# **Photon Server游戏服务器教程：游戏登录与注册操作**

### 想免费获取内部独家PPT资料库？观看行业大牛直播？点击加入腾讯GAD游戏开发行业精英群**[711501594](http://shang.qq.com/wpa/qunwpa?idkey=3fcf6147441efc8b343eb7447e6a996eabb6a33093306529c4d1e9d2ab75423d" \t "http://gad.qq.com/article/detail/_blank)**

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

5

Register //注册

6

｝

7

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

8

｛

9

Success,//成功

10

Failed //失败

11

｝

12

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

13

｛

14

Username,//用户名

15

Password, //密码

16

｝

17

//获取字典的值

18

public class DictTool

19

｛

20

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

21

｛

22

T2 value;

23

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

24

if (isSuccess)

25

｛

26

return value;

27

｝

28

else

29

｛

30

return default(T2);

31

｝

32

｝

33

｝

在客户端新建一个请求的基类Request：

1

//客户端向服务器的请求

2

public abstract class Request:MonoBehaviour

3

｛

4

public OperationCode OpCode;//请求类型

5

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

6

//服务器端返回的响应

7

public abstract void OnOperationResponse(OperationResponse operationResponse);

8

public virtual void Start()

9

｛

10

//添加请求到集合

11

GamePhotonEngine.Instance.AddRequest(this);

12

｝

13

public void OnDestroy()

14

｛

15

//从集合中删除当前请求

16

GamePhotonEngine.Instance.RemoveRequest(this);

17

｝

18

｝

在GamePhotonEngine 类中添加：

1

//所有请求的一个集合

2

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

3

//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

4

public void OnOperationResponse(OperationResponse operationResponse)

5

｛

6

//把服务器返回的请求分发给对应的子类去处理

7

OperationCode opCode =(OperationCode) operationResponse.OperationCode;

8

Request request = null;

9

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

10

if (temp)

11

｛

12

request.OnOperationResponse(operationResponse);

13

｝

14

else

15

｛

16

Debug.Log("没找到对应的响应处理对象");

17

｝

18

｝

19

public void AddRequest(Request request)

20

｛

21

RequestDict.Add(request.OpCode, request);

22

｝

23

public void RemoveRequest(Request request)

24

｛

25

RequestDict.Remove(request.OpCode);

26

｝

服务器端新建处理各个请求的 BaseHandler基类：

1

public abstract class BaseHandler

2

｛

3

public OperationCode OpCode;

4

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

5

｝

6

//新建一个默认的Handler：

7

class DefaultHandler:BaseHandler

8

｛

9

public DefaultHandler()

10

｛

11

OpCode = Common.OperationCode.Default;

12

｝

13

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

14

｛

15

throw new NotImplementedException();

16

｝

17

｝

在MyClientPeer中分派各个响应：

1

//处理客户端的请求

2

protected override void OnOperationRequest(OperationRequest operationRequest, SendParameters sendParameters)

3

｛

4

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

5

if (handler != null)

6

｛

7

handler.OnOperationRequest(operationRequest, sendParameters, this);

8

｝

9

else

10

｛

11

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

12

defaultHandler.OnOperationRequest(operationRequest, sendParameters, this);

13

｝

14

｝

在 服务器端MyGameServer中添加代码：

1

//Handler集合

2

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

3

public static MyGameServer Instance

4

｛

5

get;

6

private set;

7

｝

8

protected override void Setup()

9

｛

10

//初始化各个Handler

11

Instance = this;

12

InitHandler();

13

｝

14

public void InitHandler()

15

｛

16

DefaultHandler defaultHandler = new DefaultHandler();

17

HandlerDict.Add(defaultHandler.OpCode, defaultHandler);

18

LoginHandler loginHandler = new LoginHandler();

19

HandlerDict.Add(loginHandler.OpCode, loginHandler);

20

RegisterHandler registerHandler = new RegisterHandler();

21

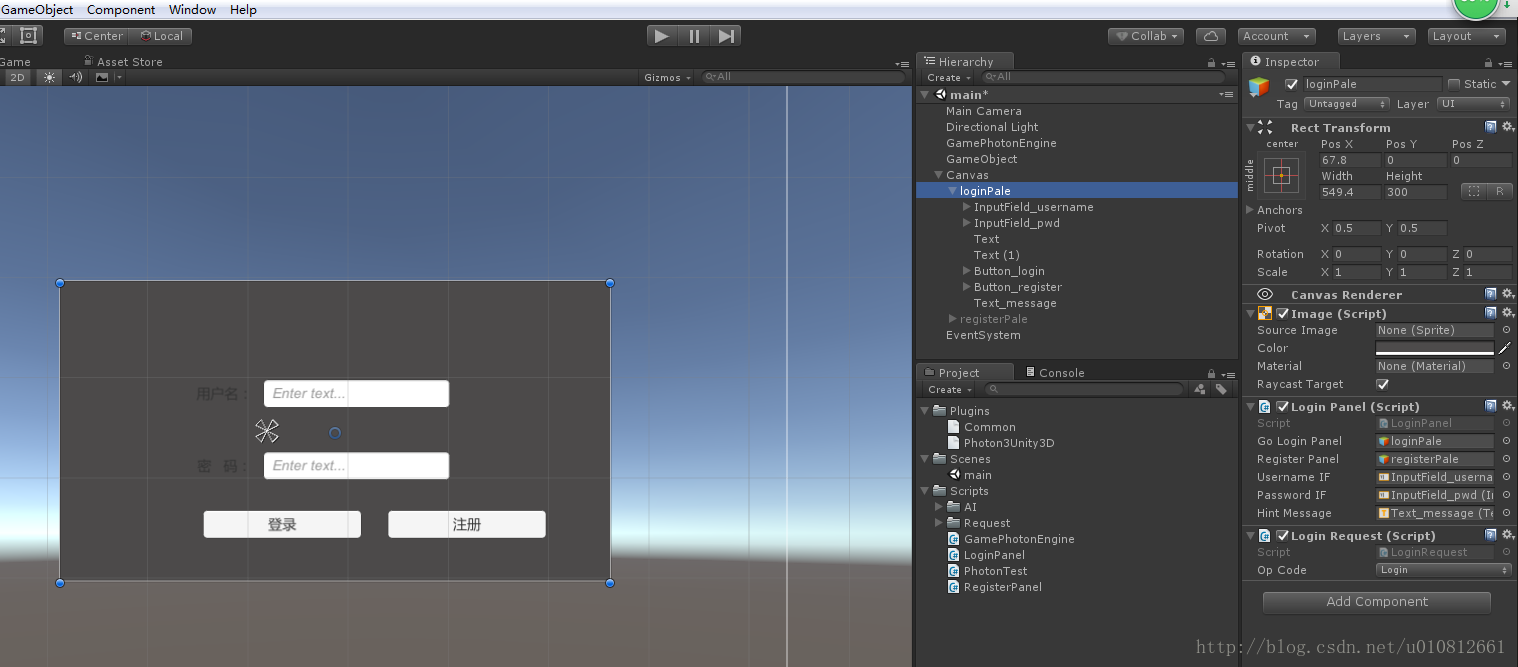
HandlerDict.Add(registerHandler.OpCode, registerHandler);

22

｝

**登录操作**

UI界面：



LoginPanel :

1

using Common;

2

using System.Collections;

3

using System.Collections.Generic;

4

using UnityEngine;

5

using UnityEngine.UI;

6

public class LoginPanel : MonoBehaviour

7

｛

8

public GameObject goLoginPanel;

9

public GameObject RegisterPanel;

10

private LoginRequest loginRequest;

11

public InputField usernameIF;

12

public InputField passwordIF;

13

public Text hintMessage;

14

void Start()

15

｛

16

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

17

｝

18

public void OnLoginBtn()

19

｛

20

hintMessage.text = "";

21

loginRequest.Username = usernameIF.text;

22

loginRequest.Password = passwordIF.text;

23

loginRequest.DefaultRequest();

24

Debug.Log("OnLoginBtn");

25

｝

26

public void OnRegister()

27

｛

28

goLoginPanel.SetActive(false);

29

RegisterPanel.SetActive(true);

30

｝

31

public void OnLoginResponse(ReturnCode returnCode)

32

｛

33

if (returnCode == ReturnCode.Success)

34

｛

35

// 跳转到下一个场景

36

// SceneManager.LoadScene("Game");

37

Debug.Log("登录成功");

38

｝

39

else

40

｛

41

hintMessage.text = "用户名或密码错误";

42

｝

43

｝

44

｝

在客户端新建一个登录操作请求LoginRequest ：

1

public class LoginRequest : Request ｛

2

[HideInInspector]

3

public string Username;

4

[HideInInspector]

5

public string Password;

6

private LoginPanel loginPanel;

7

public override void Start()

8

｛

9

base.Start();

10

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

11

｝

12

public override void DefaultRequest()

13

｛

14

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

15

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

16

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

17

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

18

｝

19

public override void OnOperationResponse(OperationResponse operationResponse)

20

｛

21

ReturnCode returnCode = (ReturnCode)operationResponse.ReturnCode;

22

loginPanel.OnLoginResponse(returnCode);

23

｝

24

｝

在服务器端新建LoginHandler：

1

class LoginHandler:BaseHandler

2

｛

3

public LoginHandler()

4

｛

5

OpCode = OperationCode.Login;

6

｝

7

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

8

｛

9

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

10

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

11

UserManager manager = new UserManager();

12

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

13

OperationResponse response = new OperationResponse(operationRequest.OperationCode);

14

if (isSuccess)

15

｛

16

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

17

peer.username = username;

18

｝

19

else

20

｛

21

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

22

｝

23

peer.SendOperationResponse(response, sendParameters);

24

｝

25

｝

注册操作：

1

public class RegisterRequest : Request ｛

2

[HideInInspector]

3

public string username;

4

[HideInInspector]

5

public string password;

6

private RegisterPanel registerPanel;

7

public override void Start()

8

｛

9

base.Start();

10

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

11

｝

12

public override void DefaultRequest()

13

｛

14

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

15

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

16

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

17

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

18

｝

19

public override void OnOperationResponse(OperationResponse operationResponse)

20

｛

21

ReturnCode returnCode = (ReturnCode)operationResponse.ReturnCode;

22

// registerPanel.OnReigsterResponse(returnCode);

23

｝

24

｝

1

class RegisterHandler:BaseHandler

2

｛

3

public RegisterHandler()

4

｛

5

OpCode = OperationCode.Register;

6

｝

7

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

8

｛

9

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

10

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

11

UserManager manager = new UserManager();

12

User user = manager.GetByUsername(username);

13

OperationResponse response = new OperationResponse( operationRequest.OperationCode );

14

if (user == null)

15

｛

16

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

17

manager.Add(user);

18

response.ReturnCode =(short) ReturnCode.Success;

19

｝

20

else

21

｛

22

response.ReturnCode = (short)ReturnCode.Failed;

23

｝

24

peer.SendOperationResponse(response, sendParameters);

25

｝

26

｝



想免费获取内部独家PPT资料库？ 观看行业大牛直播？

立即扫码加群

[点击加入](http://shang.qq.com/wpa/qunwpa?idkey=3fcf6147441efc8b343eb7447e6a996eabb6a33093306529c4d1e9d2ab75423d" \t "http://gad.qq.com/article/detail/_blank)

##### [腾讯GAD游戏开发行业精英群](http://shang.qq.com/wpa/qunwpa?idkey=3fcf6147441efc8b343eb7447e6a996eabb6a33093306529c4d1e9d2ab75423d" \t "http://gad.qq.com/article/detail/_blank)

[711501594](http://shang.qq.com/wpa/qunwpa?idkey=3fcf6147441efc8b343eb7447e6a996eabb6a33093306529c4d1e9d2ab75423d" \t "http://gad.qq.com/article/detail/_blank)

[0](http://gad.qq.com/article/detail/javascript:;) [0](http://gad.qq.com/article/detail/javascript:;)