LUA中热更新机制

1. 关于require机制

关于require路径，默认是LUA安装目录下，也可以自己添加搜索路径

package.path = 'D:/test/?.lua;' --搜索lua模块

先看一些概念性的东西

加载给定的模块.函数首先检查表package.loaded来判定modname是否已经存在.如果存在,则require返回package.loaded[modname]所存储的值否则它尝试为模块找到一个加载器(loader)

require的另一个功能是避免重复加载同一个文件两次。Lua保留一张所有已经加载的文件的列表（使用table保存）。如果一个加载的文件在表中存在require简单的返回；表中保留加载的文件的虚名，而不是实文件名。所以如果你使用不同的虚文件名require同一个文件两次，将会加载两次该文件。比如require "foo"和require "foo.lua"，路径为"?;?.lua"将会加载foo.lua两次。我们也可以通过[全局变量](http://baike.baidu.com/view/261041.htm)\_LOADED访问文件名列表，这样我们就可以判断文件是否被加载过；同样我们也可以使用一点小技巧让require加载一个文件两次。比如，require "foo"之后\_LOADED["foo"]将不为nil，我们可以将其赋值为nil，require "foo.lua"将会再次加载该文件。一个路径中的模式也可以不包含问号而只是一个固定的路径。

print("package.loaded ---------->",package.loaded["model"])

源码.

1. --require 函数的实现
2. function require(name)
3. if not package.loaded[name] then
4. local loader = findloader(name) //这一步演示在代码中以抽象函数findloader来表示
5. if loader == nil then
6. error("unable to load module" .. name)
7. end
8. package.loaded[name] = true
9. local res = loader(name)
10. if res ~= nil then
11. package.loaded[name] = res
12. end
13. end
14. return package.loaded[name]
15. end