

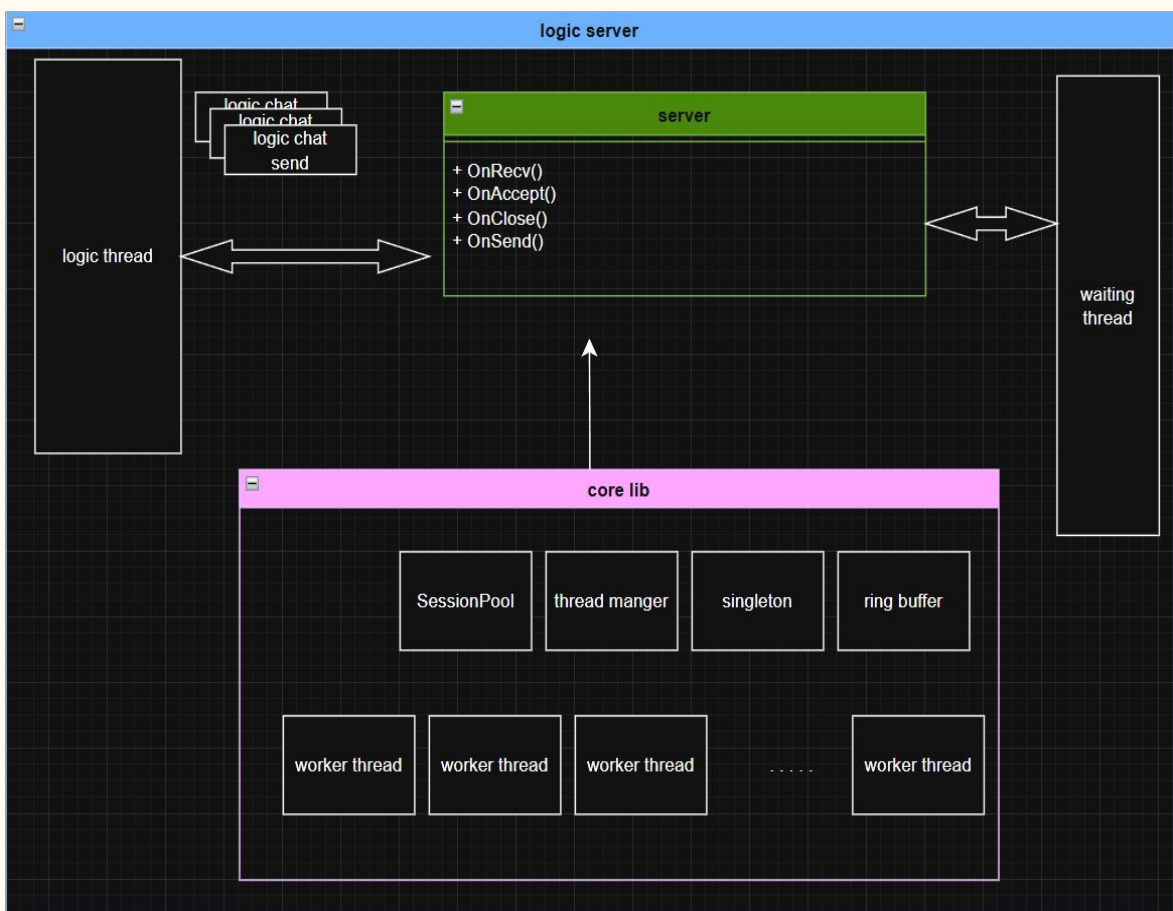
포트폴리오

문진웅

010-5120-2097

jazzmaster0415@gmail.com

git : <https://github.com/true-bear/logic-server>



<core : 정적 라이브러리>

네트워크 처리, 세션 스레드 매니저 등 전반적인 IO 및 세션 부분의 기능들을 정의하는 라이브러리 프로젝트

<server>

- 코어를 상속하고 io들어오는 패킷들을 처리하고 로직을 실행하는 서버
- 로직을 위한 로직 스레드
- 대기열을 위한 대기열 스레드 구현

 Chat Stress Client

Dummy Count: 4000 Dummy Connect Dummy Disconnect Dummy Chat Start Dummy Chat Stop

Single Connect Single Disconnect Single Chat Single Chat Stop

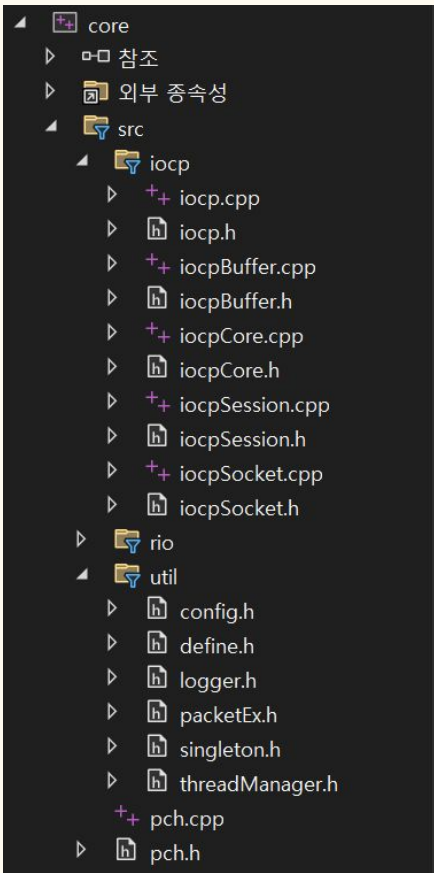
[illegible]

```

C:\Users\Wtuebaev\Desktop\Wireshark\Engine\WServer\Wsf64WRelease\WLogico
2025-05-18 14:09:23.273 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1448 : 22
2025-05-18 14:09:23.289 INFO [Chat][logic_chat.cpp:34] Send uid:1448 msg: Dummy msg 131'
2025-05-18 14:09:23.382 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1041 : 22
2025-05-18 14:09:23.398 INFO [Chat][logic_chat.cpp:34] Send uid:1041 msg: Dummy msg 132'
2025-05-18 14:09:23.488 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1247 : 22
2025-05-18 14:09:23.503 INFO [Chat][logic_chat.cpp:34] Send uid:1247 msg: Dummy msg 133'
2025-05-18 14:09:23.583 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1449 : 22
2025-05-18 14:09:23.598 INFO [Chat][logic_chat.cpp:34] Send uid:1449 msg: Dummy msg 134'
2025-05-18 14:09:23.677 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1042 : 22
2025-05-18 14:09:23.692 INFO [Chat][logic_chat.cpp:34] Send uid:1042 msg: Dummy msg 135'
2025-05-18 14:09:23.787 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1248 : 22
2025-05-18 14:09:23.803 INFO [Chat][logic_chat.cpp:34] Send uid:1248 msg: Dummy msg 136'
2025-05-18 14:09:23.882 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1043 : 22
2025-05-18 14:09:23.890 INFO [Chat][logic_chat.cpp:34] Send uid:1043 msg: Dummy msg 137'
2025-05-18 14:09:23.985 INFO [OnRecv][logic.cpp:84] [INFO ][Chat][logic_chat.cpp:34] Recv packet - uId:1450 : 22
2025-05-18 14:09:24.087 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1249 : 22
2025-05-18 14:09:24.102 INFO [Chat][logic_chat.cpp:34] Send uid:1249 msg: Dummy msg 139'
2025-05-18 14:09:24.173 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1841 : 22
2025-05-18 14:09:24.189 INFO [Chat][logic_chat.cpp:34] Send uid:1841 msg: Dummy msg 140'
2025-05-18 14:09:24.275 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1250 : 22
2025-05-18 14:09:24.291 INFO [Chat][logic_chat.cpp:34] Send uid:1250 msg: Dummy msg 141'
2025-05-18 14:09:24.376 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1044 : 22
2025-05-18 14:09:24.392 INFO [Chat][logic_chat.cpp:34] Send uid:1044 msg: Dummy msg 142'
2025-05-18 14:09:24.479 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1641 : 22
2025-05-18 14:09:24.494 INFO [Chat][logic_chat.cpp:34] Send uid:1641 msg: Dummy msg 143'
2025-05-18 14:09:24.575 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1451 : 22
2025-05-18 14:09:24.590 INFO [Chat][logic_chat.cpp:34] Send uid:1451 msg: Dummy msg 144'
2025-05-18 14:09:24.677 INFO [OnRecv][logic.cpp:84] [INFO ][Chat][logic_chat.cpp:34] Recv packet - uId:1251 : 22
2025-05-18 14:09:24.776 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1541 : 22
2025-05-18 14:09:24.792 INFO [Chat][logic_chat.cpp:34] Send uid:1541 msg: Dummy msg 146'
2025-05-18 14:09:24.878 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1045 : 22
2025-05-18 14:09:24.896 INFO [Chat][logic_chat.cpp:34] Send uid:1045 msg: Dummy msg 147'
2025-05-18 14:09:24.974 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1842 : 22
2025-05-18 14:09:24.990 INFO [Chat][logic_chat.cpp:34] Send uid:1842 msg: Dummy msg 148'
2025-05-18 14:09:25.076 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1452 : 22
2025-05-18 14:09:25.077 INFO [Chat][logic_chat.cpp:34] Send uid:1452 msg: Dummy msg 149'
2025-05-18 14:09:25.178 INFO [OnRecv][logic.cpp:84] Recv packet - uId:1542 : 22
2025-05-18 14:09:25.194 INFO [Chat][logic_chat.cpp:34] Send uid:1542 msg: Dummy msg 150'
2025-05-18 14:09:34.378 INFO [Waiting Enqueue] server.cpp:206 Session 999 pushed to waiting queue
2025-05-18 14:09:34.379 INFO [Waiting Enqueue] server.cpp:206 Session 998 pushed to waiting queue
2025-05-18 14:09:34.380 INFO [Waiting Enqueue] server.cpp:206 Session 997 pushed to waiting queue
2025-05-18 14:09:34.381 INFO [Waiting Enqueue] server.cpp:206 Session 996 pushed to waiting queue
2025-05-18 14:09:34.382 INFO [Waiting Enqueue] server.cpp:206 Session 995 pushed to waiting queue
2025-05-18 14:09:34.383 INFO [Waiting Enqueue] server.cpp:206 Session 994 pushed to waiting queue
2025-05-18 14:09:34.385 INFO [Waiting Enqueue] server.cpp:206 Session 993 pushed to waiting queue
2025-05-18 14:09:34.386 INFO [Waiting Enqueue] server.cpp:206 Session 992 pushed to waiting queue
2025-05-18 14:09:34.387 INFO [Waiting Enqueue] server.cpp:206 Session 991 pushed to waiting queue
2025-05-18 14:09:34.388 INFO [Waiting Enqueue] server.cpp:206 Session 990 pushed to waiting queue
2025-05-18 14:09:34.389 INFO [Waiting Enqueue] server.cpp:206 Session 989 pushed to waiting queue
2025-05-18 14:09:34.390 INFO [Waiting Enqueue] server.cpp:206 Session 988 pushed to waiting queue
2025-05-18 14:09:34.391 INFO [Waiting Enqueue] server.cpp:206 Session 987 pushed to waiting queue
2025-05-18 14:09:34.395 INFO [Waiting Enqueue] server.cpp:206 Session 986 pushed to waiting queue
2025-05-18 14:09:34.397 INFO [Waiting Enqueue] server.cpp:206 Session 985 pushed to waiting queue
2025-05-18 14:09:34.398 INFO [Waiting Enqueue] server.cpp:206 Session 984 pushed to waiting queue
2025-05-18 14:09:34.399 INFO [Waiting Enqueue] server.cpp:206 Session 983 pushed to waiting queue
2025-05-18 14:09:34.400 INFO [Waiting Enqueue] server.cpp:206 Session 982 pushed to waiting queue
2025-05-18 14:09:34.401 INFO [Waiting Enqueue] server.cpp:206 Session 981 pushed to waiting queue
2025-05-18 14:09:34.402 INFO [Waiting Enqueue] server.cpp:206 Session 980 pushed to waiting queue

```

00_core



/iocp

- iocpCore class

: 라이브러리의 실행 메인 로직
: 세션풀을 통한 클라이언트 연결 관리
: Accept, Send, Recv 이벤트 처리
: 서버 초기화 및 종료 처리

- iocpSession class

: IO처리를 담당하는 세션 클래스

- iocp class

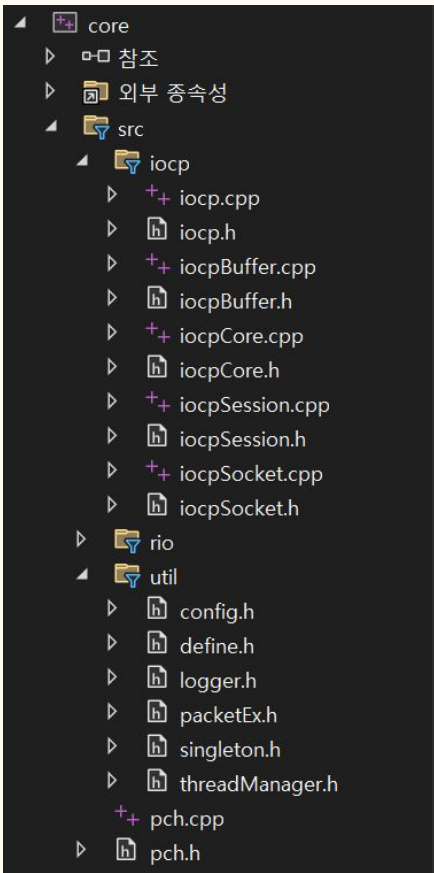
: iocp 핸들을 이용해서 WINAPI들을 모아서 래핑한 클래스

- iocpSocket class

: init, bind 등 소켓 함수의 기능을 모아놓은 클래스

- iocpBuffer: 원형 버퍼 클래스

00_core



/util

- config

: ini 파일 불러와서 서버 설정 세팅해주는 함수

- packetEx class

: 로직에 담은 패킷의 틀을 정의한 클래스

- singleton class

: 싱글톤 패턴 정의 클래스

- threadManager class

: 스레드 생성 파괴를 관리하는 클래스. 스레드 생성, 파괴를 무조건 매니저를 통해 일관적으로 하기 위함

00_core

worker thread

- GetQueuedCompletionStatusEx : io를 지정된 갯수만큼 한꺼번에 가져와서 처리
- OverlappedEx : 오버랩드 상속받아서 확장 구조체 정의 한 모델로 사용
- 타입에 따라서 OnClose, OnRecv, OnSend, OnAccept 호출

session pool

- 잦은 힙 할당 해제 대신 풀을 미리 만들어서 연결만 해주는 방식으로 구현

```
void Core::WorkerThread()
{
    while (mIsRunThread)
    {
        loop::IoEvents events;
        GQSE(events, 5);

        for (int i = 0; i < events.m_eventCount; ++i)
        {
            auto& getIoEvent = events.m_array[i];

            if (getIoEvent.lpOverlapped == nullptr)
            {
                LOG_INFO("Core", "thread shutdown");
                return;
            }

            unsigned long ioSize = getIoEvent.dwNumberOfBytesTransferred;
            OverlappedIoEx* over = reinterpret_cast<OverlappedIoEx*>(getIoEvent.lpOverlapped);

            if (!over)
            {
                LOG_ERR("Core", "Overlapped nullptr");
                continue;
            }

            LOG_ERR("Core", "Overlapped nullptr");
            continue;

            int sessionId = over->mUID;
            IO_TYPE ioType = over->mIOType;

            auto session = GetSession(sessionId);
            if (!session)
            {
                LOG_ERR("Core", "session nullptr id: {}", sessionId);
                continue;
            }
        }
    }
}
```

```
bool Core::CreateSessionPool()
{
    int total = mMaxSession + mMaxWaiting;

    for (auto i = 0; i < total; ++i)
    {
        auto session = std::make_unique<ClientSession>();
        if (!session)
            return false;

        session->SetUniqueId(i);

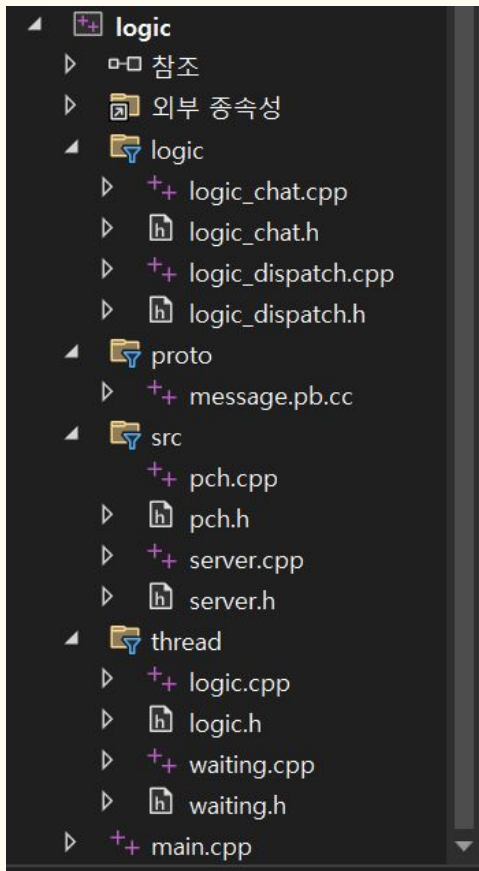
        const auto& listenSocket = mListenSocket.GetSocket();
        if (!session->AcceptReady(listenSocket, i))
            return false;

        mSessionPool.emplace(i, std::move(session));
    }

    return true;
}

ClientSession* Core::GetSession(unsigned int uid) const
{
    auto it = mSessionPool.find(uid);
    return (it != mSessionPool.end()) ? it->second.get() : nullptr;
}
```


01_logic serve



Logic

/logic

:로직 처리 및 정의 하는 부분. 추후에 패킷으로 로직을 구현시 이곳에서 하면됨

: LogicDispatch에서 함수 객체를 미리 등록해서 map으로 빠르게 찾을 수 있게 구현

```
3
4
5 void LogicDispatch::Register(size_t packetType, RecvFunc func)
6 {
7     mRecvFuncMap[packetType] = std::move(func);
8 }
9
10 void LogicDispatch::Dispatch(size_t packetType, loopSession* session, const char* data, uint16_t size)
11 {
12     if (auto iter = mRecvFuncMap.find(packetType); iter != mRecvFuncMap.end())
13     {
14         iter->second(session, data, size);
15     }
16 }
17
```

01_logic server

Logic thread

- DispatchPacket() OnRecv 처리가 된 패킷들을 boost::lockfree::queue에 넣어주는 함수

- 스레드 큐에서 빼와서 dispatch로 분기를 태워서 처리를 해주는 방식

```
37 void LogicManager::RunThread()
38 {
39     while (mRunning.load()) {
40         PacketEx* packet = nullptr;
41         if (!mPacketQueue.pop(packet))
42             continue;
43         auto session = mGetSessionObject(packet->mUID);
44         if (session)
45         {
46             const char* packetStart = packet->mData.data();
47             const char* protoStart = packetStart + sizeof(uint16_t);
48
49             PacketHeader header;
50             if (!header.ParseFromArray(protoStart, packet->mSize - sizeof(uint16_t)))
51             {
52                 LOG_ERROR("Logic", "PacketHeader 파싱 실패");
53                 continue;
54             }
55
56             PacketType type = header.type();
57             mDispatcher.Dispatch(static_cast<size_t>(type), session, packet->mData.data(), packet->mSize);
58         }
59         else
60         {
61             LOG_ERROR("Logic", "session nullptr");
62         }
63     }
64     else
65     {
66         std::this_thread::sleep_for(std::chrono::milliseconds(1));
67     }
68 }
69
70 LOG_INFO("Logic", "LogicThread 종료");
71
72 void LogicManager::DispatchPacket(const int sessionId, const char* data, uint16_t packetSize)
73 {
74     PacketEx* pkt = new PacketEx(sessionId, packetSize, data);
75     if (!mPacketQueue.push(pkt))
76     {
77         delete pkt;
78         LOG_ERROR("Logic", "packet push failed");
79     }
80
81     LOG_INFO("OnRecv", "Recv packet = uid:0 : 0", sessionId, packetSize);
82 }
```


01_logic server

```
void WaitingManager::RunThread()
{
    while (mRunning.load())
    {
        std::this_thread::sleep_for(std::chrono::milliseconds(10));

        auto session = TryDequeue();
        if (!session)
            continue;

        if (LogicServer::Get().HasFreeSlot())
        {
            LogicServer::Get().BindSession(session);
        }
        else
        {
            Enqueue(session);
            continue;
        }
    }
}

{
    std::lock_guard<std::mutex> lock(mActiveSessionLock);
    if ((int)mActiveSessionMap.size() < mMaxSession)
    {
        mActiveSessionMap[uid] = session;
        session->RecvReady();
    }
    else
    {
        WaitingPacket packet;
        auto header = new PacketHeader();
        header->set_type(PacketType::WAITING);
        header->set_length(packet.ByteSizeLong());

        packet.set_allocated_header(header);
        packet.set_message("waiting...");
        packet.set_waiting_number(WaitingManager::Get().Size());

        int size = packet.ByteSizeLong();
        std::vector<char> buf(size);
        if (!packet.SerializeToArray(buf.data(), size))
        {
            LOG_ERR("WaitingPacket", "Serialize 실패");
            return;
        }

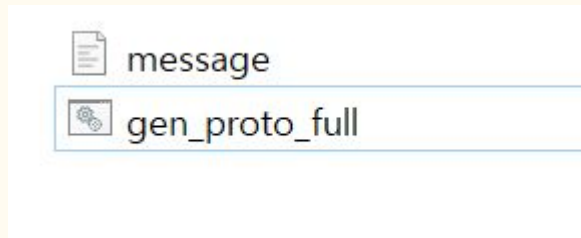
        session->SendPacket(buf.data(), size);

        WaitingManager::Get().Enqueue(session);
        LOG_INFO("Waiting Enqueue", "Session {} pushed to waiting queue", uid);
    }
}
```

Waiting thread

- OnAccept 시 세션의 여유가 있으면 수신을 걸어주고, 없으면 대기열 큐 스레드에 넣어주는 로직
- 웨이팅 스레드에서는 주기적으로 돌면서 세션 여유분 체크하고 조건에 부합하면 소켓 할당하고 수신을 걸어줌

02_protobuf



protobuf

- gen_proto_pull.bat: 윈도우용 배치 파일. message.proto를 컴파일하고 지정된 디렉토리에 복사하는 간단한 배치파일 구현
- 서버는 c++ 클라이언트는 c#으로 구현해서 간단하게 배치파일 실행으로 이기종 언어의 통신을 도와줌