## PhotonServer游戏服务器【完整版】

## 前序

#### 一、跟数据库交互

直接写，利用mysql的接口

使用nHibernate

##### Csharp直接连接mysql

找到相应版本的动态链接库 在vs中点击引用-浏览-在c盘的assemblies中选择相应的版本-选择mysql.Data.dll文件。

Documentaion文件夹中ConnectorNET.chm中提供api有助编程

定义一个字符串来存数据库的IP，端口号，数据库，用户名，密码。(端口号默认3306，如果安装时候修改了按修改的)   


##### 利用程序进行插入操作

insert into 表名（根据表中的列填写要插入的列username，possword）values(自己设置的值’aa’,’125’);   
Datetime.Now 获取当前时间   
string sql = "insert into users(username,password,registerdate) values('aaa','222','"+DateTime.Now+"')";

##### 更新操作

Update 表名 set name(要设置的列)=’自己要修改的值’ where id(条件)=2; 或者是 where name=’a’;   
string sql = "update users set registerdate='2017-10-25' where id=13";

##### 删除数据

delete from 表名 where 条件id=4  
string sql = "delete from users where name='aaa'";

##### sql中的一些函数

SQL 语法网址：[w3school](http://www.w3school.com.cn/sql/sql_func_count.asp" \t "https://blog.csdn.net/m0_37283423/article/details/_blank)  
选择 文件夹图标旁边的加号，创建一个新的sql file 可以再mysql中直接写语句

##### 获取数据的方式

IMG_257

##### 查询时添加参数

  
调用时：   
IMG_259

## 利用nhibernate

##### 1、nhibernate介绍和程序包的引入

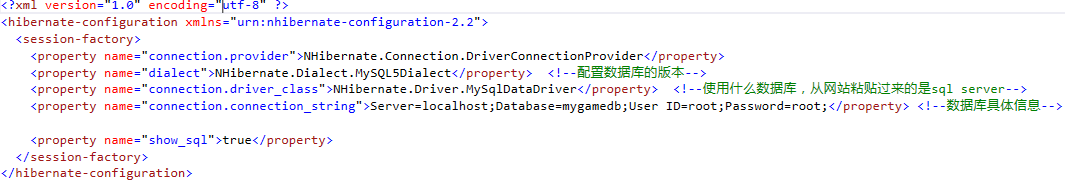
##### NHibernate是一个面向.NET环境的对象/关系数据库映射工具<br/>

如果利用nhibernate来与数据库交互的话不仅需要引入mysql.dll还需要引入nhibernate.dll，以下是nhibernate的两种引入方式（引入mysql也可以使用）：

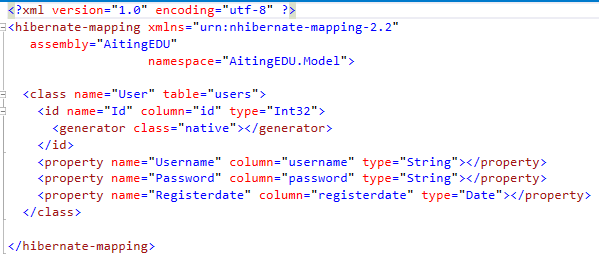
1. 到官网下载-找到Required\_Bins-找到NHibernate.dll引入即可。  
 2. 在VS中利用NUget:在资源管理右键-管理Nuget程序包-搜索nhibernate-安装即可。

##### 配置nhibernate数据库连接配置

添加一个xml文件（固定名称）：hibernate.cfg.xml

修改官网上的配置信息：修改后如下   
  
！！！需要在其文件的属性栏把（复制到输出目录）：选择为始终

##### 如何进行类和表的映射

定义一个类。添加一个映射文件。（xml文件） 文件名.hbm.xml. 在官网上找到映射文件进行复制修改如下：  
   
！！！需要在其文件的属性栏把（生成操作）：选择为嵌入  
类型根据官方文档中的进行修改例如：int应改为Int32

##### 解析数据库连接配置文件和表映射文件

进行引用：using NHibernate ;using NHibernate.Cfg;

解析hibernate.cfg.xml

var configuration=new Configuration();configuration.Configure();

也可以指定一个文件configuration.Configure(“名字”);

解析映射文件 User.hbm.xml….. configuration.AddAssembly(“AitingEDU”);

##### 创建session对话进行添加操作



## Photon Server的介绍

## 一、Photon Server工作流程介绍如图所示：  IMG_263  - 解压后有两个文件夹最为重要:  - deploy:application要在其中部署  - src-server：一些demo的源码

## 二、下载的Photon Server 默认是连接数为30如果想更大可购买，官网上有一个连接数为100的，只需下载赋值到运行Photon ServerControllor.exe同级重启Photon Server即可在Liscene info中可查看。

##### 一、创建第一个服务器端项目，并设置部署的目录

1.创建一个解决方案

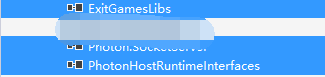
2.新建一个类库

3.在PhotonServer解压文件夹中的deploy中新建一个文件夹（名字自定义）。

4.在3创建的文件夹中再创建一个文件夹（命名为bin）

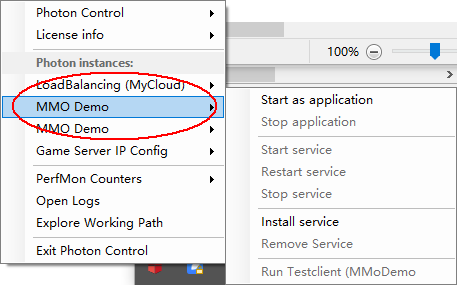
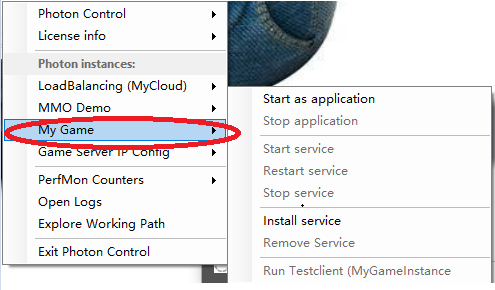
5.在创建的解决方案中的属性-生成-输出路径改为4。重新生成这个解决方案即可。

##### 二、开发ApplicationBase和ClientPeer

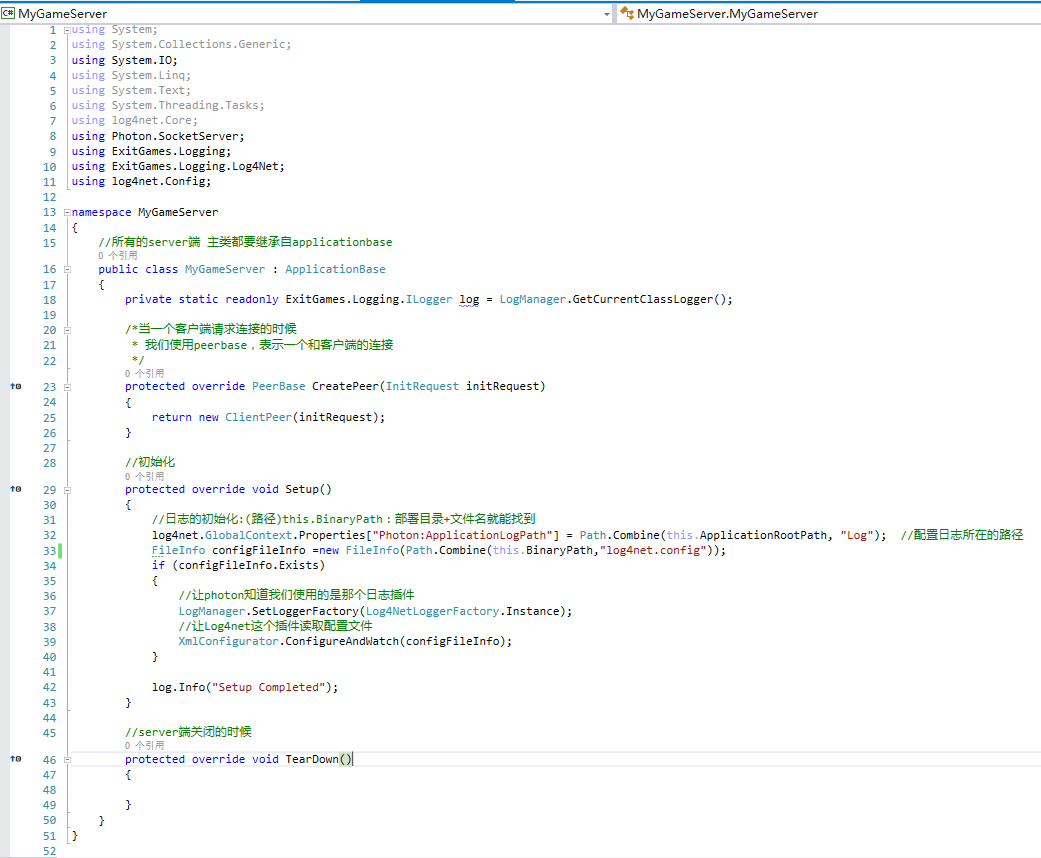
1.要添加引用:PhotonServer给我们提供的动态链接库（.dll文件）在项目的引用中右键-浏览-PhotonServer解压文件夹中的lib文件夹中 添加一下三个引用   


2.在（一）中的解决方案下删除自带的类，新建一个主类，名称与解决方案名字相同,设置为public并继承ApplicationBase（所有的server端 主类都要继承自applicationbase）。代码如下：   
   
在创建一个类命名为ClientPeer设置为public，并且继承自 Photon.SocketServer.ClientPeer。

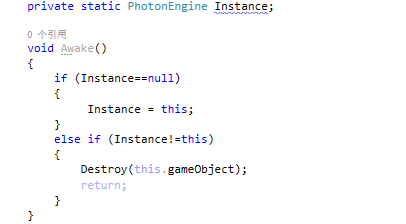
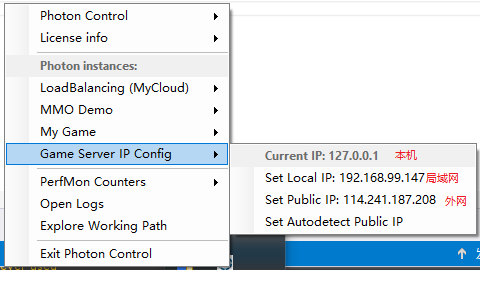
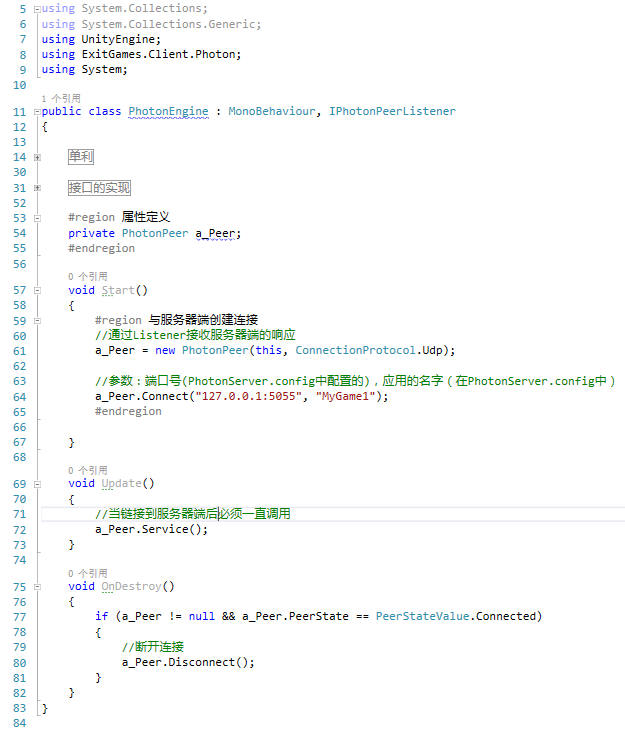
##### 部署并开启我们的第一个服务器端应用

1. 要先进行配置：找到deploy-PhotonServer.config。
2. 将其中一个配置好的代码复制一个进行修改修如下：   
   a.标签名字MMoDemo-(自己起，不能重复)   
   b.DisplayName-（自己起，这个名字将会在外部显示如图:）   
      
   c.如图所示：   
      
   d.重新启动PhotonServer如图所示：   
      
   如果服务启动失败或者有错需要在Logs(日志中找错)，上图中的Open Logs,就是打开日志。

##### 配置Photon Server中的日志Logs

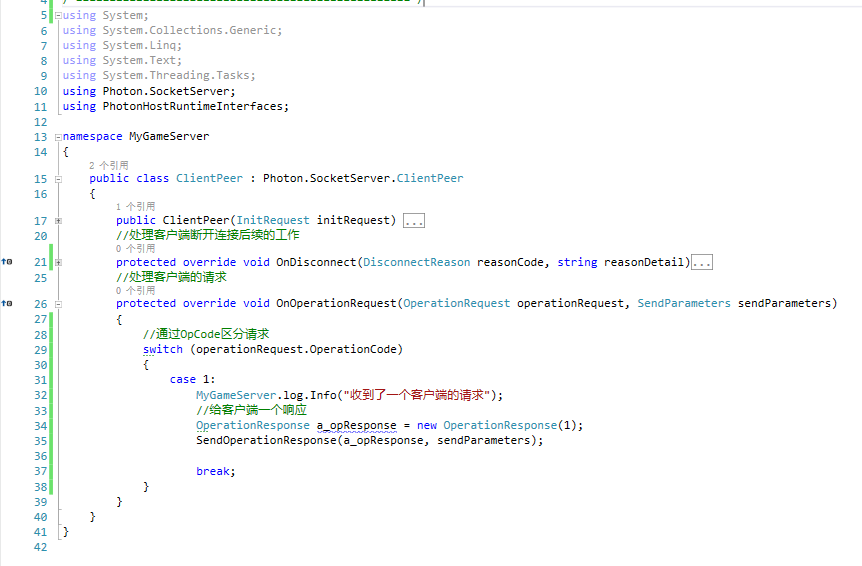
1. 需要一个插件Log4Net（在引用中点击浏览-根目录中的lib中找到）
2. Log4net的配置文件，   
   1）在官网中找：[http://logging.apache.org/log4net/](http://logging.apache.org/log4net/" \t "https://blog.csdn.net/m0_37283423/article/details/_blank)   
   2）在PhotonServer的根目录中-scr server中-随便选择一个项目-找到其中的.server为后缀的文件夹-找到log4net。将这个直接复制到MyGameServer中。并把其设置为始终复制。   
   3）对配置文件进行修改如图所示：   
   
3. 添加一个引用：ExitGames.Logging.Log4Net.dll   
   1)代码如下所示：   
     
   2）我们可以点击open在deploy-log中找到我们自己的日志文件可以查看到我们输出的日志，如果我们想同意放在win64-log中休要修改一句代码如下：   
   IMG_271  
   提示：   
   1）如果应用配置出错了：PhotonCLR.log中找。   
   2）如果程序启动异常了：Photon-MyGameInstance-20171116.log中找。   
   3）如果程序代码逻辑出错了：就在我么自己输出的日志中找。

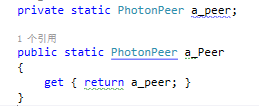
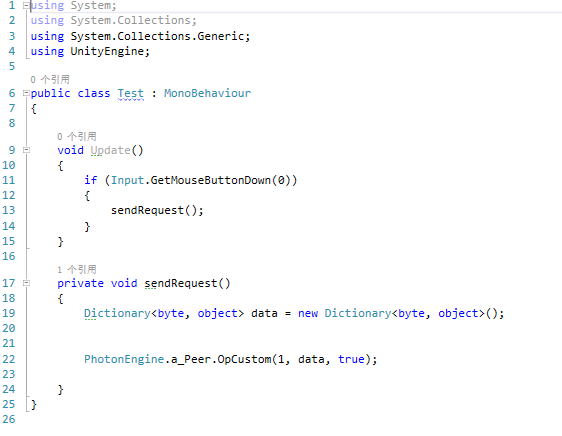
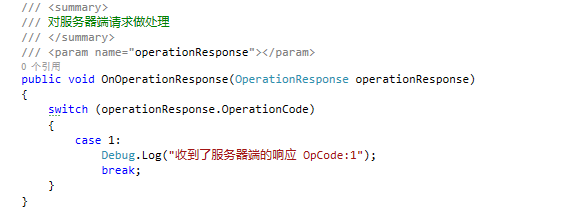
##### 创建Unity客户端

1. 引入PhotonServer的sdk到unity中：lib-Photon3Unity3d.dll   
    1）在unity中新建一个文件夹命名为：Plugins(插件unity会优先编译)。   
    2）把文件拖入。
2. 在场景中新建一个空游戏物体命名为：PhotonEngine，身上挂一个脚本PhotonEngine用来给服务器端交互。   
    1）把脚本改为单利如下：   
      
    2）添加一个引用:using ExitGames.Client.Photon;   
    3) 让我们的脚本也能实现监听功能如下所示：   
      
   4）调用监听和udp协议： PhotonPeer a\_Peer = new PhotonPeer(this,ConnectionProtocol.Udp);   
   5)创建连接：   
   IMG_274   
   a.IP选择建议图中的1.2：   
      
   6）代码参考：   
   

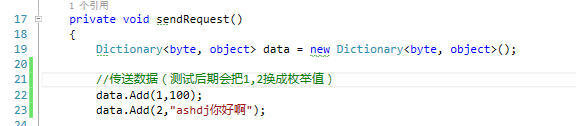
##### 创建客户端的请求

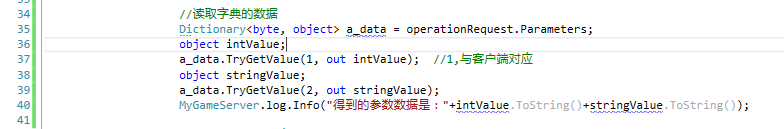
1. 通过OpCustom像服务器端发起请求
2. 服务器端通过OnOperationRequest进行处理
3. 服务器端通过OperationRespone ,SendOperationRespone给客户端进行回应。
4. 用客户端中的接口OnOperationResponse来对服务器端的回应做处理。

服务器端代码如下所示：   


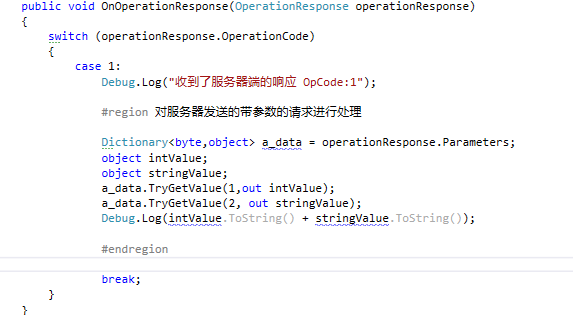
客户端代码如下所示：   
    
接口的调用   


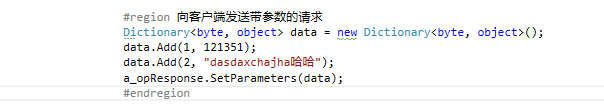
##### 客户端向服务器端发送带参数的请求

客户端代码：   


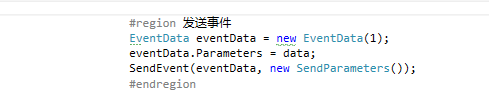
服务器代码：   


##### 服务器端向客户端发送带参数的请求

客户端代码 ：   


服务器代码：   


##### 服务器端向客户端发送事件

1. 客户端代码：   
   
2. 服务器代码：   
   

##### 交互的整体流程图

