Legado书源规则说明

> 作者: Celeter
>
> 更新时间: 2021-07-18
>
> 描述: 本说明是legado的规则说明，禁止其他未获得本人授权的第三方软件使用

概况

1、语法说明

- JSOUP之Default
  - 语法如下:

```
▲ @为分隔符,用来分隔获取规则
 ·每段规则可分为3段
 ·第一段是类型,如class,id,tag,text,children等, children获取所有子标签,不需要第二段和第三段,text可以根据文本内容获取
 ·第二段是名称,text. 第二段为文本内容的一部分
 ·第三段是位置,class,tag,id等会获取到多个,所以要加位置
 ·如不加位置会获取所有
 ·位置正着数从0开始,0是第一个,如为负数则是取倒着数的值,-1为倒数第一个,-2为倒数第二个
 ·!是排除,有些位置不符合需要排除用!,后面的序号用:隔开0是第1个,负数为倒数序号,-1最后一个,-2倒数第2个,依次
 ·获取列表的最前面加上负号- 可以使列表倒置,有些网站目录列表是倒的,前面加个负号可变为正的
 ·@的最后一段为获取内容,如text,textNodes,ownText,href,src,html,all等
 ·如需要正则替换在最后加上 ##正则表达式##替换内容, 替换内容为空时, 第二个##可以省略
 ·例:class.odd.0@tag.a.0@text||tag.dd.0@tag.h1@text##全文阅读
 ·例:class.odd.0@tag.a.0@text&&tag.dd.0@tag.h1@text##全文阅读

▲ 增加支持类似数组的写法
 ·格式如:[index,index, ...]或[!index,index, ...],其中[!`开头表示筛选方式为排除,index可以是单个索引,也可以是区间
 ·区间格式为[start:end]或[start:end:step], 其中start为0时可省略, end为-1时可省略
 ·索引(index)、区间两端(start和end)、区间间隔(step)都支持负数
 ·特殊用法tag.div[-1:0],可在任意地方让列表反向
▲ 允许索引作为@分段后每个部分的首规则, 此时相当于前面是children
 ·head@.1@text与head@[1]@text与head@children[1]@text等价
```

    - 标准规范与实现库 Package org.jsoup.select, CSS-like element selector
- JSOUP之CSS
    - 语法见https://blog.csdn.net/hou_angela/article/details/80519718
    - 必须以 `@css:` 开头
    - 标准规范与实现库 Package org.jsoup.select
    - 在线测试 Try jsoup online
- 注意：获取内容可用text,textNodes,ownText,html,all,href,src等
- 例子见最后的【书源一】的搜索页和正文页规则
- JSONPath
    - 语法见 JsonPath教程
    - 最好以 `@json:` 或 `$.` 开头，其他形式不可靠
    - 标准规范 goessner JSONPath - XPath for JSON
    - 实现库 json-path/JsonPath
    - 在线测试 Jayway JsonPath Evaluator
- 例子见最后的【书源三】的搜索页、目录页和正文页规则
- XPath
    - 语法见 XPath教程-入门、XPath教程-基础、XPath教程-高级、XPath库的说明
    - 必须以 `@XPath:` 或 `//` 开头
    - 标准规范 W3C XPATH 1.0
    - 实现库 zhegexiaohuozi/JsoupXpath
- 例子见最后的【书源二】的搜索页、详情页和正文页规则，以及目录页的下一页规则
- JavaScript
    - 可以在 `<js></js>`、`@js:`中使用，结果存在result中
    - `@js:`只能放在其他规则的最后使用
    - `<js></js>`可以在任意位置使用，还能作为其他规则的分隔符，例：`tag.li<js></js>//a`
    - 在搜索列表、发现列表和目录中使用可以用`+`开头，使用AllInOne规则
- 正则之AllInOne
    - 只能在搜索列表、发现列表、详情页预加载和目录列表中使用
    - 必须以 `:` 开头
    - 教程 Regex专题

      语法、方法、引擎
    - 例子见最后的【书源一】的目录页规则，最前面的`-`表示目录倒序，以及【书源二】的目录页规则
- 正则之OnlyOne
    - 形式 `##正则表达式##替换内容###`
    - 只能在搜索列表、发现列表、详情页预加载、目录列表之外使用
    - 例子见最后的【书源一】的详情页规则
- 注意点：该规则只能获取第一个匹配到的结果并进行替换
- 正则之净化
    - 形式 `##正则表达式##替换内容`
    - 只能跟在其他规则后面，独立使用相当于`all##正则表达式##替换内容`
    - 例子见最后的【书源一】的正文页规则

- 注意点：该规则为循环匹配替换
- 自定义三种连接符号
  - 符号：`&&`、`||`、`%%`
  - 只能在同种规则间使用，不包括js和正则
  - `&&`会合并所有取到的值，
  - `||`会以第一个取到值的为准
  - `%%`会依次取数，如三个列表，

    先取列表1的第一个，再取列表2的第一个，再取列表3的第一个，

    再取列表1的第二个，再取列表2的第二个...

2、Legado的特殊规则

- URL必知必会
  1. 请求头
     - 一般形式，如下所示

```
{
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120
Safari/537.36",
    "Accept-Language":"zh-CN,zh;q=0.9"
}
```

     - 复杂情况可使用js

```
(()=>{
    var ua = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120
Safari/537.36";
    var headers = {"User-Agent": ua};
    return JSON.stringify(headers);
})()
```

       ※其中，`ua`必须保证是`JavaScript`的`String`类型，`JSON.stringify()`才能将`header`转换为字符串。

     - 获取登录后的cookie

```
java.getCookie("http://baidu.com", null) => userid=1234;pwd=adbcd
java.getCookie("http://baidu.com", "userid") => 1234
```

     - 请求头中支持http代理,socks4 socks5代理设置

```
// socks5代理
{
  "proxy":"socks5://127.0.0.1:1080"
}
// http代理
{
  "proxy":"http://127.0.0.1:1080"
}
// 支持代理服务器验证
{
  "proxy":"socks5://127.0.0.1:1080@用户名@密码"
}
// 注意:这些请求头是无意义的,会被忽略掉
```

2. GET请求

- 一般形式如下，charset为utf-8时可省略，无特殊情况不需要请求头和webView，参数webView非空时采用webView加载

```
 https://www.baidu.com,{
    "charset": "gbk",
    "headers": {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120
Safari/537.36"},
    "webView": true
}
```

- 复杂情况可使用js

```
    var ua = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120
Safari/537.36";
    var headers = {"User-Agent": ua};
    var option = {
        "charset": "gbk",
        "headers": headers,
        "webView": true
    };
    "https://www.baidu.com," + JSON.stringify(option)
```

3. POST请求

- 一般形式如下，body是请求体，charset为utf-8时可省略，无特殊情况不需要请求头和webView，参数webView非空时采用webView加载

```
 https://www.baidu.com,{
    "charset": "gbk",
    "method": "POST",
    "body": "bid=10086",
    "headers": {"User-Agent":"Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120
Safari/537.36"},
    "webView": true
}
```

- 复杂情况可使用js

```
    var ua = "Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.120
Safari/537.36";
    var headers = {"User-Agent": ua};
    var body = "bid="+"10086";
    var option = {
        "charset": "gbk",
        "method": "POST",
        "body": String(body),
        "headers": headers,
        "webView": true
    };
    "https://www.baidu.com," + JSON.stringify(option)
```

※其中，`body`必须保证是`JavaScript`的`String`类型，变量是计算得到的尽量都用`String()`强转一下类型。

- 变量的put与get
  - `@put`与`@get`

    只能用于js以外的规则中，@put里使用JSONPath不需要加引号，其他规则需要加引号，

    例：@put:{bid:"//*[@bid-data]/@bid-data"}
  - `java.put`与`java.get`

    只能用于js中，在js中无法使用@get
- `{{}}`与`{}`规则
  - 在搜索URL与发现URL中的`{{}}`

    在{{}}里只能使用js
  - 在搜索URL与发现URL以外的`{{}}`

    可在`{{}}`中使用任意规则（正则除外？），默认为js，使用其他规则需要有明显的标志头，

    如：Default规则需要以`@@`开头，XPath需要以`@xpath:`或`//`开头，JSONPath需要以`@json:`或`$.`开头，CSS需要以`@css:`开头
  - `{}`规则

    留用了阅读2.0的规则，只能使用JSONPath，尽量避免使用
- 自定义js
  - 在js中调用java的常规方法：由于java这个关键字已经被使用，调用java开头的包名时需使用全局变量Packages 参考脚本之家

- 只调用某个public函数：，例：`io.legado.app.utils.htmlFormat(str)`、`org.jsoup.Jsoup.parse(str)`
- 直接引入java类，如下所示，引入了两个java包，java包的作用域是在`with`的范围内，其内使用java相关语法，最后在作用域外被js调用了作用域内的函数

```
var javaImport = new JavaImporter();
javaImport.importPackage(
    Packages.java.lang,
    Packages.java.security
);
with(javaImport){
    function strToMd5By32(str) {
        var reStr = null;
        var md5 = MessageDigest.getInstance("MD5");
        var bytes = md5.digest(String(str).getBytes());
        var stringBuffer = new StringBuilder();
        bytes.forEach(a=>{
            var bt = a & 0xff;
            if (bt < 16) {
                stringBuffer.append("0");
            }
            stringBuffer.append(Integer.toHexString(bt));
        });
        reStr = stringBuffer.toString();
        return reStr;
    }
}
strToMd5By32('123')
```

- 变量

```
baseUrl     // 变量-当前url,String
result      // 变量-上一步的结果
book        // 变量-书籍类,方法见 io.legado.app.data.entities.Book
cookie      // 变量-cookie操作类,方法见 io.legado.app.help.http.CookieStore
cache       // 变量-缓存操作类,方法见 io.legado.app.help.CacheManager
chapter     // 变量-当前目录类,方法见
io.legado.app.data.entities.BookChapter
title       // 变量-当前标题,String
src         // 内容,源码
```

- 下面是一些常用的函数，详见JsExtensions.kt

```
//访问网络，urlStr为url字符串，返回类型String?
java.ajax(urlStr: String)

//并发访问网络，urlList为url数组，返回StrResponse?的数组，若要获取body，需使
用.body()
java.ajaxAll(urlList: Array<String>): Array<StrResponse?>

//访问网络，urlStr为url字符串，返回Response<String>，已废弃
```

```
java.connect(urlStr: String)

//文件下载，content为十六进制字符串，url用于生成文件名，返回String文件相对路径
java.downloadFile(content: String, url: String)

//实现重定向拦截，返回[Connection.Response]
(https://jsoup.org/apidocs/org/jsoup/Connection.Response.html)
java.get(url: String, headers: Map<String, String>)
java.post(urlStr: String, body: String, headers: Map<String, String>)

//实现cookie读取，返回String
java.getCookie(tag: String, key: String?)

//base64解码，返回类型String
java.base64Decode(str: String)
java.base64Decode(str: String, flags: Int)

//base64解码，返回类型ByteArray?
java.base64DecodeToByteArray(str: String?)
java.base64DecodeToByteArray(str: String?, flags: Int)

//base64编码，返回类型String?
java.base64Encode(str: String)
java.base64Encode(str: String, flags: Int)

//md5编码，返回类型String?
java.md5Encode(str: String)
java.md5Encode16(str: String)

//格式化时间戳，返回类型String
java.timeFormat(timestamp: Long)
java.timeFormat(time: String)

//utf8编码转gbk编码，返回String
java.utf8ToGbk(str: String)

//实现字符串的URI编码，enc为编码格式，返回String
java.encodeURI(str: String) //默认enc="UTF-8"
java.encodeURI(str: String, enc: String)

//html格式化，返回String
java.htmlFormat(str: String)

//获取本地文件,path为文件的相对路径,返回File
java.getFile(path: String)

//读取本地文件，返回ByteArray?
java.readFile(path: String)

//读取本地文本文件，charsetName为编码格式
java.readTxtFile(path: String)  //自动识别charsetName不一定准，乱码时请手动指
定
java.readTxtFile(path: String, charsetName: String)

//删除文件或文件夹
```

```
deleteFile(path: String)

//zip文件解压，zipPath为压缩文件路径，返回String解压相对路径，会删除原文件只保留解压后的文件
java.unzipFile(zipPath: String)

//文件夹内所有文本文件读取，返回内容String，会删除文件夹
java.getTxtInFolder(unzipPath: String)

//获取网络zip文件中的数据，url为zip文件链接，path为所需获取文件在zip内的路径，返回文件数据String
java.getZipStringContent(url: String, path: String)

//获取网络zip文件中的数据，url为zip文件链接，path为所需获取文件在zip内的路径，返回文件数据ByteArray?
java.getZipByteArrayContent(url: String, path: String)

//解析字体,返回字体解析类QueryTTF?
java.queryBase64TTF(base64: String?)

//str支持url,本地文件,base64,自动判断,自动缓存,返回字体解析类QueryTTF?
java.queryTTF(str: String?)

//text为包含错误字体的内容，font1为错误的字体，font2为正确的字体，返回字体对应的字
java.replaceFont(text: String,font1: QueryTTF?,font2: QueryTTF?)

//输出调试日志
java.log(msg: String)

//AES解码为ByteArray?,str为传入的AES加密数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesDecodeToByteArray(str: String, key: String, transformation: String, iv: String)

//AES解码为String?,str为传入的AES加密数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesDecodeToString(str: String, key: String, transformation: String, iv: String)

//已经base64的AES解码为ByteArray?,str为Base64编码数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesBase64DecodeToByteArray(str: String, key: String, transformation: String, iv: String)

//已经base64的AES解码为String?,str为Base64编码数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesBase64DecodeToString(str: String, key: String, transformation: String, iv: String)

//加密aes为ByteArray?，data为传入的原始数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesEncodeToByteArray(data: String, key: String, transformation: String, iv: String)
```

```
//加密aes为String?，data为传入的原始数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesEncodeToString(data: String, key: String, transformation: String, iv: String)

//加密aes后Base64化的ByteArray?，data为传入的原始数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesEncodeToBase64ByteArray(data: String, key: String, transformation: String, iv: String)

//加密aes后Base64化的String?，data为传入的原始数据，key为AES解密key，transformation为AES加密方式，iv为ECB模式的偏移向量
java.aesEncodeToBase64String(data: String, key: String, transformation: String, iv: String)

/***************以下部分方法由于JAVA不支持参数默认值，调用时不能省略***************/
//设置需解析的内容content和baseUrl，返回类型AnalyzeRule
java.setContent(content: Any?, baseUrl: String? = this.baseUrl)

//输入规则rule和URL标志isUrl获取文本列表，返回类型List<String>?
java.getStringList(rule: String, isUrl: Boolean = false)

//输入规则rule和URL标志isUrl获取文本，返回类型String
java.getString(ruleStr: String?, isUrl: Boolean = false)

//输入规则ruleStr获取节点列表，返回类型List<Any>
java.getElements(ruleStr: String)
```

- url添加js参数，解析url时执行，可在访问url时处理url，例：

```
https://www.baidu.com,{"js":"java.headerMap.put('xxx', 'yyy')"}
https://www.baidu.com,{"js":"java.url=java.url+'yyyy'"}
```

- url全部参数，详见AnalyzeUrl.kt

```
data class UrlOption(
    val method: String?,
    val charset: String?,
    val webView: Any?,
    val headers: Any?,
    val body: Any?,
    val type: String?,
    val js: String?,
    val retry: Int = 0  //重试次数
)
```

3、书源之「基本」

- 书源URL(bookSourceUrl)

    - 必填
    - 唯一标识，不可重复
    - 与其他源相同会覆盖
- 书源名称(bookSourceName)
    - 必填
    - 名字可重复
- 书源分组(bookSourceGroup)
    - 可不填
    - 用于整理源
- 登录URL(loginUrl)
    - 根据需求，随机应变
    - 用于登录个人账户
- 书籍URL正则(bookUrlPattern)
    - 可不填
    - 添加网址时，用于识别书源
    - ```
      例:https?://www.piaotian.com/bookinfo/.*
      ```
- 请求头(header)
    - 根据需求，随机应变
    - 访问网址时使用

4、书源之「搜索」

- 搜索地址(url)
    - `key`为关键字标识，通常形态为`{{key}}`，运行时会替换为搜索关键字

      也可以对key进行加密等操作，如：`{{java.base64Encode(key)}}`
    - `page`为关键字标识，通常形态为`{{page}}`，page的初值为1也可以对page进行计算，

      如：`{{(page-1)*20}}`，有时会遇到第一页没有页数的情况，有两种方法：

      ① `{{page - 1 == 0 ? "": page}}`

      ② `<,{{page}}>`
    - 支持相对URL
- 书籍列表规则(bookList)
- 书名规则(name)
- 作者规则(author)
- 分类规则(kind)
- 字数规则(wordCount)
- 最新章节规则(lastChapter)
- 简介规则(intro)
- 封面规则(coverUrl)
- 详情页url规则(bookUrl)
      ```

5、书源之「发现」

- 发现地址规则(url)
  - `page`为关键字标识，通常形态为`{{page}}`，page的初值为1，也可以对page进行计算，如：`{{(page-1)*20}}`，有时会遇到第一页没有页数的情况，有两种方法：

    ① `{{page - 1 == 0 ? "": page}}`

    ② `<,{{page}}>`
  - 格式一，如：`名称::http://www。baidu.com`，发现URL可使用`&&`或换行符`\n`隔开
  - 格式二，有5个样式属性(layout_flexGrow、layout_flexShrink、layout_alignSelf、layout_flexBasisPercent、layout_wrapBefore)需要了解，详情见简书，写法如：

```
 [
    {
        "title": "今日限免",
        "url": "https://app-cdn.jjwxc.net/bookstore/getFullPage?
channel=novelfree",
        "style": {
            "layout_flexGrow": 1
        }
    },
    {
        "title": "频道金榜",
        "url": "http://app-cdn.jjwxc.net/bookstore/getFullPage?
channelBody=%7B%229%22%3A%7B%22offset%22%3A%22<,{{(page-
1)*25}}>%22%2C%22limit%22%3A%2225%22%7D%7D&versionCode=148",
        "style": {
            "layout_flexGrow": 0,
            "layout_flexShrink": 1,
            "layout_alignSelf": "auto",
            "layout_flexBasisPercent": -1,
            "layout_wrapBefore": true
        }
    },
    {
        "title": "幻想未来",
        "url": "http://app-cdn.jjwxc.net/bookstore/getFullPage?
channelBody=%7B%2222000023%22%3A%7B%22offset%22%3A%22<,{{(page-
1)*25}}>%22%2C%22limit%22%3A%2225%22%7D%7D&versionCode=148"
    }
]
```

  - 支持相对URL
- 书籍列表规则(bookList)
- 书名规则(name)
- 作者规则(author)
- 分类规则(kind)
- 字数规则(wordCount)
- 最新章节规则(lastChapter)

- 简介规则(intro)
- 封面规则(coverUrl)
- 详情页url规则(bookUrl)

6、书源之「详情」

- 预处理规则(bookInfoInit)
  - 只能使用正则之AllInOne或者js
  - 正则之AllInOne必须以`:`开头
  - js的返回值需要是json对象，例：

```
(function(){
    return {
        a:"圣墟",
        b:"辰东",
        c:"玄幻",
        d:"200万字",
        e:"第两千章 辰东肾虚",
        f:"在破败中崛起，在寂灭中复苏。沧海成尘，雷电枯竭...",
        g:"https://bookcover.yuewen.com/qdbimg/349573/1004608738/300",
        h:"https://m.qidian.com/book/1004608738"
    };
})()
```

    此时，书名规则填`a`，作者规则填`b`，分类规则填`c`，字数规则填`d`，最新章节规则填`e`，简介规则`f`，封面规则填`g`，目录URL规则填`h`

- 书名规则(name)
- 作者规则(author)
- 分类规则(kind)
- 字数规则(wordCount)
- 最新章节规则(lastChapter)
- 简介规则(intro)
- 封面规则(coverUrl)
- 目录URL规则(tocUrl)
  - 只支持单个url
- 允许修改书名作者(canReName)
  - 规则不为空且详情页书名不为空，使用详情页中的作者。否则，使用搜索页中的书名
  - 规则不为空且详情页作者不为空，使用详情页中的作者。否则，使用搜索页中的作者

7、书源之「目录」

- 目录列表规则(chapterList)
  - 首字符使用负号(`-`)可使列表反序
- 章节名称规则(ruleChapterName)
- 章节URL规则(chapterUrl)
- VIP标识(isVip)

    - 当结果为`null` `false` `0` `""`时为非VIP
- 章节信息(ChapterInfo)
    - 可调用java.timeFormat(timestamp: Long)将时间戳转为yyyy/MM/dd HH:mm格式的时间
- 目录下一页规则(nextTocUrl)
    - 支持单个url
    - 支持url数组
    - js中返回`[]`或`null`或`""`时停止加载下一页

8、书源之「正文」

- 正文规则(content)
    - 正文图片链接支持修改headers

```
let options = {
"headers": {"User-Agent":
"xxxx","Referrer":baseUrl,"Cookie":"aaa=vbbb;"}
};
'<img src="'+src+","+JSON.stringify(options)+'">'
```

    - book对象的可用属性
        - 使用方法: 在js中或`{{}}`中使用`book.属性`的方式即可获取，如在正文内容后加上`##{{book.name+"正文卷"+title}}`可以净化 书名+正文卷+章节名称（如：我是大明星正文卷第二章我爸是豪门总裁） 这一类的字符

```
bookUrl             // 详情页Url(本地书源存储完整文件路径)
tocUrl              // 目录页Url (toc=table of Contents)
origin              // 书源URL(默认BookType.local)
originName          // 书源名称 or 本地书籍文件名
name                // 书籍名称(书源获取)
author              // 作者名称(书源获取)
kind                // 分类信息(书源获取)
customTag           // 分类信息(用户修改)
coverUrl            // 封面Url(书源获取)
customCoverUrl      // 封面Url(用户修改)
intro               // 简介内容(书源获取)
customIntro         // 简介内容(用户修改)
charset             // 自定义字符集名称(仅适用于本地书籍)
type                // 0:text 1:audio
group               // 自定义分组索引号
latestChapterTitle  // 最新章节标题
latestChapterTime   // 最新章节标题更新时间
lastCheckTime       // 最近一次更新书籍信息的时间
lastCheckCount      // 最近一次发现新章节的数量
totalChapterNum     // 书籍目录总数
durChapterTitle     // 当前章节名称
durChapterIndex     // 当前章节索引
durChapterPos       // 当前阅读的进度(首行字符的索引位置)
durChapterTime      // 最近一次阅读书籍的时间(打开正文的时间)
canUpdate           // 刷新书架时更新书籍信息
order               // 手动排序
originOrder         //书源排序
variable            // 自定义书籍变量信息(用于书源规则检索书籍信息)
```

- chapter对象的可用属性
  - 使用方法: 在js中或`{{}}`中使用`chapter.属性`的方式即可获取，如在正文内容后加上`##{{chapter.title+chapter.index}}`可以净化 章节标题+序号(如 第二章 天仙下凡2)这一类的字符

```
url          // 章节地址
title        // 章节标题
baseUrl      // 用来拼接相对url
bookUrl      // 书籍地址
index        // 章节序号
resourceUrl  // 音频真实URL
tag          // 章节信息
start        // 章节起始位置
end          // 章节终止位置
variable     // 变量
```

- 如下示例，在详情页(目录页)和正文使用webView加载，例:

```
{
  "bookSourceComment": "",
  "bookSourceGroup": "  有声",
  "bookSourceName": "猫耳FM",
  "bookSourceType": 1,
  "bookSourceUrl": "https://www.missevan.com",
  "customOrder": 0,
  "enabled": false,
  "enabledExplore": true,
  "lastUpdateTime": 0,
  "ruleBookInfo": {},
  "ruleContent": {
    "content": "https://static.missevan.com/{}",
    "sourceRegex": "",
    "webJs": ""
  },
  "ruleExplore": {},
"ruleSearch": {
    "author": "author",
    "bookList": "$.info.Datas",
    "bookUrl": "https://www.missevan.com/mdrama/drama/{{$.id}},
{\"webView\":true}",
    "coverUrl": "cover ",
    "intro": "abstract",
    "kind": "{{$.type_name}},{{$.catalog_name}}",
    "lastChapter": "newest ",
    "name": "name",
    "wordCount": "catalog_name "
  },
  "ruleToc": {
    "chapterList": "@css:.scroll-list.btn-groups>a",
    "chapterName": "text",
    "chapterUrl": "href##$##,{\"webView\":true}"
  },
```

```
  "searchUrl": "https://www.missevan.com/dramaapi/search?s=
{{key}}&page=1",
  "weight": 0
}
```

- 正文下一页URL规则(nextContentUrl)
  - 支持单个url
  - 支持url数组
- WebViewJs(webJs)
  - 用于模拟鼠标点击等操作，必须有返回值（不为空，表示webjs执行成功，否则会无限循环），返回值会用于资源正则或正文中
  - 举个栗子，在webJs中执行了`getDecode()`，使正文部分解密：

```
{
  "bookSourceGroup": "阅读3.0书源合集",
  "bookSourceName": "🔥斋书苑",
  "bookSourceType": 0,
  "bookSourceUrl": "https://www.zhaishuyuan.com",
  "bookUrlPattern": "",
  "customOrder": 11,
  "enabled": false,
  "enabledExplore": false,
  "exploreUrl": "男生书库::/shuku/0_1_0_0_0_{{page}}_0_0\n男频连
载::/shuku/0_2_0_0_0_{{page}}_0_0",
  "lastUpdateTime": 0,
  "loginUrl": "",
  "ruleBookInfo": {
    "author": "@css:[property=og:novel:author]@content",
    "coverUrl": "@css:[property=og:image]@content",
    "intro": "@css:#bookintro@html",
    "kind": "@css:[property=og:novel:category]@content",
    "lastChapter": "@css:[property=og:novel:latest_chapter_name]@content",
    "name": "@css:[property=og:novel:book_name]@content",
    "tocUrl": "@css:[property=og:novel:read_url]@content",
    "wordCount": "@css:.count li:eq(3)>span@text"
  },
  "ruleContent": {
    "content": "all",
    "nextContentUrl": "",
    "webJs": "getDecode();$('#content').html();"
  },
  "ruleExplore": {
    "author": "//li[4]/a/text()",
    "bookList": "//ul[count(../ul)>10]",
    "bookUrl": "//li[3]/a/@href",
    "coverUrl":
"##/book/(\\d+)##https://img.zhaishuyuan.com/bookpic/s$1.jpg###",
    "intro": "//li[6]/text()",
    "kind": "//li[2]/text()##\\[|\\]",
    "lastChapter": "//span/a/text()",
    "name": "//li[3]/a/text()",
    "wordCount": "//li[5]/text()"
```

```
    },
    "ruleSearch": {
      "author": "//dd[2]/span[1]/text()",
      "bookList": "//*[@id=\"sitembox\"]/dl",
      "bookUrl": "//dt/a/@href",
      "coverUrl": "//img/@_src",
      "intro": "//dd[3]/html()",
      "kind": "//dd[2]/span[3]/text()",
      "lastChapter": "//dd[4]/a/text()",
      "name": "//h3/a//text()",
      "wordCount": "//dd[2]/span[4]/text()"
    },
    "ruleToc": {
      "chapterList": ":href=\"(/chapter/[^\"]*)\"[^>]*>([^<]*)</a>([^<]*)",
      "chapterName": "$2",
      "chapterUrl": "$1,{\"webView\":true}",
      "nextTocUrl": "//strong/following-sibling::a/@href",
      "updateTime": "$3"
    },
    "searchUrl": "/search/,{\n  \"charset\": \"gbk\",\n  \"method\":
\"POST\",\n  \"body\": \"page={{page}}&key={{key}}\"\n}",
    "weight": 0
  }
```

- 资源正则(sourceRegex)
  - 用于嗅探
  - 一般情况下的无脑教程如下
    - 章节链接后面加 `,{"webView":true}`，不要洒敷敷的写成 `tag.a@href,{"webView":true}` 或 `$.link,{"webView":true}`，正确写法如:`tag.a@href##$##{"webView":true},{{@@tag.a@href}},{"webView":true},tag.a@href@js:result+',{"webView":true}'` 等
    - 在有嗅探功能的浏览器（如：via、x浏览器等）中，输入章节链接。注意 `千万别带,{"webView":true}`
    - 媒体开始播放后使用浏览器的嗅探功能，查看资源的链接
    - 在资源正则里填写资源链接的正则，一般写 `.*\.(mp3|mp4).*` 这个就可以了
    - 最后在正文填写 `<js>result</js>`
  - 如下示例，在正文嗅探mp3和mp4的资源：

```
{
  "bookSourceComment": "",
  "bookSourceGroup": "  有声",
  "bookSourceName": "猫耳FM",
  "bookSourceType": 1,
  "bookSourceUrl": "https://www.missevan.com",
  "customOrder": 0,
  "enabled": false,
  "enabledExplore": true,
  "lastUpdateTime": 0,
  "ruleBookInfo": {},
  "ruleContent": {
    "content": "https://static.missevan.com/{}",
    "sourceRegex": "",
    "webJs": ""
```

```
  },
  "ruleExplore": {},
"ruleSearch": {
    "author": "author",
    "bookList": "$.info.Datas",
    "bookUrl": "https://www.missevan.com/mdrama/drama/{{$.id}},
{\"webView\":true}",
    "coverUrl": "cover ",
    "intro": "abstract",
    "kind": "{{$.type_name}},{{$.catalog_name}}",
    "lastChapter": "newest ",
    "name": "name",
    "wordCount": "catalog_name "
  },
  "ruleToc": {
    "chapterList": "@css:.scroll-list.btn-groups>a",
    "chapterName": "text",
    "chapterUrl": "href##$##,{\"webView\":true}"
  },
  "searchUrl": "https://www.missevan.com/dramaapi/search?s=
{{key}}&page=1",
  "weight": 0
}
```

9、补充说明

- 显示js的报错信息

```
(function(result){
    try{
          // 处理result
          // ...
          // 当返回结果为字符串时
          return result;
          // 当返回结果为列表时
          return list;
    }
    catch(e){
          // 当返回结果为字符串时
          return ""+e;
          // 当返回结果为列表时
          return [""+e];  //列表对应名称处填<js>""+result</js>查看
    }
})(result);
```

- 请善用调试功能
  - 调试搜索

    输入关键字，如：`系统`
  - 调试发现

    输入发现URL，如：`月票榜::https://www.qidian.com/rank/yuepiao?page={{page}}`

    - 调试详情页

      输入详情页URL，如：`https://m.qidian.com/book/1015609210`

    - 调试目录页

      输入目录页URL，如：`++https://www.zhaishuyuan.com/read/30394`

    - 调试正文页

      输入正文页URL，如：`--https://www.zhaishuyuan.com/chapter/30394/20940996`

- 无脑`{"webView":true}`很方便
- 特别注意用`JSON.stringify()`方法时，需要保证JSON对象的value都是`JavaScript`的`String`类型(在阅读3.0中)
- 附：
    - 书源一


```
{
  "bookSourceComment": "",
  "bookSourceGroup": "CSS; 正则",
  "bookSourceName": "小说2016",
  "bookSourceType": 0,
  "bookSourceUrl": "https://www.xiaoshuo2016.com",
  "bookUrlPattern": "",
  "customOrder": 0,
  "enabled": true,
  "enabledExplore": false,
  "exploreUrl": "",
  "lastUpdateTime": 0,
  "loginUrl": "",
  "ruleBookInfo": {
    "author": "##:author\"[^\"]+\"([^\"]*)##$1###",
    "coverUrl": "##og:image\"[^\"]+\"([^\"]*)##$1###",
    "intro": "##:description\"[^\"]+\"([\\w\\W]*?)\"/##$1###",
    "kind": "##:category\"[^\"]+\"([^\"]*)##$1###",
    "lastChapter": "##_chapter_name\"[^\"]+\"([^\"]*)##$1###",
    "name": "##:book_name\"[^\"]+\"([^\"]*)##$1###",
    "tocUrl": ""
},
  "ruleContent": {
  "content": "@css:.articleDiv p@textNodes##搜索.*手机访问|一秒记住.*|.*阅读下载|<!\\[CDATA\\[|\\]\\]>",
    "nextContentUrl": ""
  },
  "ruleExplore": {},
  "ruleSearch": {
    "author": "@css:p:eq(2)>a@text",
    "bookList": "@css:li.clearfix",
    "bookUrl": "@css:.name>a@href",
    "coverUrl": "@css:img@src",
    "intro": "@css:.note.clearfix p@text",
    "kind": "@css:.note_text,p:eq(4)@text",
    "lastChapter": "@css:p:eq(3)@text",
    "name": "@css:.name@text"
  },
  "ruleToc": {
```

```
    "chapterList": "-:<li><a[^\"]+\"([^\"]*)\">([^<]*)",
    "chapterName": "$2",
    "chapterUrl": "$1",
    "nextTocUrl": ""
  },
  "searchUrl": "/modules/article/search.php?searchkey=
{{key}}&submit=&page={{page}},{\n  \"charset\": \"gbk\"\n}",
  "weight": 0
}
```


- 书源二


```
{
  "bookSourceComment": "",
  "bookSourceGroup": "XPath; 正则",
  "bookSourceName": "采墨阁手机版",
  "bookSourceType": 0,
  "bookSourceUrl": "https://m.caimoge.com",
  "bookUrlPattern": "",
  "customOrder": 0,
  "enabled": true,
  "enabledExplore": false,
  "exploreUrl": "",
  "lastUpdateTime": 0,
  "loginUrl": "",
  "ruleBookInfo": {
    "author": "//*[@property=\"og:novel:author\"]/@content",
    "coverUrl": "//*[@property=\"og:image\"]/@content",
    "intro": "//*[@property=\"og:description\"]/@content",
    "kind": "//*[@property=\"og:novel:category\"]/@content",
    "lastChapter": "//*[@id=\"newlist\"]//li[1]/a/text()",
    "name": "//*[@property=\"og:novel:book_name\"]/@content",
    "tocUrl": "//a[text()=\"阅读\"]/@href"
  },
  "ruleContent": {
    "content": "//*[@id=\"content\"]",
    "nextContentUrl": ""
  },
  "ruleExplore": {},
  "ruleSearch": {
    "author": "//dd[2]/text()",
    "bookList": "//*[@id=\"sitebox\"]/dl",
    "bookUrl": "//dt/a/@href",
    "coverUrl": "//img/@src",
    "kind": "//dd[2]/span/text()",
    "lastChapter": "",
    "name": "//h3/a/text()"
  },
  "ruleToc": {
    "chapterList": ":href=\"(/read[^\"]*html)\">([^<]*)",
    "chapterName": "$2",
    "chapterUrl": "$1",
```

```
    "nextTocUrl": "//*[@id=\"pagelist\"]/*[position()>1]/@value"
  },
  "searchUrl": "/search.html,{\n  \"method\": \"POST\",\n  \"body\":
\"searchkey={{key}}\"\n}",
  "weight": 0
}
```


- 书源三


```
{
  "bookSourceComment": "",
  "bookSourceGroup": "JSon",
  "bookSourceName": "猎鹰小说网",
  "bookSourceType": 0,
  "bookSourceUrl": "http://api.book.lieying.cn",
  "customOrder": 0,
  "enabled": true,
  "enabledExplore": false,
  "header": "{\n  \"User-Agent\": \"Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106
Safari/537.36\"\n}",
  "lastUpdateTime": 0,
  "ruleBookInfo": {},
  "ruleContent": {
    "content": "$.chapter.body"
  },
  "ruleExplore": {},
  "ruleSearch": {
    "author": "$.author",
    "bookList": "$..books[*]",
    "bookUrl": "/Book/getChapterListByBookId?bookId={$._id}",
    "coverUrl": "$.cover",
    "intro": "$.shortIntro",
    "kind": "$.minorCate",
    "lastChapter": "$.lastChapter",
    "name": "$.title"
  },
  "ruleToc": {
    "chapterList": "$.chapterInfo.chapters.[*]",
    "chapterName": "$.title",
    "chapterUrl": "$.link"
  },
  "searchUrl": "/Book/search?query={{key}}&start={{(page-
1)*20}}&limit=40&device_type=android&app_version=165",
  "weight": 0
}
```