CS 기반지식 면접 기출 모음

> JAVA

OOP : 데이터를 추상화시켜 상태와 행위를 가진 객체를 만들고 객체들 간의 유기적인 상호 작용을 통해 로직을 구성하는 프로그램.

클래스 : 추상화를 거쳐 집단에 속하는 속성 & 행위를 변수와 메서드로 정의한 것

인스턴스 (객체) : 클래스에서 정의한 것을 토대로 실제 메모리상에 할당된 것

Call by value (값에 의한 호출)

> OOP 의 특성.

추상화 : 공통의 속성이나 기능을 묶어 이름을 붙이는 것.

캡슐화 : 코드를 재수정 없이 재활용 하는것.

상속 : 부모 클래스의 속성과 기능을 자식 클래스가 원하는 기능만 수정하여 사용하는것.

다형성 : 하나의 변수명, 함수명 등이 상황에 따라 다른 의미로 해석 하는것

- 오버라이딩, 오버로딩이 해당
- 오버라이딩 : 부모클래스에 메서드와 같은 이름, 매개변수를 재정의 하는것.
- 오버로딩 : 같은 이름의 함수를 여러개 정의 매개변수의 타입과 개수를 다르게 하여 매개변수에 따라 다르게 호출할 수 있게 하는 것 .

0

> 객체 지향적 설계 원칙

SRP : 단일 책임 원칙

- 클래스는 단 하나의 책임을 가져야 하며 클래스를 변경하는 이유는 단 하나의 이유여야 한다.

OCP : 계방 폐쇄 원칙

- 확장에는 열려 있어야 하고 변경에는 닫혀 있어야 한다.

LSP : 리스코프 치환 원칙

- 상위 타입의 객체를 하위 타입의 객체로 치환해도 상위 타입을 사용하는 프로그램은 정상적으로 동작.

ISP : 분리 원칙

- 인터페이스는 그 인터페이스를 사용하는 클라이언트를 기준으로 분리 해야 한다.

DIP : 의존 역전 원칙

- 고수준 모듈은 저수준 모듈의 구현에 의존해서는 안됩니다.

> Restrful API

Restful : 기본원칙을 성실히 지킨 서비스 디자인.

API 설계의 중심에 자원이 있고 HTTP Method 를 통해 처리하도록 설계

REST 6 가지 원칙

Uniform interface

Stateless

Caching

Client-Server

Hierachical system

Code on demand

Restful 하게 API 를 디자인 한다는 것은?
- 리소스와 행위를 명시적이고 직관적이게 분리한다.
- 리소스는 URL 로 표현 되는데 리소스가 가리키는 것은 명사로 표현 되어야 한다.
- 행위는 HTTP METHOD 로 표현한다.
- GET(조회) ,
- POST(생성),
- PUT(기존 Entity 전체 수정) ,
- Patch(기존 entity 일부 수정)
- DELETE(삭제)

> SDK 의 특징, SDK 의 한계와 개선방법

> Servlet 에 대해 답변해 보세요.

Servlet: 자바 플랫폼에서 웹 앱을 개발할 때 사용하는 핵심기술. 컨트롤러와 뷰의 역할 분담이 가능해진다. 자바 API 를 모두 사용할 수 있으며 다양한 서버 환경에서 실행 가능하다. 스레드를 기반으로 한다.

Servlet Container: Servlet 을 서버에서 실행하기 위한 서버 프로그램(서버는 서블릿 자체를 직접 실행할 수 없기 때문). JVM 을 내장하고 있다.

> 서블릿의 라이프 사이클에 대해 설명하세요.

init(): 컨테이너는 서블릿 인스턴스를 생성한 다음 init() 메소드를 호출한다. 이 메소드는 service()메소드 전에 실행되어야 한다. 클라이언트의 요청을 처리하기 전에 서블릿을 초기화할 기회를 주는것이다. 초기화할 코드가 있다면 init()메소드를 재정의 할 수 있다 (데이터베이스에 대한 접속, 다른 객체에 서블릿을 등록하는 등)
service(): 최초 클라이언트의 요청을 받았을때, 컨테이너는 새로운 스레드를 생성하거나 스레드 풀로부터 서블릿을 가져와서 서블릿의 service() 메소드를 호출한다. 클라이언트의 HTTP 메소드(GET, POST 등)를 참조하여 doGet() / doPost() 혹은 다른 메소드를 호출할지 판단한다. 재정의는 하지 않으며 doGet()/ doPost() 를 재정의하여 HttpServlet 의 service()가 이를 실행하도록 한다.
doGet(), doPost(): service() 메소드가 클라이언트의 HTTP 메소드(GET, POST 등) 를 참조하여 doGet()/ doPost()를 호출한다. 여기서 doGet()/ doPost() 만 언급하는 이유는 이것말고 나머지 메소드는 사용할 경우가 거의 없기 때문이다. 이 메소드 안에서 코딩작업을 하면된다. doGet()/ doPost() 둘중 하나는 반드시 재정의 해야 한다.
destroy(): 서블릿 엔진이 off 될 때 호출되는 메소드. 서블릿 종료 시 모든 자원을 반납하는 작업을 수행한다.

> CS 기반과 웹기반의 차이점은?

참고: [IT 기술면접 준비자료] 웹 아키텍처와 WAS

	2-Tier	3-Tier

개발 편의성 측면	4GL 툴 등을 사용하여 작성 용이 함.	보통 프리젠테이션 로직(주로 4GL 로 개발)과 비즈니스/데이터 접근 로직(주로 C/C++,COBOL 언어 사용)을 별도로 작성하므로 2-Tier 에 비해 개발이 불편 함
확장성 측면	좋지 않음	이기종 H/W 증설 또는 이기종 데이터베이스가 구축되어도 데이터 정합성 보장할 수있어 확장성이 뛰어남.
재사용성 측면	모든 로직이 클라이언트에 존재하고, 4GL 툴과 관련되므로 4GL 툴이 변경 시에 모든 로직을 재개발 하여야 함.	동일한 비즈니스 로직을 필요로 하는 프리젠테이션 로직을 다양하게 구현할 수 있음.비즈니스 로직을 모듈화하여 클라이언트/서버 환경과 웹환경에서 동시에 사용 가능함.
성능 측면	동시 사용자 수가 증가 함에 따라 성능이 급격히 저하 됨	동시 사용자 수가 증가해도 일정한 응답속도와 처리량을 보장 함
자원 활용 측면	H/W 자원(CPU, 메모리 등)과 데이터베이스 자원을 비효율적으로 사용 함	미들웨어에서 부하 분산, 큐잉 메커니즘을통해 효율적으로 자원 활용 함
시스템 관리 측면	모니터링 및 관리가 용이하지 못함	처리되고 있는 어플리케이션 정보, 프로그램별 처리 건수 등 다양한 모니터링이 가능하여 관리 및 모니터링이 용이 함

> TCP 와 UDP 의 비교

★ 같이보면 좋은 자료

[IT 기술면접 준비자료] 당신이 브라우저로 웹사이트에 접속할 때 일어나는 일들

특성/설명	UDP	TCP
일반 설명	단순하고, 빠르며, 애플리케이션이 네트워크 계층에 접근할 수 있도록 하는 인터페이스만 제공할 뿐 다른 것은 거의 하지 않음	애플리케이션이 네트워크 계층 문제를 걱정하지 않고 데이터를 안정적으로 송신할 수 있도록 하는, 풍부한 기능의 프로토콜
연결	비연결형. 연결 수립이 없이 데이터를 송신함	연결형. 전송 전에 연결을 먼저 맺어야 한다.
신뢰성	신뢰성이 없음. 승인이 없는 최선 노력 전송 방식	메시지 전송을 신뢰할 수 있음. 모든 데이터에 대한 승인이 있음
재전송	수행하지 않음. 애플리케이션은 손실 데이터를 탐지하고 필요할 경우 재전송해야 함	모든 데이터 전송을 관리하며, 손실된 데이터는 자동으로 재전송함
데이터 흐름 관리	없음	슬라이딩 윈도우를 이용한 흐름 제어를 함. 혼잡 회피 알고리즘을 사용함
부하	매우 낮음	낮지만 UDP 보다는 높음
전송 속도	매우 빠름	빠르지만 UDP 만큼은 아님
적합한 데이터 양	소형에서 중형 데이터(최대 수백 바이트)	소형에서 초대형 데이터까지(최대 수 기가 바이트)

애플리케이션의 유형	데이터의 완전성보다 전달 속도가 중요하고, 소량의 데이터를 송신하고, 멀티캐스트/브로트캐스트를 사용하는 애플리케이션	신뢰할 수 있는 방법으로 데이터를 송신해야 하는 대부분의 프로토콜과 애플리케이션. 대부분의 파일/메시지 전송 프로토콜을 포함함
사용 예	멀티미디어 애플리케이션, DNS, BOOTP, DHCP, TFTP, SNMP, RIP, NFS(초기버전)	FTP, Telnet, SMTP, DNS, HTTP, POP, NNTP, IMAP, BGP, IRC, NFS(나중버전)

> OSI(Open Systems Interconnection)계층을 설명해 보세요.

컴퓨터 네트워크 프로토콜 디자인을 7 개의 계층으로 구분한 것. 실제 인터넷에서 사용되는 TCP/IP 는 OSI 모델을 기반으로 구성되어있다.

Physical Layer: 실제 장치들을 연결하기 위한 전기적, 물리적 세부사항을 정의

Data link Layer: 점대점(Point to Point)간 신뢰성 있는 전송(오류제어, 흐름제어)을 보장하기 위한 계층. 주소 값은 물리적으로 할당 받는다 (MAC 주소)

Network Layer: 여러 개의 노드를 거칠 때 경로를 찾아주는 역할. 이 과정에서 Transport Layer 가 요구하는 서비스 품질을 제공.

Transport Layer: 양 끝단(End to End) 사용자들이 신뢰성 있는 데이터를 주고 받을 수 있도록 해줌(오류검출, 흐름제어, 중복검사). 상위 계층이 데이터 전달의 유효성이나 효율성을 생각할 필요가 없게 한다.

Session Layer: 양 끝단의 응용 프로세스가 통신을 관리하기 위한 방법을 제공. TCP/IP 세션을 생성 및 삭제하는 책임이 있다.

Presentation Layer: 코드간의 번역을 담당. 입력 또는 출력되는 데이터를 하나의 표현 형태로 변환하여 상위계층의 부담을 덜어준다.

Application Layer: 응용 프로세스와 직접 관계하여 일반적인 응용 서비스를 수행한다.

> Spring Framework 특징

참고: Spring 과 DB 이야기

POJO(Plain-old java object): 말 그대로 평범한 자바 오브젝트이다. 기존의 EJB 의 단점을 극복하고 장점은 그대로 취하는 방법이다.

DI(Dependency Injection): 의존하는 객체를 직접 생성하지 않고 전달 받는 방식을 의미한다. 이는 코드 재사용성을 높이고, 결합도를 낮출 수 있다. Spring 에서는 DI 를 자동으로 할 수 있는 기능이 있다.

관점지향프로그래밍(Aspect Oriented Programming): 여러 객체에 공통으로 적용할 수 있는 기능을 구분함으로써 재사용성을 높여주는 프로그래밍 기법. 스프링에서는 트랜젝션이나 로깅, 보안과 같은 여러 모듈에서 공통적으로 사용하는 기능의 경우 분리하여 관리할 수 있다.