宝塔 Linux 面板插件开发文档

插件安装位置：/www/server/panel/plugin/
文件结构：
demo_main.py　插件后端主程序(插件名称_main.py)
index.html　插件前端程序
info.json 插件信息文件
icon.png 插件图标文件
install.sh 安装卸载脚本

插件名称_main.py 内容格式
以插件名称为 demo 示例：

```
#!/usr/bin/python
# coding: utf-8
class demo_main: #必需与文件名一致
    #你的代码
    def get_list(self,args):   #如果此方法需要被前端调用，则通过 args 接收前端传过来的 POST/GET 参数
        pass;
```

info.json 内容格式：

```
{
  "title": "宝塔插件 DEMO",    //标题名称
  "name": "demo",              //名称,必需和插件目录名一致
  "ps": "宝塔插件开发示例项目",  //描述
  "versions": "1.0",           //版本号
  "checks": "/www/server/panel/plugin/demo",  //用于检测插件是否安装的文件或目录
  "author":"宝塔",      //作者
  "home":"https://www.bt.cn"    //作者主页
}
```

响应数据到前端：
允许 return 到前端的数据类型： bool、str、list、dict、int
不管响应什么类型的数据，都将自动转为 json 数据格式，若不被 json 对象支持，会直接报错

Demo 下载：https://www.bt.cn/demo.zip

注意：
1、URL 是严格大小写的
2、前端请求插件请统一使用 POST 方式
3、更详细的说明请参考 demo
4、在 POST 参数中请不要使用 action、s、name、a 这 4 个参数名

public 公共库

引用：import public

读取文件

```
#@param filename 文件名[必传]
#@param mode 文件打开模式 默认 r
#@return bool
f_body = public.ReadFile(filename,mode='r')
```

写入文件

```
#@param filename 文件名[必传]
#@param f_body 要写入的内容[必传]
#@param mode 文件打开模式 默认 w+
#@return bool
result = public.WriteFile(filename,f_body,mode='w+')
```

计算字符串的 MD5

```
#@param strings 要被计算的字符串[必传]
#@return string 小写的 MD5 结果
md5 = public.md5(strings)
```

计算文件的 MD5

```
#@param filename 文件名
#@return string 小写的 MD5 结果
file_md5 = public.FileMd5(filename)
```

写面板日志

```
#@param log_type 日志类型[必传] 如：登录面板
#@param log_body 日志内容[必传] 如：登录成功
#@return None
public.WriteLog(log_type,log_body)
```

使用 GET 方式请求 HTTP

```
#@param url URL 地址[必传]
#@param timeout 超时时间 默认 60 秒
#@return string 成功返回 http 响应内容，失败返回错误代码
http_body = public.HttpGet(url,timeout=60)
```

使用 POST 方式请求 HTTP

```
#@param url URL 地址[必传]
#@param p_data POST 内容,请传入字典(dict)[必传]
#@param timeout 超时时间 默认 60 秒
#@return string 成功返回 http 响应内容，失败返回错误代码
http_body = public.HttpPost(url,p_data,timeout=60)
```

取随机字符串

```
#@param length 要获取的随机字符串长度[必传]
#@return string
randmo_string = public.GetRandomString(length)
```

构造通用响应内容

```
#@param status 响应状态[必传] True|False
#@param return_msg 响应内容[必传]
return public.ReturnMsg(True,'操作成功!')
```

取指定文件的权限

```
#@param filename 文件名[必传]
#@return string 权限字符器 如 755
f_mode = public.GetFileMode(filename)
```

取当前使用的 web 服务器

```
#@return string 服务器类型 nginx|apache
web_server = public.GetWebServer()
```

重载当前 web 服务器

```
public.ServiceReload()
```

重载指定 PHP 版本

```
#@param version PHP 版本 如要重载 php7.2：72
public.phpReload()
```

通过管道执行 SHELL 命令

```
#@param shell_str 要被执行的 SHELL 命令[必传]
#@return list 命令执行结果 返回格式： ["正常输出","异常输出"]
result = public.ExecShell(shell_str)
```

获取本服务器 IP 地址

```
#@return string IP 地址
ip = public.GetLocalIp()
```

获取用户 IP 地址

```
#@return string IP 地址
client_ip = public.GetClientIp()
```

获取当前访问的 HOST 及端口信息

```
#@param port 返回端口[选传] True|False 默认为 False
host = public.GetHost()
port = public.GetHost(True)
```

取文件尾指定行数的内容

#@praam filename 文件名[必传]
#@param num 指定要取的行数[必传]
#@return string 获取到的内容
last_body = public.GetNumLines(filename,num)

字节单位转换(KB,MB,GB,TB)

#@param b_size 要被转换的字节数[必传]
#@return string 转换结果
size = public.to_size(b_size)

判断指定进程是否存在

#@param process_name 进程名[必传]
#@param exe 执行文件路径[选传]默认为 None 不判断，若传入值，验证执行文件路径
#@return bool
is_exists = public.process_exists(process_name,exe=None)

取服务器 mac 地址

mac_address = public.get_mac_address()

重启面板

public.restart_panel();

取当前格式化时间

#@param format 格式，默认为 %Y-%m-%d %X 2018-12-10 00:00:00
format_date = public.getDate(format='%Y-%m-%d %X')

#XSS 过滤

#@param data 要被过滤的字符串[必传]
input_body = public.checkInput(data)

获取面板自带数据库的 sqlite 数据库对象

#@param table 表名
#@return 数据库对象
db_obj = public.M('sites')

面板数据库对象方法列表

(注意：以下示例仅是数据库操作，实际添加、删除、修改请从前端调用对应接口完成操作)

获取站点 id 为 1 或 2 的网站列表

```
data = public.M('sites').where('id=? or id=?',(1,2)).field('id,name,edate,path,status').select()
```

获取所有网站列表并使用 id 降序

```
data = public.M('sites').field('id,name,edate,path,status').order('id desc').select()
```

获取符合条件的一条数据

```
find = public.M('sites').where('id=?',(id,)).field('id,name,edate,path,status').find()
```

通过网站名称获取网站 id

```
site_id = public.M('sites').where('name=?',('www.bt.cn',)).getField('id')
```

通过网站 ID 获取域名列表并使用 id 升序

```
domains = public.M('sites').where('pid=?',(site_id,)).order('id asc').field('id,pid,name,port,addtime').select()
```

删除日志

```
public.M('logs').where('id=?',(id,)).delete()
```

修改数据

```
public.M('logs').where(id=?,(id,)).save('type,log','登录','登录成功!')
```

执行 sql 语句返回受影响行

```
public.M('logs').execute(sql)
```

执行 sql 语句返回查询结果

```
public.M('logs').query(sql)
```

构造分页数据

```
#@param count 数据行数[必传]
#@param p 当前页 默认=1
#@param rows 每页行数 默认=12
#@param callback js 回调方法名称,如果您是通过 JS 调用分页数据，请传入 js 回调方法名称
#@param result 分页结构 默认='1,2,3,4,5,8',完整的是：'1,2,3,4,5,6,7,8'请根据需求调整
#@return {'page':构造好的颁布数据,'shift':偏移位置,'row':偏移量}
page_data = public.get_page(count,p=1,rows=12,callback='',result='1,2,3,4,5,8')
```

示例: 获取网站数据并构造分页

```
#取网站数量
count = public.M('sites').count()
取分页数据
page_data = public.get_page(count,p=1,rows=12,callback='get_sites_list',result='1,2,3,4,5,8')

#通过 page_data['shift'],page_data['ros']获取数据列表
site_list = public.M('sites').order('id desc')
.limit(page_data['shift'],page_data['ros']).field('id,name,edate,path,status').select()

#构造返回字典
data = {'data':site_list,'page':page_data['page']}
return data
```