60200307 우연정

목차

1. 소켓 프로그래밍

통신은 양쪽의 약속, 즉 표준이 중요하다. 파일에 쓰고 읽는 것 역시 통신의 한 형태로, 일반적인 통신은 지역적 차이를 갖는 동시적 행위이다. 소켓은 파일의 개념과 유사하며, OS마다 소켓을 통한 통신 매커니즘을 제공한다.

클라이언트에서는 Open Client - 소켓 - connect - write - close의 순서로 작업이 진행된다.

서버에서는 Open Listen - 소켓 - bind - listen - accept - read - write - read의 순서로 작업이 진행된다.

고유의 포맷은 통신을 할 때 문제가 된다. 그래서 표준화된 코드와 문법을 만들어내었다.

서버는 서비스를 제공한다. 클라이언트는 필요할 때 접속을 하고 서버는 계속 대기 해야 한다.

2. 비동기적 통신과 동기적 통신

비동기적 통신(Asynchronous Communication)은 시간적으로 다른 시간대에 가져가게 하는 것이다. 하지만 동기적 통신(Synchronous Communication)을 하려면 한 쪽은 대기 하고 있어야 한다. 서버는 보통 클라이언트가 접속을 하는 지 안하는 지를 알기 위해 일과 시간에 계속 루핑을 돌고 있다. 클라이언트는 아무 때나 접속을 했다가 서비스를 받고 다른 것을 할 수 있다.

전화 통화로 예를 들면, A가 B에게 전화를 걸고, B가 전화를 받을 때까지 A는 기다린다.

B가 전화를 받으면, 두 사람은 통화를 시작하고, 통화하는 동안에는 다른 일을 할 수 없다.

통화가 끝날 때까지는 둘 다 대화에 집중해야 한다. 동기적 통신은 요청과 응답이 순차적으로 진행되고, 요청 후 응답을 받을 때까지 다른 작업을 할 수 없다는 특징이 있다.

반면에 비동기적 통신은 문자 메시지를 예로 들 수 있다. A가 B에게 문자를 보내면, B가 답장할 때까지 A는 다른 일을 계속할 수 있다. B는 문자를 받고, 원하는 시간에 답장을 할 수 있다. 즉 A와 B 모두 대화가 계속되는 동안 다른 일을 같이 할 수 있다.

비동기적 통신의 특징은 요청을 보낸 후, 응답을 기다리지 않고 바로 다른 작업을 진행할 수 있고, 응답이 도착했을 때 해당 응답을 처리한다는 특징이 있다.

3. 클라이언트와 서버

클라이언트는 서버에게 특정 작업을 요청한다. 서버는 클라이언트의 요청에 응답하며, 필요한 서비스를 제공한다. 서버와 클라이언트는 분리되어 있어, 직접적인 호출이 어렵다. 스텁은 리모트 오브젝트의 대리인이며, 스켈레톤은 서비스 제공을 위한 대행 역할을 한다. 스텁과 스켈레톤은 소켓 및 RMI를 통해 통신한다.

우리가 짜는 프로그램에는 클라이언트에 뷰가 있다. 우리가 서버에게 작업을 부탁할 것이 있다. 서버는 컨트롤 오브젝트이다. 수강신청 프로그램에서 유저의 정보 등을 가져오는 역할이다. 예전엔 클라이언트와 서버가 같이 있었지만 지금은 분리되었다. 프로세스라는 이름의 함수가 서버에 있다. 그런데 이를 클라이언트에서 호출하면 실행되지 않는다. 서버의 내부 메모리 주소이기 때문이다. 이를 실행하기 위해서 개념적으로 stub이라는 대리자를 만들 것이다. 이 대리자는 서버의 대리인을 의미한다. 이 대리인은 서버의 객체 이름과 함수 이름은 같지만 하는 일이 다르다. 대리인이 직접 서버에 있는 함수를 호출하려면 함수가 어디 있는지를 알아야 한다. 이를 위해서 대리인을 하나 더 만든다. 이를 스켈레톤이라고 한다. 스켈레톤은 클라

이언트의 역할을 하고 프로세스 함수를 대신 호출한다. 대리인끼리 통신을 하게 만드는 것이다. stub은 리모트 오브젝트의 대리인이다. 스켈레톤은 대행으로서 서비스를 요청하고 스텁에게 제공해주는 역할이다. 스텁은 리모트 오브젝트 개수만큼 있어야 한다. 어디로 갈지 모르기 때문에 대리인이 따로 있어야 한다.
클라이언트는 스텁이 여러 개 있을 수 있지만, 내 안에 있는 스켈레톤은 하나만 있으면 된다. 스텁과 스켈레톤은 소켓과 rmi로 한다. 두 개의 다른 프로세스가 이렇게 통신한다. 스텁, 스켈레톤은 리모트 오브젝트의 대리인이 생기는 것이다. 내 컨텍스트 바깥에 있는 것과 통신하기 위한 대리인들을 세워서 하는 것이다. 통신하는 사람과 작업하는 사람은 달라야 한다. 그래야 재사용성이 있다.

<table>
<tr><th colspan="4">소켓 커뮤니케이션</th></tr>
<tr><td></td><td>클라이언트</td><td>서버</td><td></td></tr>
<tr><td rowspan="4">Open Client</td><td>소켓</td><td>소켓</td><td rowspan="4">Open Listen</td></tr>
<tr><td></td><td>bind</td></tr>
<tr><td></td><td>listen</td></tr>
<tr><td>connect -></td><td>-> accept</td></tr>
<tr><td rowspan="2">Client/Server
Session</td><td>write -></td><td>read</td><td rowspan="2"></td></tr>
<tr><td>read</td><td><- write</td></tr>
<tr><td></td><td>close -></td><td>-> read</td><td></td></tr>
<tr><td></td><td></td><td>close</td><td></td></tr>
</table>