(url, query, header, redirect)`

`url`: string

`query`: 查询, `{[key]=value,...}`, 可选, 默认为空

`header`: HTTP Header, `{[key]=value,...}`, 可选, 默认为空

`redirect`: bool, 是否自动进行重定向, 默认`true`

返回: string/nil, `NetworkReply`

发送HTTP GET请求。返回的第一个值表示是否发生错误, 没有错误时为nil, 否则是错误信息

- `httpgetbatch(urls, querys, headers, redirect)`

`urls`: `Array[string]`

`quers`: 查询, `Array[{{key}=value,...}]`, 可选, 默认为空  
`headers`: HTTP Header, `Array[{{key}=value,...}]`, 可选, 默认为空  
`redirect`: bool, 是否自动进行重定向, 默认true  
返回: string/nil, `Array[NetworkReply]`

和`httpget`类似, 但可以一次性发出一组HTTP Get请求, 需要确保`urls`、`quers`和`headers`中的元素一一对应, `quers`和`headers`也可以为空

- `httppost(url, data, header)`

`url`: string  
`data`: string, POST数据  
`header`: HTTP Header, `{{key}=value,...}`, 可选, 默认为空  
返回: string/nil, `NetworkReply`

发送HTTP POST请求。返回的第一个值表示是否发生错误, 没有错误时为nil, 否则是错误信息

- `json2table(jsonstr)`

`jsonstr`: string, json字符串  
返回: string/nil, Table

将json字符串解析为lua的Table 返回的第一个值表示是否发生错误, 没有错误时为nil, 否则是错误信息

- `table2json(table)`

`table`: Table, 待转换为json的table  
返回: string/nil, string

将lua的Table转换为json字符串 返回的第一个值表示是否发生错误, 没有错误时为nil, 否则是错误信息

- `compress(content, type)`

`content`: string, 待压缩的字符串  
`type`: 压缩方式, 可选, 默认为gzip, 目前也只支持gzip  
返回: string/nil, string

压缩字符串, 第二个返回值为压缩结果

返回的第一个参数表示是否发生错误, 没有错误时为nil, 否则是错误信息

- `decompress(content, type)`

`content`: string, 待压缩的字符串  
`type`: 压缩方式, 可选, 支持inflate和gzip, 默认为inflate

返回: string/nil, string

解压缩字符串, 第二个返回值为解压缩结果

返回的第一个值表示是否发生错误, 没有错误时为nil, 否则是错误信息

- `execute(detached, program, args)`

`detached`: bool, 是否等待程序执行结束, true:不等待

`program`: string, 执行的程序

`args`: Array[string], 参数列表

返回: string/nil, bool/number

执行外部程序。返回的第一个值表示调用参数是否正确, 没有错误时为nil, 否则是错误信息。第二个值为程序执行结果, 如果`detached=true`, 值为true/false表示程序是否启动; 如果`detached=false`, 值为外部程序的返回值

- `hashdata(path_data, ispath, filesize, algorithm)`

`path_data`: string, 文件路径或者待计算hash的数据

`ispath`: bool, 第一个参数是否是文件路径, 默认=true

`filesize`: number, 只有第一个参数为文件路径才有意义, 表示读取文件的大小, =0表示读取整个文件, 否则只读取前`filesize` bytes

`algorithm`: string, hash算法, 默认为md5, 可选:

md4,md5,sha1,sha224,sha256,sha386,sha512

返回: string/nil, string

计算文件或者数据的hash, 第一个返回值表示是否出错, 第二个返回值为hash值

- `log(...)`

打印输出到KikoPlay的“脚本日志”对话框中。支持多个参数, 如果只有一个参数且类型为Tabel, 会以json的形式将整个Tabel的内容输出

- `message(msg, flags)`

`msg`: string, 消息内容

`flags`: number, 标志, 默认为1 (NM\_HIDE), 多个标志使用|运算, 其他的标志有:

NM_HIDE=1	一段时间后自动隐藏
NM_PROCESS=2	显示busy动画
NM_SHOWCANCEL = 4	显示cancel按钮
NM_ERROR = 8	错误信息
NM_DARKNESS_BACK = 16	显示暗背景, 阻止用户执行其他操作

目前只支持资料脚本在“资料”页面顶部弹出消息，NM\_SHOWCANCEL在这里不起作用。如果没有NM\_HIDE标志，弹出的消息会一直显示，直到下一个消息出现。

- `dialog(dialog_config)`

`dialog_config`: Table, 配置对话框显示内容, 内容包括:

```
{
  ["title"]=string,  --对话框标题, 可选
  ["tip"]=string,    --对话框提示信息
  ["text"]=string,   --可选, 存在这个字段将在对话框显示一个可供输入的文本
框, 并设置text为初始值
  ["image"]=string   --可选, 内容为图片数据, 存在这个字段将在对话框内显示图
片
}
```

返回: bool, string

展示一个对话框, 第一个返回值表示用户点击接受(true)还是直接关闭(false), 第二个返回值为用户输入的文本

- `sttrans(str, to_simp)`

`str`: string, 源字符串

`to_simp`: bool, 是否转换为简体中文, true: 转换为简体中文, false: 转换为繁体中文

返回: string/nil, string

简繁转换, 这个函数只有在windows系统上有效, 其他平台上会原样返回。第一个返回值表示是否出错, 第二个返回值为转换后的结果

- `envinfo()`

返回: Table, 包含:

```
{
  ["os"]=string,      --操作系统
  ["os_version"]=string, --系统版本
  ["kikoplay"]=string  --KikoPlay版本
}
```

显示脚本环境信息

- `xmlreader(str)`

`str`: string, xml内容

返回: kiko.xmlreader



创建一个流式XML读取器。KikoPlay提供了一个简单的XML读取器（封装Qt中的QXmlStreamReader），kiko.xmlreader提供如下方法：

```
adddata(str)      --继续添加xml数据
clear()           --清空数据
atend()           --读取是否到达末尾, true/false
readnext()        --读取下一个标签
startelem()       --当前是否是开始标签, true/false
endelem()         --当前是否是结束标签, true/false
name()            --返回当前标签名称
attr(attr_name)   --返回属性attr_name的值
hasattr(attr_name) --当前标签是否包含attr_name属性, true/false
elemtext()        --读取从当前开始标签到结束标签之间的文本并返回
error()           --返回错误信息, 没有错误返回nil
```

一个示例（来自danmu/iqiyi.lua）：

```
local xmlreader = kiko.xmlreader(danmuContent)
local curDate, curText, curTime, curColor, curUID = nil, nil, nil, nil, nil
while not xmlreader:atend() do
    if xmlreader:startelem() then
        if xmlreader:name()=="contentId" then
            curDate = string.sub(xmlreader:elemtext(), 1, 10)
        elseif xmlreader:name()=="content" then
            curText = xmlreader:elemtext()
        elseif xmlreader:name()=="showTime" then
            curTime = tonumber(xmlreader:elemtext()) * 1000
        elseif xmlreader:name()=="color" then
            curColor = tonumber(xmlreader:elemtext(), 16)
        elseif xmlreader:name()=="uid" then
            curUID = "[iqiyi]" .. xmlreader:elemtext()
        end
    elseif xmlreader:endelem() then
        if xmlreader:name()=="bulletInfo" then
            table.insert(danmuList, {
                ["text"]=curText,
                ["time"]=curTime,
                ["color"]=curColor,
                ["date"]=curDate,
                ["sender"]=curUID
            })
        end
    end
    xmlreader:readnext()
end
```

- `htmlparser(str)`

`str`: string, html内容

返回: kiko.htmlparser

创建一个流式HTML读取器。KikoPlay提供了一个简单的HTML读取器，可以顺序解析HTML标签。  
kiko.htmlparser提供如下方法：

```
adddata(str)           --继续添加html数据
seekto(pos)            --跳转到pos位置
atend()                --读取是否到达末尾, true/false
readnext()             --读取下一个标签
curpos()               --返回当前位置
readcontent()          --读取内容并返回, 直到遇到结束标签</
readuntil(lb, start)   --向前读取, 直到遇到lb标签, start=true表示希望遇到开始的lb
                        --标签, =false表示希望遇到结束的lb标签
start()                --当前标签是否是开始标签, true/false
curnode()              --返回当前标签名
curproperty(prop)      --读取当前标签的prop属性值
```

简单示例 (来自library/bangumi.lua) :

```
--tagContent 为部分网页内容
local parser = kiko.htmlparser(tagContent)
while not parser:atend() do
    --遇到开始的链接标签<a>
    if parser:curnode()=="a" and parser:start() then
        parser:readnext()
        table.insert(tags, parser:readcontent())
    end
    parser:readnext()
end
```

## 数据类型

### DanmuSource

```
{
    ["title"]=string,    --条目标题
    ["desc"]=string,     --条目描述, 可选
    ["duration"]=number, --时长(s), 可选
    ["delay"]=number,    --延迟(s), 可选
    ["data"]=string,     --可以存放一些条目相关的数据, 可选
    --当DanmuSource从KikoPlay传递到脚本时, 还会提供如下信息
    ["scriptId"]=string, --脚本ID
}
```

### DanmuComment

```

{
    ["text"] = string,      --弹幕文本
    ["time"] = number,      --视频时间(ms)
    ["color"] = number,     --颜色, 0xRRGGBB
    ["fontsize"] = number,  --字体大小, 0: 正常, 1: 小, 2: 大
    ["type"] = number,      --弹幕类型, 0: 滚动, 1: 顶部, 2: 底部
    ["date"] = string,      --发送时间, unix时间戳(s)转换为字符串
    ["sender"] = string     --发送用户
}

```

## AnimeLite

```

{
    ["name"] = string,      --动画名称, 注意KikoPlay通过name标识动画, name相同即为同一部
动画
    ["data"] = string,      --脚本可以自行存放一些数据
    ["extra"] = string,     --附加显示数据, 这个信息不会由KikoPlay传递到脚本, 仅用户向用
户展示
    ["scriptId"] = string   --脚本ID, 这里可以指定其他脚本的ID, 后续的获取详细信息等任务将
会由指定的其他脚本完成。为空则默认为当前脚本
}

```

## EpInfo

```

{
    ["name"] = string,      --分集名称
    ["index"] = number,     --分集编号 (索引)
    ["type"] = number       --分集类型
}

```

分集类型包括 EP, SP, OP, ED, Trailer, MAD, Other 七种, 分别用1-7表示, 默认情况下为1 (即EP, 本篇)

注意, KikoPlay通过分集类型和分集索引两个字段标识一个动画下的分集, 如果脚本返回的分集列表中有重复, KikoPlay会自动重新编号

## Character

```

{
    ["name"] = string,      --人物名称
    ["actor"] = string,     --演员名称
    ["link"] = string,      --人物资料页URL
    ["imgurl"] = string     --人物图片URL
}

```

## Anime

```
{
  ["name"]=string,      --动画名称
  ["data"]=string,      --脚本可以自行存放一些数据
  ["url"]=string,       --条目页面URL
  ["desc"]=string,      --描述
  ["airdate"]=string,   --放送日期，格式为yyyy-mm-dd
  ["epcount"]=number,   --分集数
  ["coverurl"]=string,  --封面图URL
  ["staff"]=string,     --staff
  ["crt"]=Array[Character], --人物
  ["scriptId"]=string   --脚本ID
}
```

脚本传递到KikoPlay时，staff的格式为：job1:staff1;job2:staff2;....

KikoPlay传递到脚本时，staff的格式为：

```
{
  ["job1"]=staff1,
  ["job2"]=staff2,
  .....
}
```

提供封面图URL和人物的图片URL后，KikoPlay会自动从相应URL下载图片。如果不需要KikoPlay下载，URL可以是本地文件路径，但此时KikoPlay不会在数据库中保存URL

## MatchResult

```
{
  ["success"]=bool,     --是否成功关联
  ["anime"]=AnimeLite,  --关联的动画信息
  ["ep"]=EpInfo         --关联的剧集信息
}
```

## LibraryMenu

```
{
  ["title"]=string,     --菜单标题
  ["id"]=string         --菜单ID
}
```

## ResourceItem

```
{
  ["title"]=string,    --标题
  ["size"]=string,     --大小, 建议是xxx(GB/MB/KB)形式
  ["time"]=string,     --发布时间
  ["magnet"]=string,   --磁力链接 (或者其他aria2支持的链接)
  ["url"]=string       --资源页面
}
```

## NetworkReply

```
{
  ["statusCode"]=number, --http状态
  ["hasError"]=bool,     --是否出错
  ["errInfo"]=string,    --错误信息
  ["content"]=string,    --响应内容
  ["headers"]={          --响应头
    [key]=value,
    ....
  }
}
```