# **Photon Server游戏服务器教程（二） Log日志文件配置与输出**

**一、Log日志的打开**

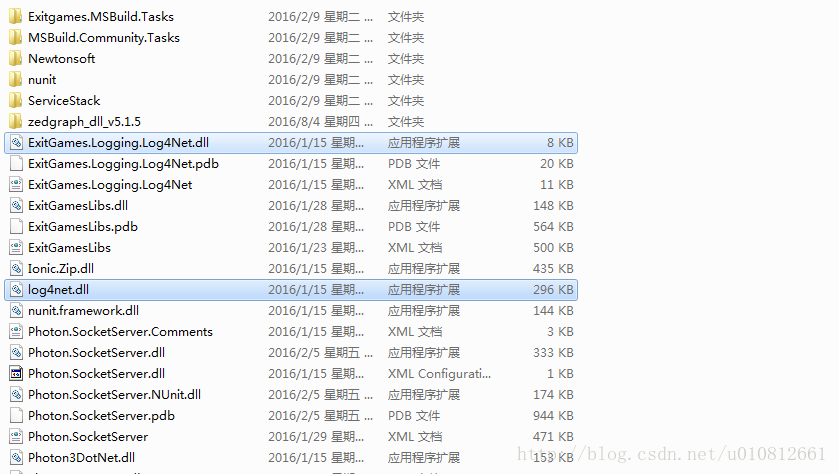
在Photon中点击Open Logs选项，打开BareTail软件，相当于一个记事本，用于Log日志的输出。

打开以后的界面

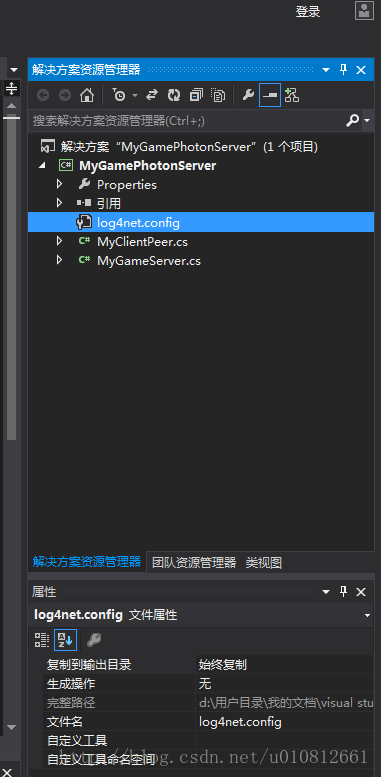


**二、Log日志的输出**

1.引用log4net.dll 和ExitGames.Logging.Log4Net.dll，位于lib文件夹下。



2.添加配置文件，由于有固定的格式，因此我们在src-server\Mmo\Photon.MmoDemo.Server文件夹下复制一份log4net.config文件放在我们项目的根目录下，修改属性为始终复制。



配置文件路径 Photon:ApplicationLogPath，修改为自己的程序名MyGame。

&lt;file type="log4net.Util.PatternString" value="%property｛Photon:ApplicationLogPath｝\\MyGame.Server.log" /&gt;

3.在主类MyGameServer中添加一个静态只读字段log并初始化

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Photon.SocketServer;

using ExitGames.Logging;

using ExitGames.Logging.Log4Net;

using System.IO;

using log4net.Config;

namespace MyGamePhotonServer｛

//所有的server端 主类都要集成自applicationbase

public class MyGameServer:ApplicationBase{

public static readonly ILogger log = LogManager.GetCurrentClassLogger();

/// 刚一个客户端请求连接的

protected override PeerBase CreatePeer(InitRequest initRequest)｛

log.Info("一个客户端连接过来了。。。。");

return new MyClientPeer(initRequest);

｝

/// 初始化

protected override void Setup()｛

// 日志的初始化

log4net.GlobalContext.Properties["Photon:ApplicationLogPath"] = Path.Combine(

Path.Combine(this.ApplicationRootPath, "bin\_Win64"), "log");

FileInfo configFileInfo = new FileInfo( Path.Combine( this.BinaryPath ,"log4net.config"));

if (configFileInfo.Exists)｛

LogManager.SetLoggerFactory(Log4NetLoggerFactory.Instance);//让photon知道使用的是Log4NetLog插件

XmlConfigurator.ConfigureAndWatch(configFileInfo);//让log4net这个插件读取配置文件

｝

log.Info("初始化完成！");

｝

/// server端关闭的时候

protected override void TearDown()｛

log.Info("服务器应用关闭了");

｝

｝

｝

运行后的日志输出结果

