60200307 우연정

목차

1. MVC

뷰와 컨트롤러는 한 세트이다. 뷰가 하나 생기면 컨트롤러도 하나 생긴다. 뷰는 통합된 서비스이다. 예를 들어 예약, 구매, 찾기 등 고객이 원하는 것을 내주는 서비스이고, 여러 단계일 수도 있다. 로그인은 서비스라고 하기 곤란하다. 로그인은 뭔가를 하기 위한 과정이다.

뷰와 컨트롤은 차이가 있다. 뷰는 사용자에게 고객이 원하는 결과를 제공하는 것이 하나의 서비스 단위이다. 고객이 원하는 것을 여러 가지로 나누는데, 뷰는 시각, 청각, 촉각 등 지각에 관한 기능을 제공해준다. 컴퓨터에 오감과 연결되는 디바이스를 만들어놓았다. 즉 뷰는 오감과 소통할 수 있는 기능을 제공해주고, 컨트롤러는 업무 처리에 관한 것이다. 처리한다는 것은 데이터를 가공하는 것이다. 데이터를 뷰에서 원하는 형태로 바꿔주는 것을 말한다.

모델은 데이터를 관리해준다. 파일은 시간적인 차이를 극복해서 우리가 데이터를 쓸 수 있도록 한다. 즉 어떠한 데이터가 시간적 지속성을 갖게 만든다. 공간적 지속성을 갖게 하는 것은 통신이라고 한다. 지속성은 과거와 미래가 소통한다는 것이다. 파일 시스템은 옛날에 만든 데이터를 지금 쓸 수 있다. 일반적으로 파일은 시간적 지속성을 위해서 만들어놓은 것이다.

서비스가 생기면 뷰와 컨트롤은 세트로 생긴다. 뷰는 서비스의 사용자 인터페이스(지각적 기능)를 담당하고, 컨트롤은 서비스의 업무(domain knowledge, 업무적 기능)를 담당한다. 데이터의 구조와 서비스의 구조는 다르다. 데이터의 구조는 재사용성과 독립성을 중심으로 분류해서 넣는다. 만약 은행이라면 계좌(모델) 하나로 입금과 출금(서비스)을 할 수 있다. 이체는 두 개의 계좌를 쓰지만 서비스는 하나이다. 이런 식으로 서비스와 데이터 모델의 mapping 관계는 n:n의 관계이다. 모델과 파일은 1:1의 관계이다.

2. Client-Server

클라이언트(뷰) - 서버(컨트롤러 - 모델 - 파일)로 이루어진 구조이다. 컨텍스트가 달라지면 통신으로 해결한다. 소프트웨어적으로 통신할 때 가장 낮은 레벨에 있는 것이 소켓이다. 소켓이라는 전화기가 OS에 붙어 있는 것으로 비유할 수 있다. 클라이언트 쪽에 스텁을 만들었고, 서버 쪽에 스켈레톤을 만들었다. 스텁은 컨트롤을 대신해주는 대리인이고, 스켈레톤은 통신 담당자이다. 스켈레톤은 서버 당 하나가 있다.

스텁은 스켈레톤에게 세션을 받아낸다. 세션은 컨트롤을 내부에서 연결해준다. 서비스가 늘어나면 늘어난만큼 스텁, 뷰, 세션, 컨트롤이 늘어난다. 같은 컨트롤이면 스텁을 공유할 수도 있다. 그러나 동시에 서비스를 받으면, 뷰 두 개가 하나의 컨트롤을 쓰면 데이터가 깨질 수 있다. (공유 자원에 관한 문제가 생긴다.)

클라이언트 서버 모델에서 서비스 세트는 뷰, 스텁, 세션, 스켈레톤, 컨트롤에 해당되며, 데이터 세트는 모델과 파일에 해당된다.

우리는 스켈레톤을 서버 소켓으로 구현했다. 스텁은 클라이언트 소켓으로 구현했다. 서버 소켓이 워커 세션을 만들었다. 일반적으로 세션은 연결된 상태로 유지되고 있는 하나의 통신 서비스를 말한다. 일반적으로 연결이 끊어지면 세션이 끝났다고 한다. 보통 하나의 세션은 하나의 서비스를 준다. 여기에는 커넥션 베이스와 커넥션리스 두 가지가 있다.

커넥션 베이스는 하나의 서비스가 끝날 때까지 통신을 유지하고 있는 것을 말한다. 이체를 할 때 돈이 옮겨질 때까지 통신을 유지하고 있는 것을 예로 들 수 있다.
반면에 커넥션 리스는 한번 연결하고 나서 끊어진다. 이렇게 되면 여러 가지 문제가 생긴다.

스텁과 세션 사이에서 통신할 때 주고받는 언어의 문법을 프로토콜이라고 한다. 우리는 object name, method name, argument라는 문법을 만들었었다. 이를 String으로 쓰고 delimeter를 end of line(null)으로 막았다. String-null이 우리의 가장 기본적인 문법이었다. 이는 단어를 끊어낼 때 쓰는 기본적인 방법이다.

우리는 단어를 인식할 수 있어야 하고 그 구조가 만들어져야 한다. 즉 단어를 만들려면 알파벳이 있어야 하고, 단어와 단어를 끊어내는 방법이 있어야 한다. 알파벳은 유니코드로 썼고, 그 사이는 null로 띄었다. 구조는 객체 이름, 메소드 이름, 아규먼트 리스트로 했다.

3. DB 연결

데이터 베이스도 독립적 컨텍스트를 가지고 있다. 전혀 다른 서버가 또 생기는 것이다. 독립적인 데이터베이스가 있는 쪽을 DB 서버라고 하며, 상대적으로 기존 클라이언트 서버 구조에서 서버는 애플리케이션 서버라고 한다.

윈도우에서는 파일이 2종류가 있다. 프로그램 파일과 데이터 파일이다. 프로그램 파일은 상태가 없다. 상태가 없다는 것은 데이터가 없다는 뜻이다. 프로그램 파일은 form만 가지고 있다. 그 상태에서 내 데이터를 연결하면 데이터 파일이 되는 것이다. 이와 똑같이 애플리케이션 서버는 프로그램이고, DBMS 서버는 데이터이다. 즉 실시간으로 연결할 수 있도록 프로그램을 분리해놓은 것이다. 이를 Enterprise application이라고 한다. 또한 아무나 와서 접속할 수 있도록 클라이언트도 분리되어 있다. 클라이언트가 접속할 때마다 세션이 같이 늘어난다. 컨트롤은 같이 늘어날 수도 있고 늘어나지 않을 수도 있다.

DBMS 서버와 애플리케이션 서버가 통신할 때에도 스텁, 스켈레톤, 세션, 프로토콜의 개념이 그대로 적용된다고 할 수 있다. 소켓으로 연결하며, 포트 넘버와 프로토콜은 DBMS 서버가 원하는 것을 쓴다. 포트 넘버와 프로토콜은 1:1 관계이다. 왜냐하면, 담당 스켈레톤(서버)가 다르기 때문이다. rmi용, 웹용, ftp용, 이메일용 등 프로토콜, 언어, 서버가 다 다르면 스켈레톤도 다르다.

jdbc는 자바가 데이터 베이스를 연결할 때 쓰는 프로토콜이다. 자바에서 쓰는 표준화된 자바의 데이터베이스 인터페이스 함수 이름이다. 프로그램을 짜서 던져준다. 데이터 베이스의 프로그램을 string으로 짜서 글씨로 준 것이다. 왜냐하면, 내부에서의 프로그램은 무조건 메모리

주소로 통신한다. 그러나 컨텍스트가 다른 곳으로 넘어가는 순간 전기 신호로 바뀌어서 진짜 데이터를 보내야 한다. 컨텍스트가 달라지면 내부의 데이터 표현 방법이 달라지기 때문이다. 명령어도 진짜 명령어로 보내야 한다. 이때 표준화된 String으로 보낸다. 프로그램을 짤 때 우리는 String으로 짜지만, 컴파일러가 모두 주소로 바꾼다.

애플리케이션 서버와 DBMS 서버간의 프로토콜은 jdbc:sql이며, 클라이언트와 애플리케이션 서버 간의 프로토콜은 rmi이다.

컨텍스트가 달라지면 표준화된 프로토콜을 이용해서 서로 통신하는데, 대부분이 String 기반이다. (rmi 제외: rmi는 jvm과 jvm이 통신하기 때문에 바이트 코드를 그대로 보내도 상관없다.) 애플리케이션 서버의 스켈레톤은 rmi 서버가 된다.

4. Web

웹 서버와 일반적인 클라이언트 서버의 차이점은? 웹 서버에서 view는 웹 브라우저이다. 웹 브라우저와 애플리케이션 서버 간의 프로토콜은 http이다. 웹의 장점은 뷰를 가진 클라이언트 애플리케이션을 깔지 않아도 된다는 것이다. 또한 애플리케이션은 만드는 사람에 따라 프로토콜이 모두 다르다. 하지만 웹 프로토콜은 표준화시켜서 http로 쓴다.

가장 큰 근본적인 차이점은 페이지의 위치이다. 클라이언트 서버 프로그램에서는 뷰에 페이지(데이터가 첨가되지 않은 form)가 있다. form은 뷰에 그대로 있고 데이터만 db에서 가져온다.
반면에 웹 서버는 페이지가 애플리케이션 서버에 있다. 세션이 해당 페이지를 읽어온다. 즉 서버에 있는 페이지를 읽어서 컨트롤에 있는 데이터와 합쳐서 웹 브라우저로 내려보낸다. 실제로는 웹 서버(http)와 애플리케이션 서버가 나누어져 있다. http 서버는 페이지를 가지고 있다. 이와 달리 클라이언트 서버는 페이지가 뷰에 있다. 페이지를 그릴 때는 html로 그린다.

웹은 극명한 한계가 있다. OS에 native user interface graphics를 쓰지 못하고, 어떠한 컴퓨터에도 있는 공통적인 graphics를 쓴다. html에서 모든 명령을 웹 서버에서 가져온다. 즉 어떠한 기능이 추가되거나 변경될 때마다 웹 서버로 가서 해당 기능을 가져와서 플러그인하기 때문에 속도가 매우 느리다. 그래서 원래 웹은 무언가 기능을 하는 용도가 아니어서 정상적인 웹은 애플리케이션 서버와 DB 서버가 없는 구조이다.
원래 웹 서버는 기능과 관계없이 하이퍼 텍스트 멀티미디어 문서를 네트워크로 연결해서 쓸 수 있도록 하는 것을 목적으로 만들어졌다. 처음에는 웹 서버와 DB 서버를 연결했지만, 점점 업무가 복잡해지면서 웹에서 기능을 쓰기 위해 '웹 애플리케이션 서버'가 웹 서버와 DB 서버 사이에 있게 된 것이다.
웹 서버는 뷰를 완전하게 만들어서 합성하는 것을 담당하기 때문에 form(형태, 양식)을 가지고 있다. 웹 애플리케이션 서버에서 데이터를 만들어서 웹 서버가 합성해서 주는 것이다.
웹인 경우 클라이언트가 없어도 된다. 웹 브라우저 하나만 있으면 된다. 이러한 경우를 'thin 클라이언트'라고 한다. 그리고 클라이언트-서버 구조는 'rich 클라이언트'라고 한다.

3-tier의 모델이 이렇게 여러 개의 종류가 있다.