60200307 우연정

목차

1. 클라이언트 서버 프로그램에서 함수를 호출하는 법
 우리가 예전에 만들었던 수강신청 프로그램은 메모리 주소를 알고 있어서 바로 데이터를 보낼 수 있다. 함수를 실행하기 위해서는 오브젝트 주소, 메소드 주소, 뿐만 아니라 메소드를 호출하기 위한 파라미터들의 값을 가지고 와야 한다. 기존 프로그램에서는 오브젝트 주소, 멧드 주소, 아규먼트 주소/값 이 세 개만 알았으면 됐지만, 클라이언트 서버 프로그램에서는 외부에서 주소라는 것을 알기가 쉽지 않다. 내가 뭘 할지는 알아야 하므로 어떤 함수를 호출해야 될지는 서로 약속을 해야 한다. 근데 그것이 무엇인지를 찾아야 한다는 이슈가 발생한다.

 그래서 서버의 스켈레톤과 클라이언트 쪽의 컨트롤이 약속을 했다. 보통 컨텍스트가 다를 때 우리가 어떤 이름을 만들 때는 스트링, 아스키 캐릭터로 만든다. 표준화된 캐릭터를 만드는 것이다. 스트링은 표준화된 코드를 쓴다. integer나 float은 표준화된 게 아니다. 기계마다 컴파일러마다 다를 수가 있다. 내부 포맷이 인텔 계열과 암 계열 등등 다르다면 이런 하드웨어에 실제로 인티저를 표현하는 방법은 다 다르다. 우리가 뭔가 데이터를 주고받을 때 인티저가 기계마다 틀리니까 몇 바이트가 될지 알 방법이 없다. 즉 여기서 나온 것처럼 지금 일단 클라이언트의 컨트롤과 서버의 스켈레톤이 객체의 이름을 정해놓아야 한다.

 즉 클라이언트와 서버 간에는 특정 객체와 메서드를 어떻게 식별하고 호출할 것인지에 대한 방식을 정해야 하는데, 주로 문자열 기반의 식별자가 사용되며, 이는 특정 메서드의 이름 혹은 엔드포인트 역할을 한다.

 그리고 뷰가 컨트롤을 호출할 때 그러니까 함수의 이름을 알아야 한다. 객체는 주소는 다를 수 있지만 이 함수의 상대 주소는 같아야 한다. 그런데 객체는 얘가 어디 있을지는 모른다. 그래서 공통으로 사용할 수 있는 이름 하나를 도출을 해 놔야 한다.
 분산 프로그램에서는 오브젝트를 쓰는 방법이 두 가지가 있어다. 아주 크게 나누면, 첫 번째는 우리가 보통 프로그램을 짤 때는 A클래스라는 A객체에서 B객체로 연결할 때 어소시에이션을 한다. 이때 B의 주소를 알아야 하고, 함수 이름, 파라미터를 다 알아야 한다.

 서버는 비 클래스이고 서비스를 호출당하는 쪽이다. 클라이언트 서버 프로그램은 클라이언트와 서버가 계속 바뀔 수 있다. 비가 에이를 노출하면 동시에 클라이언트자 서버인 것이다. 항상 오브젝트들은 대부분이 서비스를 사용하기도 하고, 서비스를 제공하기도 한다. 에이에서 비를 호출하면 원래는 우리가 비라는 클래스와 비의 함수를 다 알아야지만, 에이가 함수 정보를 임포트 하면 사용할 수 있다.

 컴파일러가 비 클래스(서버)에 있는 이 함수 이름, 리턴 타입, 파라미터 타입을 비 클래스에서 읽어와 에이 클래스에서 내가 함수를 호출하게 되면, 내부에서 사용할 땐 객체의 이름으로 사용한다. 즉 함수를 호출하는 클래스 내부에서 포인터를 지칭하는 이름을 하나 만들어 쓴다.

 그러면 비라는 오브젝트를 쓸 때 클래스와 함수 이름만 알면 호출할 수 있다. 클래스가 아니라 실제로 뉴를 하기 때문에 안에다가 내가 쓸 수 있는 포인트를 하나 만들어 놓고 이걸 주소를 여기다 집어넣는 것이다.
  뉴 해서 만들어진 비 클래스에 인스턴스 b의 오브젝트의 주소를 여기다 집어넣어놨어. B에

다가 얘가 주소를 담고 있는 포인터이다. 이렇게 어소세이션 하는데 이걸 역할이라고 한다. UML 입장에서 보면 어소시에이션을 해서 에이 클래스가 보는 비 클래스의 이름은 b라고 보고 있는 것이다.

내가 어떤 클래스를 호출하고 싶다면 오브젝트 이름과 함수만 알면 된다. 클래스가 뭔지는 뭐 뉴 할 때 알아야 되겠지만, 일단은 만들어 놓고 난다면 그다음에는 그 오브젝트의 주소와 함수 이름만 알면 된다.

두 개의 다른 프로그램을 따로 개발하면, 서버(실행되어야 하는 함수)의 오브젝트 주소와 함수 이름, 아규먼트를 알아야 된다. 근데 이것은 임포트를 하면 알게 된다. 임포트 하면 뭘 알게 함수 이름을 알게 되잖아요.

함수 이름만 꺼내놓은 것을 인터페이스라고 한다. 그리고 일반적으로 그것을 API라고 한다.

2. 자바 리플렉션

“자바 리플렉션은 자바에서 제공하는 API로, 런타임에 클래스, 인터페이스, 메서드, 변수 등의 메타데이터 정보를 알아내거나, 객체를 생성하고, 메서드를 호출하는 등의 동적인 작업을 할 수 있게 해주는 기술이다.”[네이버 지식백과] 이는 클라이언트-서버 모델에서 서버가 동적으로 메서드를 호출할 때 사용할 수 있다. 자바 리플렉션은 클래스 객체를 통해 클래스의 정보를 알아낼 수 있고, 클래스 객체로 동적으로 객체를 만들어낼 수 있다. 클래스의 객체를 생성할 때 new 키워드를 사용하지 않고, Class 객체를 통해 인스턴스를 생성할 수 있다. 리플렉션을 사용하면, 특정 객체의 private 메서드에 접근하거나, 수정할 수 있다. 또한, 메서드를 동적으로 호출할 수 있다.