60200307 우연정

목차

1. Remote Method Invocation

Remote는 원격, 즉 컨텍스트가 다르다는 것을 의미한다. 컨텍스트가 다르다는 것은 주소 체계가 다르다는 것이다. 바깥으로 나가면 서로가 내부의 내용을 이해할 수 없게 되는 것이다. 컨텍스트가 없으면 문장이 해석되지 않는다. 컨텍스트는 특정 개체에 의미를 부여하기 위한 복합적인 관계이다. 상대주소를 쓰기 때문에 객체를 알아야지만 함수를 실행할 수 있다. 함수가 실행되는 객체의 메모리가 함수의 실행을 한정하기 때문이다.
프로세스는 실행되는 프로그램을 말한다. 실행 중인 프로그램이 만들어지려면 CPU가 실행 가능한 명령어들이 들어간다. 혹은 OS를 통해서 명령을 내리는 경우도 있다.
결론은 머신 인스트럭션은 시피유가 실행한다. 그래서 프로세스는 OS나 하드웨어에 종속적이다.

가장 작은 단위의 컨텍스트는 객체이다. 프로세스는 여러 개의 객체로 구성된다. 통신은 정보를 주고 받는 것이다. 기본적으로 객체와 객체의 통신에서 정보를 주고 받는데, 메소드를 호출함으로써 서로 통신을 한다. 원래는 클래스 안에 들어있지만, 메소드는 개념적으로 객체 안에 들어가 있다. 함수가 하나밖에 없다고 할지라도, 그 함수가 호출될 때, 객체가 컨텍스트를 한정한다. 그 메소드가 쓰는 메모리를 객체가 new되어서 제공한다. 즉 함수가 속해있는 클래스의 인스턴스가 그 메소드의 컨텍스트를 만들어내는 것이다. 메소드는 데이터를 쓸 때 객체의 상대주소를 쓴다. 객체의 주소는 프로세스의 상대주소로 만들어진다.

프로세스 내에 있는 것들은 모두 프로세스의 상대주소로 계산할 수 있다. 그런데, 프로세스 바깥에 함수나 객체가 있다면 그것은 다른 프로세스이 객체나 함수의 주소를 알 방법이 없다. 서로 특수한 약속을 하지 않는 이상 불가능하다. 물리적으로 공유할 수 있는 장치가 전혀 없는 경우이다. 이러한 경우는 네트워크 카드 등을 통해서 물리적인 전기신호를 통해서 데이터가 움직인다.

우리가 한 프로그래밍은 논리적인 개념을 가지고 소켓을 만들어서 사용했다. 소켓은 소프트웨어이다. 실제로, 데이터는 오에스를 통해서 하드웨어를 통해서 네트워크로 간다. 네트워크에서 이를 받아 하드웨어, 오에스 순으로 간다. 물리적인 하드웨어 내부에서 통신을 하더라도 실제로 오에스에서 하드웨어로, 네트워크로 갔다가 다시 하드웨어, 오에스에서 다른 프로세스로 간다. 즉 내부의 프로세스 통신과 외부의 프로세스 통신의 개념이 동일해진다.

localhost는 내가 내 아이피 주소를 그대로 쓴다는 뜻이다. 즉 물리적으로 내 내부에 있는 프로세스끼리 통신을 하겠다는 뜻이다.

1-1. Inter-Process Communication

내부의 객체와 객체의 통신을 인트라 프로세스 커뮤니케이션 통신이라고 한다. 인터 프로세스 커뮤니케이션은 프로세스와 프로세스 간의 통신을 말한다. 인터넷은 네트워크와 네트워크 간의 네트워크를 말한다. 초기의 컴퓨터가 프로토콜이 모두 달라서 네트워크 연결이 되지 않았고, 이를 표준화 시키기 위해서는 네트워크와 네트워크 간의 프로토콜을 표준화해야 통신이

가능하다. 이것이 인터넷의 시작이다.

인터 프로세스 통신을 하기 위해서는 프로세스의 주소가 붙어야 한다. 인트라 프로세스 통신에서는 프로세스의 기존 메모리 주소가 있어야 한다. 메모리의 상대 주소를 가지고 계산할 수 있다. method invocation에서 함수 호출을 하면 두 개의 함수가 공유할 수 있는 곳에 파라미터의 위치를 정해놓고 그 주소를 알려주는 것이다.

객체의 이름까지 알면 그것의 상대 주소를 가지고 함수를 계산할 수 있다. 이는 우리가 프로그램을 만들 때 인터페이스를 가지고 함수의 내부주소, 상대주소를 알게 된 것이다. 같은 객체의 경우 함수의 주소는 객체의 상대주소로 계산할 수 있는 것처럼 만들었다. RMI는 결국 인터 프로세스 통신을 하는 것인데, 프로세스 간에 통신으로 데이터가 오가는 것이다. 데이터를 주고받는 주체는 함수(메소드)이다. 메소드가 없으면 데이터를 주고받을 수 없다. 프로세스에는 시피유와 오에스의 명령어가 들어가 있기 때문에 시피유가 다른 환경에서는 프로세스가 돌아갈 수 없다.

클라이언트 서버 프로그램에서 서버에서는 자신의 IP주소, 포트 번호, 객체 이름을 노출시켜야 한다. 이를 provided interface라고 한다. 파라미터 타입도 있어야 하는데, 인터페이스 클래스도 같이 노출해야 한다. 프로그램할 때 쓰기 위해서이다. 즉 서버가 서비스를 하려면 provided interface와 인터페이스 클래스 이 두 개를 노출해야 한다. 그렇게 해야 서비스, 클래스를 쓸 수 있게 되는 것이다. 인터페이스 클래스는 프로그래밍을 할 때 클래스라는 개념으로, 누군가가 import해서 쓰게 된다.(include) provided interface도 프로그램으로 읽게 할 수 있지만 import하진 않는다. 프로그램은 아니고 일종의 텍스트 파일이다.(xml, html, json) 텍스트 파일로 만들어서 프로그램이 시작될 때 클라이언트 쪽 프로세스가 읽게 만든다.(read) 텍스트 파일로 만드는 이유는, 그것을 프로그램에 코드로 넣으면, 다른 서버를 쓰고 싶을 때 컴파일에 문제가 생기지 않게 하기 위해 코드를 바꿔야 하기 때문이다. 인터페이스 클래스는 이러한 경우 프로그램이 바뀌어야 하기 때문에 import한다.

2. JVM

시피유와 오에스가 다른 환경에서 프로세스를 돌아가게 만들기 위해서 자바 버츄얼 머신(JVM)이 필요하다. 자바 버츄얼 머신의 역할은 명령어가 오에스에서 읽을 수 있도록 번역하는 역할을 한다. 그리고 프로세스를 모두 자바 버츄얼 머신에서 돌아가는 기계어로 만든다. 즉 표준화시키는 역할을 한다. 자바 버츄얼 머신이 만드는 기계어는 바이트 코드이다. 이는 클래스 파일(.class)에 들어가 있다. 자바 파일(.java)을 컴파일하면 클래스 파일이 된다. 바이트 코드는 자바 버츄얼 머신의 기계어이다. 표준화된 자바 버츄얼 머신의 기계어가 만들어져서 자바 버츄얼 머신한테 명령을 내리면 해당 기계용으로 바꿔주는 것이다. 자바 버츄올 머신은 기계당 하나만 있으면 된다.

소켓 대신에 기계어와 기계어 사이에 RMI라는 개념을 붙인 것이다. 하드웨어는 표준화되었지만 소프트웨어를 어떻게 만드느냐에 따라서 어떤 서비스를 제공할 것인지가 달라진다.