1. DB 연동

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명 텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명

저번 주, DB 세팅으로 visual studio에서 DB 작업을 진행할 수 있었다.

이를 이용하여 게임서버프로그래밍 과목의 텀프로젝트에 DB를 연동하고, 후에 졸업 작품에 같은 방식으로 연동 할 예정이다.

우선 직접 SQL을 이용하지 않기 위해 저장 프로시저를 만들어 주었다.

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Insert\_player – player 테이블에 데이터를 추가하는 프로시저

텍스트, 폰트, 스크린샷, 번호이(가) 표시된 사진

자동 생성된 설명

Select\_All – player 테이블의 모든 값을 확인하기 위해 만든 프로시저

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

Select\_player – player 테이블에서 player\_name 값으로 하나의 player 데이터를 추출하여 확인하는 프로시저

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

Update\_player – player 테이블의 한 데이터를 Update하기 위한 프로시저

폰트, 텍스트, 스크린샷, 그래픽이(가) 표시된 사진

자동 생성된 설명

서버에서 DB관련 작업을 담당하는 class도 만들어 주었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

DBMGR은 생성될 때 DB로 연결하고, 삭제할 때 DB와 연결을 끊는 작업을 하게 했다.

싱글톤으로 하지 않은 이유는 DBMGR이 한 객체이고 다수의 Thread가 접근하여 할 때 공용 데이터에 접근하는 행위이기 때문에 위험하여 lock과 같은 작업들이 필요하다. 이런 작업은 성능에 큰 영향을 주기 때문에 DB로 연결하는 Handle을 Thread마다 따로 두려고 했다.

 Thread마다 DBMGR를 생성하는 모습

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

connect()와 disconnect()작업은 위와 같이 ODBC관련 문서와 ‘게임서버프로그래밍’ 과목에서 배운 것을 바탕으로 제작했다.

이제 DBMGR을 통해 모든 DB작업을 할 것이다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

미리 만든 4개의 저장 프로시저를 ODBC를 통해 호출하는 함수를 만들었다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

예시로 인자가 있는 호출이면 wchar\_t형 문자열 끝에 붙여 SQLExecDirect를 호출하면 해당하는 프로시저에 원하는 파라미터를 포함시켜 실행할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그리고 값 반환되는 프로시저라면 SQLBindCol()를 이용하여 해당하는 변수에 값을 불러온다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

이제는 Get\_SELECT\_PLAYER() 함수를 통해 해당하는 Player\_name이 있는지 판별하여 로그인 여부를 결정할 수 있다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

그리고 클라이언트가 종료할 때 Set\_UPDATE\_PLAYER로 DB에 현재 클라이언트의 플레이어 정보를 Update작업 하도록 하여 다음번에 접속하면 종료 전 상태의 플레이어로 시작할 수 있다.

기말고사과 텀프로젝트등 학업의 이유로 졸업 작품에 관한 작업을 많이 진행하지 못했지만, 다음 주차부터 작업속도를 높일 것이다.