# X Game Core服务器开发框架

## 简介

X Game Core 简称 XGC 是我积累下的一套跨平台的服务器开发框架。该框架包含以下几个部分

* common：通用底层库，内含多个游戏服务器开发常用的组件
* net：基础网络组件，基于asio封装而成。
* net\_module：服务器模块，基于Net的更丰富的包装，用于快速建立服务器间的数据连接。可用于GateServer和GameServer之间通过一个Sock建立的虚拟数据链路。
* database：数据库组件，用于快速上手MySql数据库操作。
* nosql：Redis的C++封装。
* core：基于游戏中逻辑的封装
* framework：一个服务器框架模块，封装了一些常用的服务器功能

## Common 通用底层库

* auto\_handle.h :自动列表
* csv\_reader.h : csv 文件读取封装
* Ini\_reader.h : ini 文件读取封装
* datetime.h : 时间日期和时间片的封装
* debuger.h : 调试技术相关的封装
* encoding.h: utf8 – gbk字符编码转换
* exception.h: 主要用于windows下的异常捕获
* filewatcher.h: 文件监视器组件，用于监视文件改动。
* freelist.h : 空闲列表的封装
* geometry.h : size，rect，point的封装
* getopt：命令行解析
* http.h : windows下的HTTP客户端，使用wininet API 编写
* logger.h : 一个日志系统
* md5.h : MD5校验码的封装。
* pugixml.hpp : 一个xml读取封装，读取和操作速度比TiXml快，使用方便。（第三方）
* serialization.h : 缓冲序列化器
* sha1.h : SHA1 算法封装
* thread\_pool.h : 线程池封装
* timer.h : 一个固定时间间隔的定时器
* xbuffer.h : 缓冲区封装
* xsystem.h : 跟系统相关的一些算法封装
* xvariant.h : 变体变量的封装

### auto\_handle

auto\_handle是自动列表的模式。该模式在对象生成时，向管理器请求一个句柄，并在对象销毁时归还。如此做的好处是，当你需要记录对象指针的时候可以只记录其句柄。并在需要的时候向该对象的管理器查询该句柄指向的对象指针。此模式可以有效杜绝野指针问题。

此封装使用句柄Chunk，可以动态增长，并有效控制内存，且不影响句柄分配效率。该模块适用于句柄数量较稀疏的环境中。否则会有效率损失。

### csv\_reader

csv\_reader 是csv格式文件的读取封装（不支持写）。支持excel中单元格多行的情况，支持且自动识别utf8编码，文件处理速度快，接口使用方便。

### ini\_reader

ini\_reader 是ini格式文件的读取封装（不支持写）。在ini的基础上，该类对ini格式做了扩展。

1. 可以多个节使用相同名称
2. 可以多个键使用相同名称
3. 可以引用外部文件
4. 可以使用%Section.Key%引用已定义过的值
5. 文件处理速度块。

该类支持且自动识别utf8编码，文件处理速度快，接口使用方便。

### datetime

datetime 是日期时间的封装，当算法需要处理时间相关的问题是，使用该类会更加简单。datetime::to\_string 可以将存储的时间转换为字符串，也可以使用datetime::convert将字符串转为时间值。

timespan 是时间差的封装，可以与datetime 一起做时间相关的运算。

### encoding

编码的封装，有较为方便的接口实现utf8 <==> mbs 的编码转换

### filewatcher

文件监视器，用于监视文件变更（修改、访问、新增、删除等）。加入了合并机制，一段时间内的事件可合并上报。

### serialization

缓冲区序列化器，用于将数据序列化至缓冲，可通过重载operator<< 函数的方式，扩展序列化的类型。现已内置了基本数据类型和STL容器的序列化重载。

### thread\_pool

一个线程池的封装，使用C++11 特性编写。接口类似于std::bind，返回值使用std::future

### timer

一个固定时间间隔的定时器。其效率要比基于堆的定时器更高，在适当的选择精度后其延迟可以达到游戏内需求。

### xbuffer

缓冲区封装包含了多种缓冲区类型

1. 缓冲区
   1. static\_buffer 静态缓冲，类内部有固定大小的缓冲区，缓冲内存可分配在栈上。
   2. reference\_buffer 引用缓冲，自身并不持有缓冲，只有缓冲区信息（指针，大小）。
   3. shared\_memory\_buffer 共享内存，使用系统共享内存
2. 缓冲读写
   1. separate\_buffer\_recorder 分离的缓冲读写器，读写指针不使用缓冲内存。
   2. union\_buffer\_recorder 联合的缓冲读写器，读写指针使用缓冲内存
3. 缓冲类型
   1. linear\_buffer 线性缓冲
   2. ring\_buffer 环形缓冲

### xsystem

系统相关的算法封装

1. get\_module\_name ：获取执行文件的文件名
2. get\_module\_path ：获取执行文件的文件路径
3. get\_absolute\_path ：获取相对与执行文件的路径
4. get\_relative\_path ：获取两个目录对比下来的相对路径
5. is\_absolute\_path ：判断一个路径是否绝对路径
6. make\_dirs ：创建目录（可递归创建）
7. list\_directory : 列出指定目录下的目录和文件（可递归）
8. get\_process\_memory\_usage ：获取进程使用的内存
9. get\_system\_memory\_usage ： 获取系统使用的内存
10. get\_process\_cpu\_usage ：获取进程的CPU占用情况
11. get\_system\_cpu\_usage ： 获取系统的CPU占用情况

## Core 逻辑底层

### XAttribute

XObject属性的封装管理类

### XClassInfo

XObject RTTI的封装

### XObject

XObject是Core的顶层类，其他游戏内对象都应从XObject派生。

XObject使用了自动列表技术，每一个XObject的派生类在产生实例的时候都会被分配一个句柄，通过存储对象句柄而不是指针可以有效防止野指针的产生。

XObject提供了属性系统的初始化和相关接口。所有XObject对象的派生类都可以访问自己和其父类的属性。属性系统简化了属性的定义，并提供了属性版本的管理，使游戏在增删属性定义后的版本继续兼容之前版本的属性存档。

XObject提供了一个事件模型，保证游戏对象产生的事件可以送达对其事件感兴趣的其他对象。同时，事件也会向其组织结构的上层传递。

### XObjectNode

XObjectNode 提供了子对象的管理功能

### XBag

XBag 是一个背包管理的封装。该背包管理支持事务，在操作失败的情况下可以对已操作的物品进行回滚。

### XGameMap

XGameMap 是一个格子地图的管理类封装。专门用于管理M \* N的格子地图。该类中对碰撞数据（按格子）和视野（按m \* n一组）进行了区分。使其可以在视野管理上更为高效。

XGameMap 还内置了一套可配置的事件系统，包含计数器和定时器再配合XObject的事件模型，可以配置出复杂的地图玩法。

XGameMap 也提供了多种地图内对象的捡取算法，现在有基于视野、圆形、扇形、圆环、扇环、矩形的检取函数。

XGameMap也提供了客户端事件的触发机制，可以将场景状态进行记录。

## Framework 服务器框架层

### ServerBase

使用net\_module库作为网络支撑，使用database库作为数据库支撑。使用配置文件来配置服务器信息，数据库信息以及网络联通情况。

通过使用Framework在不写一行代码的情况下较为容易的搭建出整个服务器架构，使开发人员可以将注意力放在逻辑层的实现。

### ServerParams

一个集中式的变量存储功能。ServerParams分客户端和服务器，服务器会将上传的变量持久化，并同步给其他客户端。该模块主要用于全局变量的同步。

### ServerCommand

服务器指令的封装。可进行多级解析，对较复杂的指令可使用别名功能进行简化。

### ServerRefresh

服务器刷新功能，将刷新功能转发至脚本，用脚本修改较为灵活的特性定义一次刷新所需要做的行为。角色上线时对刷新的判定应由逻辑层来完成。

### ServerEvent

服务器事件模型，分为同步事件提交（EmmitEvent）和异步事件提交（PostEvent），主要用于成就系统、延迟执行等环境下。

## 其他

其他技术选型

1. 脚本使用luajit
2. 脚本和C++通讯使用LuaBridge（有修改）
3. Windows下内存泄漏检查使用VLD
4. 网络库底层使用asio

其他代码说明

1. Sample\network目录下是一个文件服务器和对应的客户端。主要用于网络流量压力测试！
2. Sample\chat目录下是一个完整的聊天服务器，它还可以用于测试网络连接压力测试。
3. Sample\servers目录下是一个服务器架构示例，包含gate\_server, game\_server, 和client。用于演示framework的使用。