RELAY CHAT

201804005 김민서

나만의 채팅 프로토콜의 프로젝트 제목은 RELAY CHAT으로 정했다. 단순하게 여러 클라이언트가 자유롭게 통신하고 릴레이 경주처럼 대화하기 떄문이다. 프로토콜은 크게 메시지 전송을 하는 SendMessage() 메시지 확인을 하는 SendMessageView() 서버에 접속 중인 클라이언트를 조회하는 ShowCurrentClient(), ShowAccessLog()로 나누어진다. 클라이언트간 메시지 전송 메시지 포맷은 이렇다.

C: string.Format("{0}<{1}>", receiver, message);

S: SendMsgToClient((List<string> msgList, string sender)) 로 이루어진다.

클라이언트는 Format() 메소드를 이용하여 문자열을 출력한다. Format() 메소드를 이용하여 문자열을 출력하는 방법은 중괄호를 이용하여 코드에서 사용하는 변수나, 서식을 넣어준다. 중괄호 내부에 넣는 숫자는 첨자라고 하는데 첨자의 시작은 0부터 시작한다. 문자열이 끝나고 콤마(,)뒤에 순서대로 0, 1에 맞게 넣을 변수들을 입력한다. 즉 메시지를 전송할떄 수신자와 메시지를 전송한다. 서버는 메시지 리스트와 송신할 클라이언트 정보를 리스트로 받고 다른 클라이언트에게 전송한다. 클라이언트에서 보낸 메시지를 확인하는 메시지 포맷은 이렇다.

C:string.Format("[{0}] 수신자 : {1}, 메시지 : {2}", DateTime. Now.ToString("yyyy-MM-dd HH:mm:ss"), receiver, message)

S: SendMessageView()= string.Format("[{1}] {2}",DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"), sender,message); 로 이루어진다.

메시지 전송과 마찬가지로 클라이언트는 서버에 Format() 메소드를 이용해서 수신자와 메시지에 현재 시각정보를 함께 서버에 전송한다. 그러면 서버는 클라이언트에게 받은 메시지의 정보를 읽고 SendMessageView()로 나타내준다. ShowCurrentClient()는 현재 접속 중인 클라이언트의 이름을 보여주고 ShowAccessLog()는 접속 일시까지 보여준다. 메시지 포맷은 이렇다.

C: ClienntName = name

S: Show CurrentClient= ClientData currentClient = newClientData(newClient);

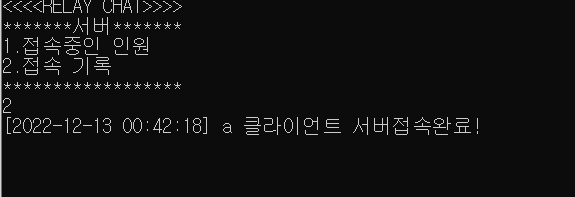
S: Show accessLog = string.Format("[{0}] {1} Access Server", DateTime.Now.ToString("yyyy-MM-dd HH:mm:ss"), client.clientName);

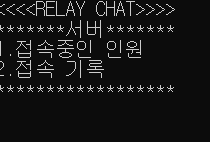
우선 클라이언트는 이름을 입력하고 서버에 접속한다. Show CurrentClient는 새로운 클라이언트 데이터를 받으면 현재 클라이언트 데이터에 입력해서 현재 어느 클라이언트가 서버에 접속했는지 알려준다. Show accessLog는 접속한 클라이언트가 정확히 언제 접속했는지 보여준다. 마찬가지로 Format()메소드를 이용해서 현재 일시 데이터와 클라이언트 이름을 받아서 등록한다.

relay chat은 1:N 채팅 서비스다. 여러 클라이언트가 자유롭게 통신 가능하고 다른 클라이언트에게 보낸 메시지도 확인 가능하다. 메시지에는 수신자와 내용 일시도 정확하게 기록되어있다. 또한 서버는 현재 접속해 있는 클라이언트가 무엇인지 알 수 있고 언제 접속했는지 기록도 볼 수 있다.

사용자 인터페이스는 이렇다.

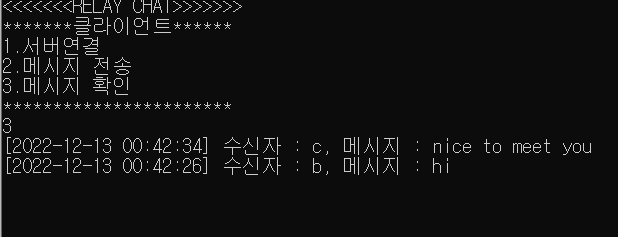
-서버-





-클라이언트-





실행방법도 매우 간단하다. 첫번쨰로 클라이언트는 서버에 먼저 접속해야한다. 이름을 입력하고 클라이언트에 접속했으면 다음에 대화할 클라이언트도 서버에 접속하고 메시지를 주고받을 수 있다. 채팅을 하다 자신이 보낸 메시지가 궁금하다면 메시지를 확인할 수 있다.

결론적으로 배운점도 많지만 아쉬움이 남는 프로젝트다. 클라이언트에 ReceiveMessageVIew() 기능이 작동되지 않았고 처음에는 서버에 채팅로그를 구현에서 클라이언트끼리 주고받는 모든 대화를 한눈에 보기좋게 만들고 싶었는데 이부분이 해결되지 못했다. 프로젝트를 진행하면서 c#에 대해 이론적인 지식이 많이 부족하다고 느꼈고 코딩할수록 다양한 오류를 발견하고 수정해보면서 많은것을 알게되었다. ConcurrentDictionary클래스는 추가하거나 제거하려고 시도하기 전에 키의 존재 여부를 먼저 확인하는 코드가 필요하지 않게 하는 편리한 몇 가지 메서드를 제공한다. key는 클라이언트의 IP의 네번째 옥텟, Value는 연결된 클라이언트 객체를 의미한다. ConcurrentBag은 평소 자주 사용하는 List을 병렬처리하여 추가하거나 삭제하는 경우에 데이터가 안전하게 보관되도록 만든 클래스다. 자주 쓰이는 Format() 메소드는 C#의 string 출력 방법이다. C#에서는 string을 출력하는 여러가지 방법이 존재하는데 Format() 메소드는 지정된 형식에 따라 개체의 값을 문자열로 변환하여 다른 문자열에 삽입한다.

그리고 하나의 PC에서 사용하므로 루프백IP를 사용하고 클라이언트를 구분하기 위해 127.0.0.1, 127.0.0.2, 127.0.0.3으로 클라이언트를 지정했다. 서버에접속할때서버의 IP주소를 직접 입력해서 접속해도 접속이 가능하지만 서버에서 LocalEndPoint로 클라이언트의 IP를 받아오면 클라이언트의 IP주소가아닌 접속할때 사용한 루프백IP를 받아올 수 있으며 하나의 PC에서 서버에서 구별할 수 있는 여러개의 클라이언트를 접속 시킬 수 있게된다. 이외에도 AsyncCallback의 매개변수에 콜백메서드를 등록, 클라이언트가 메시지를 보내서 서버가 해당 메시지를 읽게 됐을 때 콜백메서드가 실행되는것과 상대방 이름과 메시지를 보내면 해당 이름에 맞는 클라이언트 번호를 키로 딕셔너리에 접근하는 방식등 꼭 알아야 하는 핵심 개념들을 알게되서 유익했다. 이를통해 네트워크프로그래밍과 c# 소켓통신 부분에대해서 깊게 공부해야겠다고 생각했다.

.