

# SiteD 插件开发

(指南)

v34-66

插件是一个标签 + 脚本的 xml 文件

插件可接入现有的数据接口；或解析网站数据并转换为预定格式；等任何数据格式

建议与插件示例对照阅读

开发好后，上传到插件中心即可安装调试

by noear

# 目录

一、入门（开发第一个插件） .....	3
二、关于插件.....	6
1. 插件的各种叫法.....	6
2. 独立插件的安装方式（更多插件可百度：SITE D 插件） .....	7
三、格式概要.....	7
1. 插件描述.....	7
2. 数据节点.....	8
3. 解析脚本.....	8
4. 注意事项.....	8
四、格式及引擎能力详解.....	8
1. 插件节点类型简介.....	8
2. 插件格式说明（?表示随意） .....	9
3. 全局变量 SITE D 结构说明（可用于引擎版本的向下兼容处理） .....	14
4. 引擎能力说明.....	15
五、插件数据节点说明.....	16
1. 节点及其数据说明.....	16
2. 节点跳转示意图（基于 EXPR 和 DTYPE） .....	17
3.1. 输出格式[DTYPE=1]::漫画（或有目录图片） .....	17
3.2. 输出格式[DTYPE=2]::轻小说.....	18
3.3. 输出格式[DTYPE=3]::动画（或有目录视频） .....	19
3.4. 输出格式[DTYPE=4]::图片（或无目录的漫画） .....	20
3.5. 输出格式[DTYPE=5]::商品（周边、漫画书之类） .....	20
3.6. 输出格式[DTYPE=6]::资讯.....	21
3.7. 输出格式[DTYPE=7]::视频（或无目录的动画） .....	22
3.8. 输出格式[DTYPE=8]::查询工具.....	22
4. ABOUT 节点下的 ITEM 说明.....	23
5. BOOK[DTYPE=8]节点下的 ITEM 说明.....	24
六、插件节点数据的获取程序.....	24
1. BUILDURL, PARSEURL, CALL::URL, PARSE 的内部逻辑（伪代码） .....	24
2. CALL::的补充说明.....	25
七、ADD 节点的示例说明.....	25
示例 1：直接解析父节点的数据.....	25
示例 2：借父节点 URL 构建目标 URL，并解析.....	26
示例 3：为父节点增加数据源.....	26
八、插件更新接口规范.....	26
附录.....	26
* 调试说明.....	26
* 相关约定.....	27
* 联系方式.....	27

# 一、入门（开发第一个插件）

把下面的源码复制到记事本，保存为：xxx.sited（注意后缀名）；传到测试设备上并使用多多猫app打开（可参考：二.3独立插件的安装方式）。

```
<?xml version="1.0" encoding="utf-8"?>
<sited ver="1" debug="1" engine="27" schema="1">
  <meta>
    <ua></ua>
    <title>东方二次元</title>
    <intro>动漫产业联盟 | Oriental ACG，正版授权，IP运营</intro>
    <author>东方二次元</author>
    <url>http://comic.oacg.cn</url>
    <expr>\.oacg\.cn</expr>
    <encode>utf-8</encode>
  </meta>
  <main dtype="1" durl="http://comic.oacg.cn">
    <home>
      <hots cache="1d" title="首页" method="get" parse="hots_parse" url="http://comic.oacg.cn" />
      <tags title="标签">
        <item title="恋爱"

url="http://comic.oacg.cn/index.php?m=Index&a=type_theme&theme_id=1&pageidx=@page"/>
      </tags>
    </home>
    <search cache="1d" method="get" parse="search_parse"
      url="http://comic.oacg.cn/index.php?m=Index&a=searchlistdata&tag=@key"/>
    <tag cache="0" method="get" parse="tag_parse" addPage="-1"
      header="Accept=/*;X-Requested-With=XMLHttpRequest" />

    <book cache="1d" method="get" parse="book_parse" expr="manga-" />
    <section cache="1" method="get" parse="section_parse" header="cookie;referer" />
  </main>
  <script>
    <require>
      <item url="http://sited.noear.org/addin/js/cheerio.js" lib="cheerio" />
    </require>
    <code>
      <![CDATA[

function urla(u) {
  var host = "http://comic.oacg.cn";
  if (u.indexOf("http")<0) {
    if (u.substr(0, 1) != '/')

```

```

        u = host + '/' + u;
    else
        u = host + u;
    }
    return encodeURI(u);
}

```

```

function hots_parse(url,html) {
    var $ = cheerio.load(html);
    var list = [];

    $('.in-sign-list li').each(function(){
        var slf = $(this);
        var img = slf.find('img');

        var bm = {};
        bm.name = img.parent().attr('title');
        bm.url = urla(img.parent().attr('href'));
        bm.logo = urla(img.attr('src'));

        list.push(bm);
    });
    return JSON.stringify(list);
}

```

```

function search_parse(url,html) {
    var jsonObj = JSON.parse(html);
    var jList = jsonObj.comic_arr;
    var list = [];

    jList.forEach(function(item){
        var bm = {};
        bm.name = item.comic_name;
        bm.url = 'http://comic.oacg.cn/index.php?m=Index&a=comicinfo&comic_id=' + item.id;
        bm.logo = 'http://le.cdn.pc.comicq.cn' + item.comic_pic_240_320;

        bm.author = item.painter_user_nickname;
        bm.newSection = item.comic_last_orderidx;
        bm.updateTime = "";
        bm.status = "";
        list.push(bm);
    });

    return JSON.stringify(list);
}

```

```

}

function tag_parse(url,html){
    var jsonObj = JSON.parse(html);
    var jList = jsonObj.comic_arr;
    var list = [];

    jList.forEach(function(item){
        var bm = {};
        bm.name = item.comic_name;
        bm.url = 'http://comic.oacg.cn/index.php?m=Index&a=comicinfo&comic_id=' + item.id;
        bm.logo = 'http://le.cdn.pc.comicq.cn' + item.comic_pic_240_320;

        bm.author = item.painter_user_nickname;
        bm.newSection = item.comic_last_orderidx;
        bm.updateTime = "";
        bm.status = "";
        list.push(bm);
    });

    return JSON.stringify(list);
}

```

```

function book_parse(url, html) {
    var $ = cheerio.load(html);
    var data={};

    data.name = $('h2.works-intro-title').text().trim();
    data.author = $('p.works-intro-digi em').eq(0).text().trim();
    data.logo = $('p.works-cover img').attr('src');
    data.intro = $('p.works-intro-short').text();
    data.updateTime = "";
    data.sections = [];

    $('p.works-chapter-list span>a').each(function(){
        var al = $(this);

        if(al.attr('href')){
            var sm = {url:url+al.attr('href'),name:al.text()};
            data.sections.splice(0, 0, sm);
        }
    });

    return JSON.stringify(data);
}

```

```

}

function section_parse(url,html) {
    var $ = cheerio.load(html);
    var list = [];
    $('#page_area img').each(function(){
        var al = $(this);
        list.push(al.attr('data-original'));
    });

    return JSON.stringify(list);
}

]]>
</code>
</script>
</sited>

```

## 二、关于插件

### 1.插件的各种叫法

#### 1.1.插件按托管分

##### a.托管插件

发布在插件中心，任何人可下载。通过插件中心可以发现、下载和更新。

##### b.独立插件

发布在第三方服务器，不能通过插件中心发现。插件网址生成 [专属二维码]，通过扫描安装；或者通过输入框打开网址安装。

#### 1.2.插件按受众分

##### a.开放插件

插件里的内容，可以分享，可以进入垃圾街，可以查看源网页，可以 Pin 到桌面。

##### b.私密插件

标识 private="1"的插件。不能分享二维码，不能进入垃圾街，不能被别人看到收藏。

## 1.3.插件按数据分

- 1.漫画插件（或有目录的图片）
- 2.轻小说或小说插件
- 3.动画插件（或有目录的视频）
- 4.图片插件（或无目录的漫画）
- 5.商品插件（周边、漫画书等可购买的商品）
- 6.资讯插件（或无目录的小说）
- 7.视频插件（或无目录的视频）
- 8.工具插件

## 2.独立插件的安装方式（更多插件可百度：SiteD 插件）

### a.使用手机浏览器安装

在网络上找到 sited://开头的磁链接，浏览器会启动多多猫 app 并安装插件。

### b.使用多多猫 app 首页输入框安装

输入 sited://开头的磁链接地址，并回车

输入.sited 或.sited.xml 的文件网址，并回车（.sited.xml 为插件源码；.sited 为加密插件）

### c.使用文件管理器安装

从网络下载.sited 文件（如果是压缩包，需要先解压），在文件管理器打开，打开时选择多多猫 app 打开（不同的文件管理器操作各有不同，个人推荐 [腾讯的文件管理器]）

### d.使用 QQ 安装

从 QQ 群共享下载.sited 文件（如果是压缩包，需要先解压），点 [用其他应用打开]，会启动多多猫 app 并打开。

## 三、格式概要

插件的开发分三部分内容

### 1.插件描述

插件采用标准 xml 作为文件规范

分三部分：头部节点、主体节点、脚本节点

头部节点：用于描述插件的相关信息（开发者、对应网站、标题、登录等）

主体节点：用于描述数据和样式

脚本节点：用于描述处理代码（支持 javascript 脚本）

## 2.数据节点

用于配置具体的数据连接网址

## 3.解析脚本

用于解析获取到的数据，并转换为目标数据

\*如果不需要解析则设置："parse"="@null"（一般应用于 url 输出格式与最终输出一致时）

## 4.注意事项

### 4.1.宏定义使用

@page 表示分页页码，一般用于@url, @args

@key 表示搜索关键字，一般用于@url, @args

@null 表示不用对应的处理，一般用于@parse, @method

### 4.2.函数的输入输出注意

所有函数的输入参数为 String 类型

所有函数的返回为 String 类型

## 四、格式及引擎能力详解

### 1.插件节点类型简介

主要节点	说明
------	----



sited	根节点
>meta	头部节点（标注元信息）
>main	主体节点
>script	脚本节点
节点类型	说明
::NodeSet	数据节点集合类型（不能有任何属性）
::Node	数据节点类型（必须要有属性）
::Add	附加节点类型（附加在 Node 下面） 或者共享 Node 的请求结果，无 url，有 parse（Node 有 url，有 parse） 或者为 Node 增加数据来源，有 url，有 parse（Node 无 url，无 parse）
::Item	配置节点类型（附加在 Node 下面） 为 Node 添加静态数据

## 2. 插件格式说明（?表示随意）

	下级	类型	
sited	插件根节点		
	@ver	Int	插件版本
	@vip?	Int	用户等级限制 0:普通用户可用（默认） 1:VIP 用户可用 2:SVIP 用户可用
	@debug?	1 或 0	是否开启调试模式
	@private?	1 或 0	是否为私密插件（私密插件不可分享）
	@engine	Int	需要支持的引擎版本号
	@schema	Int	插件规范版本号（v27 支持） 0 为测试规范（旧格式） 1 为正试规范（新格式）
	@head?	String	指定头部节点名称（v27 支持） 默认为：meta
	@body?	String	指定主体节点名称（v27 支持） 默认为：main
	@script?	String	指定脚本节点名称（v27 支持） 默认为：script
	@sds?	Uri	插件自动更新接口（参考 POJO 格式说明）

			检查时，@sds 末尾附上插件的 url 托管在插件中心的，不需要添加
>meta	头部节点		
	>ua?	String	访问插件的默认 UserAgent
	>guid	GUID	插件唯一标识（采用小写无间隔），此标识为插件更新提供权限保障。示例： 18d797d2697c48b983b1d6a2e2b867d3
	>title	String	插件名称（最好不要超过 5 个汉字）
	>author	String	插件开发者（最好不要超过 6 个字母）
	>contact	String	插件开发者联系方式： mail 或 sited 磁链或 phone
	>intro	String	插件介绍（显示在插件中心列表）
	>alert?	String	插件提醒（用户打开插件时，跳出提醒）
	>url	String	源网站地址
	>expr	String	匹配目标网址的正则表达式
	>logo?	Uri	插件图标（显示在插件中心列表）
	>encode	String	目标网址的默认编码
>main	主体节点		
	@dtype	[1,2,3,4,5,6,7]	数据类型（对应严格的数据格式） 1:漫画；2:轻小说；3:动画； 4:图片（无目录）；5:周边； 6:资讯；7:视频（无目录）
	@durl	String	S 按钮打开的源网站地址（可选）
	@btype?	[1,2,3,4,5,6,7]	业务类型（根据情况进行业务分类） 1:漫画；2:轻小说；3:动画； 4:图片（无目录）；5:周边； 6:资讯；7:视频（无目录）
	@btag?	String	对 btype 的真实描述（3 个字以内） 例如：漫画；小说，FM，电台...  原为：dtag
	@trace?	String	一个 HTTP 请求的 url 用于记录插件的点击
>>set*		::NodeSet	节点集合类型。不可以有任何属性！
>>node*		::Node	节点类型

@cache?	[1m, 1d, 1, 0]	缓存 (1=永久; 1d=1 天; 1m=1 分钟; 0=不缓存)
@title?	String	节点标题
@ua?	String	节点请求 HTTP 时使用的 UA 默认使用 site 或引擎给定的值
@expr?	String	正则表达式; 满足条件的, 由当前节点处理 一般用在 book 节点上
@btn?	String	按钮名称 (仅在有界面需求的地方可用)
@dtype?	[1,2,3,4,5,6,7]	当前节点的内容类型:: (用于混合插件) 1:漫画; 2:轻小说; 3:动画; 4:图片 (无目录); 5:周边; 6:资讯; 7:视频 (无目录)
@style?	[11,12]	界面样式风格:: (dtype[3,7]的扩展) 11:视频 (默认) 12:音乐
@update?	[0,1,2]	数据更新方式 (v28) 0: 每次先清空 (默认) 1: 添加到尾部 2: 插入到开头
@downAll?	1/0	是否提供全部下部下载功能 仅在 book[1,2,3]上可用 当 sections 内容为混合时, 设为 0
@showNav?	1/0	是否显示提供上下集导航功能 仅在 section[1,2,3]上可用
@showImg?	2/1/0	默认, 不添加属性 0: 无图 1: 小图 2: 大图 仅在 hots、updates 和 tag 上可用
@showWeb?	1/0	是否显示 S 按钮 private=1 是, 默认为 0 private=0 时, 默认为 1
@screen?	v/h	v:显示为竖屏; h: 显示为横屏 (只在视频类插件有效)
@options?	String	阅读选项 ("模式,方向,缩放,屏幕") 示例: "0,0,0,1"

			模式：0 流水，1 点击 方向：0 上下，1 左右，2 右左 缩放：0 适应屏幕，1 适应图片， 2 原始大小 屏幕：0 横屏，1 竖屏
@w?	宽比重		只在首页热门节点有效
@h?	高比重		只在首页热门节点有效
@method?	get 或 post 或 @null		请求方式 @null，表示不需请求；直接 parse(url)
@args?	String		为 post 提供参数支持，格式如下： k1=v1;k2=v2（其它用途时不计格式） 支持@page,@key 宏定义
@parseUrl?	fun(url,html) ->url;url;CALL::url		通过 Url 和它的 Html 输出新的 url 作为 parse 的目标网址，如果多个 url 以“;”隔 开（CALL::url 的请求结果仍由 parseUrl 处理）
@parse	fun(url,html) ->参考输出格式		解析函数 支持@null 宏定义（表示不进行解析）
@check?	fun(url,cookie) ->"1"或"0"		检查是否已登录 只在 login 节点有效
@auto?	"1"或"0"		是否自动检查登录状态并跳出登录窗口 只在 login 节点有效（具体参考示例）
@buildUrl?	fun(url,key?,page?) fun(url,str?)		通过 Url 生成新的目标 Url dtype=8 时，fun(url,str)
@buildWeb?	fun(url)		通过 Url 生成新的目标展示网址
@buildRef?	fun(url) ->url		通过 Url 生成新的当前 Url 的 Referer
@buildArgs?	fun(url,str?)		通过 Url 生成 post 参数，格式如下： k1=v1;k2=v2（其它用途时不计格式） dtype=8 时，fun(url,str)
@buildHeader?	fun(url) ->header		通过 Url 生成新的当前 Url 的 Header，格 式如下： k0 \$\$ k1:v1 \$\$ k2=v2
@buildCookie?	fun(url,html,cookie) ->cookie		通过 Url 和已有 cookie 生成新的 cookie 格式如下： k1=v1;k1=v1
@header?	String		要添加的 http Header，格式如下： k0 \$\$ k1:v1 \$\$ k2=v2

			(例: cookie \$\$ referer)
	@encode?	String	目标网址编码
	@url?	Uri	目标网址 支持@page 宏定义
	@run?	[web]	@run=web 时, 用浏览器打开 url
	@addPage?	Int	分页的增加值 (一般用于分页加载) 用于调节@page 的大小 (+-) 例: -2
	@addKey?	String	搜索的增加值 (一般用于搜索) 用于为@key 增加用户输入之外的键
	@addCookie?	String	默认添加的 Cookie 例: a=1;b=2 (以“;”隔开)
	.....		
	>item	Item	静态子项目 (当不通过 url 获取数据时) 一般为 tags 或 hots 配置静态数据 *.节点名称不可更改
	>@title?	String	目标标题
	>@group?	String	分组标题 (一类分组的第一项里添加)
	>@url?	Uri	目标网址 (可使用@key@page 宏定义)
	>@txt?	String	辅助文本
	.....		
	>add*	<b>::Add</b>	动态子项目 1.直接解析父节点的数据, >需要@parse 2.借父节点 Uri 构建目标 Uri, 并解析 >需要@parse, @buildUrl 3.为父节点增加数据源 >需要@parse, @url *.节点名称可具体命名
	>@parse	String	解析函数 (@null 表示不进行解析)
	>@url?	Uri	目标网址
	>@buildUrl?	fun(url) ->url	构建目标网址函数 根据上级 url 换成另一个 url
	>@...		... like Node (不支持@url)
<b>&gt;script</b>	脚本节点		
	>require	<b>::Node</b>	外部脚本文件
	>>item	<b>::Item</b>	

	>>@url	Uri	脚本网址
	>>@encode?	String	编码
	>>@lib	String	本地引擎内置的脚本库 如果有，则直接通过本地加载
	>code	CData	脚本代码

### 3.全局变量 SiteD 结构说明（可用于引擎版本的向下兼容处理）

属性	说明
ver	引擎版本号
udid	设备 ID 号
uid	用户 ID
usex	用户性别
uvip	用户 vip（0：不是；1：vip；2：svip）
ulevel	用户等级
函数	说明（val 统一为字符串）
get(key)	获取当前插件的一个配置值（v29 支持）
set(key,val)	设置当前插件的一个配置值
API	说明（val 统一为字符串）
get('g_location')	获取当前位置信息（v32 有效） {lat,lng,country,countryCode,province,city,cityCode}

注：使用前需要在插件 JS 代码里申明变量（便于向下兼容），示例：

```
var SiteD={}
function section_parse(url,html){
    if(SiteD.ver && SiteD.ver>=24){
        ...
    }else{

    }
}
```

## 4.引擎能力说明

引擎	说明
v34	使用新的 header 格式: key:val \$\$ key:val \$\$ key:val (旧的: key=val; key=vak;)
v33	增加 WEB::url 播放支持
v32	1.为 book[8]增加 ss 输出, 控制进入 section[1,2,3]节点的扩展 2.原 reward 更名为: about (即打赏改为关于、反馈、介绍之用) 3.增加 SiteD.get(' g_location') 4.增加 book[7]、section[3]的 parse 输出 web://的支持 (浏览器打开)
v31	增加 subtag 节点 (二级分类) (v31 支持)
v30	Add 增加 parseUrl(支持 CALL::),Search,Tag 增加 Add
v29	为插件提供存储能力 SiteD.get(key);SiteD.set(key,val);
v28	增加 dtype=8 全新插件
v27	增加新的插件规范
v26	1.section[1,2,3]增加 showNav 控制属性; 2.book[1,2,3]增加 donwAll 控制属性 3.section[2]、book[6]支持 t=10 (原始大小图片, 居中), 增加 c 颜色值
v25	1.增加@style 2.section[3]、book[7].parse 增加支持: [{url,type,mime,logo}] 格式 3.section[1]、book[4].parse 增加支持: {bg,list["",""]}& {bg,list[{url,time}]}格式 4.method 支持 @null (表示不用请求 http)
v24	1.@parseUrl 支持 CALL::url 和 url 的组合; CALL::结果仍由 parseUrl 处理 2.为 SiteD 注入新的 uid,usex 等信息 3.section[3]、book[7].parse 增加支持返回: [{url,type,mime}] 格式
v23	增加 reward 打赏节点; 增加@txt 属性
v22	hots, updates, tag , 全部支持 showImg=0,1,2; 并都支持 w,h 属性
v21	支持 302 跳转 url 的获取;
v20	1.hots、updates 支持跳转到 tags (提供更自由的数据组合) 2.book,section 数据支持 sited://格式跳转 (可提供插件中心作为插件) 3.支持无图的 tag 内容样式(showImg=0) 4.支持有图的 update 内容样式(showImg=1) 5.完善 vip 限制 6.支持 updates 通过 item 配置,增加 logo 的数据接收 7.增加 bookViewModel.isSectionsAsc 输出
v19	完成插件 WEB 登录功能, 并验证
v18	小说插件输出改为: [{d:',t:1},...]格式
v17	增加 object 节点配置
v16	增加 buildCookie;增加处理分支;

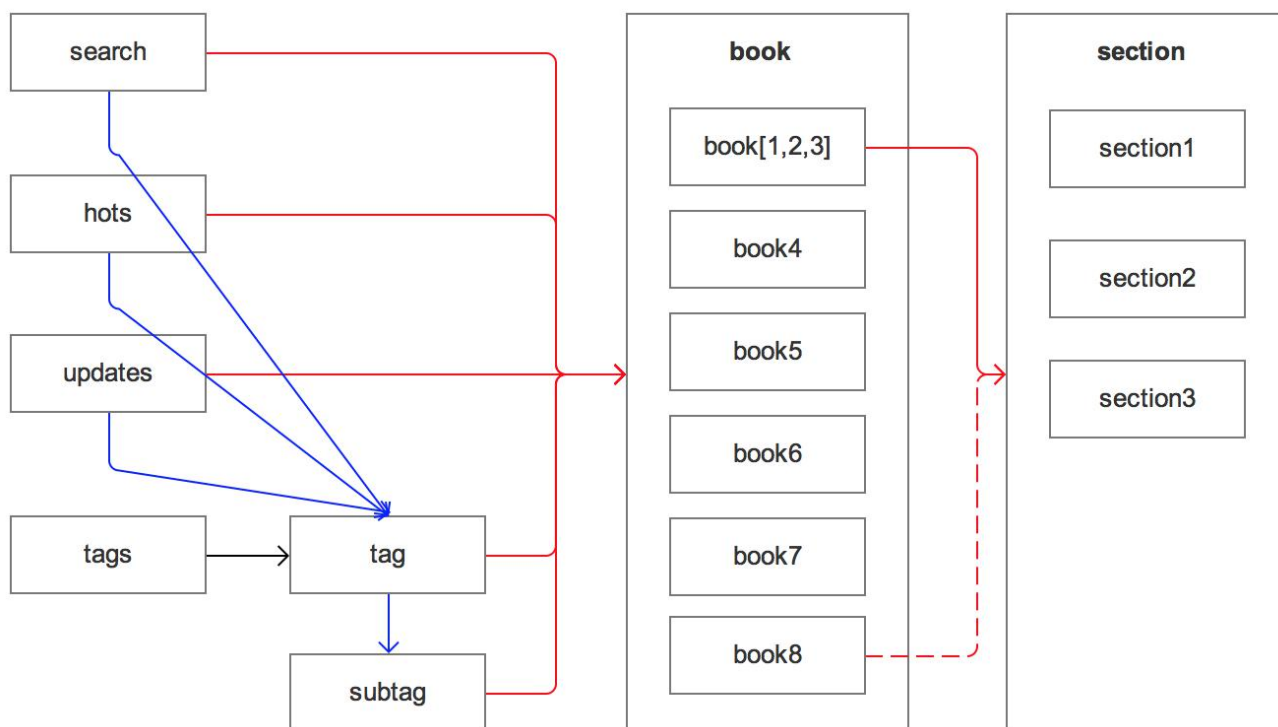
## 五、插件数据节点说明

### 1. 节点及其数据说明

	下级	类型	
<b>meta</b>			
>login		Node	:: 登录
>about		Node	:: 关于（简介、提供反馈渠道、让人赞助等）
<b>main</b>			
>home		Set	站点首页数据
	>hots	Node	:: 热门（可使用>item 进行静态配置） 注：item 支持跳转到 book 或 tag
	>>updates?	Add	当 hots 与 updates 在同一个 url 里使用（只填写 parse）
	>>tags?	Add	当 hots 与 tags 在同一个 url 里使用（只填写 parse）
	>updates	Node	:: 更新（可使用>item 进行静态配置） 注：item 支持跳转到 book 或 tag 注：当 showImg="1"时，使用有图模板
	>tags	Node	:: 分类（可使用>item 进行静态配置）
>search		Node	:: 搜索
>tag		Node	:: 分类（处理某一个分类的数据） 注：当 showImg="0"时，使用无图模板
>subtag		Node	:: 二级分类（处理某一个二级分类的数据） 注：当 showImg="0"时，使用无图模板 注：必须要要有 expr 注：与 tag 输出相同
>book		Node	:: 书本（处理某一本书的数据） 注：支持 sited://的 url 接收
	>sections?	Add	book 的数据与 sections 不在同一个 url 时使用
>section		Node	:: 章节（处理某一话的数据）



## 2.节点跳转示意图（基于 expr 和 dtype）



黑色线为自然推进

蓝色线为 expr 推进

红色线为 expr+dtype 推进

### 3.1.输出格式[dtype=1]::漫画（或有目录图片）

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[[name:"", url:"", logo:""]]
>updates.parse	[[name:"", url:"", logo?:"", newSection:"", updateTime:"yyyy-MM-dd"]]
>tags.parse	[[title:"", group:"" url:""]] 注: group, 为非必须项（只能在分组的第一项里出现）
>search.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:"", btag:""]]
>tag.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:""]]
>book.parseUrl?	url 或 CALL::URL（多个时用";"隔开） url = 目标 url（其请求结果转到 parse 处理） CALL::url = 中转解析 url（其请求结果仍回到 parseUrl 处理）  目标 url 需要中转解析，或分布在多个 url 上时使用。

>book.parse	{name:"", author:"",intro:"",logo:"",updateTime:"yyyy-MM-dd", isSectionsAsc:1 0,sections:[{name:"", url:""}]} 注: url 为空时, 则为分组
>section.parseUrl?	url;url;url (目标 url 字符串; 多个时用";"隔开) section 的数据分布在多个网页; 或目标 url 需要中转解析时使用
>section.parse	["", ""] (图片地址的数组) 或者, 带背景音乐的 {bg:"",list:["", ""]} (需引擎 v25 支持) 或者, 带背景音乐+带时间位置的 {bg:"",list:[{url:"",time:xxx},{...}]} (xxx 毫秒)  bg=背景音乐 url, 用于支持有声漫画

### 3.2.输出格式[dtype=2]:轻小说

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[{name:"", url:"",logo:""}]
>updates.parse	[{name:"", url:"", logo?: "",newSection:"",updateTime:"yyyy-MM-dd"}]
>tags.parse	[{title:"",group:"" url:""}] 注: group, 为非必须项 (只能在分组的第一项里出现)
>search.parse	[{name:"", url:"",logo:"",author:"",status:"",newSection:"",updateTime:"", btag:""}]
>tag.parse	[{name:"", url:"",logo:"",author:"",status:"",newSection:"",updateTime:""}]
>book.parseUrl?	url 或 CALL::URL (多个时用";"隔开) url = 目标 url (其请求结果转到 parse 处理) CALL::url = 中转解析 url (其请求结果仍回到 parseUrl 处理)  目标 url 需要中转解析, 或分布在多个 url 上时使用。
>book.parse	{name:"", author:"",intro:"",logo:"",updateTime:"yyyy-MM-dd", isSectionsAsc:1 0,sections:[{name:"", url:""}]} 注: url 为空时, 则为分组 注: isSectionsAsc,指示 sections 是否为 ASC 排序
>section.parseUrl?	url;url;url (目标 url 字符串; 多个时用";"隔开) section 的数据分布在多个网页; 或目标 url 需要中转解析时使用
>section.parse	[{d:"xxx", t:1, c:"#999999",b:1,i:1,u:1,w:10,h:10}] d: 数据

	t: 类型 (1,文本; 8,视频; 9,图片; 10,原始大小图片[居中]) c?:颜色  b?:是否加粗 i?:是否斜体 u?:是否下划线 w?: 视图宽 (当 t=8, 10 时有效) h?: 视图高 (当 t=8, 10 时有效)
--	--

### 3.3.输出格式[dtype=3]::动画 (或有目录视频)

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[[name:"", url:"", logo:""]]
>updates.parse	[[name:"", url:"", logo:"", newSection:"", updateTime:"yyyy-MM-dd"]]
>tags.parse	[[title:"", group:"", url:""]] 注: group, 为非必须项 (只能在分组的第一项里出现)
>search.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:"", btag:""]]
>tag.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:""]]
>book.parseUrl?	url 或 CALL::URL (多个时用";"隔开) url = 目标 url (其请求结果转到 parse 处理) CALL::url = 中转解析 url (其请求结果仍回到 parseUrl 处理)  目标 url 需要中转解析, 或分布在多个 url 上时使用。
>book.parse	{name:"", author:"", intro:"", logo:"", updateTime:"yyyy-MM-dd", isSectionsAsc:1 0, sections:[[name:"", url:""]]} 注: url 为空时, 则为分组
>section.parseUrl?	url 或 CALL::URL (多个时用";"隔开) url = 目标 url (其请求结果转到 parse 处理) CALL::url = 中转解析 url (其请求结果仍回到 parseUrl 处理)  目标 url 需要中转解析, 或分布在多个 url 上时使用。
>section.parse	url 或者 [[url:"", type:"", mime:"", logo:""]] (新格式可更好支持下载::仅 v24 支持) 常见的 type - mime

	.mp4 - video/mp4 .mp3 - audio/x-mpeg  更多 type - mime 去网上搜索； @style=12 时需要 logo
--	---

### 3.4.输出格式[dtype=4]::图片（或无目录的漫画）

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[[{name:"", url:"",logo:""}]]
>updates.parse	[[{name:"", url:"", logo?:"",newSection:"",updateTime:"yyyy-MM-dd"}]]
>tags.parse	[[{title:"",group:"", url:""}]] 注： group， 为非必须项（只能在分组的第一项里出现）
>search.parse	[[{name:"", url:"",logo:"",author:"",status:"",newSection:"",updateTime:"", btag:""}]]
>tag.parse	[[{name:"", url:"",logo:"",author:"",status:"",newSection:"",updateTime:""}]]
>book.parseUrl?	url 或 CALL::URL（多个时用";"隔开） url = 目标 url（其请求结果转到 parse 处理） CALL::url = 中转解析 url（其请求结果仍回到 parseUrl 处理）  目标 url 需要中转解析， 或分布在多个 url 上时使用。
>book.parse	引擎 v25 支持的新格式： ["",""]（图片地址的数组） 或者， 带背景音乐的 {bg:"",name?:"",logo?:"",list:["",""]} 或者， 带背景音乐+带时间位置的 {bg:"",name?:"",logo?:"",list:[{url:"",time:xxx},{...}]}  bg=背景音乐 url， 用于支持有声漫画 name,logo（v26 新加）

### 3.5.输出格式[dtype=5]::商品（周边、漫画书之类）

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[[{name:"", url:"",logo:""}]]

>updates.parse	[[name:"", url:"", logo?:"", newSection:"", updateTime:"yyyy-MM-dd"]]
>tags.parse	[[title:"", group:"", url:""]] 注: group, 为非必须项 (只能在分组的第一项里出现)
>search.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:"", btag:""]]
>tag.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:""]]
>book.parseUrl?	url 或 CALL::URL (多个时用";"隔开) url = 目标 url (其请求结果转到 parse 处理) CALL::url = 中转解析 url (其请求结果仍回到 parseUrl 处理)  目标 url 需要中转解析, 或分布在多个 url 上时使用。
>book.parse	{name:"", shop:"", intro:"", logo:"", buyUrl:"" pictures:["", ""]} (v26)  注: buyUrl, 可购买的点击 url

### 3.6.输出格式[dtype=6]::资讯

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[[name:"", url:"", logo:""]]
>updates.parse	[[name:"", url:"", logo?:"", newSection:"", updateTime:"yyyy-MM-dd"]]
>tags.parse	[[title:"", group:"", url:""]] 注: group, 为非必须项 (只能在分组的第一项里出现)
>search.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:"", btag:""]]
>tag.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:""]]
>book.parse	[[d:"xxx", t:1, c:"#999999", b:1, i:1, u:1, w:100, h:100]] 或 {name?:"", logo?:"", list: [[d:"xxx", t:1, c:"#999999", b:1, i:1, u:1, w:10, h:10]] } (v26 支持) d: 数据 t: 类型 (1,文本; 8,视频; 9,图片; 10,原始大小图片[居中]) c?:颜色 b?:是否加粗 i?:是否斜体 u?:是否下划线

	w?: 视图宽 (当 t=8, 10 时有效) h?: 视图高 (当 t=8, 10 时有效)
--	--

### 3.7.输出格式[dtype=7]::视频 (或无目录的动画)

处理事件	输出格式
>login.check	"1"或"0"
>hots.parse	[[name:"", url:"", logo:""]]
>updates.parse	[[name:"", url:"", logo:"", newSection:"", updateTime:"yyyy-MM-dd"]]
>tags.parse	[[title:"", group:"", url:""]] 注: group, 为非必须项 (只能在分组的第一项里出现)
>search.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:"", btag:""]]
>tag.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:""]]
>book.parseUrl?	url 或 CALL::URL (多个时用";"隔开) url = 目标 url (其请求结果转到 parse 处理) CALL::url = 中转解析 url (其请求结果仍回到 parseUrl 处理)  目标 url 需要中转解析, 或分布在多个 url 上时使用。
>book.parse	url 或者 [[url:"", type:"", mime:"", logo:""] (新格式可更好支持下载::仅 v24 支持) 或者 {name:"", logo:"", list: [[url:"", type:"", mime:"", logo:""]]} (v26) 常见的 type - mime .mp4 - video/mp4 .mp3 - audio/x-mpeg  更多 type - mime 去网上搜索; @style=12 时需要 logo

### 3.8.输出格式[dtype=8]::查询工具

处理事件	输出格式
>login.check	"1"或"0"

>hots.parse	[[name:"", url:"", logo:""]]
>updates.parse	[[name:"", url:"", logo?:"", newSection:"", updateTime:"yyyy-MM-dd"]]
>tags.parse	[[title:"", group:"", url:""]] 注: group, 为非必须项 (只能在分组的第一项里出现)
>search.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:"", btag:""]]
>tag.parse	[[name:"", url:"", logo:"", author:"", status:"", newSection:"", updateTime:""]]
>book.parseUrl?	url 或 CALL::URL (多个时用";"隔开) url = 目标 url (其请求结果转到 parse 处理) CALL::url = 中转解析 url (其请求结果仍回到 parseUrl 处理)  目标 url 需要中转解析, 或分布在多个 url 上使用。
>book.parse	[[d:"xxx", t:1, c:"#999999", b:1, i:1, u:1, w:100, h:100]] 或 {name?:"", logo?:"", list: [[d:"xxx", t:1, c:"#999999", b:1, i:1, u:1, w:10, h:10]]} d: 数据 t: 类型 (1,文本; 9,图片; 10,原始大小图片[居中]) c?:颜色 b?:是否加粗 i?:是否斜体 u?:是否下划线 w?:图片宽 (当 t=10 时有效) h?: 图片高 (当 t=10 时有效) url:点击后要打开的网址 ss?: 0 或 1, url 是否通过 section[1,2,3]节点打开 (v32 支持)

#### 4. about 节点下的 item 说明

属性	输出效果 (每个属性各占一行)
title	加粗文本
url	下划线且可复制文本
logo	图片
::expr	::url 或 logo 点击后的用浏览器打开 expr (expr 即为链接地址)
txt	普通文本
无属性	空一行

## 5. book[dtype=8]节点下的 item 说明

显示属性	输出效果（每个属性各占一行）
title	加粗文本
txt	普通文本
无属性	空一行
输入框属性	
key	输入框（key 为输入框的 id） 或通过：buildUrl(url,str)获取输入值； 或通过：parse(url,str)获取输入值（当 method=@null 时）。  str 为 json 格式，例：{"key1':'xxx',key2:'yyy'"}"
hint?	输入框的提示文字
isScan?	0 或 1。输入框增加扫描支持

## 六、插件节点数据的获取程序

### 1.buildUrl, parseUrl, CALL::url, parse 的内部逻辑（伪代码）

```
public void getNodeData(url,callback){
    if(@buildUrl){
        url = @buildUrl(url); //or @buildUrl(url,page) //or @buildUrl(url,key,page)
    }

    if(@check(url ,cookie)==false){
        return;
    }

    var fun=(url)->{
        var html = $http(url);
        if(@parseUrl){
            var urls = @parseUrl(url,html);
            for(var u1 in urls){
```



```

        if(u1.start('CALL:')){
            fun (u1.remove('CALL:'));
        }else{
            var h1 = $http(u1);
            callback(@parse(u1,h1));
        }
    }
}
}
}
};

fun(url);
}

```

## 2.CALL::的补充说明

形式	说明
CALL::url	get 请求 url
CALL::GET::url	get 请求 url （请使用上面的简写方式）
CALL::POST::url	post 请求 url
注： 如需 header 请在当前节点的 buildHeader 里做分流处理	

## 七、Add 节点的示例说明

### 示例 1：直接解析父节点的数据

```

<home>
  <hots cache="1d" title="Hot" method="get" parse="hots_parse" url="http://xxx.xxx.xxx" >
    <updates parse="updates_parse" />
    <tags parse="tags_parse" />
  </hots>
  <updates title="Updates" /> //只起到占位作用
  <tags title="Tags" /> //只起到占位作用
</home>

```

## 示例 2：借父节点 Url 构建目标 Url，并解析

```
<book cache="1d" method="get" parse="book_parse" expr="\VcomicV">  
  <sections cache="1d" method="get" parse="book_s_parse" buildUrl="book_s_url" />  
</book>
```

## 示例 3：为父节点增加数据源

```
<tags title="分类">  
  <tags cache="1d" method="get" parse="tags_parse" url="http://m.x.x/cat?type=game"/>  
  <tags cache="1d" method="get" parse="tags_parse" url="http://m.x.x/cat?type=sjyx"/>  
  <tags cache="1d" method="get" parse="tags_parse" url="http://m.x.x/cat?type=ylxtd"/>  
</tags>
```

# 八、插件更新接口规范

例：

@sds=http://api.xxx.xxx/update?

>url=http://m.bbb.com

则请求时地址为：

<http://api.xxx.xxx/update?http://m.bbb.com>（即@sds 末尾附上插件的 url）

要求输入数据为：

{ver:1,url: ""}（ver 为最新版本号；url 为最新版下载地址）

\*客户端会比较新的版本，如果服务端更新则使用 url 进行下载

## 附录

### \*.调试说明

#### 1.安装好插件

- 2.进入多多猫 app 高级设置，长按标题可开启开发者模式（仅 Android 版本支持）
- 3.可在 `/android/data/org.noear.ddcat/files` 查看日志文件

## \*.相关约定

- 1.插件文件名以 `.sited.xml` 结尾
- 2.加密后的插件文件名以 `.sited` 结尾
- 3.解析函数名用节点名\_功能名  
例如： `book_parse`、`book_buildUrl`、`book_buildRef`、`book_buildArgs`  
再如： `book1_parse`、`book4_parse`（混合插件分流处理时）  
再如： `book1_parse_xxx`、`book_parse_yyy`
- 4.认真填写 `guid`、`expr`、`author` 和 `intro` 节点的内容

## \*.联系方式

QQ 群： 316410970  
示 例： QQ 群共享下载  
开发者中心： <http://sited.noear.org/dev>（学习、示例、发布、管理插件）