Scut快速开发（3）

# 开发环境

Scut Lib版本：5.2.3.2

**需要安装的软件**

1. IIS和消息队列(MSMQ)
2. 数据库，Sql2005以上版本
3. VS2010开发工具（.Net Framework 4.0以上）
4. Python2.6（ScutGame官网下载IronPython2.6.1 RC1 for .NET 4.0插件）

**工具**

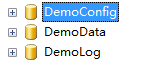
1. 协议工具（目录Source\Tools\ContractTools）

# 代码框架

项目划分三层：实体层，组件层，业务逻辑层（脚本层）；模型层主要是数据实体映射，自定义缓存结构；组件层主要负责实现中层层扩展功能；业务逻辑层主要负责实现游戏功能；

## 数据库创建

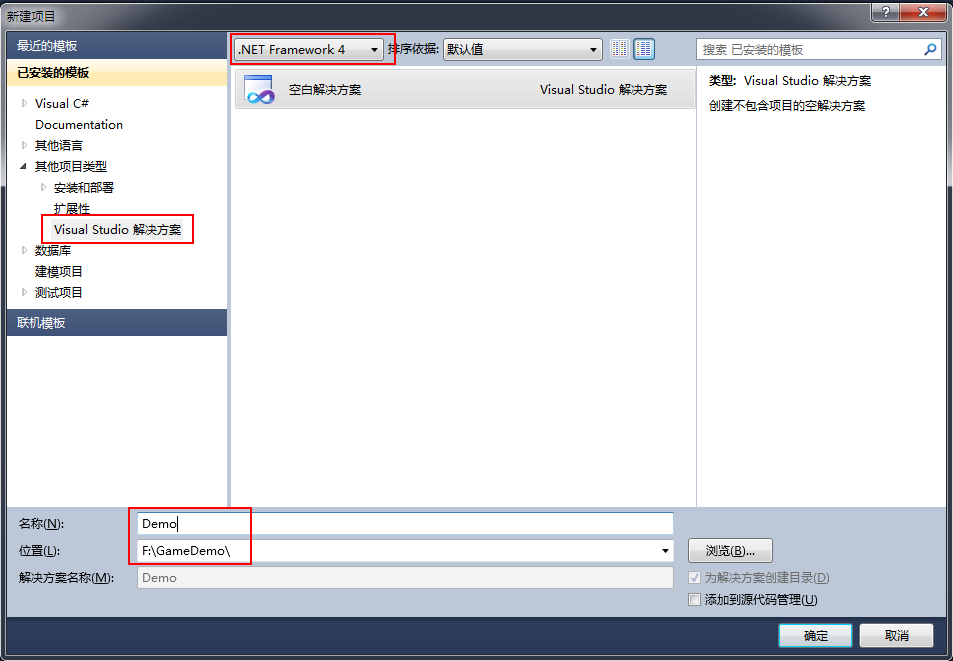
游戏划分成三个库：DemoConfig库（负责存储游戏配置数据） 、DemoData库(负责存储游戏玩家数据)和DemoLog库(游戏玩家日志记录数据)；如图



## 项目搭建

### 创建解决方案

打开VS2010 在菜单上选择 文件 -> 新建 -> 项目；弹出“新建项目”对话窗口，在左则展开“其它项目类型”，选择“Visual Studio解决方案”，选择“.Net Framework 4.0”后，输出项目名称及位置，再点击“确定”；如图：



### 创建项目

在“资源管理器”中添加新的几个项目Model、Lang、Com、Bll、HostServer；**Model项目：**负责从数据库中导出表的数据实体类映射；

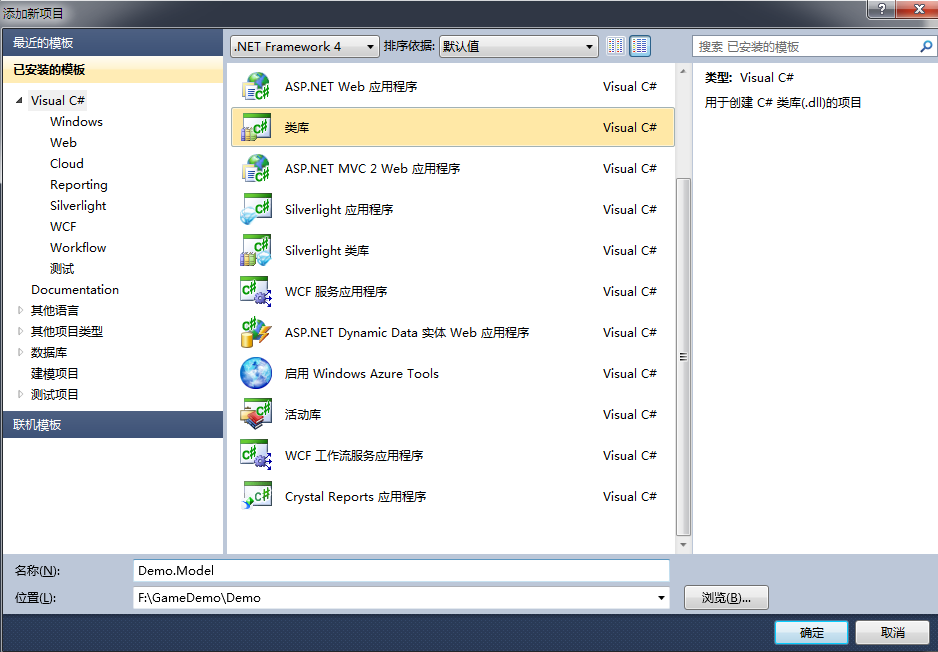
**Lang项目：**负责多语言包；

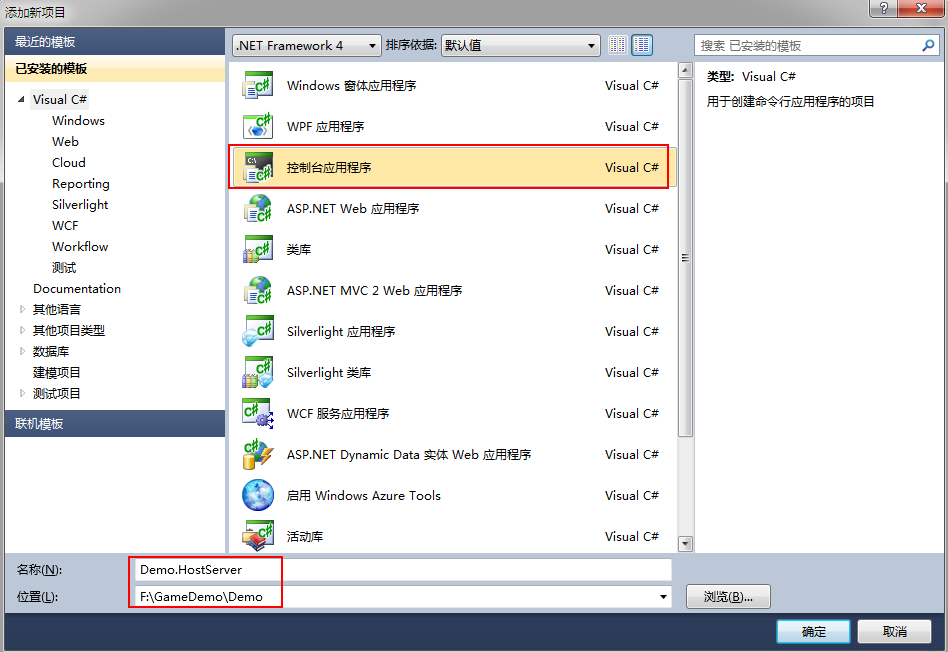
**Com项目：**负责中间层组件扩展处理，及中间层业务实体类；

**Bll项目：**负责业务逻辑处理；

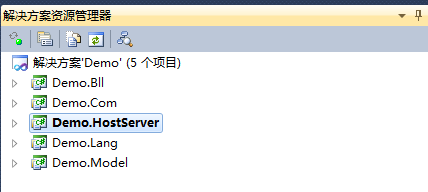
**HostServer项目：**控制台启动程序，及脚本（Python）业务逻辑处理；

如图：

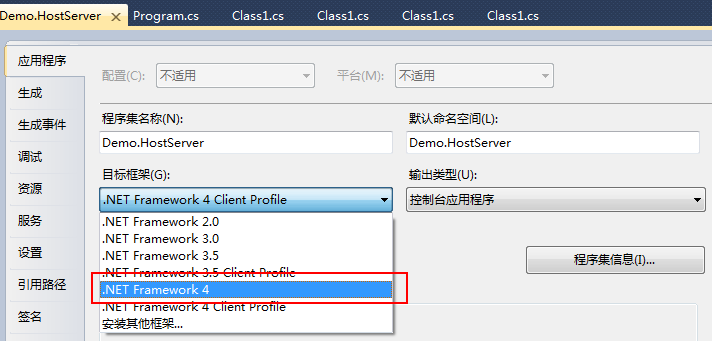




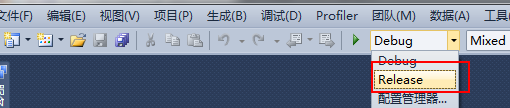
创建项目结果如下：



设置控制台程序“HostServer”属性为“.Net Framework 4.0”



设置成“Release”编译方式



### Model项目

**组件引用**

|  |  |
| --- | --- |
| ****项目**** | **引用路径** |
| Model | **Lib\protobuf-net.dll**  **Lib\ZyGames.Framework.Common.dll**  **Lib\ZyGames.Framework.dll**  **Lib\ZyGames.Framework.Game.dll**  [组件详情](#_Scut开发包) |

**目录划分**

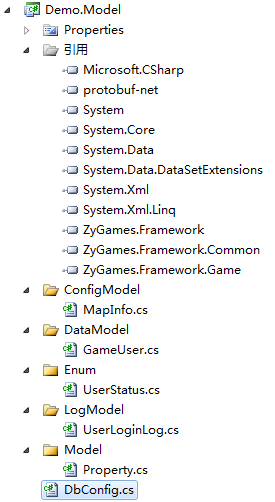
**划分配置库（ConfigModel）、玩家信息库（DataModel）、玩家日志库（LogModel）和自定义类型数据实体（Model）目录存储数据实体类； Enum目录存储自定枚举；**

**DbConfig类**

**在生成实体类时，生成配置ConnectKey项中使用**

|  |
| --- |
| public class DbConfig  {  public const string Config = "DemoConfig";  public const string Data = "DemoData";  public const string Log = "DemoLog";  public const int GlobalPeriodTime = 0;  public const int PeriodTime = 0;  public const string PersonalName = "UserId";  }  //UserId：是玩家库中表的主键字段名称; 红色部分是需要修改的 |

项目建立完成如图：



**实体静态注入配置**

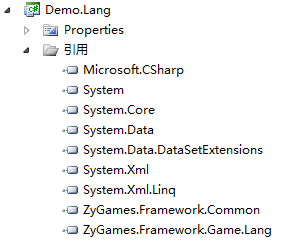
用记事本打开Demo.Model.csproj文件，在结尾增加如下配置:

|  |
| --- |
| <Project>  ... ...  <UsingTask TaskName="ZyGames.Framework.Common.Build.WeavingEntityTask" AssemblyFile="bin\$(Configuration)\ZyGames.Framework.Common.dll" />  <Target Name="AfterBuild">  <WeavingEntityTask SolutionDir=".\\bin\$(Configuration)" FilePattern="Demo.Model.dll" />  </Target>  ... ...  </Project> |

### Lang项目

**组件引用**

|  |  |
| --- | --- |
| ****项目**** | **引用路径** |
| Lang | **Lib\ZyGames.Framework.Common.dll**  **Lib\ZyGames.Framework.Game.Lang.dll** |



负责处理多语言包配置，需要实现中层提供的语言包；以下是定义类

**LanguageManager类**

|  |
| --- |
| public class LanguageManager  {  private static object thisLock = new object();  private static Dictionary<LangEnum, IGameLanguage> \_langTable = new Dictionary<LangEnum, IGameLanguage>();  private static LangEnum \_langEnum;  static LanguageManager()  {  \_langEnum = (ConfigUtils.GetSetting("LanguageType", "0")).ToEnum<LangEnum>();  LanguageHelper.SetLang(\_langEnum);  }  public static IGameLanguage GetLang()  {  return GetLang(\_langEnum);  }  public static IGameLanguage GetLang(LangEnum langEnum)  {  IGameLanguage lang = null;  if (!\_langTable.ContainsKey(langEnum))  {  lock (thisLock)  {  if (!\_langTable.ContainsKey(langEnum))  {  switch (langEnum)  {  case LangEnum.ZH\_CN:  \_langTable.Add(langEnum, new GameZhLanguage());  break;  default:  throw new Exception("Language is error.");  }  }  }  }  lang = \_langTable[langEnum];  return lang;  }  } |

**IGameLanguage接口**

|  |
| --- |
| public interface IGameLanguage : ILanguage  {  #region  /// <summary>  /// 君主帐号  /// </summary>  int SystemUserId { get; }  /// <summary>  /// 玩家名称  /// </summary>  string KingName { get; }  string Date\_Yesterday { get; }  string Date\_BeforeYesterday { get; }  string Date\_Day { get; }  string St1002\_GetRegisterPassportIDError { get; }  string St1005\_NickNameOutRange { get; }  string St1005\_NickNameExistKeyword { get; }  string St1005\_NickNameExist { get; }  string St1006\_PasswordTooLong { get;}  string St1006\_ChangePasswordError { get;}  string St1006\_PasswordError { get;}  string St1066\_PayError { get; }  #endregion  } |

**GameZhLanguage类**

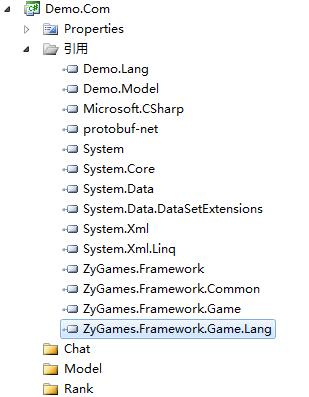
|  |
| --- |
| class GameZhLanguage : BaseZHLanguage, IGameLanguage  {  public int SystemUserId  {  get { return 1000000; }  }  public string KingName  {  get { return "系统"; }  }  public string Date\_Yesterday { get { return "昨天"; } }  public string Date\_BeforeYesterday { get { return "前天"; } }  public string Date\_Day { get { return "{0}天前"; } }  public string St1002\_GetRegisterPassportIDError { get { return "获取注册通行证ID失败!"; } }  public string St1005\_NickNameOutRange { get { return "您的昵称输入有误，请重新输入!"; } }  public string St1005\_NickNameExistKeyword { get { return "您输入的昵称存在非法字符，请重新输入!"; } }  public string St1005\_NickNameExist { get { return "您输入的昵称已存在，请重新输入!"; } }  public string St1006\_PasswordTooLong { get { return "输入错误，请输入4-12位数字或字母!"; } }  public string St1006\_ChangePasswordError { get { return "修改密码失败!"; } }  public string St1006\_PasswordError { get { return "密码格式错误!"; } }  public string St1066\_PayError { get { return "充值失败"; } }  } |

### Com项目

**组件引用**

|  |  |
| --- | --- |
| ****项目**** | **引用路径** |
| **Com** | **Lib\protobuf-net.dll**  **Lib\ZyGames.Framework.Common.dll**  **Lib\ZyGames.Framework.dll**  **Lib\ZyGames.Framework.Game.Lang.dll**  **Lib\ZyGames.Framework.Game.dll**  Model  **Lang** |

划分中间件业务实体（Model），聊天组件（Chat）与排行榜（Rank）等目录，如图：

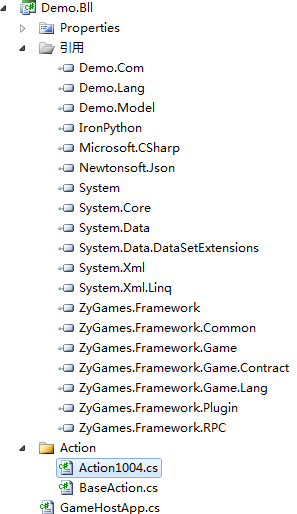


需要使用中间层的功能，请参考《中间层使用文档》

### ****Bll项目****

**组件引用**

|  |  |
| --- | --- |
| ****项目**** | **引用路径** |
| **Bll** | **LibNewtonsoft.Json.dll**  **Lib\IronPython.dll**  **Lib\ZyGames.Framework.Common.dll**  **Lib\ZyGames.Framework.dll Lib\ZyGames.Framework.Plugin.dll**  **Lib\ZyGames.Framework.RPC.dll**  **Lib\ZyGames.Framework.Game.Lang.dll**  **Lib\ZyGames.Framework.Game.dll**  **Lib\** **ZyGames.Framework.Game.Contract.dll**  Model  **Lang**  **Com** |



**功能划分**

创建Action目录划分接口协议处理逻辑；主要提供中间层定义的固定协议接口，如：登录（1004）与建角（1005）及充值中间层接口

**ActionIDDefine类**

|  |
| --- |
| **public class ActionIDDefine**  **{**  **///<summary>**  **///客户端注册Socket**  **///</summary>**  **public const Int16 Cst\_Action100 = 100;**  **///<summary>**  **///错误日志**  **///</summary>**  **public const Int16 Cst\_Action404 = 404;**  **///<summary>**  **///注册通行证ID获取接口**  **///</summary>**  **public const Int16 Cst\_Action1002 = 1002;**  **///<summary>**  **///用户注册**  **///</summary>**  **public const Int16 Cst\_Action1003 = 1003;**  **///<summary>**  **///用户登录**  **///</summary>**  **public const Int16 Cst\_Action1004 = 1004;**  **///<summary>**  **///创建角色**  **///</summary>**  **public const Int16 Cst\_Action1005 = 1005;**  **}** |

**BaseAction类**

|  |
| --- |
| **public abstract class BaseAction : AuthorizeAction**  **{**  **protected BaseAction(short actionID, HttpGet httpGet)**  **: base(actionID, httpGet)**  **{**  **}**  **protected override bool IgnoreActionId**  **{**  **get**  **{**  **//排除不需要登录授权的协议接口**  **return actionId == ActionIDDefine.Cst\_Action404;**  **}**  **}**  **}** |

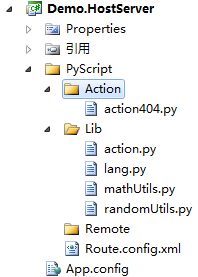
### HostServer项目

**组件引用**

|  |  |
| --- | --- |
| ****项目**** | **引用路径** |
| **HostServer** | **LibNewtonsoft.Json.dll**  **LibNLog.dll**  **Lib\protobuf-net.dll**  **Lib\IronPython.dll**  **Lib\IronPython.Modules.dll**  **Lib\Microsoft.Dynamic.dll**  **Lib\Microsoft.Scripting.dll**  **Lib\ServiceStack.dll**  **Lib\** **ServiceStack.Common.dll**  **Lib\** **ServiceStack.Interfaces.dll**  **Lib\** **ServiceStack.Redis.dll**  **Lib\** **ServiceStack.Text.dll**  **Lib\ZyGames.Framework.Common.dll**  **Lib\ZyGames.Framework.dll Lib\ZyGames.Framework.Plugin.dll**  **Lib\ZyGames.Framework.RPC.dll**  **Lib\ZyGames.Framework.Game.Lang.dll**  **Lib\ZyGames.Framework.Game.dll**  **Lib\** **ZyGames.Framework.Game.Contract.dll**  Model  **Lang**  **Com**  **Bll** |

**功能划分**

**划分PyScript目录，存放Python脚本文件；层次如图：**

****

**Action目录：处理请求与响应的脚本，可以协议生成器工具中Copy部分模板；**

**Lib目录：Python中间层脚本，复制Scut开发包中的PythonLib目录；**

**Remote：应用程序之间内通讯，访问时有IP访问限制；**

**Route.config.xml：是请求路由配置表，格式如下：**

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <config>  <!--Python安装类库路径-->  <lib path="D:\Python\Lib" />  <route-list>  <!--配置Action路由  action：映射的Action代码  path：指定执行的脚本路径  ignoreAuthorize:是否不需要登录授权，true：不需要登录授权  -->  <route action="404" path="Action\action404.py"/>  <route action="1009" path="Action\action1009.py"/>  </route-list>  </config> |

**GameHostApp类**

|  |
| --- |
| class GameHostApp : GameSocketHost  {  private static GameHostApp instance;  static GameHostApp()  {  instance = new GameHostApp();  }  private GameHostApp()  {  }  public static GameHostApp Current  {  get { return instance; }  }  protected override void OnConnectCompleted(object sender, ConnectionEventArgs e)  {  Console.WriteLine("Client:{0} connect to server.", e.Socket.RemoteEndPoint);  }  protected override void OnRequested(HttpGet httpGet, IGameResponse response)  {  try  {  ActionFactory.Request(httpGet, response, null);  }  catch (Exception ex)  {  Console.WriteLine("{0}", ex.Message);  }  }  protected override void OnStartAffer()  {  try  {  //时º¡À间?间?隔?更¨¹新?库a  int cacheInterval = 600;  GameEnvironment.Start(cacheInterval, () => true);  Console.WriteLine("The server is staring...");  }  catch (Exception ex)  {  TraceLog.WriteError("The server start error:{0}",ex);  }  }  protected override void OnServiceStop()  {  GameEnvironment.Stop();  }  } |

**Program类**

|  |
| --- |
| static void Main(string[] args)  {  try  {  GameHostApp.Current.Start();  }  catch (Exception ex)  {  Console.WriteLine(ex.Message);  TraceLog.WriteError("HostServer error:{0}", ex);  }  finally  {  Console.WriteLine("Press any key to exit the listener!");  Console.ReadKey();  GameHostApp.Current.Stop();  }  } |

**宿主程序Config配置**

|  |
| --- |
| <?xml version="1.0"?>  <configuration>  <configSections>  <section name="zyGameBaseBll" type="ZyGames.Framework.Game.Configuration.ZyGameBaseBllSection,ZyGames.Framework.Game"/>  <section name="zyGameBase-GM" type="ZyGames.Framework.Game.Command.GmSection,ZyGames.Framework.Game"/>  <section name="sdkChannel" type="ZyGames.Framework.Game.Sns.Section.SdkChannelSection,ZyGames.Framework.Game"/>  </configSections>  <appSettings>  <!--必须配置  Port:监听端口  Code:产品代码  ServerId:产品游服代码  -->  <add key="Product.Code" value="1" />  <add key="Product.Name" value="" />  <add key="Product.ServerId" value="1" />  <add key="Game.Port" value="9701" />  <!--可选配置  MessageQueuePath:消息队列中创建指定的专用名称  Action.AssemblyName:Python开发不用设置, 当使用C#语言开发的Action接口必须设置  SignKey:请求签名Key  PublishType:发布版本类型，Debug:请求出错会有信息通知客户端（正式发布可删除）  EnableGM:开启可使用GM命令（正式发布可删除）  Python\_IsDebug:设置python可以调试（正式发布可删除）  PythonRootPath:修改Python执行的相对路径（正式发布可删除）  -->  <add key="Game.Action.AssemblyName" value="ZyGames.Demo.Bll"/>  <add key="MessageQueuePath" value=".\private$\DemoCmdSql"></add>  <add key="Product.SignKey" value="44CAC8ED53714BF18D60C5C7B6296000"/>  <add key="PublishType" value="Debug"/>  <add key="EnableGM" value="true"/>  <add key="Python\_IsDebug" value="true" />  <add key="PythonRootPath" value="..\..\PyScript" />  <!--通用组件配置开始  PayDB:充值模块功能使用的数据库配置(ConnectionString:连接串,Server:服务器,Acount:加密后的登录帐号与密码)  Snscenter:用户中心模块使用的数据库配置(ConnectionString:连接串,Server:服务器,Acount:加密后的登录帐号与密码)  注（Acount原串："Uid=sa;Pwd=123" 密钥：BF3856AD）  -->  <add key="PayDB\_ConnectionString" value="Data Source={0};Database=PayDB;{1}; Pooling=true;" />  <add key="PayDB\_Server" value="." />  <add key="PayDB\_Acount" value="39B30ED8D3FA3A3E5B3B4CA4F9039D55" />  <add key="Snscenter\_ConnectionString" value="Data Source={0};Database=snscenter;{1}; Pooling=true;" />  <add key="Snscenter\_Server" value="." />  <add key="Snscenter\_Acount" value="39B30ED8D3FA3A3E5B3B4CA4F9039D55" />  </appSettings>  <connectionStrings>  <add name="DemoConfig" providerName="" connectionString="Data Source=.;Database=DemoConfig;Uid=sa;Pwd=123; Pooling=true;"/>  <add name="DemoData" providerName="" connectionString="Data Source=.;Database=Demo1Data;Uid=sa;Pwd=123; Pooling=true;"/>  <add name="DemoLog" providerName="" connectionString="Data Source=.;Database=Demo1Log;Uid=sa;Pwd=123; Pooling=true;"/>  </connectionStrings>  <!--业务层自定义配置-->  <zyGameBaseBll>  <login defaultType="ZyGames.Framework.Game.Sns.Login36you,ZyGames.Framework.Game">  <retailList>  <add id="0000" type="ZyGames.Framework.Game.Sns.Login36you,ZyGames.Framework.Game" args="Pid,Pwd,DeviceID"/>  </retailList>  </login>  </zyGameBaseBll>  <zyGameBase-GM>  <command>  <!--GM:cache-->  </command>  </zyGameBase-GM>  <startup>  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>  </startup>  </configuration> |

## 定义协议

打开协议生成器工具，增加一个“Demo”项目方案，接着在增加协议（或从现有项目中Copy相同的接口协议），接着定义客户端提供的请求参数和服务器下发的参数（支持多行的格式，字段类型：Record与End组合）；定义好客户端与服务器之前通讯的接口后，使用自动生成的服务端Python代码（客户端使用Lua脚本代码）复制到已创建的Python接口文件，如图：

