

云水

MyGitee - <https://gitee.com/lsgx/>
MyGithub - <https://github.com/lsgxevea/>

博客园

首页

新随笔

联系

订阅

管理

随笔 - 1424 文章 - 0 评论 - 73

修改Luci界面

修改Luci界面

参考 <https://blog.csdn.net/hui523hui523hui523/article/details/38943693>

参考 <https://www.jianshu.com/p/bfb93c4e8dc9>

参考 https://blog.csdn.net/weixin_43883277/article/details/98725690

参考 https://blog.csdn.net/weixin_43883277/article/details/99677581

参考 https://blog.csdn.net/weixin_43883277/article/details/98505104

参考 https://blog.csdn.net/weixin_43883277/article/details/99865510

LuCI 基础

Controller 位于: /usr/lib/luas/luci/controller/ 下——定义模块的入口
Model 位于: /usr/lib/luas/luci/model/cbi/ 下——配置模块实际的代码

第一步：定义模块入口：

```
eg: module("luci.controller.控制器名/路径", package.seeall)function index()
```

```
entry(路径, 调用目标, _("显示名称"), 显示顺序)
end
```

控制器名/路径：

不带路径的控制器名默认存在于usr/lib/luas/luci/controller/下，否则以controller/为根目录
entry表示添加一个新的模块入口，官方给出了entry的定义，其中后两项都是可以空的：

```
entry(path, target, title=nil, order=nil)
```

path：

如果这样写{"click", "here", "now"}, 那么就可以在浏览器里访问<http://192.168.x.1/cgi-bin/luci/click/here/now>来访问这个脚本。我们也可以根据需要按如下方式编写{"admin", "一级菜单名", "菜单项名"}, 系统会自动在对应的菜单中生成菜单项。比如想在"网络"菜单下创建一个菜单项，那么一级菜单名可以写为"network"。

target：


- 调用目标分为三种，分别是执行指定方法Action、访问指定页面Views以及调用CBI Module
- 第一种可以直接调用指定的函数，比如点击菜单项就直接重启路由器等等，比如写为call("function_name"), 然后在lua文件下编写名为function_name的函数就可以调用了。
 - 第二种可以访问指定的页面，比如写为template("myapp/mymodule")就可以调用usr/lib/luas/luci/view/myapp/mymodule.htm文件了
 - 而如果要编写配置页面，那么使用第三种方法无非是最方便的，比如写为cbi("myapp/mymodule")就可以调用usr/lib/luas/luci/model/cbi/myapp/mymodule.lua文件了。


公告

昵称: lsgxevea
园龄: 5年1个月
粉丝: 160
关注: 0
[+加关注](#)

2020年11月						
日	一	二	三	四	五	六
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5
6	7	8	9	10	11	12

搜索





常用链接

[我的随笔](#)
[我的评论](#)
[我的参与](#)
[最新评论](#)
[我的标签](#)

随笔分类

a10(4)
AI(1)
algorithm(1)
Android(37)
ans(55)
bitcoin(4)
c++(103)
c++11(38)
ccna(11)
cmake(4)
cnns(18)
cocos(1)
CSharp(29)

```
title和order

module("luci.controller.LuoYeLuCI", package.seeall)

function index()
    entry({"admin", "network", "LuoYeconfig"}, cbi("LuoYeCBI"), _("LuoYeTest"), 100)
end
```

第二步：配置CBI Module

1. 首先要需要映射与存储文件的关系

```
m = Map("配置文件文件名", "配置页面标题", "配置页面说明")
```

第一个参数即为配置文件存储的文件名，不包含路径。
第二与第三个参数则是用在来页面上显示的

2. 接下来需要创建与配置文件中对应的Section

Section分为两种，NamedSection和TypedSection，前者根据配置文件中的Section名，而后者根据配置文件中的Section类型 <http://luci.subsignal.org/trac/wiki/Documentation/CBI>

3. 创建配置文件

文件需要存储在/etc/config（如果配置文件不存在的话，访问配置页面将会报错）
内容格式如下：

```
config login
    option username "
    option password "
    option ifname 'eth0'
    option domain "
```

LuCI 页面修改

简单的文件配置，路由上路径主要是/usr/lib/lua/luci/下子目录：/controller/、/model/cbi/、/view/，或者根目录下的/www/中。可以在路上修改查看效果。
如果想要编译自定义LuCI页面的固件，请尝试修改如下OpenWRT源码结构路径内的LuCI文件。

```
xxx/package/feeds/luci/luci/luci/libs/web/root/etc/config/luci --- AA 版本packages/feeds/luci/中
xxx\feeds\luci\luci\luci\libs\web\root\etc\config\luci --- AA 版本的feeds/luci文件夹中
xxx/feeds/luci/modules/base/root/etc/config/luci --- BB 版本中feeds中，bb版本open修改了luci配置文件
再者就是修改 xxx/dl/ 下的源码压缩包，或者 xxx/build_dir/$target/下源码
```

注意：如果在xxx/feeds 修改可能需要执行 ./scripts/feeds install luci 更新

主题Logo替换

源码路径：xxx/feeds/luci/luci/luci/themes/bootstrap/htdocs/luci-static/bootstrap/logo.jpg
路由路径：/www/luci-static/bootstrap/logo.jpg

Tips：
由于版本的更新，文件路径可能变更，此处列出的为BB版本的例子。如果找不到可以用命令手动在/feeds/中查找：find ./ -name logo.jpg。此处为bootstrap主题Logo，其他主题的Logo修改类似。

页面脚标信息

源码路径：xxx/feeds/luci/luci/luci/themes/bootstrap/luasrc/view/themes/bootstrap/footer.htm
路由路径：/usr/lib/lua/luci/view/themes/bootstrap/footer.htm
修改位置：

DataSeture(6)
DesignPattern(39)
ditel(39)
done(2)
eve-ng(4)
exam(6)
FlightController(1)
FortiGate(14)
fpga(2)
gcc(14)
gnss(53)
Hacker(28)
illustration(6)
ios(9)
Java(46)
Juila(1)
Linux(167)
lua(32)
nasm(1)
niaoyun(12)
nodejs(58)
nodemcu(3)
ns(183)
openwrt(33)
powershell(25)
Python(6)
QT(119)
ruby(1)
Rust(7)
sangforAD(2)
Scheme(116)
SCORM(10)
scratch(2)
SICP(1)
solidworks(3)
STL(16)
stm32(1)
taobao(6)
TCP/IP(9)
unity3d(6)
weixin(8)
win32(18)

随笔档案

2020年11月(39)
2020年10月(39)
2020年9月(90)
2020年8月(22)
2020年7月(29)
2020年6月(11)
2020年5月(50)
2020年4月(49)
2020年3月(39)
2020年2月(10)
2020年1月(18)
2019年12月(26)
2019年11月(30)
2019年10月(12)
2019年9月(40)
2019年8月(28)
2019年7月(50)
2019年6月(32)

```
Powered by <%= luci.__appname__ .. " (" .. luci.__version__ .. ") " %>
<%=luci.version.distversion%>
```

status状态栏信息

源码路径：xxx/feeds/luci/modules/admin-full/luasrc/view/admin_status/index.htm
路由路径： /usr/lib/ua/luci/view/admin_status/index.html
修改位置：找到类似的代码段修改.

```
<%=System%>

<%=Hostname%><%=luci.sys.hostname() or "?"%>
<%=Model%><%=pcdata(model or "?" )%>
<%=Firmware Version%>
<%=pcdata(luci.version.distname)%> <%=pcdata(luci.version.distversion)%> /
<%=pcdata(luci.version.luciname)%> (<%=pcdata(luci.version.luciversion)%>)

<%=Kernel Version%><%=luci.sys.exec("uname -r")%>
<%=Local Time%>-
<%=Uptime%>-
<%=Load Average%>
```

=====

在使用OpenWrt路由器的过程中，经常需要根据需要改改配置文件然后重新启动服务什么的，一般的做法是SSH登录路由器后台，使用vi编辑器修改文件，然后使用/etc/init.d/xxxx restart 来重启服务，次数多了就会觉得很繁琐，光SSH输入密码就够麻烦的，所以不妨自己写一个luci界面在路由器web后台修改配置文件并完成重启这一系列操作。下面以Pdnssd为例进行介绍

首先看一下实现效果

- 2019年5月(19)
- 2019年4月(24)
- 2019年3月(33)
- 2019年2月(19)
- 2019年1月(43)
- 2018年12月(49)
- 2018年11月(30)
- 2018年10月(16)
- 2018年9月(21)
- 2018年8月(36)
- 2018年7月(42)
- 2018年6月(35)
- 2018年5月(15)
- 2018年4月(41)
- 2018年3月(33)
- 2018年2月(25)
- 2018年1月(32)
- 2017年12月(89)
- 2017年11月(129)
- 2017年10月(72)
- 2017年9月(7)

最新评论

- 1. Re:TPFanControl.ini
学习了
--吃了3碗还是不够哇
- 2. Re:Qt编写自定义控件插件路过的坑及注意事项
看了半天，最后一句才是重点，哈哈
--FisherCloud
- 3. Re:Git Submodule使用完整教程
好文感谢，看完基本submodule就搞懂了。就是有一点，全是黑色的字体，而且很多冗余的信息，本来扫一眼就知道怎么回事的东西要慢慢看半天。
--mungsoup
- 4. Re:uboot命令行界面中刷入固件
你那个波特率可能写错了，正常Linux调试默认为115200
--Orson_Yan
- 5. Re:Qt基本控件及三大布局
总结的不错
--wang444455555

阅读排行榜

- 1. Docker 创建镜像、修改、上传镜像(143404)
- 2. Git Submodule使用完整教程(75337)
- 3. git diff命令详解(48468)
- 4. Qt基本控件及三大布局(45021)
- 5. IPV6地址格式分析(41833)

评论排行榜

- 1. Git Submodule使用完整教程(7)
- 2. babel从入门到入门(6)
- 3. libuv 简单使用(5)
- 4. MIT Scheme 的基本使用(4)
- 5. react 入门与进阶教程(3)

推荐排行榜

OpenWrt

状态 ▾ 系统 ▾ 服务 ▾ 网络 ▾ 退出

Pdn

nsd

Pdn

nsd

可以实现类似ChinaDNS的效果，通过TCP协议进行DNS解析可以有效避免DNS污染，默认上游服务器为114DNS，可以在【DHCP/DNS】中设置【DNS转发】为【127.0.0.1#5053】即可将所有DNS请求交给pdnsd进行解析，由于pdnsd自带缓存，所以很快哦。注意SSR如果你勾选了TCP解析DNS也会开启一个pdnsd监听7453接口，并与该页面的pdnsd冲突，点击【保存/应用】可以重启pdnsd，另外你可以使用dig来检测DNS解析情况，方法dig @127.0.0.1 -p 5053 www.facebook.com

启用 ☒

```
global {
    perm_cache=512;      # dns缓存大小，单位KB，建议不要写的太大
    cache_dir="/var/pdnsd"; # 缓存文件的位置
    server_ip = 0.0.0.0;  # pdnsd监听的网卡，0.0.0.0是全部网卡
    server_port=5053;     # pdnsd监听的端口，不要和别的服务冲突即可
    status_cli = on;
    paranoid=on;          # 二次请求模式，如果请求主DNS服务器返回的是垃圾地址，就向备用服务器请求
    query_method=tcp_only; # 请求模式，推荐使用仅TCP模式，UDP模式一般需要二次请求
    neg_domain_pol = off;
    par_queries = 400;     # 最多同时请求数
    min_ttl = 1d;          # DNS结果最短缓存时间
    max_ttl = 1w;          # DNS结果最长缓存时间
    timeout = 10;          # DNS请求超时时间，单位秒
}

/* 上游DNS服务器的配置 */
server {
    label = "routine";    # 这个随便写
    ip = 114.114.114.114; # 这里为主要上级 dns 的 ip 地址，建议填写一个当地最快的DNS地址
    timeout = 5;          # DNS请求超时时间
    reject = 74.125.127.102; # 一下是脏IP，也就是DNS污染一般会返回的结果，如果收到如下DNS结果会触发二次请求（TCP协议一般不会碰到脏IP）
    74.125.155.102,
    74.125.39.102,
    74.125.39.113,
    209.85.229.138,
    128.121.126.139,
    159.106.121.75,
    169.132.13.103,
    192.67.198.6,
    202.106.1.2,
    202.181.7.85,
    203.161.230.171,
    203.98.7.65,
    207.12.88.98,
    208.56.31.43,
    209.145.54.50,
    209.220.30.174,
    209.36.73.33,
    211.94.66.147,
    213.169.251.35,
    216.221.188.182,
    216.234.179.13,
    243.185.187.39,
```

保存&应用

保存

复位

Powered by

LuCI (git-16.018.33482-3201903)

/

OpenWrt Chaos Calmer 15.05.1

- 1. Git Submodule使用完整教程(10)
- 2. javaee, javaweb和javase的区别以及各自的知识体系(6)
- 3. c++11 智能指针 unique_ptr、shared_ptr与weak_ptr(6)
- 4. c++11 类默认函数的控制：“=default”和“=delete”函数(6)
- 5. react 入门与进阶教程(4)

要实现一个luci界面至少需要如下三个文件

- A: /etc/config/pdnsd
- B : /usr/lib/lua/luci/controller/pdnsd.lua
- C: /usr/lib/lua/luci/model/cbi/pdnsd.lua

其中

A的作用是存储你在luci界面上的控件中填入的数值，比如用户名密码，是否可用，选择的选项等等，是一个uci配置文件，可以使用shell的uci命令进行读取

B的作用是通告OP系统你的自定义界面的显示位置，相邻排序，并指向C文件

C是最核心的文件，里面记录了你luci页面的控件布局，控件的显示内容，和控件触发事件的执行脚本

在创建每个文件之前，首先要了解一个概念，就是luci是MVC架构的，而且MVC的理念在luci上得到了最充分最明显的阐释，是理解MVC架构非常形象生动的例子，下面我就简单介绍一下luci的MVC架构。

首先请看下面的两个截图，这是两个不同OP的路由器安装同一个luci ipk安装包之后的界面，界面风格差别很大，不知道的还以为这是两个不同的软件。但其实不是，因为他们来源于同一个ipk安装包，不仅代码一模一样，里面控件的类型和数量也是一致的。而且执行效果也是一样的。

状态系统服务网络退出

D: OKBU: OK

影梭ChinaDNS

基本设置服务器管理访问控制

影梭 - 基本设置

运行状态

透明代理	运行中
SOCKS5代理	运行中
端口转发	运行中

透明代理

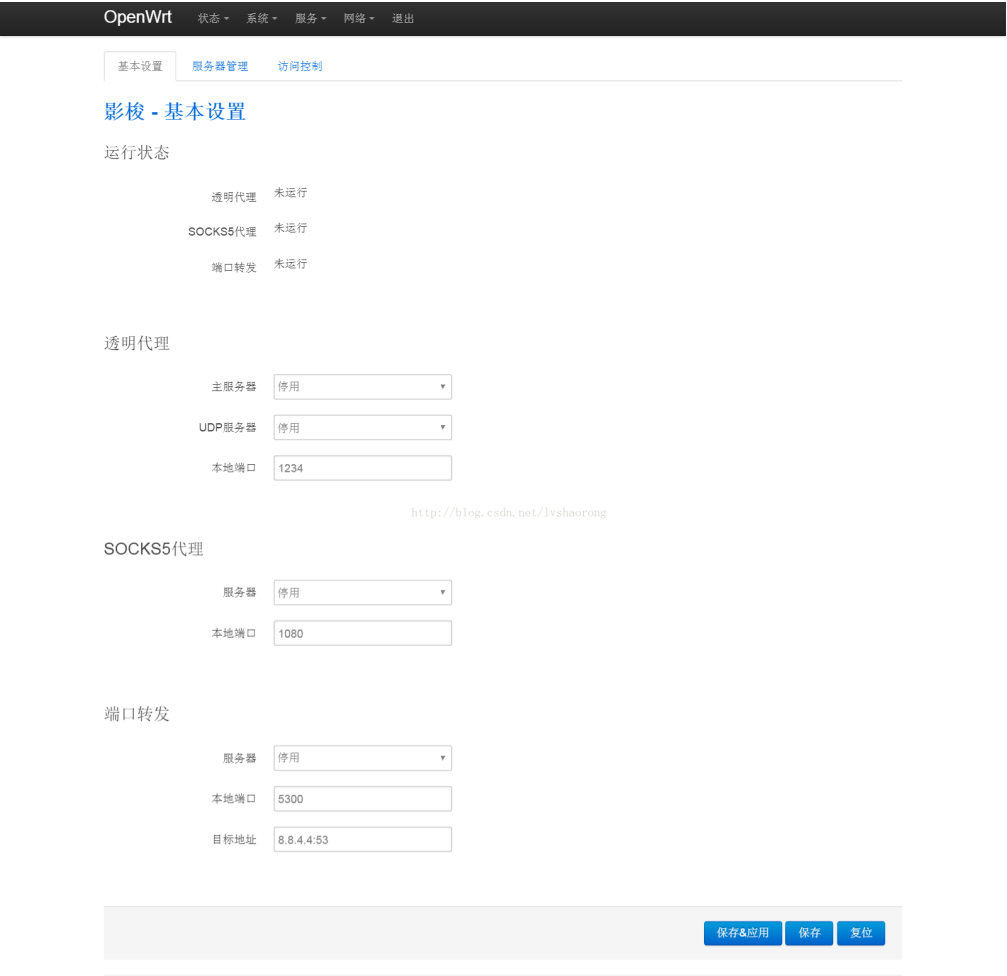
主服务器	DOVPS1_443
UDP服务器	http://blog.csdn.net/qq_34446317 DOVPS1_1380_UDP
本地端口	1080

SOCKS5代理

服务器	DOVPS1_443
本地端口	1390

端口转发

服务器	DOVPS1_1390_UDP
本地端口	5300
目标地址	8.8.4.4:53



这上面左面应该是潘多拉固件或者DreamBox等等固件的截图，而右面是OP默认界面的原图，右面的加入了一些bootstrap样式。

那为什么同样的luci代码会产生如此风格迥异的界面，并且自动的适配路由器当前固件的风格呢，这就是luci使用lua脚本使用MVC架构的奥义了。同时我们也知道，luci的开发不是像网页一样使用html，css,js进行开发，而是使用lua脚本和uci接口，下面就来仔细介绍一下luci的简单开发过程。

首先我们创建上面必需的三个文件，做一个简单的界面

```
/etc/config/pdnsd
```

```
config arguments
```

由于这个文件是记录你配置好的参数的，由于我们现在并不需要记录什么参数，所以写个uci的开头就够，其中“arguments”是一个固定标志

```
/usr/lib/lua/luci/controller/pdnsd.lua
```

```
module("luci.controller.pdnsd", package.seeall)
function index()
    if not nixio.fs.access("/etc/config/pdnsd") then
        return
    end
```

```
entry({"admin", "services", "pdnsd"}, cbi("pdnsd"), _("Pdnsd")).dependent = true
end
```

这个文件的含义是：首先检查有没有/etc/config/pdnsd文件，如果有就继续向下执行，没有就当什么都没有发生过

第一行中，固定格式是“luci.controller.我的项目名”，因为我们所有的lua文件和/etc/config下面的配置文件都是以pdnsd来进行命名的，所以应该填“luci.controller.pdnsd”

倒数第二行是一个固定格式

entry(路径, 调用目标, _("显示名称"), 显示顺序)

路径：{"admin", "services", "pdnsd"} 意思是登录用户可见，“服务”主菜单下，“名字叫pdnsd的配置文件”，如果你想要放到“系统”菜单下，就把“services”改成“system”

调用目标：指向/usr/lib/lua/luci/model/cbi/pdnsd.lua这个控制文件

显示名称：显示为“Pdnsd”，当然这里也可以填UTF-8的中文，用来在主菜单上显示

显示顺序：这里没填，使用系统默认的显示顺序

这个文件规定的规则显示如下，就是下图中的“Pdnsd”选项



/usr/lib/lua/luci/model/cbi/pdnsd.lua

```
local fs = require "nixio.fs"
m=Map("pdnsd", translate("Pdnsd"), translate("Pdnsd可以实现类似ChinaDNS的效果，通过TCP协议进行DNS解#
s=m:section(TypedSection, "arguments", "")
s.addremove=false
s.anonymous=true
return m
```

其中，第1行是引用依赖库，这个依赖库是用来读取和写入文件的，目前不写这一行也可以，但是要实现读写配置文件的功能，这个库是必不可少的

第2行是固定格式

m = Map("配置文件文件名", "配置页面标题", "配置页面说明")

第一个参数:上一步我们新建配置文件/etc/config/pdnsd.这里就是建立与配置文件的联系.

第二个参数: 主标题.

第三个参数: 副标题，注意不能出现双引号""，否则会报错，中文要使用UTF-8编码

第3行:

m:section(类型, "arguments", "")

在一个配置文件中可能有很多Section,所以我们需要创建与配置文件中我们想要的Section的联系. "arguments"就是section的uci名字，就是我们在/etc/config/pdnsd中写的那个config arguments

有两种方式可以选择:NamedSection(name,type,title,description)和TypedSection(type,title,description),前者根据配置文件中的Section名, 而后者根据配置文件中的Section类型.我们选用了第二种.

第4行: 设定不允许增加或删除Section, 这样能保证页面的清爽, 不然会多出一些奇奇怪怪的控件

第5行: 设定是否显示Section的名称, 建议为true

这里顺带提一句, lua脚本中使用--来作为注释的标记, 而不是常见的//或者#; 另外luci可以显示中文, 但是你在编辑lua文件的时候必须指定“UTF-8”编码, 否则出来的中文是乱码

把这三个文件通过winscp上传到路由器对应的目录上, 无需重启路由器, 刷新一下路由器后台就可以看到效果, 如下图



这样一个简单的页面就出来了, 只有主标题和副标题, 没有任何其他的控件

下面我们添加一个简单的checkbox, 用来控制是否让配置立即生效, 只需修改C文件

```
--Alex<1886090@gmail.com>
local fs = require "nixio.fs"
m=Map("pdnsd", translate("Pdnsl"), translate("Pdnsl可以实现类似ChinaDNS的效果, 通过TCP协议进行DNS解
s=m:section(TypedSection, "arguments", "")
s.addremove=false
s.anonymous=true
view_enable = s:option(Flag, "enabled", translate("Enable"))
return m
```

仅添加了一行

```
view_enable = s:option(Flag,"enabled",translate("Enable"))
```

Flag是checkbox控件的意思, 如果改成Value就是文本框, 相当于html中的input标签

“enabled”是/etc/config/pdnsd这个uci文件的字段名, 所以我们可以顺带在这个文件里加入一个默认值, 如下:

```
config arguments
    option enabled '1'
```

0就是默认不选, 也就是进入luci界面之后checkbox不勾选, 1就是默认勾选

translate (“Enable”) 就是checkbox前面的提示文字, 如果你安装了luci-i18n等ipk, 会自动翻译成你选择的语言, 目前实现的效果如下

OpenWrt 状态 系统 服务 网络 退出

Pdnsd

Pdnsd可以实现类似ChinaDNS的效果，通过TCP协议进行DNS解析可以有效避免DNS污染，默认上游服务器为114DNS，可以在【DHCP/DNS】中设置【DNS转发】为【127.0.0.1#5053】即可将所有DNS请求交给pdnsd进行解析，由于pdnsd自带缓存，所以很快哦。注意SSR如果你勾选了TCP解析DNS也会开启一个pdnsd监听7453接口，并与该页面的pdnsd冲突，点击【保存/应用】可以重启pdnsd。另外你可以使用dig来检测DNS解析情况，方法dig @127.0.0.1 -p 5053 www.facebook.com

☒ 启用
 http://blog.csdn.net/lvshaorong

Powered by LuCI (git-16.018.33482-3201903) / OpenWrt Chaos Calmer 15.05.1

页面中果然多了一个“启用”checkbox，此时如果你点击“保存&应用”就会把该checkbox的状态记录到/etc/config/pdnsd文件中，如果勾选就是1，不勾选为0或者不显示该项

下面再来添加一个大文本框并显示我们要修改的文本文件。

```
--Alex<1886090@gmail.com>
local fs = require "nixio.fs"
m=Map("pdnsd", translate("Pdnsd"), translate("Pdnsd可以实现类似ChinaDNS的效果，通过TCP协议进行DNS解
s=m:section(TypedSection, "arguments", "")
s.addremove=false
s.anonymous=true
view_enable = s:option(Flag, "enabled", translate("Enable"))
view_cfg = s:option(TextView, "1", nil)
    view_cfg.rmemory = false
    view_cfg.rows = 43

    function view_cfg.cfgvalue()
        return nixio.fs.readfile("/etc/pdnsd.conf") or ""
    end
return m
```

上面的代码中TextView就是大文本框的意思，相当于html中的text标签，

.rmemory = false是设置该文本框不许留空，如果为空在“保存&应用”中会出现相应的提示

.rows = 43是这个大文本框的高度为43行，如果内容超出会出现滚动条

要实现进入该界面的时候该文本框自动显示/etc/pdnsd.conf这个配置文件的内容，需要重写大文本框自带的.cfgvalue()方法，这样浏览器进入pdnsd页面之后luci后台就会自动调用该控件的.cfgvalue()方法，把文件内容呈现在浏览器上。

nixio.fs.readfile("文件路径")会读取一个文件的全部内容，返回值就是该文件的内容 or ""意思是如果前面那句报错，就输出空字符串

现在已经实现了文件的读取功能，那么如何实现文件的写入功能呢？我们需要实现TextView控件的另一个自带方法，如下

```
local fs = require "nixio.fs"

function sync_value_to_file(value, file)
    value = value:gsub("\r\n?", "\n")
```

```

local old_value = nixio.fs.readfile(file)
if value ~= old_value then
    nixio.fs.writefile(file, value)
end

end

m=Map("pdnsd", translate("Pdnsd"), translate("Pdnsd可以实现类似ChinaDNS的效果，通过TCP协议进行DNS解析"))
s=m:section(TypedSection, "arguments", "")
s.addremove=false
s.anonymous=true

view_enable = s:option(Flag, "enabled", translate("Enable"))
view_cfg = s:option(TextValue, "1", nil)
view_cfg.rmempty = false
view_cfg.rows = 43

function view_cfg.cfgvalue()
    return nixio.fs.readfile("/etc/pdnsd.conf") or ""
end
function view_cfg.write(self, section, value)
    sync_value_to_file(value, "/etc/pdnsd.conf")
end

return m

```

上面代码中又重写了.write(self,section,value)这个方法，其中value这个参数就是当前TextValue这个控件中显示的文字，于是我又写了一个函数 sync_value_to_file(文本，文件路径)方法来将控件中的内容写入到文件中去。

而那个信函数主要作用就是先把windows中的换行符“\r\n”换成linux的换行符“\n”

然后检查一下新旧文本有没有发生变化，如果发生变化才调用库函数nixio.fs.writefile(file, value)将文本写入到文件中。

截至目前，我们已经实现了使用luci页面读写配置文件的功能了。

可是我们还想在配置文件发生修改之后自动重启后台的服务，应该怎么做呢，首先我们记得我们上面的checkbox记录了启用与否，并记在了/etc/config/pdnsd这个文件里。那么我们可以根据这个文件记录的内容，如果是1就restart pdnsd这个服务并enable，如果是0就stop这个服务并disable。那么我们先写一个shell脚本来读取uci文件的记录值并完成这一系列动作，我新建了一个/etc/pdnsd_init.sh 并添加执行权限，内容如下

```

echo luci for pdnsd
local vt_enabled=`uci get pdnsd.@arguments[0].enabled 2>/dev/null`
echo $vt_enabled
if [ "$vt_enabled" = 1 ]; then
    logger -t alex restarting pdnsd
    echo "restarting pdnsd"
    /etc/init.d/pdnsd enable
    /etc/init.d/pdnsd restart
else
    logger -t alex stopping pdnsd
    echo "stopping pdnsd"
    /etc/init.d/pdnsd disable
    /etc/init.d/pdnsd stop
fi

```

一个很普通的shell脚本，其中echo语句只有使用控制台执行的时候才输出，而logger语句会输出到OpenWrt主界面【状态】-【系统日志】中去，方便记录log和分析bug

上面的uci get pdnsd.@arguments[0].enabled 就是获取uci配置文件中enabled这个字段的值，因为一个config可以有多个section，每个section下面都可以有一个叫enabled的变量，所以要填写arguments[0]。比如像如下这种uci文件，就会有多个section（uci和lua中nil就是null的意思）

```
config transparent_proxy
    option main_server 'nil'
    option udp_relay_server 'nil'
    option local_port '1234'

config socks5_proxy
    option server 'nil'
    option local_port '1080'

config port_forward
    option server 'nil'
    option local_port '5300'
    option destination '8.8.4.4:53'
```

然后使用终端执行/etc/pdnsd_init.sh系统就会根据/etc/config/pdnsd中记录的值来判断是启动还是停止pdnsd服务了，下面我们要把这个shell文件放到luci中执行，我的方法是放在之前写的文件写入方法中直接执行，这样应该是不太对的方法，不过由于我刚刚接触luci，也没发现别的方法，如果有大神发现可以告诉我。

```
function sync_value_to_file(value, file)
    value = value:gsub("\r\n?", "\n")
    local old_value = nixio.fs.readfile(file)
    if value ~= old_value then
        nixio.fs.writefile(file, value)
    end
    os.execute("/etc/pdnsd_init.sh >/dev/null")
end
```

其中，os.execute("shell命令")就可以执行我们自定义的代码，然后点击“保存&应用”luci页面会转圈圈，提示“正在应用更改...”“然后”配置已应用“，其中“正在应用更改...”取决于上面自定义shell脚本的运行时间，我试过如果在shell脚本中添加一句“sleep 10s”会让“正在应用更改...”的时间明显延长。

那么为什么要加上>/dev/null这个呢？其实非常必要，因为luci需要一个规定格式的输出，如果不写这句，你点击“保存&应用”之后就会出现白屏错误，然后就没有下文了，如下



所以必须把我们自己shell脚本的输出全部丢弃，只需添加>/dev/null就好，注意2>/dev/null是没有用的。至此，我们的通过luci修改配置文件并控制启动、停止的功能就全部完成了。

参考文章：<http://www.right.com.cn/forum/thread-183560-1-1.html>

http://blog.csdn.net/icy_river/article/details/48179649


===== End

分类: openwrt

好文要顶

关注我

收藏该文



[lsgxeva](#)

[关注 - 0](#)


[粉丝 - 160](#)

+加关注

« 上一篇: [开发环境构建](#)
» 下一篇: [SNMP OID Search](#)

posted @ 2020-09-24 21:34 lsgxeva 阅读(107) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

 **登录后才能发表评论, 立即 [登录](#) 或 [注册](#), [访问网站首页](#)**
博客园派送云上免费午餐, AWS注册立享12个月免费套餐