# 创建脚本服务器——Demo

**Demo服务器提交的svn版本：44**

## 配置修改

### 1 添加DemoServerConfig.pb

在server/project/config/下添加DemoServerConfig.pb，将GameServerConfig.pb的内容复制过来，再修改

port: 8006

game\_id: 3

game\_name: "demo"

其中port，game\_id，game\_name不要跟其他服务器配置重复，game\_name不要用中文。game\_id，game\_name会在脚本中生成对应的def\_game\_id，def\_game\_name变量。

### 2 修改GateServerConfig.pb

在GateServerConfig.pb中添加

game\_addr {

ip: "127.0.0.1"

port: 8006

}

**注：上面步骤现在通过配置AllConfig.xlsx，用python自动生成**

### 3 修改《启动服务器.bat》

在《启动服务器.bat》中添加

start GameServer.exe ../config/DemoServerConfig.pb

这样就可以启动demo服务器了！

## 定义房间

**在server/project/script/game/目录下添加demo文件夹，以后所有demo服务器相关脚本都放在这下面。**

### 1 添加demo\_room\_manager.lua

用于实现自己的房间。

demo\_room\_manager = base\_room\_manager:new()  **说明：继承自base\_rooms**

function demo\_room\_manager:init(tb) **说明：按照自己的需求初始化房间**

base\_room\_manager.init(self, tb) **说明：注意先调用基类的函数**

end

function demo\_room\_manager:auto\_sit\_down(player) **说明：重载快速坐下**

base\_room\_manager.auto\_sit\_down(self, player) **说明：注意先调用基类的函数**

**do something…**

end

function demo\_room\_manager:sit\_down(player, table\_id\_, chair\_id\_) **说明：重载坐下**

**do something…**

end

function demo\_room\_manager:stand\_up(player) **说明：重载站起**

**do something…**

base\_room\_manager.stand\_up(self, player) **说明：注意后调用基类的函数**

end

### 2 修改register.lua

在server/project/script/game/register.lua中的tb\_game\_obj表中添加

demo = function ()

require "game/lobby/base\_rooms"

local rooms = base\_rooms:new()

rooms:init({

{table\_count = 1, chair\_count = 2, ready\_mode = GAME\_READY\_MODE\_NONE, room\_limit = 0},

})

return rooms

end,

这样就使用新定义的demo房间了！

## 添加消息

### 1 添加common\_msg\_demo.proto

在common/pb/下添加common\_msg\_demo.proto，加入两个测试消息

message CS\_Demo {

enum MsgID { ID = 13000; Target = 2; }

optional string test = 1;

}

message SC\_Demo {

enum MsgID { ID = 13001; }

optional string test = 1;

}

### 2 修改client\_msg\_reg.lua

在reg\_gate\_dispatch\_msg.lua中添加

readMsgFile("common\_msg\_demo.proto")

### 3 执行server\_pbcreate.bat

将生成的c++文件加入vs中pb\_server工程，如果全部是lua处理protobuffer，也可以不加工程。

## 处理消息

### 1 添加on\_demo.lua

添加客户端发来消息的处理函数

function on\_cs\_demo(player, msg)

send2client\_pb(player.guid, "SC\_Demo", { test = "hello world"

}, player.gate\_id)

end

### 2 添加register.lua

注册客户端发送来的消息处理函数

register\_client\_dispatcher("CS\_Demo", "on\_cs\_demo")

### 3 注册Demo消息

在server/project/script/game/register.lua中的tb\_reg\_msg\_dispatcher表中添加

demo = function ()

require "game/demo/register"

end,

这样demo服务器就可以处理客户端发送过来的消息了！