

# OpenWRT 系统的网页配置分析与界面开发

潘楚文<sup>1+</sup>

<sup>1</sup>(广东海洋大学 信息学院 电气工程及其自动化 1112 班)

## Analysis on the OpenWRT of Web Configuration System and Interface Developing

Chuwen Pan<sup>1+</sup>

<sup>1</sup>(Class Electrical Engineering and Automation 1112, Information Insitute, GuangDong Ocean University)

+ Corresponding author: Phone: +86-13590044487, E-mail:540981964@qq.com,

<http://www.gdou.edu.cn>

**Abstract:** In this paper stresses the relationship openWRT system configuration process luci web architecture and one of the modules, in-depth analysis of the structure of each module, and to understand how the process luci as cgi to generate pages and control elements of the website.

**Key words:** CGI; lua; LuCI; MVC; openWRT; uhttpd

**摘 要:** 以本文主要讲了 openWRT 系统的配置网页处理程序 luci 的架构以及其中各模块的关系, 深入分析各个模块结构, 并了解 luci 作为 cgi 处理程序如何生成网页和控制网页中的元素。

**关键词:** CGI; lua; LuCI; MVC 框架; openWRT; uHTTPd

## 1 前言

openWRT 系统现在的版本的网页配置站点是由 uhttpd 和 LuCI 组成, 作为“FFLuCI”诞生于 2008 年 3 月份, 目的是为 OpenWrt 固件从 Whiterussian 到 Kamikaze 实现快速配置接口。Lua 是一个小巧的脚本语言, 很容易嵌入其它语言。轻量级 LUA 语言的官方版本只包括一个精简的核心和最基本的库。这使得 LUA 体积小、启动速度快, 从而适合嵌入在别的程序里。UCI 是 OpenWrt 中为实现所有系统配置的一个统一接口, 英文名 Unified Configuration Interface, 即统一配置接口。LuCI, 即是这两个项目的合体, 可以实现路由的网页配置界面。

最初开发这个项目的原因是没有一个应用于嵌入式的免费, 干净, 可扩展以及维护简单的网页用户界面接口。大部分相似的配置接口太依赖于大量的 Shell 脚本语言的应用, 但是 LuCi 使用的是 Lua 编程语言, 并将接口分为逻辑部分, 如模板和视图。LuCI 使用的是面向对象的库和模板, 确保了高效的执行, 轻量的安装体积, 更快的执行速度以及最重要的是有更好的可维护性。与此同时, LuCI 从 MVC-Webframework 衍生出一个包含了很多库、程序以及 Lua 程序用户接口的集合, 但是 LuCI 仍然专注于实现网页用户界面并从 OpenWrt 'Kamikaze' 8.09.开始成为 OpenWrt Kamikaze 官方的一份子。

## 2 Lua 简介

Lua 是一种脚本语言, 是一种可扩展的语言。自从诞生时起, Lua 就得到了很好的发展, 广泛应用于工业领域, 比如机器人技术、图象处理、化学等等。

Lua 可以对、函面向对象语言数式程序设计提供很好的支持。它的目标是可以用作一种强大的、轻型的配置语言。Lua 目前已经被实现为一个扩展库，是用 ANSI C/C++编写的。作为一种扩展语言，Lua 没有"main"函数的概念，它仅仅是嵌入一个宿主程序进行工作，可以称之为嵌入式编程或者简单的说是宿主编程。这个宿主程序可以调用函数来执行 Lua 的代码片断，可以设置和读取 Lua 的变量，可以注册 C 函数让 Lua 代码调用。Lua 程序可以是文本格式的脚本语言，有解析器解析执行，也可以编译成二进制代码后再用解析器执行，也可以用第三方工具打包成独立程序运行，也可以编译成 C 语言用 include 调用，使用非常广泛。

### 3 CGI 介绍

CGI(Common Gateway Interface)即公共网关接口，是 WWW 技术中最重要的技术之一，有着不可替代的重要地位。CGI 是外部应用程序（CGI 程序）与 Web 服务器之间的接口标准，是在 CGI 程序和 Web 服务器之间传递信息的规程。CGI 规范允许 Web 服务器执行外部程序，并将它们的输出发送给 Web 浏览器。LuCI 就是一个 CGI 程序，工作过程如下：

- 1.浏览器通过 HTML 表单或超链接请求指上一个 CGI 应用程序的 URL。
- 2.web 服务器收发到请求并执行指定所 CGI 应用程序。
- 3.CGI 应用程序执行所需要的操作，通常是基于浏览者输入的内容。
- 4.CGI 应用程序把结果格式化为网络服务器和浏览器能够理解的文档（通常是 HTML 网页）。
- 5.网络服务器把结果返回到浏览器中。

### 4 uHTTPd 简介

uHTTPd 是 OpenWrt 的/LUCI 开发者从头开始编写一个 Web 服务器。它的目的是实现一个高效，稳定的服务器，适用于轻量级的任务与嵌入式设备，适合集成到 OpenWrt 的 web 配置框架（UCI）。特别是，它是 luci 默认的 Web 服务器，用于通过 Web 界面配置 OpenWrt。此外，它具有现今的 Web 服务器所需要的功能。默认情况下，根链接页面是/www/index.html，/www/index.html 会重定向到/cgi-bin/luci，这样配置可以使 LuCI 通过嵌入到/luci 的解析器执行。/cgi-bin/luci 是 LuCI 默认的 CGI 入口，/cgi-bin/是由 uHTTPd 配置文件 /etc/config/uhttpd 中的 cgi\_prefix 变量决定，用于存放 CGI 程序，CGI 程序放到其他路径是不会被 web 服务器执行的。如果访问根链接要重定向到其他页面，或者不需要重定向，就要刷新浏览器的缓存，否则它可能会保持重定向到/cgi-bin/luci 直到浏览器缓存过期，还有一点，uHTTPd 与浏览器建立的 TCP/IP 连接是长连接，可能会是浏览器保持重定向。

index.html 如下

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="refresh" content="0; URL=/cgi-bin/luci" />
</head>
<body style="background-color: black">
<a style="color: white; text-decoration: none" href="/cgi-bin/luci">LuCI - Lua Configuration Interface</a>
</body>
</html>
```

luci 的代码如下：

```
#!/usr/bin/lua-- 执行命令的路径
require"luci.cacheloader"-- 导入 cacheloader 包
require"luci.sgi.cgi"--导入 sgi.cgi 包
luci.dispatcher.indexcache="/tmp/luci-indexcache"--cache 缓存路径地址
luci.sgi.cgi.run() -- 执行 run 方法，此方法位于/usr/lib/lua/luci/sgi/cgi.lua 中
```

## 5 LuCI 环境搭建

首先搭建运行环境，这里只讲在 ubuntu 下搭建运行环境，也可以直接在跑 openWRT 系统路由器上运行。下面是在 ubuntu 下建立 LuCI 运行环境

```
sudo apt-get install lua5.1
sudo apt-get install lua5.1-policy-dev
cd ~
mkdir openwrt
cd openwrt
svn co http://svn.luci.subsignal.org/luci/branches/luci-0.11/
cd luci-0.11
make
make runuhttpd
```

浏览器输入

```
http://127.0.0.1:8080/cgi-bin/luci/
```

这样就能看到效果了，写好的 lua 文件直接放到 host/usr/lib/lua/luci/的正确位置上马上就能在浏览器中看到效果，如果修改过里面的文件要删除/tmp/下与 luci 有关的文件和目录才能看到修改后的效果。可以使用一下两条命令：

```
rm /tmp/luci-*
rm -r /tmp/luci-*
```

编辑器很多都可以，可以直接用 gedit，eclipse 的 lua 专用版也可以，能够支持 lua 语法高亮就行了。

## 6 Luci 框架

根据官方描述，LuCI 采用的是 MVC 框架，MVC 是一种使用 MVC（Model View Controller 模型-视图-控制器）设计创建 Web 应用程序的模式，在 luc 目录里面可以看到有 mode、view 和 controller 这三个目录，我们可发主要是针对这三个目录进行的。

### 6.1 controller介绍

在 controller 目录下，每个.lua 文件中，都有一个 index()函数，Createtree()函数就是要找到 controller 目录下所有的.lua 文件，并找到其中的 index()函数执行，从而生成一个 node-tree。这样做的 io 操作太多，为了效率，第一次执行后，把生成的 node-tree 放在 treecache 文件中，treecache 文件放在/tmp 目录里，以后只要没有更新（一般情况下，服务器里的.lua 文件是不会变的），直接读该文件即可，lua 文件更新了就删除/tmp/下与 luci 有关的文件和目录。

Index()函数主要调用 entry()函数，形如 entry(path,target,title,order)，entry()函数根据这些创建一个 node（根节点为/cgi-bin/luci），并把它放在全局 node-tree 的相应位置，后面的参数都是该 node 的属性，还可以有其他的参数。参数说明如下：

path 形如{admin,network,wireless}，是 lua 的 table，描述的是节点的访问路径（URL）。

target 描述了用户访问 path 相应节点时的行为。

title 定义显示在菜单中目录的 title。

order 定义了同层次菜单中的显示顺序。

其中最重要的就是 target, target 主要有: alise、firstchild、call、cbi、form、template。这几个总体上可以分成两类, 前两种主要用于链接其它 node, 后一个则是主要的操作、以及页面生成。下面分别描述。

alise(node)链接到指定的节点, firstchild()链接到它的第一个子节点, 下面举个例子说明, 在浏览器输入 <http://192.168.1.1>, 浏览器获取到 [www/index.html](http://www/index.html), index.html 自动链接到 [cig-bin/luci/](http://cig-bin/luci/), 这应该会索引到 rootnode 节点。而该节点本身是没有内容显示的, 所以它用 alias('admin')方法, 自动链接到 admin 节点, 即 [cig-bin/luci/admin/](http://cig-bin/luci/admin/)。再比如, admin 节点本身也没有内容显示, 它用 firstchild()方法, 自动链接到它的第一个子节点 entry({"admin", "services"}, firstchild(), \_("Services"), 40), 即 [admin/services](http://admin/services), 它页面内容是它的第一个节点的内容, 第一个节点是 entry({"servicectl"}, alias("servicectl", "status")).sysauth="root", 即 [admin/services/status](http://admin/services/status), 所以访问 <http://192.168.1.1> 最终显示的是<http://192.168.1.1/cig-bin/luci/admin/services/status>页面的内容。

Call()函数会调用 controller 里的函数(一般是当前 lua 文件里的 action\_function), 主要通过 openwrt 系统的 uci、network、inconfig 等工具对系统进行设置, 如果需要还会生成新界面。cbi()和 form()对应到 [./host/usr/lib/lua/luci/](#)相应的目录下的 html 和 lua 文件, lua 一般是描述 UCI 配置文件的结构, LUCI 负责生成, 解析 XHTML 表单以及读写 UCI, 它们利用 model 中定义的模板 map, 生成 html 文件; template()利用 view 中定义的 htm (一种类似 html 的文件且可以嵌入 lua 语句), 直接生成界面。template()、cbi()和 form()可以动态生成界面的。

## 6.2 Mode模块编写

这里主要介绍 cbi 模块, cbi 模块几乎都是利用 map 模版生成 html, cbi 模块的内容会被 luci.cbi 和 luci.il8n 的 translate 函数自动扩展成 html。下面介绍 cbi 模块的几个常用的类:

class Map (config, title, description)

cbi:模块的根类

**config:** /etc/config 里面的配置文件名, 必需存在, 若没有匹配的配置文件会出错。

**title:**在页面显示的标题

**description:**在页面显示的描述

:section (sectionclass, ...)

创建新的选项

**sectionclass:**选择对象所属的类

创建选择的类所需的参数

class TypedSection (type, title, description)

根据类型选择一组 UCI 选项的对象类

.addremove = false

允许用户删除或增加选项

.anonymous = false

显示选项的名称

class ListValue (option, title, description)

:value (key, value = key)

增加一个选择项

其他选项参考官方资料<http://luci.subsignal.org/trac/wiki/Documentation/CBI>, 下面通过例子说明:

controller

```
module("luci.controller.routerzte", package.seeall)
```

```
function index()
```

```
if nixio.fs.access("/etc/config/routerzte") then
```

```
local page
```

```

page = entry({"admin", "services", "routerzte"}, cbi("routerzte"), _("routerzte"), 10)
page.i18n = "routerzte"
page.dependent = true
end
end

```

```
mode/cbi
```

```
require("luci.tools.webadmin")
```

```
m = Map("routerzte", translate("routerzte"), translate("802.1x supplicant on Linux.."))
```

```
function m.on_commit(self)
```

```
os.execute("/etc/init.d/routerzte start")
```

```
end
```

```
s = m:section(TypedSection, "service", translate("Start option"), translate("Configure routerzte start option"))
```

```
s.addremove = false
```

```
s.anonymous = true
```

```
s:option(Flag, "enable", translate("802.1x-Client_enable"), translate("Enable or disable routerzte
```

")).default="0"

```
s:option(Flag, "boot", translate("Start at boot"), translate("Start routerzte when the route is
```

```
booting")).default="0"
```

```
s:option(Flag, "ipv6", translate("Enable IPv6"), translate("Enable or disable Napt66 Kernel")).default="0"
```

```
s = m:section(TypedSection, "routerzte", translate("Config routerzte "), translate("The options below are all of
802.1x-client's arguments."))
```

```
s.addremove = true
```

```
s.anonymous = true
```

```
s:option(Value, "username", translate("Username"))
```

```
pw=s:option(Value, "password", translate("Password"))
```

```
pw.password=true
```

```
nic=s:option(ListValue, "interface", translate("Port"))
```

```
nic.anonymous = true
```

```
for k, v in pairs(luci.sys.net.devices()) do
```

```
    nic:value(v)
```

```
end
```

```
return m
```

```
host/etc/config/routerzte
```

```
config 'service'
```

```
    option 'enable' '1'
```

```
    option 'boot' '1'
```

```
    option 'ipv6' '1'
```

```
config 'routerzte'
```

```
    option 'username' 'user'
```

```
    option 'interface' 'eth0'
```

```
    option 'password' '123'
```

产生的页面如下图:

# routerzte


802.1x supplicant on Linux..

## Start option

Configure routerzte start option


802.1x-Client\_enable

☒

 Enable or disable routerzte


Start at boot

☒

 Start routerzte when the route is booting

Enable IPv6

☒


 Enable or disable Napt66 Kernel

## Config routerzte


The options below are all of 802.1x-client's arguments.

用户名

密码



端口



添加

删除

图 1

### 6.3 view下面的html简介

这个是最好理解的例: passwd.htm

```
<%+header%>
<h2><a id="content" name="content"><%=system%></a></h2>
<h3><%=reboot%></h3>
<p><%=a_s_reboot1%></p>
<%-
local c = require("luci.model.uci").cursor():changes()
if c and next(c) then
-%>
<p class="warning"><%=a_s_reboot_u%></p>
<%-
end
if not reboot then
```

```
-%>
<p><a href="<%=controller%/admin/system/reboot?reboot=1"><%=a_s_reboot_do%></a></p>
<%- else -%>
<p><%=a_s_reboot_running%></p>
<script type="text/javascript">setTimeout("location='<%=controller%/admin'", 60000)</script>
<%- end -%>
<%=footer%>
```

<%=header%> <%=footer%> 加载公用的头部和尾部

<%= lua code-%>嵌入 lua 代码

<%=:i18n%>使用 i18n 处理显示

<%=lua 变量%>使用 lua 变量

<%=lua code%>嵌入 lua 代码

到 veiw 目录下看看里面的 htm 文件，那些就是很好的例子。

## 7 结语

Luci 可以实现很多功能，在官方页面可以查看其 API 接口，不认识的函数可以到官方页面查一下说明，同时也可以查看源代码，多看看里面的例子会学得更多。

## 参考文献：

[1] Mr. Roberto Ierusalimschy, Programming in Lua



潘楚文(1991-)广东海洋大学，信息学院，电气工程及其自动化 1112 班，兴趣方向为嵌入式系统开发，自动化控制。