1. 协议目录: Data/Protobuf

通常1个功能定义1个proto文件；

工具: 协议.bat

流程:

1. 写好proto文件

2. 执行 协议.bat，服务器会生成到LoadPBCProto.lua中，客户端会生成到客户端对应目录

3. 服务器在CmdDef.lua里面注册RPC, 并拷贝到Data/Document/协议文档.xls

4. 客户端从协议文档.xls中拷贝RPC到cmdDef.js里面

2. 配置文件目录: Data/Config

CSV 需要C++读取的配表

XML 生产LUA/JS文件用的配表

工具: 导表.bat

流程:

1. 服务器定义好表，填临时数据

2. 执行 导表.bat，会把生成的LUA文件拷贝到服务器对应目录，JS文件拷贝到客户端对应目录

3. 客户端如果需要添加配表字段，就修改xml文件，再跑流程

3. GM指令文件：Data/Document/GM.txt

目的: 方便测试用

4. 服务器通讯:

通讯类文件约定: (文件名)Rpc.lua

发送消息给客户端例子: CmdNet.PBSrv2Clt(会话ID, 协议名, 消息体)

5. 服务器注册/取消计时器:

local nTickID = goTimerMgr:Interval(秒, functin() 业务 end)

goTimerMgr:Clear(nTickID)

注意：不需要了的计时器一定要调用Clear释放，否则会泄露。

6. 客户端调用协议例子:

var tcpNet = require("tcpNet");

var PBBuilder = require("protoBuilder");

var msg = PBBuilder.message("global.TestPack");

msg.set("i1", 0x7FFFFFFF);

msg.set("i2", 0x7FFFFFFFFFFFFFFF);

msg.set("s1", "字符串测试 test1");

tcpNet.pbCall("TestPackReq", msg);

7. 客户端监听协议例子:

var clientPBRpc = require("clientPBRpc");

clientPBRpc.TestPackRet = function (cmdNo, source, data) {

console.log("TestPacRet\*\*\*", data.i1, data.i2.toNumber(), data.s1);

};

知识点:

1. Protobuf （通讯协议）

2. Common/LuaClass.lua （模拟面向对象）

服务器组成：

网关服务器 GateServer

逻辑服务器 LogicServer 业务都在这里

路由服务器 RouterServer 服务器之间路由

全局服务器 GlobalServer 全局服

日志服务器 LogServer

LogicServer目录结构:

ConfCheck：

启动的时候需要对配置文件进行校验时写在这里。

Global：

全局系统目录(联盟，邮件，商城，离线数据，排行榜，聊天等等)。数具常驻内存，定时保存脏数据。服务器启动的时候加载数据。

Module：

玩家身上的系统(国库，妃子，大臣等等)。

Player：

玩家自身，所有的Module对象会挂在玩家对象身上。玩家上线的时候会创建玩家对象和Module对象，并加载自己和所有Module对象的数据。离线时会保存自己和Moudle的脏数据，并将玩家对象和Module对象清出内存。

Common目录：

定义了一些公用的工具模块

批处理相关：

提交SVN

提交.bat

启动服务器批处理：

启动.bat

关闭服务器批处理:

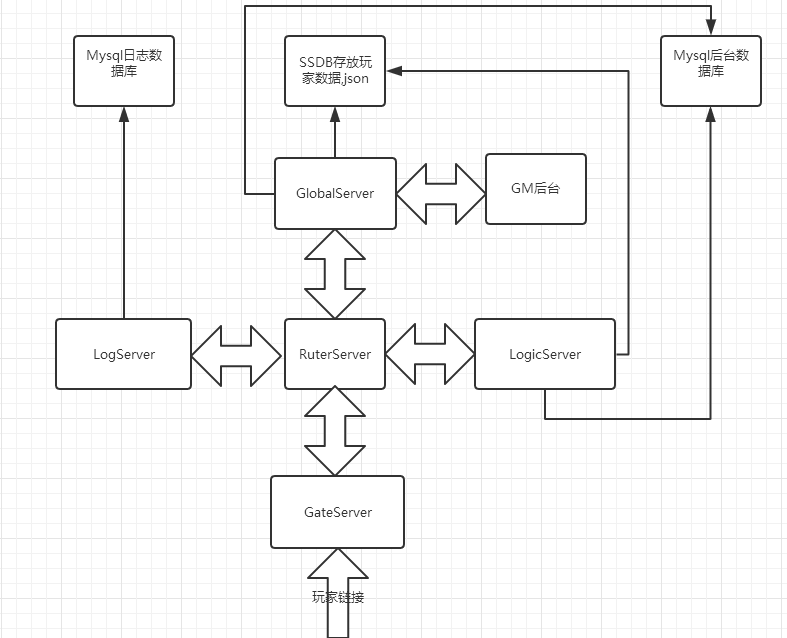
关闭.bat

生成协议批处理:

协议.bat

导表批处理:

导表.bat



LuaTrace 和 Print区别:

LuaTrace: 无论是debug或者release都会打印

Print: debug会打印 release不会打印

平时调试(debug)不重要的打印用Print，如果上线后(release)也需要打印的用LuaTrace

目的就是上线后屏蔽掉不必要的打印，打印耗性能