Lua开发规范

更新时间：2017-10-25

1 介绍

本Lua开发规范客户端服务器端通用，尽量贴近C#/C++的编程习惯。

2 模块

2.1 基于对象的设计

Lua没有提供语言层面对面向对象的支持，但基于模块的设计仍然可以达到面向对象的效果。在Lua中一个文件就是一个模块，我们使用对象来封装相关功能，但不使用继承而是用组合来复用代码。

2.2 模块的定义

```
local Player = {} -- module table

Player.MAX_LEVEL = 99 -- public static const module variable

-- private function
local function somePrivateFunc(self)
    print(self.level)
end

-- constructor
function Player.new(name, level)
    local obj = {
        name = name,
        level = level
    }
    return setmetatable(obj, {__index = M})
end

-- public member method
function Player:sayHello()
    print(self.name .. "say hello")
    somePrivateFunc(self)
end

-- public static method
function Player.createFromTable(t)
    return Player.new(t.name, t.level)
end

return Player
```

2.3 定义模块规则

- 定义一个local的table作为模块的容器，比如 `Player`
- 模块中常量的定义紧跟着 `Player` 的定义之后
- 私有函数必须放在public函数之前
- 公有成员函数使用冒号定义法
- 公有静态函数使用点号定义法

- 模块 `Player` 中的变量相当于静态常量，所以是只读的
- 最后一行必须是 `return Player`
- 容器名简单起见可以统一使用 `M` 这个名字，但不是必须

2.4 模块与class的对应关系

以下这段C#代码对应上面的Lua代码：

```
public class Player {
    public static int readonly MAX_LEVEL = 99;

    public string name { get; private set; }
    public int level { get; private set; }

    public Player(string name, int level) {
        this.name = name;
        this.level = leve;
    }

    private void SomePrivateFunc() {
        print(this.level);
    }

    public void SayHello() {
        print(this.name + " say hello");
        SomePrivateFunc();
    }

    public static Player CreateFromTable(Dict<string,object> t) {
        return new Player(t["name"].ToString(), t["level"].ToInt());
    }
}
```

3 全局变量和函数

禁止使用全局变量和函数，所有的变量和函数必须定义在模块内部。

4 命名

4.1 文件名

文件名由一个或多个单词组成，首字母大写，单词之间不加任何字符。目录名规则相同。

示例：

```
Util.lua
Model/Player.lua
System/BattleSystem.lua
```

4.2 变量名

- 模块容器名使用CamelCaseName，首字母大写，可以统一用 `M`
- 局部变量, 成员变量使用camelCaseName，首字母小写
- 当局部变量是require返回值时可以根据模块名灵活使用首字大写或小写（因为标准库及一些第三方库使用首字母小写）

- 常量名全大写并用下划线分隔

4.3 函数名

函数名用camelCaseName，首字母小写

4.4 require

- require模块必须赋值给一个local变量，例如：`local os = require("os")`
- require模块含有目录的使用点号，例如：`local Player = require("Model.Player")`