# Luci2(OpenWrt web管理界面) api

一直以来OpenWrt都是采用Lua写的web管理界面LuCI，（开机速度慢不说，居然比不过腾达等弱路由器开机速度）。 LuCI需要使用多个Lua扩展(如 ubus, luci.model.uci, nixio.fs, 等等)去存取系统信息和设置. 不幸的是这种解决方案在慢CPU和低内存的低配机器设备上是个灾难，此方案相当消耗资源且并不能很好的工作。

这导致开发了LuCI2, 一个不同架构的新的web管理节目。它不再使用Lua，而是使用静态HTML页面加JavaScript XHR方法。 这意味着从OpenWrt设备中下载后在客户端(浏览器)中构建HTML页面, 通过ubus存取各种系统数据(通过uhttpd-mod-ubus提供基于HTTP的接口API)。

如上所述, LuCI2通过ubus和OpenWrt子系统通信(包括如network、service以及其它)。遗憾的是并非每个主要的OpenWrt工具都自己注册了ubus,如LuCI2不能使用opkg(安装包管理)。 LuCI2通过提供在附加ubus名称空间的rpcd插件解决了这个问题。前面说的opkg它在ubus中注册了一个新的luci2.opkg路径来访问。

综上所述, LuCI2包括两个方面: 打包的HTML/CSS/JS文件 (htdocs) 和一些在OpenWrt环境下运行的附加小工具。

**菜单：**首先需要知道的是浏览器接收到的关于LuCI2菜单并不固定写死在任何文件。代替的是通过ubus使用luci.ui路径和menu方法。可以通过使用以下命令来查看，ubus call luci2.ui menu '{ "ubus\_rpc\_session": "invalid" }'

**模板：**每一个LuCI2子页面必须有一个存放在/www/luci2/template/目录下的模板。它们提供了非常简单的内容替换区的HTML文件。需要注意的是它们不包含任何变量的引用，这是JavaScript的功劳：用JavaScript去读取并写入内容。在为本地化系统(i18n)这些文件中唯一特定的语法类似下面的标记:<div id=’map’></div>。

**视图：**从模板中分离，每个LuCI2子页面也需要一个定义并存放在/www/luci2/view/目录下的一个视图，视图是使用了子页面特定对象的L.ui.view扩展的JavaScript文件。Javascript中需要提供execute方法的实现，该方法将在膜拜加载后被执行。可选的也可以提供子页面的title和description属性。

L.ui.view.extend({

title: L.tr('Foo'), /\* 可选的标题 \*/

description: 'Bar', /\* 可选的描述 \*/

execute: function() {

var deferred = $.Deferred(); /\* 创建一个延迟对象 \*/

deferred.resolve(); /\* 立即Resolve,它不做任何事可以返回失败 \*/

return deferred.promise(); /\* 返回Promise对象 (延迟对象的子集) \*/

}

});

**通过ubus通讯：**在开始之前，需要知道的是LuCI2提供了一些存取UCI系统的一些帮助。如果写一个简单的管理/etc/config/下配置文件不需要完全知道ubus的调用方法，则可以跳过该部分。构建一些更复杂的LuCI2视图前最好先弄明白使用到的ubus调用。完整的对象和方法列表可以通过运行ubus -v list命令来得到。下面这个简单的例子调用了log对象和write方法，它需要提供一个event参数传递进去。使用ubus命令行国内根据是，需要如下所示命令:

ubus call log write '{ "event": "Foo" }'

LuCI2为ubus通讯提供了一个叫做L.rpc.declare的工具，它可以如魔法般的帮助JavaScript访问ubus方法。注意，定义声明方法的时候方法并不被执行，参数也未传递，这是为以后调用准备的一个方法。下面这个示例的方法调用了log对象的write方法:

var writeToLog = L.rpc.declare({

object: 'log',

method: 'write',

params: [ 'event' ]

})

定义了一次这个方法后，可以在任意时间通过如下示例简单调用:

writeToLog('Foo');

在上面的例子中执行结果被忽略了，如果视图需要处理ubus返回的数据时不能忽略执行结果。

var readSystemInfo = L.rpc.declare({

object: 'system',

method: 'info',

params: [‘a’,’b’] //可选

expect: { memory: { } } /\* 可选, 只提取结果的一部分memory \*/

})

readSystemInfo().then(function(memory) {

console.log(memory);

});

# Ais-rg（AC85）

Ac85是基于luci2框架和jsrender，采用模板与数据相互绑定，双向流动的管理界面，相比传统的luci2界面，更高效，代码更简洁，紧凑，传统luci2是通过js将页面的内容动态生成，模板只提供一个最外层的div，例如<div id=’map’></div>，然后对应的js生成dom之后插入到该模板内。这种做法的优点是dom结构由js完成，通过查看网页源码是无法获得其html内容，保密性好，但其缺点也非常显著，由于js构建dom，非常消耗js引擎，同时伴有阻塞，对浏览器的渲染而言不流畅（相对于静态html），也不易于维护和调试。

Ais-rg完美解决了这个问题，在原有的luci2框架的基础上，使用jsrender，这是前端jQuery模板的一种，类似于目前流行的angular，react和vue。可以预先自定义一些固定格式的HTML标签，在需要显示数据时，再传入真实数据组装并展示在Web页中；避免了在JS中通过“+”等手动分割、连接字符串的复杂过程；针对高性能和纯字符串渲染进行了优化；无需依赖DOM和jQuery；在<script id="temp" type="text/x-jsrender">标签里面可以清晰的看到整个dom结构，而js将实际的数据填充到对应的绑定位置。

在实际开发中，有如下几种模板。

1. 字符串常量，<h4 class="title-topic"><%:DDNS Settings%></h4>，渲染的结果就是<h4 class="title-topic">DDNS Settings</h4>。
2. 动态数据，<input type="checkbox" id="ddnsEn" data-link='ddnsEn' />，渲染的结果是<input type="checkbox" id="ddnsEn" /> 它的值由ddnsEn来决定。
3. 事件绑定，<button class="card-btn green" data-link='{on ~applyDdnsEn}'><%:Apply%></button>，点击后执行applyDdnsEn函数，对于button来讲on事件就是默认的点击，其他事件需在on后面加上事件名，例如<select data-link=’{on “change“ ~change}’>。
4. 带有参数的事件绑定，<img style=’cursor:pointer’ src=’lcui2/icons/xxx.png’ data-link=’{on ~edit}’ id{:#index}>，表示在执行edit函数，传入id作为参数，id为遍历时的index下标。
5. 动态绑定css样式，<div id=’xx’ data-link=”css-display{:(xx?’’:’none’)}”>，根据xx的值来判断是否显示。
6. 循环，很多情况下需要构造一个表格或者下拉菜单等，js得到的数据如数组，对象，将其传入模板，然后通过循环语句得到目标结果，例如

<select type=’text’ data=link=’xxx’ id=’yy’>

{^{for ~array}}

<option value=’{{>val}}’>{{>txt}}</option>

{{/for}}

</select>

表示从js获取的array循环构造option，option包含的值val，txt均来自于遍历的每一位，用js语法来解释就是：

for(let item of array){  
 let opt = $(‘<option />’).val(item.val).text(item.txt);

opt.appendTo($(‘#yy’))

}。

1. 单选框radio，在html内就有input type=’radio’这样的标签，当name值相同时，决定了从众多选项里面选择一个值，在ais-rg系统里面为了绑定这一个选出的值，设计了radiogroup这样的模板，它包裹整个radio集合，其值为从radio里面选择的值，例如，

{^{radiogroup val}}

<label><input type=’radio’ name=’xx’ value=’0’>0</label>

<label><input type=’radio’ name=’xx’ value=’1’>1</label>

{{/radiogroup}}

表示从这两个radio选出的值决定了js绑定在radiogroup的val的值。

1. 动态变量，类比字符串常量，dom内部的值由js提供，两种情况，第一由循环得到的变量，例如

{^{for array}}

<td class=’text-center’>{{>txt}}</td>,

{{/for}}

表示循环得到的对象身上的txt变量值去充填这个td。第二非循环，例如

<p>{^{:model}}</p>，直接用js提供的值model，若需要使用三木运算，{^{:(xx?:)}}。

1. 条件判断，根据某个条件是否成立来构造dom，注意，这里是要么构造目标dom，要么不构造，而非显示隐藏，例如，

{^{if xxx}}

<div>xxxxx</div>

{{/if}}，表示当xxx为真时，构造里面的dom，否则不构造。If和else搭配的情况是

{^{if xxx}}

<div>…</div>

{{else}}

<div>\\\<div>

{{/if}} 注意：没有else if的情况，一律用else代替。

1. 绑定转换类型后的值，将js提供的值进行转换，然后绑定到模板，例如，<input type=’checkbox’ data-link=’{toBool:xxx:toInt}’>，表示将js提供的xxx，先转换成number，再转换成bool，xxx若为0，转换为false，为1转换为true。
2. 混合绑定，在一个标签里面绑定多个内容如css样式，页面值，属性，方法，例如<select data-link=”

css-display{:(xx?’’:’none’)}

disabled{:yy==true}

on ‘change’ ~change

id{:id}

{:zz:}

”></select>，表示当xx为真时显示，yy为真的时候disabled属性展示，change时触发change函数，id属性为js的id,绑定js的值为zz。

Js部分：Ais-rg采用的js书写规范是基于传统的luci2，luci2使用的是L.ui.view.extend({

//业务逻辑代码

})，

Ais-rg采用的类似于上述，L.ui.alpha\_view.extend({}),下面详细介绍逻辑功能代码。

任何一个页面js文件都有\_data和package属性，\_data是绑定到模板上的数据源，它是一个对象；package是一个数组，其成员代表了该页面使用到的数据库，即/etc/config下面的文件。

exacute函数，是整个js文件执行的入口，内部往往有一个初始化函数init，用来初始化\_data这个对象。

save 函数，用来保存页面的配置。

Validation 函数，用来验证页面值的合法性。

Refresh 函数，重新刷新页面。

Add函数，添加配置

Modify函数，修改某一项配置。

Delete函数，删除某一项配置。

简而言之整个流程就是首先从config里面拿到初始值，初始化页面，然后部署各种函数，对页面增删查改的操作，最后保存。主要的api如下。

1. 清除所有验证信息 delete L.cbi.validation.message。
2. 编译验证类型 var vstack = L.cbi.validation.compile('hostname');

var v = vstack[0].apply(self.\_data['data']['domain']);

if(!v) return;

1. 设置页面的绑定值 $.observable(self.\_data['data']).setProperty(key,val);
2. /etc/config下增加内容 var sid = L.uci.add('packagename',type);

L.uci.alpha\_set('packagename',sid,option,val);

5. /etc/config下删除整某一个整体配置 L.uci.remove('packagename',sid);

6. /etc/config下删除某一个配置下的option L.uci.alpha\_set('packagename',sid,option,'');

7. /etc/config下设置 L.uci.alpha\_set('packagename',sid,option,val);

8. 获取/etc/config内容 L.uci.sections('packagename',[.type]);

9. 保存设置 save: function(){  
 var self = this;

L.ui.loading(true);

L.uci.save().then(function() {

if (L.globals.commit\_button) {

L.ui.loading(false);

self.render()

} else {

L.commitFunc()

}

}

10. 模板数据绑定 var template = $.templates("#temp"); template.link('#map',self.\_data,eventBindings)

11. 重带某些功能 L.system.initRestart('minidlna')。

12. 使用ubus获取数据，首先定义接口xxx: L.rpc.declare(),然后L.rpc.batch(),调用self.xxx()后，L.rpc.flush().then(function(res){}),结果包含在res里面，res是一个数组，每一位表示的是对应调用接口后得到的数据，顺序和接口调用的顺序一致。

# UI.js部分

Ui.js是构造web页面UI组件的主要文件，实质是通过js创建dom到页面，这些dom拥有各种属性的方法，创造组件需要调用L.ui的各种方法，下面列举主要的ui组件。

1. loading(enable[,msg])，弹出加载层，用于页面数据加载或者提交数据到后台这个过程的遮罩蒙版,enable为布尔值，为真时弹出，为假时消失，msg为可选项，代表加载层的标题，默认为‘Loading data…’。
2. dialog(title,content,options)，弹出对话框，title为标题，为false时对话框消失；content为具体dom结构；options是一个配置对象，style属性是这个对话框的类型，confirm，close，wait可选，选择之后可配置对应的回调函数，

style==confirm时，confirm: function(){} cancel: function(){}

style==close时，close: function(){}

wide属性是设定对话框宽度的。

1. reconnect([,info])，重连设备,info可选，表示重连弹出层的提示语。
2. login(invalid), 登录，invalid为false或undefined时，提示用户名密码错误。
3. getHash(‘view’), 获取当前页面view的哈希值。
4. setHash(‘view’,hash), 设置页面（跳转到）hash路径。
5. changeView(), 切换页面，在每次hash变化时触发。
6. loadTheme()，加载主题文件，htm,css,js
7. load()， 登录后执行，加载当前页面的数据文件。
8. renderMainMenu(),，初始化全局变量。
9. menu\_aisnav，构造函数，创建主界面菜单栏，包含handleClick，注册菜单点击事件和renderNodes渲染菜单节点。
10. mobilemenu\_aisnav，构造函数，属性同上。
11. viewmenu\_aisnav，构造函数，创建侧边导航栏，包含handleClick,导航栏点击事件，renderNodes渲染导航栏节点，render渲染入口。
12. table({columns, caption}), 创建表格,columns表示每一行的内容格式，它是一个数组，其成员对象是每一行具体的每一列，里面的属性决定了具体某行某列的样式，例如columns: [{align:’center’,’width’:’200px’,key:’ip4’,caption:’ip’},{},{}],表示每一行的第一列的样式居中对齐，长度200px，内容为ip4的值，表头为ip；

然后通过new操作之后得到实例化对象，调用row(obj)或rows(objs)即可，obj代表一个一维数组或者对象，若定义时的colums的key存在，按照key的值从该obj对象里取值，若不存在key，则按照obj数组的顺序依次取值，

例如var table = new L.ui.table(columns: [{key:’ip’},{key:’mac’}]), var table2 = new L.ui.table(columns:[{width: 4},{align: ‘center’}]); 在table调用rows时，传入[{ip:’192.168.1.1’,mac:’11:11:11:11:11:11’},{ip:’192.168.1.2’,mac:’22:22:22:22:22:22’}], 在table2调用rows时传入[[192.168.1.1,11:11:11:11:11:11],[192.1468.1.2,22:22:22:22:22:22]],即可。

1. grid,创建栅格系统，原理同table，增加createCell方法，可指定单元格类名。
2. form,创建类似表单结构，原理同grid。。
3. hlist({item,separator}), 创建一个文本列表，以特殊标志分隔，item参数必须是一个一维数组，separator是分隔符，可选，默认为‘|’，用分隔符将item的成员分开，形成一个文本列表。
4. progress({value,max,format})，创建进度条的组件,其参数分别代表value当前值，max最大值和format格式化比值。progress\_home是ais的进度条原理与luci2进度条一样，采用ais自身样式。
5. staus，类似hlist，区别是不使用分隔符而是采用换行符将itme内容分开。
6. devicebadge, 创建设备标志以及详情。
7. icon(src,alt,title),创建图标，参数分别是图标路径，文字和title。
8. button(label,style,title),创建按钮，参数分别是文本内容，boostrap的button风格（默认default）和title。

# Luci2.js 部分

Luci2.js 是整个系统基础js文件，它本质是一个构造函数，包含了非常复杂的功能，当页面加载时会实例化这个构造函数，生成一个L的对象，页面上的绝大多数操作都是调用这个对象身上的方法，而系统加载的其他js文件，都是在这个实例化对象身上拓展而来的如ui.js,cbi.js,rpc.js，最终调用的时候会变成L.ui.xxx,L.cbi.xxx,L.rpc.xxx,这样的方法，下面详细介绍luci2.js的常用api。

1. String的原型链上拓展padLeft方法，接受两个参数total和pad，total代表向左充填到多少位，比如一个str=‘123’，total为5时表示需要将该字符串左补齐充填至5位，pad表示占位字符，‘123’.padLeft(5,’|’),得到的结果是‘||123’，如果total小于字符串自身长度，就表示从倒数第几位开始截取该字符串，‘12345’.padLeft(3,’-’),得到’345’。
2. String的原型链上拓展format方法，表示格式化字符串，类似于python中的fomat, ‘%d %s’.format(2,’666’)这样的替换作用，格式化包括，%b（二进制）,%c(ASCII),%d(整数），%u(绝对值),%f(浮点类型)，%o(八进制), %s(字符串),%x(十六进制小写), %X(十六进制大写)，%t(时间) ，%m(内存)。
3. callSuper(func,…res), 调用祖先的方法,第一个参数表示需要调用的方法名，后面的参数表示在func执行时需要的参数。
4. require(name), 引入js文件，name是js文件名，例如L.require(‘home.home.js’)。
5. i18n，加载语言包。
6. tr(msg), 翻译函数，将msg翻译成对应的系统语言。
7. setHash(key,value), 设置页面哈希值，已键值对形式呈现，每个键值对以,分隔，例如L.setHash(‘view’,’home/home’), 会得到location.href = #id:xxx,view:home/home,进而达到跳转页面效果。
8. getHash(key), 获取页面哈希值，例如L.getHash(‘view’)得到当前页面的路径。
9. toArray(value), 转换一个值为数组，当value是number或bool时，返回[value]这样一位的数组；当value是string类型时，返回的该字符串按照空白字符串分隔后的数组，当value为object时，返回该对象属性的集合，其他情况一律返回空数组。
10. toObject(value), 与toArray相似，转换一个值为对象，number或bool时，返回{value: true}，string时返回一个对象，该对象将字符串每一位作为属性，属性值全为true；array时与string情况一样，其他情况全为空对象。
11. filterArray(array,item), 从一个数组里面剔除item元素，array不为数组时返回空数组。
12. parseIpv4(str), 判断str是不是一个标准的ipv4地址，不是地址返回空数组，是则返回字符串按照’.’分割的数组，例如’192.168.1.1’转化为[192,168,1,1]。
13. parseIpv6cidr6(str), 判断str是否为ipv6或ipv6前缀，返回十六位的数组。
14. checkSegment(str)，判断网关。
15. parseIpv6(str),判断是否为标准ipv6地址，返回十六位数组。
16. isNetmask(addr), 判断是否为掩码。
17. commitFunc, 提交保存函数。
18. globals, 全局对象，由所有全局变量组成，把这些全局变量作为属性添加到这个全局对象上面，变量值对应其属性值。

# Cbi.js 部分

Cbi.js 是另一种构造ui组件的js文件，将常用的表单组件封装，同时提供多种方法对表单值的操作。具体表单组件如下。

L.cbi.DummyValue

L.cbi.InputValue (datatype:1.range(1,100),2.ip4addr,3.ip6addr,4.port)

L.cbi.CheckboxValue

L.cbi.DynamicList

L.cbi.ListValue

L.cbi.PasswordValue

L.cbi.DeviceList

L.cbi.ComboBox

L.cbi.MultiValue

L.cbi.HiddenValue

L.cbi.ButtonValue

L.cbi.NetworkList

L.cbi.WlanFreqValue 工作频率

L.cbi.NamedSection

L.cbi.TypedSection

L.cbi.TableSection

L.cbi.GridSection

L.cbi.SingleSection

Validation作为验证方法，其验证类型为，

'integer'

'uinteger'

'float'

'ufloat'

'ipaddr'

'ip4addr'

'ip6addr'

'netmask4'

'netmask6'

'cidr4'

'cidr6'

'ipmask4'

'ipmask6'

'port'

'portrange'

'macaddr'

'host'

'hostname'

'network'

'wpakey'

'uciname'

'range'

'min'

'max'

'and'

'or'

‘device’

# Uci.js 部分

Uci.js 是跟数据库打交道的js，提供多种方法方便我们能对/etc/config下面的文件的增删查改,其调用方法为L.uci.xxx(params)。下面介绍其主要方法。

1. load(packages), 加载数据库。
2. unload(packages), 删除数据库。
3. add(conf,type,name), 在数据库中添加一个新的config，参数分别代表名称，类型和sid。
4. remove(conf,sid),在数据中删除某一个config，参数同上。
5. sections(conf,type,cb)，获取该数据库中的某一个config内容，返回一个对象，若只传conf，将返回整个数据库对象，cb为获取之后的回调函数。
6. get(conf,sid,opt), 获取config下面对应sid的option值。
7. set(conf,sid,opt,val), 设置config里面的sid中option为val。
8. alpha\_set(), ais系统的别名，效果和set一样。
9. unset(conf,sid,opt) , 删除config里面sid的option。
10. save(), 保存配置。