# **Photon Server游戏服务器教程：游戏登录与注册操作**

Photon引擎是一款实时的Socket服务器和开发框架，快速、使用方便、容易扩展。服务端架构在windows系统平台上，采用C#语言编写。客户端SDK提供了多种平台的开发API，包括DotNet、Unity3D、C/C++以及ObjC等。下面就给大家介绍下在使用Photon引擎去实现游戏登录与注册操作功能。

为了在客户端与服务器端使用共同的code，创建共有引用Common：

public enum OperationCode:byte //区分请求和响应的类型

｛

Default,//默认请求

Login, //登录

Register //注册

｝

public enum ReturnCode:short //服务器返回的类型

｛

Success,//成功

Failed //失败

｝

public enum ParameterCode:byte //区分传送数据的时候，参数的类型

｛

Username,//用户名

Password, //密码

｝

//获取字典的值

public class DictTool｛

public static T2 GetValue&lt;T1, T2&gt;(Dictionary&lt;T1,T2&gt; dict, T1 key) ｛

T2 value;

bool isSuccess = dict.TryGetValue(key, out value);

if (isSuccess) ｛ return value; ｝

else ｛ return default(T2); ｝

｝

｝

在客户端新建一个请求的基类Request：

//客户端向服务器的请求

public abstract class Request:MonoBehaviour ｛

public OperationCode OpCode;//请求类型

public abstract void DefaultRequest();//默认的请求

//服务器端返回的响应

public abstract void OnOperationResponse(OperationResponse operationResponse);

public virtual void Start() ｛

//添加请求到集合

GamePhotonEngine.Instance.AddRequest(this);

｝

public void OnDestroy() ｛

//从集合中删除当前请求

GamePhotonEngine.Instance.RemoveRequest(this);

｝

｝

在GamePhotonEngine 类中添加：

//所有请求的一个集合

private Dictionary&lt;OperationCode, Request&gt; RequestDict = new Dictionary&lt;OperationCode, Request&gt;();

//客户端向服务器发起一个请求以后服务器处理完以后 就会给客户端一个响应

public void OnOperationResponse(OperationResponse operationResponse) ｛

//把服务器返回的请求分发给对应的子类去处理

OperationCode opCode =(OperationCode) operationResponse.OperationCode;

Request request = null;

bool temp= RequestDict.TryGetValue(opCode, out request);

if (temp) ｛

request.OnOperationResponse(operationResponse);

｝ else ｛

Debug.Log("没找到对应的响应处理对象");

｝

｝

public void AddRequest(Request request) ｛

RequestDict.Add(request.OpCode, request);

｝

public void RemoveRequest(Request request)｛

RequestDict.Remove(request.OpCode);

｝

服务器端新建处理各个请求的 BaseHandler基类：

public abstract class BaseHandler ｛

public OperationCode OpCode;

public abstract void OnOperationRequest(OperationRequest operationRequest, SendParameters sendParameters,MyClientPeer peer);

｝

//新建一个默认的Handler：

class DefaultHandler:BaseHandler｛

public DefaultHandler() ｛

OpCode = Common.OperationCode.Default;

｝

public override void OnOperationRequest(Photon.SocketServer.OperationRequest operationRequest, Photon.SocketServer.SendParameters sendParameters, MyClientPeer peer) ｛

throw new NotImplementedException();

｝

｝

在MyClientPeer中分派各个响应：

//处理客户端的请求

protected override void OnOperationRequest(OperationRequest operationRequest, SendParameters sendParameters)｛

BaseHandler handler = DictTool.GetValue&lt;OperationCode, BaseHandler&gt;(MyGameServer.Instance.HandlerDict, (OperationCode)operationRequest.OperationCode);

if (handler != null) ｛

handler.OnOperationRequest(operationRequest, sendParameters, this);

｝ else ｛

BaseHandler defaultHandler = DictTool.GetValue&lt;OperationCode, BaseHandler&gt;(MyGameServer.Instance.HandlerDict, OperationCode.Default);

defaultHandler.OnOperationRequest(operationRequest, sendParameters, this);

｝

｝

在 服务器端MyGameServer中添加代码：

//Handler集合

public Dictionary&lt;OperationCode, BaseHandler&gt; HandlerDict = new Dictionary&lt;OperationCode, BaseHandler&gt;();

public static MyGameServer Instance｛

get;

private set;

｝

protected override void Setup()｛

//初始化各个Handler

Instance = this;

InitHandler();

｝

public void InitHandler()｛

DefaultHandler defaultHandler = new DefaultHandler();

HandlerDict.Add(defaultHandler.OpCode, defaultHandler);

LoginHandler loginHandler = new LoginHandler();

HandlerDict.Add(loginHandler.OpCode, loginHandler);

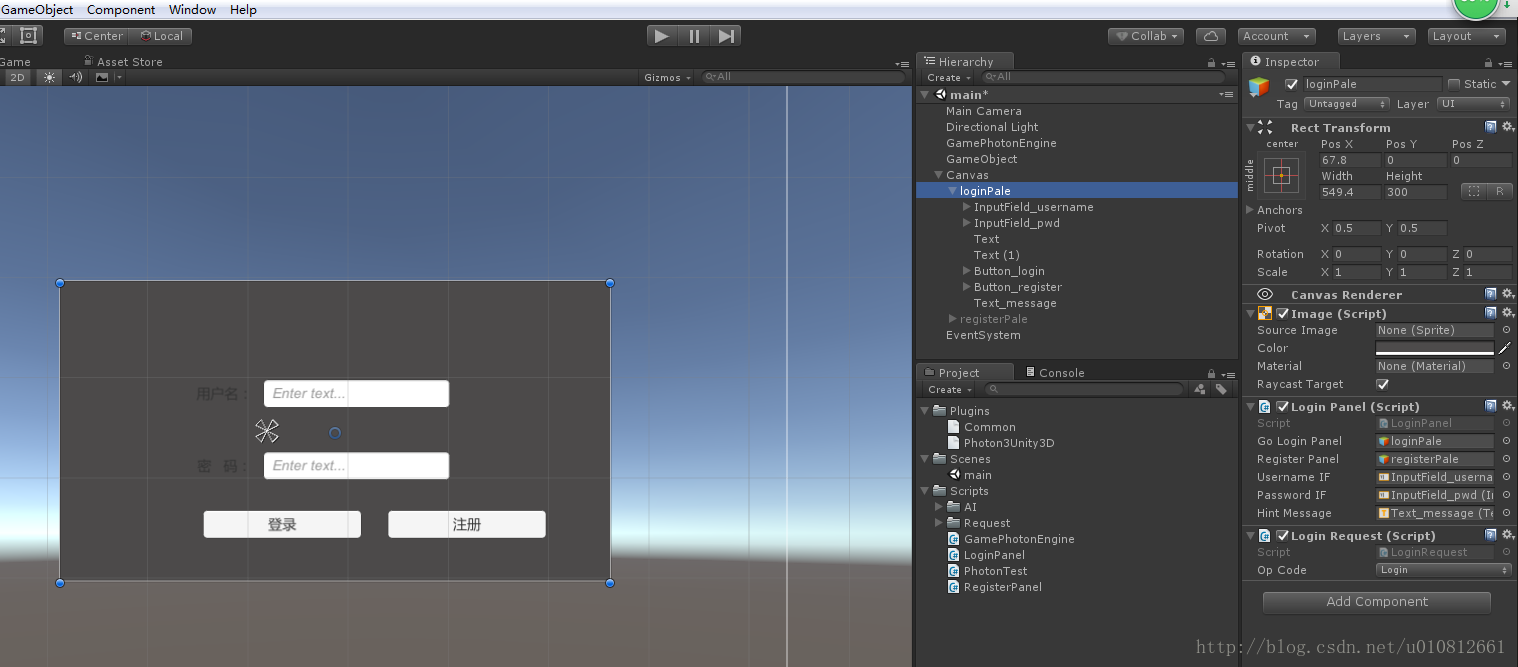
RegisterHandler registerHandler = new RegisterHandler();

HandlerDict.Add(registerHandler.OpCode, registerHandler);

｝

**登录操作**

UI界面：



LoginPanel :

using Common;

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI;

public class LoginPanel : MonoBehaviour｛

public GameObject goLoginPanel;

public GameObject RegisterPanel;

private LoginRequest loginRequest;

public InputField usernameIF;

public InputField passwordIF;

public Text hintMessage;

void Start() ｛

loginRequest = GetComponent&lt;LoginRequest&gt;();

｝

public void OnLoginBtn() ｛

hintMessage.text = "";

loginRequest.Username = usernameIF.text;

loginRequest.Password = passwordIF.text;

loginRequest.DefaultRequest();

Debug.Log("OnLoginBtn");

｝

public void OnRegister() ｛

goLoginPanel.SetActive(false);

RegisterPanel.SetActive(true);

｝

public void OnLoginResponse(ReturnCode returnCode) ｛

if (returnCode == ReturnCode.Success) ｛

// 跳转到下一个场景

// SceneManager.LoadScene("Game");

Debug.Log("登录成功");

｝ else ｛

hintMessage.text = "用户名或密码错误";

｝

｝

｝

在客户端新建一个登录操作请求LoginRequest ：

public class LoginRequest : Request ｛

[HideInInspector]

public string Username;

[HideInInspector]

public string Password;

private LoginPanel loginPanel;

public override void Start() ｛

base.Start();

loginPanel = GetComponent&lt;LoginPanel&gt;();

｝

public override void DefaultRequest() ｛

Dictionary&lt;byte,object&gt; data = new Dictionary&lt;byte,object&gt;();

data.Add((byte)ParameterCode.Username, Username);

data.Add((byte)ParameterCode.Password, Password);

GamePhotonEngine.Peer.OpCustom((byte)OpCode, data, true);

｝

public override void OnOperationResponse(OperationResponse operationResponse) ｛

ReturnCode returnCode = (ReturnCode)operationResponse.ReturnCode;

loginPanel.OnLoginResponse(returnCode);

｝

｝

在服务器端新建LoginHandler：

class LoginHandler:BaseHandler ｛

public LoginHandler() ｛

OpCode = OperationCode.Login;

｝

public override void OnOperationRequest(OperationRequest operationRequest, SendParameters sendParameters, ClientPeer peer)｛

string username = DictTool.GetValue&lt;byte, object&gt;(operationRequest.Parameters, (byte)ParameterCode.Username) as string;

string password = DictTool.GetValue&lt;byte, object&gt;(operationRequest.Parameters, (byte)ParameterCode.Password) as string;

UserManager manager = new UserManager();

bool isSuccess = manager.VerifyUser(username, password);

OperationResponse response = new OperationResponse(operationRequest.OperationCode);

if (isSuccess) ｛

response.ReturnCode = (short)Common.ReturnCode.Success;

peer.username = username;

｝ else ｛

response.ReturnCode = (short)Common.ReturnCode.Failed;

｝

peer.SendOperationResponse(response, sendParameters);

｝

｝

注册操作：

public class RegisterRequest : Request ｛

[HideInInspector]

public string username;

[HideInInspector]

public string password;

private RegisterPanel registerPanel;

public override void Start() ｛

base.Start();

registerPanel = GetComponent&lt;RegisterPanel&gt;();

｝

public override void DefaultRequest() ｛

Dictionary&lt;byte, object&gt; data = new Dictionary&lt;byte, object&gt;();

data.Add((byte)ParameterCode.Username, username);

data.Add((byte)ParameterCode.Password, password);

GamePhotonEngine.Peer.OpCustom((byte)OpCode, data, true);

｝

public override void OnOperationResponse(OperationResponse operationResponse) ｛

ReturnCode returnCode = (ReturnCode)operationResponse.ReturnCode;

// registerPanel.OnReigsterResponse(returnCode);

｝

｝

class RegisterHandler:BaseHandler｛

public RegisterHandler() ｛

OpCode = OperationCode.Register;

｝

public override void OnOperationRequest(OperationRequest operationRequest, SendParameters sendParameters, ClientPeer peer) ｛

string username = DictTool.GetValue&lt;byte, object&gt;(operationRequest.Parameters, (byte)ParameterCode.Username) as string;

string password = DictTool.GetValue&lt;byte, object&gt;(operationRequest.Parameters, (byte)ParameterCode.Password) as string;

UserManager manager = new UserManager();

User user = manager.GetByUsername(username);

OperationResponse response = new OperationResponse( operationRequest.OperationCode );

if (user == null)｛

user= new User()｛Username=username,Password=password｝;

manager.Add(user);

response.ReturnCode =(short) ReturnCode.Success;

｝ else ｛

response.ReturnCode = (short)ReturnCode.Failed;

｝

peer.SendOperationResponse(response, sendParameters);

｝

｝