웹 개발 면접시 인성 질문, 자기소개, 프로젝트 내용 질문을 제외한 JAVA와 관련된 기술 예상 면접 질문을 정리해 보았습니다.

1) 자바의 특징에 대해 말해보시오.

1) 상속 (Inheritance) : 부모 클래스의 변수와 메서드를 자식 클래스가 전부 물려받는 것입니다.

2) 캡슐화 (Encapsulation, Information Hiding) : 객체의 변수 및 메서드를 외부 객체가 함부로 건드리지 못하게 감싸는 개념입니다 . 클래스의 getter/setter 등을 통해 은닉합니다.

3) 다형성(Polymorphism)

- Poly(다양한) + morphism(변형,변신) 의 합성어입니다.

- 하나의 객체가 여러가지 타입을 가질 수 있는 것을 의미합니다.

- 자바에서는 한 레퍼런스 변수가 다른 형태의 객체를 참조할 수 있음을 말합니다.

- 오버로딩, 오버라이딩, 업캐스팅, 다운캐스팅 등의 방법이 있습니다.

4) 추상화 (Abstraction)

- 자바에서 공통의 속성, 기능을 묶어 이름을 붙이는 것을 의미합니다.

- 자바에서 추상화는 추상클래스, 인터페이스를 통해 구현됩니다.

- 추상클래스 : 추상메서드를 하나 이상 포함한 추상 클래스

- 인터페이스 : 추상메서드만을 포함한 추상 클래스, 다중 상속 가능

1) OOP(객체 지향 언어)

: 부품에 해당하는 객체들을 먼저 만들고, 이것들을 하나씩 조립해 전체 프로그램을 완성하는 개발 기법

2) "가비지 컬렉션"에 의한 메모리 자동 관리

3) "멀티 쓰레드"를 지원한다.

4) JVM 위에서 동작하기 때문에 특정 OS에 종속적이지 않고 이식성이 좋으며, 보안성이 좋다.

5) 다양한 Open 라이브러리들이 존재한다.

2) 자바를 만든 사람에 대해 아시나요?

: "제임스 고슬링"

3) 변수란?

: "하나의 값을 저장할 수 있는 메모리 공간"

4) 객체와 클래스의 차이점에 대해 설명해 보시오.

* 클래스(Class) : 현실 세계의 객체의 속성과 동작을 추려내 필드와 메서드로 정의한 것으로 "아직 메모리가 할당되지 않은 상태"

vs

* 객체(Object) : 이 Class라는 설계도를 기반으로 실제 메모리가 잡힌 것을 의미하며 이런 객체를 조합해 전체 프로그램을 완성해

나가는 방식을 OOP(객체지향 프로그래밍)이라고 한다.

5) 객체 지향 PG이란? 또 그 특징은?

: 현실세계의 객체를 필드와 메서드로 정의한 Class를 기반으로 실제 메모리가 잡혀 만들어진 부품과 같은 객체들을 조합해

전체 프로그램을 완성해 나가는 개발 기법으로

특징)

* 캡슐화, 은닉화 : 외부 객체에서 구현방식은 알 수 없도록 숨기고 별도로 접근할 수 있는 getter/setter 메서드를 통해 접근하도록 하는 방식
* 상속 : 부모 Class를 자식이 접근할 수 있도록 물려 받는 방식
* 다형성 : 부모 클래스 타입으로 해당 부모를 상속받는 여러 자식 class를 대입할 수 있는 성질

등을 들 수 있다.

6) 다형성이란?

: 서로 다른 클래스로부터 만들어진 객체지만 같은 부모의 Class 타입으로 이들을 관리할 수 있는(=대입될 수 있는) 성질

7) 자바의 메모리 영역(간단하게 설명)

1. 메서드 영역 : static 변수, 전역변수, 코드에서 사용되는 Class 정보 등이 올라간다.

, 코드에서 사용되는 class들을 로더로 읽어 클래스별로 런타임 필드데이터, 메서드 데이터 등을 분류해 저장한다.

1. 스택(Stack) : 지역변수, 함수(메서드) 등이 할당되는 LIFO(Last In First Out) 방식의 메모리
2. 힙(Heap) : new 연산자를 통한 동작할당된 객체들이 저장되며, 가비지 컬렉션에 의해 메모리가 관리되어 진다.

8) 추상메서드? 추상 클래스?

* 추상메서드 : 메서드의 정의부만 있고 구현부는 있지 않은 메서드
* 추상 클래스 : 추상메서드를 적어도 하나 이상 가지고 있는 클래스로 자식클래스에서 오버라이딩(재정의)가 필요한 추상메서드를 가지고 있기

때문에 객체화 할 수 없다.

9) 인터페이스(Interface)란? 또 왜 사용하나?

: 인터페이스는 모든 메서드가 구현부가 없는 추상메서드로 이루어진 클래스로, abstract 키워드를 붙이지 않아도 자동으로 모든 메서드는 추상메서드로

정의가 된다. 또한 변수도 자동으로 final static 키워드가 붙게 된다.

왜 인터페이스를 사용하는가?

: 팀작업시 개발코드 부분과 객체가 서로 통신하는 접점 역할을 지원하게 되는데, 이는 개발코드에선 객체의 내부 구조를 모르더라도 인터페이스의

메서드 명만 알고 있으면 되기 때문이다. 이를 통해 얻을 수 있는 장점은 해당 메서드를 통해 나오는 결과물을 알고 있기 때문에 다른 팀의

작업을 기다리고 있지 않아도 되며, 또한 해당 객체가 수정될 경우 개발 코드 부분은 수정을 하지 않아도 된다.

또한, 부가적으로 객체를 파일에 쓰기 위해 Serializable 인터페이스를 구현하거나, Collections.sort()를 하기 위해서 Comparable 인터페이스를

상속하는 것, Cloneable 을 구현하는 것처럼 특정 작업을 하겠다라는 "Mark"역할을 해주기도 한다.

10) 프로세스(Process) 와 쓰레드(Thread)의 차이점에 대해 아는가?

* 프로세스 : OS가 메모리 등의 자원을 할당해준 실행중인 프로그램을 가리킨다. 이때, 각각의 프로세스는 서로 메모리 공간을 독자적으로 갖기 때문에

서로 메모리 공간을 공유하지 못한다. 따라서 공유하기 위해서는 IPC(InterProcess Communication)과 같은 방식이 필요하다.

* 쓰레드 : 쓰레드는 프로세스 내에서 프로세스의 자원을 가지고 실제로 일하는 "일꾼"과 같으며 각 쓰레드는 독자적인 Stack 메모리를 갖고 그 외의

자원(메모리)는 프로세스 내에서 공유하게 된다.

11) 컬렉션프레임워크(CollectionFramework)에 대해 아는만큼 말해 보시오.

* Collection 인터페이스
* List 인터페이스 : 배열과 유사하되, 추가할때마다 자동으로 Boundary를 늘려주는 구조로, 중복된 데이터를 허용하며, 순서가 존재한다.

ex) - ArrayList : 배열로 구현됬으며, 인접해 있기 때문에 데이터 조회에 매우 빠르다 하지만, 빈번한 삽입, 삭제시 새로 배열을 만들고 데이터를

옮겨야 하기 때문에 LinkedList에 비하여 속도가 느리다.

- LinkedList : 링크 구조로 되어 있기 때문에 조회는 ArrayList에 비해 느리지만, 삽입 삭제시 링크를 끊고 새로 추가되는 데이터에 링크만

연결하면 되기 때문에 삽입, 삭제에 유리하다.

- Vector : 구현 방식은 ArrayList와 유사하지만 Vector를 개선한 것이 ArrayList이다. 또한 Vector의 경우에는 ArrayList와 달리

Synchronized(동기화)가 걸려 있어 여러 쓰레드에서 동시에 접근할 수 없다.

* Set 인터페이스 : 집합처럼 중복된 데이터를 허용하지 않으며, 순서가 없다. 또한, 객체 내부의 중복된 데이터를 배제하고 싶은 경우

Object 클래스의 equals 메서드와 hashCode 메서드의 재정의가 반드시 필요하다.

ex) - HashSet

- TreeSet : 순서가 있는 HashSet으로 이진 트리 구조로 만들어 졌다. 순서에 맞게 정렬되어 저장되기 위해서 Comparable을 구현해야한다.

* Map 인터페이스 : key와 value 쌍으로 데이터를 저장하며, key는 중복될 수 없고, value는 중복 저장이 가능하다.

ex) - HashMap

- TreeMap

- Properties : key value 쌍으로 저장되지만 value의 타입이 String만 가능하다.

- Hashtable : HashMap과 구조는 같으며, 단지 Synchronized(동기화) 되어져 있다는 점이 다른점이다.

12) 쿠키(Cookie)와 세션(Session)의 공통점과 차이점은?

* 공통점 : 둘 다 사용자의 데이터를 저장한다.
* 차이점
* 쿠키: 쿠키는 Client 컴퓨터에 저장했다 서버 요청시 네트워크를 타고 서버로 전달되기 때문에 보안에 취약하다.
* 세션 : 세션은 서버에 저장되고 브라우저 단위로 관리된다. 캐시에 비해 보안성이 좋다.

13) Request 전송 방식에는 어떤 것들이 있는지 아시나요?

* Get 방식 : URL의 쿼리문자열에 데이터를 같이 전달하는 방식으로 데이터 길이에 제한이 있고, 보안에 취약하다.
* POST 방식 : 헤더에 데이터를 넣어 보내기 때문에 보안에 조금 더 유리하고 데이터 길이에 제한이 없다. 하지만, Get에 비해 다소 느리다.
* DELETE 방식 : RESTFUL에서 삭제 기능을 할 때 주로 사용된다.
* PUT/PUSH 방식 : RESTFUL에서 수정 작업을 할 때 주로 사용된다.

14) RESTFUL이란?

: 해당 URL만 보더라도 바로 어떤 작업을 하는지를 알 수 있도록 하나의 데이터는 하나의 URL을 갖도록 작업하는 방식

15) Spring에서 DI란 무엇인지 아시나요?

: DI는 Dependency Injection(의존성 주입)의 약자로, 객체들 간의 의존성을 줄이기 위해 사용되는 Spring의 IOC 컨테이너의 구체적인 구현 방식입니다.

DI는 기존처럼 개발코드 부분에서 객체를 생성하는 것이 아니라, 팩토리 패턴처럼 객체의 생성과, 데이터를 주입만 담당하는 Factory에 해당 하는

별도의 공간에서 객체를 생성하고 데이터간의 의존성을 주입해 개발코드에서는 이를 가져다 씀으로서 의존성을 줄이는 방식입니다. 이때,

Factory 패턴의 Factory Class의 역할을 스프링의 환경설정 파일이 담당합니다.

16) Spring의 AOP란?

: AOP는 Aspect Oriented Programming 관점 지향 프로그래밍의 약자로, 기존의 OOP(객체 지향 프로그래밍)에서 기능별로 class를 분리했음에도 불구하

고, 여전히 로그, 트랜잭션, 자원해제, 성능테스트 메서드 처럼 공통적으로 반복되는 중복코드가 여전히 발생하는 단점을 해결하고자 나온 방식으로

이러한 공통 코드를 "횡단 관심사"라 표현하며 개발코드에서는 비지니스 로직에 집중하고 실행시에 비지니스 로직 앞, 뒤 등 원하는 지점에

해당 공통 관심사를 수행할 수 있게 함으로서 중복 코드를 줄일 수 있는 방식입니다.

17) Filter와 Interceptor 방식의 차이?

:

18) 디자인 패턴 아는 것?

1) 싱글톤(SingleTone Pattern) : 대표적으로 Calendar 객체나 dataSource 객체처럼 객체가 하나만 생성되어야 하는 경우

전체 코드에서 하나의 객체만 존재할 수 있도록 이미 생성된 객체가 있으면 그 객체를 사용하도록 하는 방식입니다.

2) 팩토리 패턴(Factory pattern) : 객체간 의존성을 줄이기 위해 객체의 생성과 데이터 주입만 담당하는 Factory Class를 정의하고 개발 코드 부분에서는

생성된 객체를 가져다 사용함으로서 의존성을 줄이는 방식입니다.

3) 옵저버 패턴(Observer Pattern) : 기후 정보처럼 RSS 수신시 하나의 객체가 변하면 다른 객체에 객체가 변했다는 사항을 알려주어야 할 경우에 주로

사용됩니다.

19) MVC 패턴이란?

* Model : data 처리와 접근을 담당
* View : Client에 보여지는 화면을 담당
* Controller : Model과 View를 제어

하는 3가지 부분으로 나눔으로서, 데이터와 화면간의 의존관계를 벗어날 수 있게하는 개발 기법입니다.

20) 프로젝트 개발 순서??

대강...

1) 요구사항 분석

기획 및 스토리 보드 작성

2) WBS(Work Breakdown Structure) 작성 : 작업 분해도로 프로젝트 범위와 최종산출물을 세부요소로 분할한 계층적 구조도

3) 논리 ERD 작성

4) 물리 ERD 작성

5) 개발

6) Testing

7) 유지보수

21) 오버로딩과 오버라이딩의 차이?

* 오버로딩 : 메서드 명은 동일하지만, 매개 변수 타입과 개수를 다르게 해 선언하는 방식
* 오버라이딩 : 상속한 자식에서 부모의 메서드를 재정의하는 방식

22) Servlet vs JSP

* Servlet : 자바 언어로 웹 개발을 위해 만들어진 것으로, Container가 이해할 수 있게 구성된 순수 자바코드로만 이루어진 것
* JSP : html 기반에 JAVA 코드를 블록화하여 삽입한 것으로 Servlet을 좀 더 쉽게 접근할 수 있도록 만들어 진 것

23) Wrapper Class의 사용이유를 아나요?

: 기본 data 타입은 객체가 아니어서 Object로 받는 다형성을 지원할 수가 없다. 하지만, 메서드에서 실재로 기본데이터 타입을 다형성으로

넘겨주어야 하는 경우가 빈번히 발생하는데 이때, 기본 데이터 타입을 객체로 변환시켜 전달하기 위해 사용되며

최근에는 AUTO Boxing, AUTO UnBoxing이 지원된다.

24) DataBase에서 Index란?

: Table에 대한 동작 속도를 높여주는 자료구조로서 빠른 검색을 가능하게 해준다.

25) private, protected, public, default 제어자에 대해 설명해 보시오

* private : 같은 class 내부에서"만" 접근이 가능하다.
* public : 어디서든 자유롭게 접근이 가능하다.
* protected : 같은 class 내부 + 상속받은 자식에서는 부모 class에 접근이 가능하다.
* default : 아무 것도 선언하지 않은 경우로 같은 패키지 내부에서만 접근이 가능하다.

26) SI가 무엇을 하는 건지 알고 오셨나요?

: System Integration의 약자로 시스템 통합 사업으로 고객의 기존 전산시스템을 통합하거나 새로운 시스템을 구축하는 작업입니다.

27) SW 개발시 가장 비중을 크게 두어야 할 부분은 어디라고 생각하나요?

: Testing 부분입니다.

28) 자바의 제네릭이란??

: 클래스 내부에서 사용할 데이터 타입을 인스턴스(객체) 생성시에 결정짓는 방식

-> 여기까지는 제가 예상하는 면접 질문을 올리고 이 아래부터는 실제로 제가 기술면접 당시 받았던 질문들입니다.

[ 기술 면접 질문 ]

1. CVS나 SVN에 대해서 아는대로 설명해 보시오.
2. 64bit CPU와 32bit CPU의 OS적 관점에서의 차이를 설명해 보시오.
3. 프로세스와 쓰레드의 차이점에 대해서 설명해 보시오.( 메모리 구조 포함 )
4. ‘데드락’ 이란 무엇이고 이를 해결하기 위한 방법을 설명해 보시오..

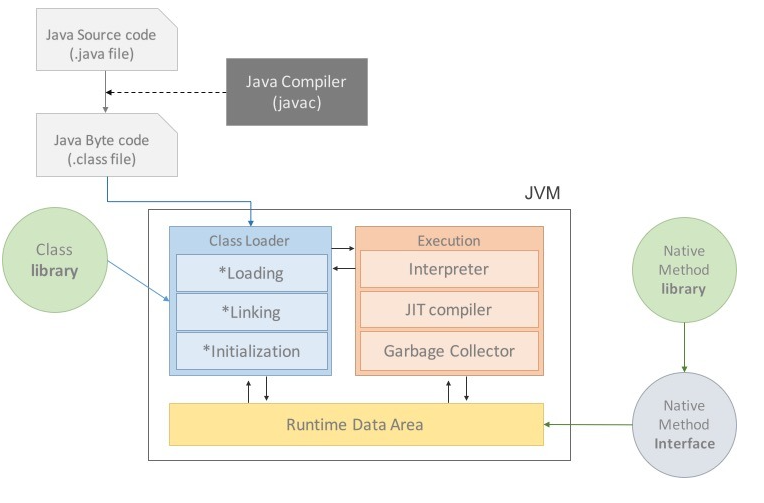
둘 이상의 프로세스가 다른 프로세스가 점유하고 있는 자원을 서로 기다릴 때 무한 대기에 빠지는 상황

1. 변수 명명법이 중요한 이유에 대해서 설명하고 예를 들어 보시오.
2. 자바의 JVM의 역할에 대해서 설명해 보시오.

자바의 가상머신. 가상머신이란 프로그램 실행하기 실행하기 위해 물리적 머신과 유사한 머신을 소프트웨어로 구현한 것. 자바 애플리케이션을 클래스 로더를 통해 읽어 들여 자바 API 와 함께 실행하는 것. JAVA와 OS 사이에서 중개자 역할을 수행하여 JAVA와 OS에 구애받지 않고 재사용 가능하게 해줌. 중요한 메모리관리, 가비지 컬렉션 수행함. JVM은 스택기반의 가상머신.

자바실행과정~

* + 프로그램식행되면 JVM은 OS로부터 필요로 하는 메모리 할당받음
  + JVM은 메모리 용도에 따라 여러영역으로 나누어 관리
  + 자바 컴파일러(javac)가 자바 소스코드(.java)를 읽어들여 자바 바이트코드(.class)로 변환
  + Class loader를 통해 class파일들을 JVM으로 로딩함
  + 로딩된 class파일들은 Execution engine 통해 해선됨
  + 해석된 바이트코드는 Runtime Data Areas에 배치되어 실질적인 수행이 이루어지게 됨



1. 자바의 특징에 대해서 말해 보시오.
2. Linux에서 톰캣 환경설정을 잡는 것에 대해 설명해 보시오.
3. WAS와 웹서버의 차이점은?

**1. 웹 서버와 WAS( Web Application Server )**

* 웹 서버는 **정적**인 컨텐츠( html, css, js )를 제공하는 서버입니다.
  + ex) Apache, Nginx
* WAS는 DB 조회나, 어떤 로직을 처리해야 하는 **동적**인 컨텐츠를 제공하는 서버입니다.
  + ex) Tomcat, Jeus

즉, 웹 서버와 WAS의 차이는 **어떤 타입의 컨텐츠를 제공하느냐**의 차이입니다.

웹 서버와 WAS는 각각 독립적으로 존재할 수 있습니다.

대부분의 WAS는 정적인 컨텐츠를 제공해주고 있기 때문에, 웹 서버 없이 WAS만 존재할 수 있습니다.

그래서 WAS는 웹 서버를 포함하는 개념이라고 생각해도 될 것 같습니다.

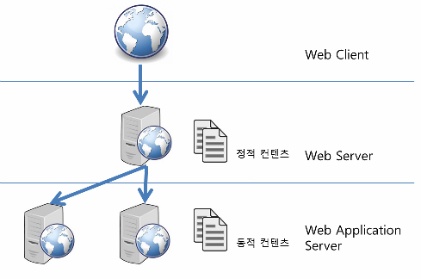
WAS 자체로 웹 서버의 역할을 수행하기 때문에 무조건적으로 웹 서버를 앞단에 두어야 할 이유는 없습니다.

그런데 일반적으로 웹 서버를 WAS 앞 단에 배치하게 되는데, WAS만 사용하면 될 것이지 왜 웹 서버를 앞단에 배치하는 구조를 사용할까요?

**2. 웹 서버 사용 이유**

**1) WAS가 해야 할 일의 부담을 줄이기 위해서 입니다.**

WAS 앞에 웹 서버를 둬서 웹 서버에서는 정적인 문서만 처리하도록 하고, WAS는 애플리케이션의 로직만 수행하도록 기능을 분배하여 서버의 부담을 줄이기 위한 것입니다.



위의 그림처럼 WAS앞에 웹 서버를 둠으로써 서버의 부담을 줄일 수 있습니다.

웹 서버에서는 플러그인 형태로 WAS를 연결하면 일 처리를 나눌 수 있겠죠.

**2) WAS의 환경설정 파일을 외부에 노출시키지 않도록 하기 위해서입니다.**

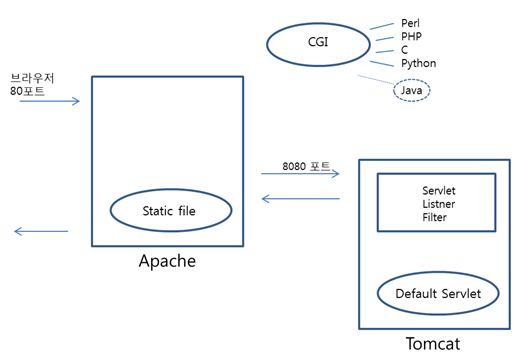
클라이언트와 연결하는 포트가 직접 WAS에 연결이 되어 있다면 중요한 설정 파일들이 노출될 수 있기 때문에 WAS 설정 파일을 외부에 노출시키지 않도록 하기 위해서 웹 서버를 앞단에 배치시킵니다.

웹 서버와 WAS에 접근하는 포트가 다르기 때문에, WAS에 들어오는 포트에는 방화벽을 쳐서 보안을 강화할 수도 있습니다.

**3. 아파치( Apache )와 CGI, 그리고 톰캣( Tomcat )**

자바 웹 애플리케이션을 개발 할 때 주로 사용하는 조합이 아파치와 톰캣일 것입니다.

그러면 다른 언어들은 톰캣 같은 WAS가 없을까요?



아파치에는 **CGI**( Common Gateway Interface )라는 것을 제공합니다.

CGI는 이름 그대로 **인터페이스로서, 웹 서버 상에서 프로그램을 동작시키기 위한 방법을 정의한 프로그램**( 또는 스크립트 )입니다.

즉 PHP, Perl, Python 등의 언어들은 CGI를 구현해놓았기 때문에, 아파치에서 다양한 언어로 짜여진 각 프로그램을 실행할 수 있습니다.

예를 들어 아파치에 PHP 모듈을 설치했을 경우, 요청이 왔을 때 아파치는 HTTTP 헤더를 분석하고 파싱하여 PHP로 파라미터를 넘겨줍니다.

그러면 PHP에서는 파라미터를 받아 응답 할 HTML 문서를 만들어서 아파치에 전달하죠.

HTML 문서를 전달 받은 아파치는 CSS, JS, img 등 정적인 자원들과 함께 브라우저로 반환합니다.

그런데 자바는 CGI로 구현되어 있지 않습니다.

자바 자체가 무겁고, Common 라이브러리와 JEE라는 플랫폼이 존재하기 때문에 아파치에서 굳이 CGI를 제공하지 않은 것 같습니다.

그렇기 때문에 톰캣은 Default Servlet을 통해 정적인 파일을 제공해주기 때문에 웹 서버의 역할을 할 수 있는 것입니다.

( 그럼에도 웹 서버를 두는 이유는 앞서 언급을 했습니다. )

1. Jquery와 Ajax에 대해 아는가?
2. 비동기와 동기 방식의 차이점에 대해서 말해보시오.(네트워크 동기,비동기 아님)

동기: 결과가 주어질때까지 아무것도 못하고 대기. 설계가 간단, 직관적

비동기: 동기보다 복잡, 결과가 주어지는데 시간이 걸리더라도 다른 작업 할 수 있으므로 자원을 효율적으로 사용함

1. 개발시에 중요하다 생각하는 요소를 3가지 기술해 보시오.
2. 스프링의 MVC에 대해서 설명하시오.

* 사용자 인터페이스, 데이터 및 논리 제어를 구현하는데 널리 사용되는 소프트웨어 디자인 패턴이다.
* 소프트웨어의 비즈니스 로직과 화면을 구분하는데 중점을 둔다.
* 간단하게 Model, View, Controller로 분리하여 개발하는 디자인 패턴이다.

**Model**

* 데이터와 비즈니스 로직을 관리한다.
* 애플리케이션이 포함해야할 데이터가 무엇인지를 정의한다.
* 일반적으로 POJO로 구성된다.
* Java Beans

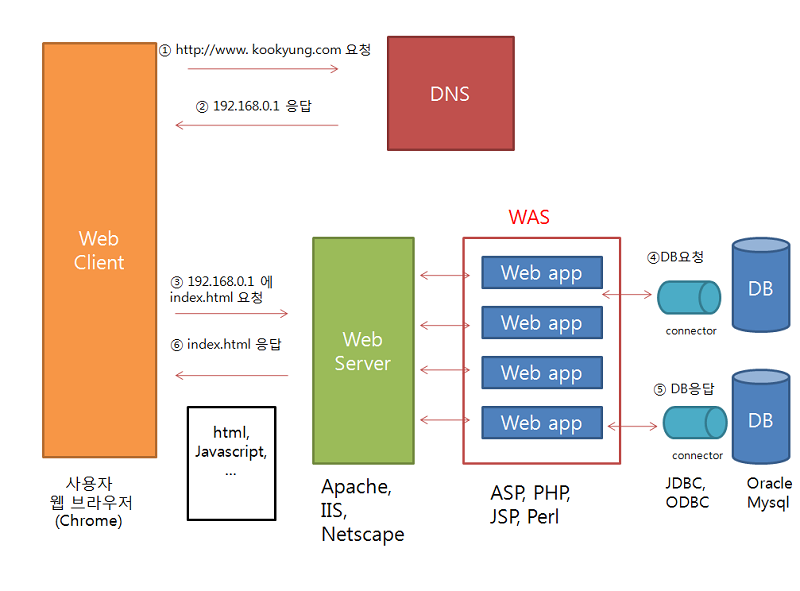
**View**

* 레이아웃과 화면을 처리한다.
* 애플리케이션의 데이터를 보여주는 방식을 정의한다.
* JSP
* JSP 이외에도 Thymeleaf, Groovy 등 여러 Template Engine이 있다.

**Controller**

* VIew와 Model 사이의 인터페이스 역할을 한다.
* 애플리케이션 사용자의 입력에 대한 응답으로 Model 및 View를 업데이트하는 로직을 포함한다.
* Model/View에 대한 사용자 입력 및 요청을 수신하여 그에 따라 적절한 결과를 Model에 담아 View에 전달한다.
* 즉, Model Object와 이 Model을 화면에 출력할 View Name을 반환한다.
* Controller —> Service —> Dao —> DB

1. AOP란 무엇이고 왜 사용하는지
2. ‘에자일’ 방법론에 대해서 아는가?
3. 스프링 환경설정 혼자 잡을 수 있는가? 대강 어떻게 해야하는지 설명해 보시오.
4. 웹서버 내부 구동 방식에 대해 설명할 수 있는가?



1. 스프링 DI란?
2. UML 그려본 적 있는가?
3. Node js나 Angular JS를 사용해 본 적이 있는가?
4. 캐시와 세션의 공통점과 차이점에 대해 말해보시오.
5. 디자인 패턴 아는 것들만 간략히 설명해 보시오.
6. DataBase에서 index관련 질문이었는대 잘 모르겠어서..기억이..
7. 크롬이나 파이어폭스에서 개발도구를 사용해 디버깅을 해보았는가?
8. JDBC는 무엇인가?
9. 스프링을 사용하지 않고 MVC를 JSP에서 만들어 보았는가?
10. DB 옵티마이저에 대해 아는가?

프로젝트 수행내용(주제, 역할, 구현 기능 관련)

객체지향?

아우터조인과 이너조인의 정의와 차이점

자바 인터페이스의 정의와 장점

멀티쓰레드의 장단점

백앤드 프론트 앤드 정의

Jquery/ JavaScript 차이?

Spring/JSP 차이점

Ajax 설명

Spring MVC 패턴 설명

API 오픈소스 활용

**1. Java**

Java는 네트워크상에서 쓸 수 있도록 미국의 Sun마이크로시스템가 개발한 객체지향 프로그래밍 언어

특징

a.     JVM만 설치하면 컴퓨터의 운영체제에 상관없이 작동한다.( 운영체제에 독립적)

b.     기본 자료형을 제외한 모든 요소들이 객체로 표현

c.     객체 지향 개념의 특징인 캡슐화, 상속, 다형성이 잘 적용된 언어

d.     Garbage Collector를 통한 메모리 관리기능

e.     멀티쓰레드(Multi-thread)를 지원한다.

**2. OOP(객체지향 프로그래밍)**

OOP란 Object-Oriented Programming의 약어로써 객체지향 프로그래밍을 의미

데이터를 객체로 취급하여 프로그램에 반영한 것이며, 순차적으로 프로그램이 동작하는 기존의 것들과는 다르게 객체와 객체의 상호작용을 통해 프로그램이 동작하는 것을 말한다.

OOP 특징

a.     객체지향 프로그래밍은 코드 재사용성이 높다.

b.     코드의 변경이 용이

c.      직관적인 코드 분석

d.     개발속도 향상

e.     상속을 통한 장점 극대화

**3. Object**

Object(객체)는 OOP에서 데이터(변수)와 그 데이터에 관련되는 동작(함수). 즉 절차, 방법, 기능을 모두 포함한 개념

**예)**

기차역에서 승차권을 발매하는 경우, 실체인 ‘손님’과 동작인 ‘승차권 주문’은 하나의 객체이며, 실체인 ‘역무원’과 동작인 ‘승차권 발매’도 하나의 객체이다.

같은 성질, 같은 구조와 형태를 가지는 객체는 등급으로 정의하고 등급에 속하는 객체는 그 등급의 인스턴스라고 한다.

**4. Overloading vs Overriding**

Overloading(오버로딩)

-       같은 이름의 메소드를 여러 개 정의하는 것.

-       매개변수의 타입이 다르거나 개수가 달라야 한다.

-       Return type과 접근 제어자는 영향을 주지 않음

Overriding(오버라이딩)

-       상속에서 나온 개념

-       상위 클래스(부모 클래스)의 메소드를 하위 클래스(자식 클래스)에서 재정의

**5. JDBC**

-       Java Data Base Connection의 약자로 JAVA 언어를 통해 데이터베이스에 접근할 수 있는 프로그래밍

**6. Interface, Abstract**

Interface

-       일종의 추상 클래스

-       오직 추상메서드와 상수만을 멤버로 갖는다.

-       Implements 키워드를 사용

-       상속의 관계가 없는 클래스간 서로 공동되는 로직을 구현하여 쓸 수 있도록 한다.

-       Extends는 하나의 클래스만 상속 가능하나 Interface는 다중 상속이 가능하다.

Abstract

-       추상메소드를 하나 이상 가진 클래스

-       자신의 생성자로 객체 생성 불가능

-       하위 클래스를 참조하여 상위 클래스의 객체를 생성

-       하위 클래스를 제어하기 위해 사용

Interface vs Abstract

**공통점**

-       New 연산자로 인스턴스 생성 불가능

-       프로토타입만 있는 메서드를 갖는다.

-       사용하기 위해서는 하위클래스에서 확장/구현해야 한다.

**차이점**

-       사용하는 키워드가 다르다.

-       Abstract는 일반 메서드를 사용할 수 있지만, Interface는 메서드 선언만 가능하다

**7. Call by Reference, Call by Value**

Call by References     – 매개변수의 원래 주소에 값을 저장하는 방식, 클래스 객체를 인수로 전달한 경우

Call by Value            – 인수로 기본 데이터형을 사용. 주어진 값을 복사하여 처리하는 방식. 메서드 내의 처리 결과는 메서드 밖의 변수에 영향을 미치지 않늗다.

**8. Static의 의미**

- 클래스가 로딩될 때, 메모리 공간을 할당하는데 처음 설정된 메모리 공간이 변하지 않음을 의미

- 객체를 아무리 많이 만들어도 해당 변수는 하나만 존재(객체와 무관한 키워드)

**9. Garbage Collection(가비지 컬렉션)**

- 시스템에서 더 이상 사용하지 않는 동적 할당된 메모리 블록을 찾아 자동으로 다시 사용 가능한 자원으로 회수 하는 것. 시스템에서 가비지 컬렉션을 수행하는 부분을 가비지 컬렉터라 부른다. 고로 자바에서는 메모리 문제를 신경쓰지 않아도 된다!

**10. Primitive type과 Reference type**

Primitive type – 변수에 값 자체를 저장(정수형, 실수형, 문자형, 논리형) – Wrapper Class를 통해 객체로 변환 가능

Reference type – 메모리상에 객체가 있는 위치를 저장

종류 – Class, Interface, Array 등

**11. Wrapper Class**

Primitive type으로 표현할 수 있는 간단한 데이터를 객체로 만들어야 할 경우가 있는데 그러한 기능을 지원

**12. Thread**

Thraed(쓰레드)      – 프로세스내에서 동시에 실행되는 독립적인 실행 단위를 말함. 장점으로는 자원을 많이 사용하지 않고 구현이 쉬우며 범용성이 높다.

Process(프로세스)  – 운영체제에서 실행중인 하나의 프로그램(하나 이상의 쓰레드를 포함한다.)

Thread 장점

-       빠른 프로세스 생성

-       적은 메모리 사용

-       쉬운 정보 공유

Thread 단점

-       교착상태에 빠질 수 있다.

**\*교착상태** : 다중 프로그래밍 체제에서 하나 또는 그 이상의 프로세스가 수행할 수 없는 어떤 특정 시간을 기다리고 있는 상태

Thread와 Process 차이

여러 분야에서 ‘과정’ 또는 ‘처리’라는 뜻으로 사용되는 용어로 컴퓨터 분야에서는 ‘실행중인 프로그램’이라는 뜻으로 쓰인다. 이 프로세스 내에서 실행되는 각각의 일을 스레드라고 한다. 프로세스 내에서 실행되는 세부 작업 단위로 여러 개의 스레드가 하나의 프로세스를 이루게 되는 것이다.

**13. 접근 제한자(public > protected > default > private)**

Public – 접근 제한이 없다.(같은 프로젝트 내에 어디서든 사용 가능)

Protected – 같은 패키지 내, 다른 패키지에서 상속 받아 자손 클래스에서 접근 가능

Default – 같은 패키지 내에서만 접근 가능

Private – 같은 클래스 내에서만 접근 가능

**14. Stack, Queue**

Stack

-       LIFO(Last In First Out)의 후입선출 구조

-       push(); 를 이용한 데이터 입력, pop();을 이용한 데이터 출력

-       예) 시스템 스택 : 함수의 호출과 복귀 순서는 스택의 구조를 응용하여 관리.

-       역순 문자열 만들기, 수식의 괄호 검사, 수식의 후위 표기법 변환

Queue

-       FIFO(First In First Out)의 선입선출 구조

-       enQueue();를 이용한 데이터 입력, deQueue();를 이용한 데이터 출력

-       예) 우선순위가 같은 작업 예약(인쇄 대기열), 선입 선출이 필요한 대기열(티켓 카운터)

\* Linear Queue(선형큐)는 메모리 재사용이 불가능 이러한 문제점을 보완하여 Circular Queue(원형 큐)가 나옴

**15. Singleton Design Pattern(싱글톤 디자인 패턴, 싱글톤 패턴)**

- 클래스 인스턴스가 하나만 만들어지도록 하고, 그 인스턴스에 대한 전역 접근을 제공한다.

**16. String과 StringBuffer의 차이**

자바의 기본 데이터 타입인 int, float, char 등과 다르게 String은 데이터 타입이 아닌 클래스 객체입니다. 또한 String은 불변의 객체입니다. 한번 String name =”길동”; 라고 선언하게 되면 먼저 String 객체타입인 name이라는 인스턴스를 만들고 메모리에 “길동”을 올려버립니다. 그리고 name이 “길동”을 참조하는 레퍼런스가 되는 것입니다.

중요한점은 이제 이 name에 새로운 내용을 추가하겠습니다. name += “홍”; 와 같은 소스를 적용하게 되면 name인스턴스가 가리키는 값은 “길동홍”가 됩니다. 우리는 이런식으로 흔히 코딩을 했을 텐데 아까 위에서 String이 불변 객체라고 설명을 했듯이 한번 선언된 내용에 추가적으로 바뀌지 않습니다. String은 final char배열 형태이기 때문에 내용의 추가와 삭제가 되지 않거든요. 그럼 도대체 어떻게 name변수의 값이 바뀌는 건지 답은 내부에 있습니다.

name = name + “홍”; 구문이 실행될 때 실제로는 스트링 버퍼를 새로 생성해서 name이 가리키는 “길동”을 만들어주고 스트링 버퍼의 append 함수를 이용하여 “홍” 를 붙여줍니다. 그렇게 완성된 스트링 버퍼값을 메모리에 올리고 name은 다시 이 값을 참조하게 됩니다. 스트링 버퍼는 char타입의 배열로 되어 있어서 한글자 한글자를 append할 수 있다는 것을 아셔야합니다. 그럼 그와중에 생겨난 메모리 안의 “길동” 이라는 값과 “홍”라는 값은 가비지 컬렉터가 가지고 있다가 필요없어서 버립니다. 그 짧은 순간에 이런 일처리가 진행이 되어서 스트링 버퍼를 사용하는 것이 스트링 객체를 사용하는 것보다 빠르다는게 제 생각입니다.

**17. 자바의 메모리 영역**

- 메서드 영역 : static 변수, 전역변수, 코드에서 사용되는 Class 정보 등이 올라간다. 코드에서 사용되는 class들을 로더로 읽어 클래스별로 런타임 필드데이터, 메서드 데이터 등을 분류해 저장한다.

- 스택(Stack) : 지역변수, 함수(메서드) 등이 할당되는 LIFO(Last In First Out) 방식의 메모리

- 힙(Heap) : new 연산자를 통한 동적 할당된 객체들이 저장되며, Garbage 컬렉션에 의해 메모리가 관리되어 진다.

**18. DAO와 DTO**

- DAO : Data Access Object의 약자로 간단히 데이터베이스의 데이터에 접근을 위한 객체이다. 데이터베이스에 접근을 하기위한 로직과 비즈니스 로직을 분리하기 위해서 사용한다. DB를 사용해 데이터를 조회하거나 조작하는 기능을 전담하도록 만든 오브젝트를 말한다.

- DTO : Data Transfer Object의 약자로 VO(Value Object)로 바꾸어 말할 수 있는데 계층간 데이터 교환을 위한 JavaBean을 말합니다. 여기서 말하는 계층이란 Controller, View, Business Layer, Persistent Layer를 말하며 각 계층간 데이터 교환을 위한 객체를 DTO 또는 VO라고 부릅니다. 그런데 VO는 동일한 개념이지만 read only 속성을 가집니다.

**19. 변수 명명법**

- 헝가리언 표기법 : 자료형을 식별자에 같이 포함

ex) inum; int int\_num; 인터페이스명.

- 파스칼 표기법 : 식별자가 한 단어나 혹은 여러 단어로 조합(언더바X), 각단어의 첫문자만 대문자로

ex) KorScore

- 캐멀 표기법 : 모든 단어를 공백없이 조합(언더바X) – 첫단어의 첫문자는 소문자로

ex) korScore

- 스네이크 표기법 : ex)eng\_score

**1. Servlet, JSP**

Servlet – Container가 이해할 수 있도록 구성된 자바코드로 이루어진 것(Html in JAVA)

JSP(Java Server Page) – html 기반에 자바코드를 블록화하여 삽입한 것(JAVA in Html)

**2. Get과 Post 방식**

Get방식

- 클라이언트에서 서버로 데이터를 전달할 때, 주소 뒤에 “이름”과 “값”이 결합된 쿼리 스트링 형태로 전달.

- 주소창에 쿼리 스트링이 그대로 보여지기 때문에 보안성이 떨어진다.

- 길이에 제한이 있다.(= 전송 데이터의 한계가 있다.)

- Post방식보다 상대적으로 전송 속도가 빠르다

Post방식

- 일정 크기 이상의 데이터를 보내야할 때 사용한다.

- 서버로 보내기 전에 인코딩하고, 전송 후 서버에서는 다시 디코딩 작업을 한다.

- 주소창에 전송하는 데이터의 정보가 노출되지 않아 Get방식에 비해 보안성이 높다.

- 속도가 Get방식보다 느리다.

- 쿼리 스트링(문자열) 데이터 뿐만 아니라, 라디오 버튼, 텍스트 박스 같은 객체들의 값도 전송가능

Get과 Post 차이점

- Get은 주로 웹 브라우저가 웹 서버에 데이터를 요청할 때 사용

- Post는 웹 브라우저가 웹 서버에 데이터를 전달하기 위해 사용

- Get을 사용하면 웹 브라우저에서 웹 서버로 전달되는 데이터가 인코딩되어 URL에 붙는다.

- Post방식은 전달되는 데이터가 보이지 않는다.

- Get방식은 전달되는 데이터가 255개의 문자를 초과하면 문제가 발생할 수 있다.

- 웹서버에 많은 데이터를 전달하기 위해서는 Post방식을 사용하는 것이 바람직하다.

**3. Session과 Cookie**

Session과 Cookie 사용 이유

현재 우리가 인터넷에서 사용하고 있는 HTTP 프로토콜은 연결 지향적인 성격을 버렸기 때문에 새로운 페이지를 요청할 때마다 새로운 접속이 이루어지며 이전 페이지와 현재 페이지 간의 관계가 지속되지 않는다. 이에 따라 HTTP프로토콜을 이용하게 되는 웹 사이트에서는 웹 페이지에 특정 방문자가 머무르고 있는 동안에 그 방문자의 상태를 지속시키기 위해 쿠키와 세션을 이용한다.

Session

- 특정 웹사이트에서 사용자가 머무르는 기간 또는 한 명의 사용자의 한번의 방문을 의미한다.

- Session에 관련된 데이터는 Server에 저장된다.

- 웹 브라우저의 캐시에 저장되어 브라우저가 닫히거나 서버에서 삭제시 사라진다.

- Cookie에 비해 보안성이 좋다.

Cookie

- 사용자 정보를 유지할 수 없다는 HTTP의 한계를 극복할 수 있는 방법

- 인터넷 웹 사이트의 방문 기록을 남겨 사용자와 웹 사이트 사이를 매개해주는 정보이다.

- Cookie는 인터넷 사용자가 특정 웹서버에 접속할 때, 생성되는 개인 아이디와 비밀번호, 방문한 사이트의 정보를 담은 임시파일로써, Server가 아닌 Client에 텍스트 파일로 저장되어 다음에 해당 웹서버를 찾을 경우 웹서버에서는 그가 누구인지 어떤 정보를 주로 찾았는지 등을 파악할 때 사용된다.

- Cookie는 Client PC에 저장되는 정보이기 때문에, 다른 사용자에 의해서 임의로 변경이 가능하다.( 정보유출 가능, Session보다 보안성이 낮은 이유 )

**Q**. 보안성이 낮은 Cookie대신 Session을 사용하면 되는데 안하는 이유?

**A**. 모든 정보를 Session에 저장하면 Server의 메모리를 과도하게 사용하게 되어 Server에 무리가 가게 된다.

**4. MVC 패턴**

MVC란?

- 객체지향프로그래밍에서, MVC란 사용자 인터페이스를 성공적이며 효과적으로 데이터 모형에 관련 시키기 위한 방법론 또는 설계 방식중 하나이다. MVC방식은 자바, Smalltalk

- MVC 패턴은 목적 코드의 재사용에 유용한 것은 물론, 사용자 인터페이스와 응용프로그램 개발에 소요되는 현저하게 줄여주는 형식이라고 많은 개발자들이 평가

MVC 구성요소

Model – 소프트웨어 응용과 그와 관련된 고급 클래스 내의 논리적 데이터 기반 구조를 표현. 이 목적 모형은 사용자 인터페이스에 관한 어떠한 정보도 가지고 있지 않다.

View – 사용자 인터페이스 내의 구성요소들을 표현(사용자에게 보여지는 화면)

Controller – Model과 View를 연결하고 있는 클래스를 대표, Model과 View 내의 클래스들 간의 정보교환하는 데 사용

**데이터베이스 용어정리**

**1. 데이터베이스**

- 어느 한 조직에서 업무처리를 위해 다수의 응용시스템 혹은 다수의 사용자들이 공용으로 사용하기 위해 저장된 운영 데이터의 집합.

**2. 데이터베이스 언어**

- 정의어(DDL : Data Definition Language) : 데이터베이스 구조를 정의,수정,삭제하는 언어. (create, alter, drop)

- 조작어(DML : Data Manipulation Language) : 데이터베이스 내의 자료 검색, 삽입, 갱신, 삭제를 위한 언어. (select, delete, update, insert)

- 제어어(DCL : Data Control Language) : 데이터에 대해 무결성 유지, 병행 수행 제어, 보호와 관리를 위한 언어.(commit, rollback, grant, revoke)

**3. 스키마(Schema)**

- 데이터베이스의 전체적인 구조와 제약조건에 대한 명세를 기술 정의한 것을 말한다.

**4. 무결성**

- 개체 무결성 : 기본키는 NULL값이 올 수 없으며, 중복될 수 없다.

- 참조 무결성 : 외래키는 NULL값이 올 수 있으며, 부모 테이블의 기본키에 종속되어야 한다.

- 도메인 무결성, 고유(Unique)무결성, NULL 무결성, 키 무결성이 있다.

**5. 뷰**

- 하나이상의 테이블로부터 유도되어 만들어진 가상테이블, 실제로 기억공간을 차지하지 않으며 논리적 독립성을 제공하고 데이터 접근 제어로부터 보안성을 향상시킴

- 정의된 기본 테이블이 삭제되면 뷰도 자동적으로 제거된다.

- 한번 정의된 뷰는 변경할 수 없으며, 삭제한 후 다시 생성해야 한다.(replace)

- 검색은 일반 테이블과 동일한 방식으로 한다.

- 뷰에 대한 삽입, 삭제, 갱신에 대한 제약이 따른다. 컬럼에 기본키가 포함되어 있지 않으면 삽입, 갱신, 삭제가 되지 않는다.

**6. 인덱스 생성시 고려할 사항**

- 테이블의 전체 데이터 중 적은 양을 조회할 때 사용한다.

- 테이블에 데이터가 적을수록 인덱스의 효율은 떨어진다.

- 데이터의 유일성이 높을수록, 데이터의 범위가 넓을수록 인덱스의 효율은 올라간다.

- NULL이 적은 컬럼이 인덱스 효율이 좋다.

- 결합 인덱스의 경우 자주 사용되는 컬럼을 앞쪽에 배치한다.

**데이터 베이스 질문들**

**1. SELECT문은 무엇입니까?**

- Select문은 사용자들이 데이터베이스 내 테이블에서 값들을 선택할 수 있도록 합니다. 데이터베이스 테이블에서 어떤 값들을 선택하는 가는 SQL질의 내 다양한 조건에 달려있습니다.

**2. 전체 이름이 아닌 이름의 일부로 값을 비교할 수 있는 방법은 무엇입니까?(like 절)**

- SELECT \* FROM people WHERE name LIKE ‘%na%’; 이렇게 하면 name에 문자열 na를 포함하는 name레코드들을 가진 레코드셋을 반환합니다.

**3. Insert문은 무엇입니까?**

- 데이터베이스에 정보를 삽입하는 문입니다.

**4. 데이터베이스에서 레코드는 어떻게 삭제합니까?(DELETE문)**

- DELETE문은 데이터베이스에서 레코드 또는 특정 컬럼의 값을 삭제합니다.

**5. 테이블에서 어떻게 중복이 없는(별개) 항목을 얻을 수 있습니까?(distinct)**

- DISTINCT에 대해서 설명을 하자면 중복된 결과를 제거하고 하나만 원하고자 할 때 쓰이는 문법이다. 데이터베이스 테이블에 특정 값에 a,a,b,c가 있으면 a,b,c만 나오게 할 수 있다.

**6. 어떤 순서로 정렬된 결과를 얻는 방법은 무엇입니까?(order by)**

- 프로그램에서 ORDER BY 키워드를 사용하여 정렬하고 정렬된 결과를 반환하여 정렬을 수행하는 수고를 덜 수 있습니다. 키워드는 ORDER BY 렬에 사용됩니다. ORDER BY를 이용함으로써 오름차순으로 정렬되고 ‘DESC’값을 주게 되면 내림차순이 된다.

**7. 테이블의 전체 레코드 수를 어떻게 얻을 수 있습니까?**

- 예를 들어 SELECT COUNT(\*) FROM employees WHERE age > 40 에서 처럼 count키워드를 사용하게 되면 카운트(수)를 얻을 수 있습니다.

**8. GROUP BY는 무엇입니까?**

- GROUP BY 키워드는 집계함수(SUM같은)가 호출되 때마다 모든 컬럼 값들의 합계를 반환하기 때문에 SQL에 추가되었습니다. GROUP BY기능 없이 컬럼 값들의 개별 그룹마다 합계를 구하는 것은 불가능합니다.

**9. 테이블을 드롭(DROP)하는 것과 자르는 것(Truncate), 그리고 테이블 내 모든 레코드를 삭제(Delete)하는 것의 차이점은 무엇입니까?**

- DELETE TABLE은 로그되는 작업이기 때문에 삭제되는 각 행은 트랜잭션 로그에 기록되고 이것은 작업을 느리게 합니다. TRUCATE TABLE 역시 테이블 내 행들을 삭제하지만 삭제되는 각 행을 기록하지 않고 대신 테이블의 데이터베이스 할당 해제를 기록하여 작업이 빠릅니다. TRUNCATE TABLE는 롤백할 수 없습니다.

- DELETE 명령어는 데이터는 지워지지만 테이블 용량은 줄어들지 않는다. 원하는 데이터만 지울 수 있다. 삭제 후 RollBack 가능하다.

- TRUNCATE 명령어는 용량이 줄어들고, 인덱스 등도 모두 삭제된다. 테이블은 삭제하지는 않고 데이터만 삭제한다. 한꺼번에 다 지워야 한다. 삭제후 절대 되돌릴 수 없다.

- DROP 명령어는 테이블 전체를 삭제,공간, 객체를 삭제한다. 삭제 후 절대 되돌릴 수 없다.

**10. Where 절과 Having절의 다른 점은 무엇입니까?**

- Having절은 그룹함수의 그룹의 조건으로 사용되고, where 절은 select할 데이터에 조건을 주는 역할입니다.

**11.기본키(Primary Key)와 유일키(Unique Key)의 차이점은 무엇입니까?**

-  기본키는 널을 허용하지 않지만 유일키는 모든 컬럼 중 유일하게 하나에 대한 NULL을 허용합니다. 그래서 unique키는 개체하나하나를 구분할 기본키가 될 수 없다.

**12. 쿼리 순서**

- SELECT -> FROM -> WHERE -> GROUP BY -> ORDER BY

**. Framework**

- 특정 형태의 소프트웨어 문제를 해결하기 위해 상호 협력하는 클래스 프레임과 인터페이스 프레임의 집합.

- 특정한 틀을 만들어 놓고 거기에 살을 붙여 놓음으로써 프로그램을 만들어 작업시간을 줄여주는 것이다.

- 프레임워크는 특정 개념들의 추상화를 제공하는 여러 클래스나 컴포넌트로 구성된다.

- 프레임워크는 이렇게 추상적인 개념들이 문제를 해결하기 위해 같이 작업하는 방법을 정의한다.

- 프레임워크는 좀 더 높은 수준에서 패턴을 조작한다.

\* 프레임워크가 중요한 이유는 객체지향 개발을 하게 되면서 개발자의 취향에 따라 다양한 프로그램이 나오게 되었다. 프로그램 개발에 투입되는 개발자도 점점 늘어남에 따라 전체 시스템의 통합성, 일관성이 부족하게 되었기 때문이다. 그래서 개발자의 자유를 제한하기 위해 프레임워크를 도입했다.

프레임워크가 가져야할 특징

a. 개발자들이 따라야할 가이드라인을 가진다.

b. 개발할 수 있는 범위가 정해져 있다.

c. 개발자를 위한 다양한 도구들이 지원된다.

프레임워크의 장/단점

장점  –  개발 시간을 줄일 수 있고 오류로부터 자유로울 수 있다.

단점  –  프레임워크에 너무 의존하면 개발 능력이 떨어져서 프레임워크 없이 개발하는 것이 불가능해진다.

**2. Spring Framework(스프링 프레임워크)**

- 자바(JAVA) 플랫폼을 위한 오픈소스(Open Source) 애플리케이션 프레임워크(Framework)

- 자바 엔터프라이즈 개발을 편하게 해주는 오픈소스 경량급 애플리케이션 프레임워크

- 자바개발을 위한 프레임워크로 종속 객체를 생성해주고, 조립해주는 도구

- 자바로 된 프레임워크로 자바SE로 된 자바 객체(POJO)를 자바EE에 의존적이지 않게 연결해주는 역할

**3. 스프링 특징 간단히**

- 크기와 부하의 측면에서 경량

- 제어 역행(IoC)이라는 기술을 통해 애플리케이션의 느슨한 결합을 도모

- 관점 지향 프로그래밍(AOP)을 위한 풍부한 자원

- 애플리케이션 객체의 생명주기와 설정을 포함하고 관리한다는 점에서 일종의 컨테이너(Container)라고 할 수 있음.

- 간단한 컴포넌트로 복잡한 애플리케이션을 구성하고 설정할 수 있음.

**4. 스프링 특징 자세히**

a. 경량 컨테이너로서 자바 객체를 직접 관리 => 각각의 객체 생성, 소멸과 같은 라이프 사이클을 관리하며 스프링으로부터 필요한 객체를 얻어올 수 있다.

b. 스프링은 POJO(Plain Old Java Object) 방식의 프레임워크. => 일반적인 J2EE 프레임워크에 비해 구현을 위해 특정한 인터페이스를 구현하거나 상속을 받을 필요가 없어 기존에 존재하는 라이브러리 등을 지원하기에 용이하고 객체가 가볍다.

c. 스프링은 제어의 역행(IoC : Inversion of Control)을 지원 => 컨트롤의 제어권이 사용자가 아니라 프레임워크에 있어서 필요에 따라 스프링에서 사용자의 코드를 호출한다.

d. 스프링은 의존성 주입(DI : Dependency Injection)을 지원 => 각각의 계층이나 서비스들 간에 의존성이 존재할 경우 프레임워크가 서로 연결시켜준다.

e. 스프링은 관점 지향 프로그래밍(AOP : Aspect-Oriented Programming)을 지원 => 따라서 트랜잭션이나 로깅, 보안과 같이 여러 모듈에서 공통적으로 사용하는 기능의 경우 해당 기능을 분리하여 관리할 수 있다.

f. 스프링은 영속성과 관련된 다양한 서비스를 지원 => MyBatis나 Hibernate등 이미 완성도가 높은 데이터베이스 처리 라이브러리와 연결할 수 있는 인터페이스를 제공한다.

g. 스프링은 확장성이 높음 => 스프링 프레임워크에 통합하기 위해 간단하게 기존 라이브러리를 감싸는 정도로 스프링에서 사용이 가능하기 때문에 수많은 라이브러리가 이미 스프링에서 지원되고 있고 스프링에서 사용되는 라이브러리를 별도로 분리하기도 용이하다.

**5. Spring MVC 구조의 처리과정**

1) DispatcherServlet : 어플리케이션으로 들어오는 모든 Request를 받는 관문이다. Request를 실제로 처리할 Controller에게 전달하고 그 결과값을 받아서 View에게 전달하여 적절한 응답을 생성할 수 있도록 흐름을 제어한다.

2) HandlerMapping : Request URL 각각 어떤 Controller가 실제로 처리할 것인지 찾아주는 역할

3) Controller : Request를 직접 처리한 후 그 결과를 다시 DispatcherServlet에게 돌려준다.

4) ModelAndView : Controller가 처리한 결과와 그 결과를 보여줄 View에 관한 정보를 담고있는 객체이다.

5) ViewResolver : View관련 정보를 갖고 실제 View를 찾아주는 역할을 한다.

6) View : Controller가 처리한 결과값을 보여줄 View를 생성한다.

시즈원소프트

자바/JSP/SPRING

메서드 오버로딩/로버라이딩

오버로딩의 정의는 **자바의 한 클래스 내에 이미 사용하려는 이름과 같은 이름을 가진 메소드가 있더라도 매개변수의 개수 또는 타입이 다르면, 같은 이름을 사용해서 메소드를 정의할 수 있다.**

**메소드의 이름이 같고, 매개변수의 개수나 타입이 달라야 한다.** 주의할 점은 **'리턴 값만' 다른 것은 오버로딩을 할 수 없다는 것이다.**

오버라이딩: **부모 클래스로부터 상속받은 메소드를 자식 클래스에서 재정의**

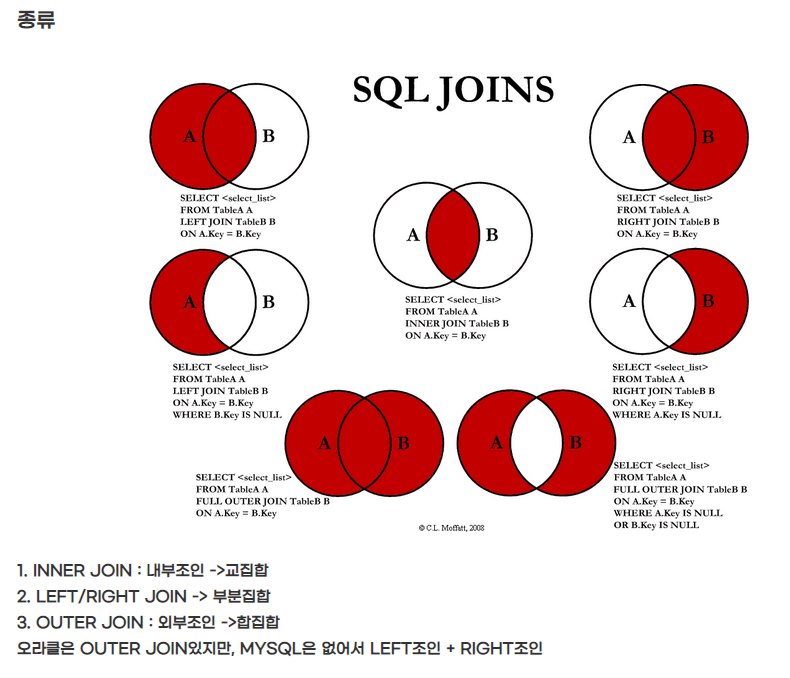
오버라이딩은 부모 클래스의 메소드를 재정의하는 것이므로, 자식 클래스에서는 **오버라이딩하고자 하는 메소드의 이름, 매개변수, 리턴 값이 모두 같아야 한다**

DB 조인. 셀프조인, 인터조인, 레프트조인, 아웃조인…

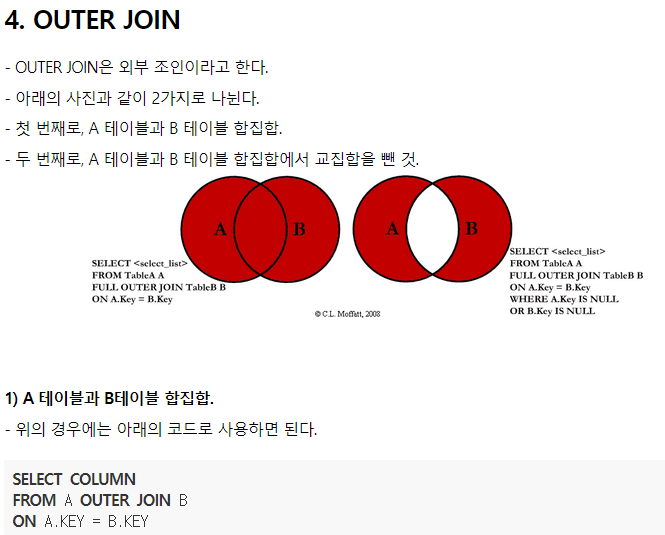
조인: 둘 이상의 테이블을 연결해서 데이터를 검색하는 방법

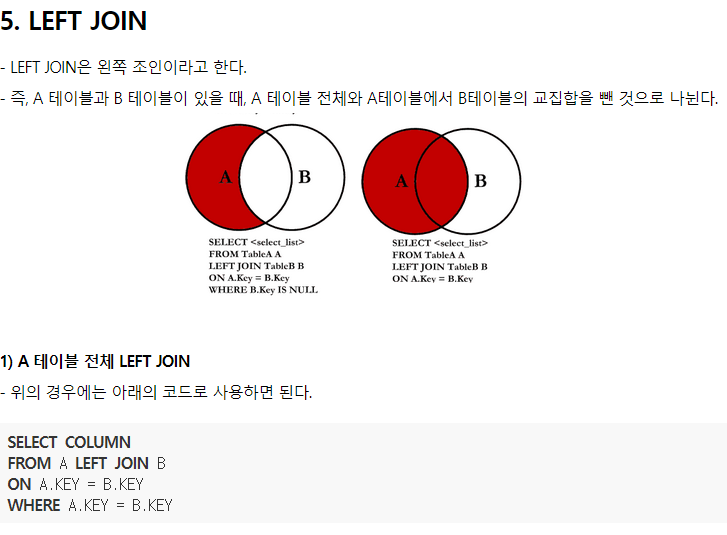
연결하려면 테이블들이 적어도 하나의 컬럼을 공유하고 있어야함

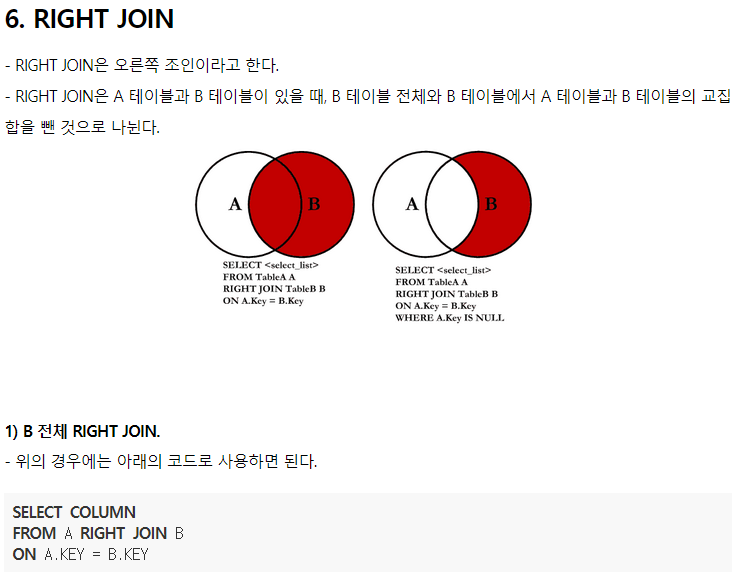
공유하고 있는 컬럼을 PK, FK값으로 사용











**트렌젝션**

데이터베이스의 상태를 변화시키기 해서 수행하는 작업의 단위를 뜻한다.

간단하게 말해서 아래의 질의어(SQL)를 이용하여 데이터베이스를 접근 하는 것을 의미한다.

SELECT

INSERT

DELETE

UPDATE

착각하지 말아야 할 것은, 작업의 단위는 질의어 한문장이 아니라는 점이다.

작업단위는 많은 질의어 명령문들을 사람이 정하는 기준에 따라 정하는 것을 의미한다.

게시판을 예로 들어보자.

게시판 사용자는 게시글을 작성하고, 올리기 버튼을 누른다. 그 후에 다시 게시판에 돌아왔을때,게시판은 자신의 글이 포함된 업데이트된 게시판을 보게 된다.

이러한 상황을 데이터베이스 작업으로 옮기면, 사용자가 올리기 버튼을 눌렀을 시, Insert 문을 사용하여

사용자가 입력한 게시글의 데이터를 옮긴다. 그 후에, 게시판을 구성할 데이터를 다시 Select 하여 최신 정보로

유지한다. 여기서 작업의 단위는 insert문과 select문 둘다 를 합친것이다. 이러한 작업단위를 하나의 트랜잭션이라 한다.

관리자나 개발자가 하나의 트랜잭션 설계를 잘하는 것이 데이터를 다루는 것에 많은 이점이 있다.

트랜잭션의 특징

트랜잭션의 특징은 크게 4가지로 구분된다.

원자성 (Atomicity)

일관성 (Consistency)

독립성 (Isolation)

지속성 (Durability)

첫번째로, 원자성은 트랜잭션이 데이터베이스에 모두 반영되던가, 아니면 전혀 반영되지 않아야 한다는 것이다. 트랜잭션은 사람이 설계한

논리적인 작업 단위로서, 일처리는 작업단위 별로 이루어 져야 사람이 다루는데 무리가 없다.

만약 트랜잭션 단위로 데이터가 처리되지 않는다면, 설계한 사람은 데이터 처리 시스템을 이해하기 힘들 뿐만 아니라, 오작동 했을시 원인을 찾기가 매우 힘들어질것이다.

두번째로, 일관성은 트랜잭션의 작업 처리 결과가 항상 일관성이 있어야 한다는 것이다.

트랜잭션이 진행되는 동안에 데이터베이스가 변경 되더라도 업데이트된 데이터베이스로 트랜잭션이 진행되는것이 아니라,

처음에 트랜잭션을 진행 하기 위해 참조한 데이터베이스로 진행된다. 이렇게 함으로써 각 사용자는 일관성 있는 데이터를 볼 수 있는 것이다.

세번째로, 독립성은 둘 이상의 트랜잭션이 동시에 실행되고 있을 경우 어떤 하나의 트랜잭션이라도, 다른 트랜잭션의 연산에 끼어들 수 없다는 점을 가리킨다.

하나의 특정 트랜잭션이 완료될때까지, 다른 트랜잭션이 특정 트랜잭션의 결과를 참조할 수 없다.

네번째로, 지속성은 트랜잭션이 성공적으로 완료됬을 경우, 결과는 영구적으로 반영되어야 한다는 점이다.

트랜잭션의 Commit, Rollback 연산

Commit이란 하나의 트랜잭션이 성공적으로 끝났고, 데이터베이스가 일관성있는 상태에 있을 때, 하나의 트랜잭션이 끝났다라는 것을

알려주기위해 사용하는 연산이다. 이 연산을 사용하면 수행했던 트랜잭션이 로그에 저장되며, 후에 Rollback 연산을 수행했었던 트랜잭션단위로 하는것을 도와준다.

Rollback이란 하나의 트랜잭션 처리가 비정상적으로 종료되어 트랜잭션의 원자성이 깨진경우, 트랜잭션을 처음부터 다시 시작하거나, 트랜잭션의 부분적으로만 연산된 결과를 다시 취소시킨다.

후에 사용자가 트랜잭션 처리된 단위대로 Rollback을 진행할 수도 있다.

락

Lock

Lock이란 트랜잭션 처리의 순차성을 보장하기 위한 방법입니다. 트랜잭션이란 DB의 나누어지지 않는 최소한의 처리 단위입니다. 트랜잭션에 대해서 모호하시거나 개념의 정립이 필요하시다면 제가 이전에 포스팅한 [데이터베이스] 트랜잭션과 격리성을 먼저 참조하시기바랍니다.

그리고 중요한 것은 DBMS마다 Lock을 구현하는 방식과 세부적인 방법이 다릅니다. 따라서 DBMS를 효과적으로 이용하기 위해서는 해당 DB의 Lock에 대한 이해가 요구됩니다.

Lock의 종류

Lock의 종류로는 공유(Shared) Lock과 베타(Exclusive) Lock이 있습니다. 공유락은 다른 말로 Read Lock이라고 불리며 베타락은 Write Lock이라고도 불립니다.

공유(Shared) Lock

공유 Lock은 데이터를 읽을 때 사용되어지는 Lock입니다. 이런 공유 Lock은 공유 Lock 끼리는 동시에 접근이 가능합니다. 즉, 하나의 데이터를 읽는 것은 여러 사용자가 동시에 할 수 있다라는 것입니다. 하지만 공유 Lock이 설정된 데이터에 베타 Lock을 사용할 수는 없습니다.

베타(Exclusive) Lock

베타 Lock은 데이터를 변경하고자 할 때 사용되며, 트랜잭션이 완료될 때까지 유지됩니다. 베타락은 Lock이 해제될 때까지 다른 트랜잭션(읽기 포함)은 해당 리소스에 접근할 수 없습니다. 또한 해당 Lock은 다른 트랜잭션이 수행되고 있는 데이터에 대해서는 접근하여 함께 Lock을 설정할 수 없습니다.

Lock의 설정 범위(Level)

Lock의 설정 범위(Level)에 대해서 알아보도록 하겠습니다.

데이터베이스

데이터베이스 범위의 lock은 전체 데이터베이스를 기준으로 lock 하는 것입니다. 즉, 1개의 세션만이 DB의 데이터에 접근이 가능합니다. 해당 기능은 일반적으로는 사용하지 않습니다. 사용하는 때가 있다면 DB의 소프트웨어 버전을 올린다던지 주요한 DB의 업데이트에 사용합니다.

파일

데이터베이스 파일을 기준으로 lock을 설정합니다. 파일 이란 테이블, row 등과 같은 실제 데이터가 쓰여지는 물리적인 저장소 입니다. 해당 범위의 Lock은 잘 사용되지는 않습니다.

테이블

테이블 수준의 Lock은 테이블을 기준으로 Lock을 설정합니다. 이는 테이블의 모든 행을 업데이트 하는 등의 전체 테이블에 영향을 주는 변경을 수행할 때 유용합니다. 즉, DDL(create, alter, drop 등) 구문과 함께 사용되며 DDL Lock이라고도 합니다.

페이지와 블럭

파일의 일부인 페이지와 블록을 기준으로 Lock을 설정합니다. 잘 사용되지는 않습니다.

컬럼

컬럼 기준의 Lock은 컬럼을 기준으로 Lock을 설정할 수 있습니다. 하지만 이 형식은 Lock 설정 및 해제의 리소스가 많이 들기 때문에 일반적으로 사용되지는 않습니다. 지원하는 DBMS도 많지 않습니다.

행(Row)

행 수준의 Lock은 1개의 행(Row)를 기준으로 Lock 설정을 합니다. DML에 대한 Lock으로 가장 일반적으로 사용하는 Lock입니다.

블로킹(Blocking)

블로킹은 Lock간(베타 - 베타, 베타 - 공유)의 경합이 발생하여 특정 Transaction이 작업을 진행하지 못하고 멈춰선 상태를 말합니다. 위에 설명했듯이 공유락 끼리는 블로킹이 발생하지 않지만 베타락은 블로킹을 발생시킵니다. 블로킹을 해소하기 위해서는 이전의 트랜잭션이 완료(커밋 OR 롤백)되어야 합니다. 뒤에 들어온 트랜잭션은 이전 트랜잭션이 마무리되어야 이후 진행이 가능합니다. 이런 경합은 성능에 좋지 않은 영향을 미칩니다. 따라서 경합을 최소화 할 필요가 있습니다.

blocking

DB를 사용하는 프로그래밍을 진행하면서 몇가지 주의사항을 알아보도록 하겠습니다.

한 트랜잭션의 길이를 너무 길게하는 것은 경합의 확률을 올립니다.

처음부터 설계할 때 같은 데이터를 갱신하는 트랜잭션이 동시에 수행되지 않도록 해야합니다.

트랜잭션 격리성 수준을 불필요하게 상향 조정하지 않습니다. (참고 : 트랜잭션 격리성 수준)

쿼리를 오랜시간 잡아두지 않도록 적절한 튜닝을 진행합니다.

이외에 DBMS에 따라서 lock 대기 시간 등을 설정할 수 있습니다.

교착상태(DeadLock)

교착상태는 두 트랜잭션이 각각 Lock을 설정하고 다음 서로의 Lock에 접근하여 값을 얻어오려고 할 때 이미 각각의 트랜잭션에 의해 Lock이 설정되어 있기 때문에 양쪽 트랜잭션 모두 영원히 처리가 되지않게 되는 상태를 말합니다. 예를 들어 보면, game\_master, game\_detail 테이블이 있습니다. 트랜잭션 A가 game\_master 테이블에 5번 Row를 수정했고 이제 game\_detail 테이블에 5번 Row를 이어서 수정하려고 합니다. 동시에 트랜잭션 B는 game\_detail 테이블의 5번 Row를 수정하고 이어서 game\_master 테이블의 5번 Row를 수정하려고 합니다. 이 경우 트랜잭션 A는 game\_master 테이블의 5번 Row에 배타 락을 설정했고 트랜잭션 B는 game\_detail 테이블의 5번 Row에 배타 락을 설정하였습니다. 그리고 교차로 트랜잭션 A는 game\_detail의 5번 row의 Lock 설정을 하려고 하고 트랜잭션 B는 game\_master의 5번 row에 Lock 설정을 하려고 합니다. 하지만 이미 각 row들은 서로다른 트랜잭션에 의해서 배타락 설정이 되어있습니다. 따라서 Lock이 해제되기를 서로 기다립니다. 하지만 이 Lock은 풀리지 않을 서로의 트랜잭션 기다리므로 영원히 풀리지 않을것입니다.

이미지로 나타내면 아래와 같습니다.

deadlock

그래서 교착상태가 발생하면 DBMS가 둘 중 한 트랜잭션에 에러를 발생시킴으로써 문제를 해결합니다. 교착상태가 발생할 가능성을 줄이기 위해서는 접근 순서를 동일하게 하는것이 중요합니다. 즉, 위의 예제라면 프로그래밍을 할 때 game\_master를 업데이트 한 후 game\_detail을 업데이트 한다와 같은 규칙을 정해 테이블 접근의 교차가 일어나지 않도록 하는것이 중요할 것입니다.

배열. 리스트 사용

배열

 데이터가 많아지고 **그룹 관리의 필요**에 따라 배열을 사용한다.

 **고정된 크기**를 갖는 **같은 자료형**의 원소들이 **연속적**인(논리적 저장 순서와 물리적 저장 순서가 일치) 형태로 구성된 자료구조

* 인덱스에 따라 값을 유지하므로 원소가 삭제되어도 빈자리가 남게되어 메모리가 낭비된다.
* 처음 크기를 10으로 지정한다면 5개의 데이터만 저장하더라도 실제 배열의 크기는 10이다.

 **인덱스(index)** : 각 원소의 번호로 0번부터 시작하며, 해당 원소에 접근한다.

 데이터 갯수가 확실하게 정해져 있고, 접근이 빈번한 경우 배열이 효율적이다.

 **cache hit** 가능성이 커져 성능에 큰 도움이 된다.

* cache hit : CPU가 참조하고자 하는 메모리가 캐시에 존재하고 있는 경우

 고정이고 연속적인 만큼 인덱스로 **random access**가 가능하다.

* 접근, 수정 O(1)으로 빠르게 조회가 가능하다.

 하지만 삽입과 삭제의 경우 연속적인 형태 유지를 위해 shift 연산을 해야하므로 O(n)이 된다.

리스트

 배열의 문제점을 해결하기 위한 자료구조

 **빈틈없는 데이터의 적재**라는 장점을 가진다.

* 원소를 삭제했을 때 삭제된 데이터 뒤 원소로 빈틈없이 연속적으로 위치시킨다.

 리스트의 핵심은 원소들 간의 순서로 **순서가 있는 데이터의 모임**이 리스트이며 리스트를 다른 이름으로 시퀀스(sequence)라고도 부른다.

 배열에서 인덱스는 유일무이한 식별자이지만 리스트에서는 몇 번째 데이터인지 정도의 의미를 가진다.

 빈 엘리먼트는 허용하지 않는다.

 순차성을 보장하지 못하기 때문에 spacial locality 보장이 되지 않아 **cash hit가 어렵다.**

 spacial locality : 프로그램 실행 시 접근하는 메모리 영역은 이미 접근이 이루어진 영역의 근처일 확률이 높다는 프로그램 성격 표현

 언어별로 list를 지원하는 것이 다르다

* 최근 언어들은 리스트를 기본으로 제공
* C : 리스트 지원 X
* JavaScript : 배열에 리스트 기능 포함
* Python : 기본 리스트, 배열 지원 X
* Java : 배열과 리스트 모두 지원, ArrayList와 LinkedList로 나뉨

 ArrayList와 LinkedList는 구현 방법에 따라 나뉜다.

* **ArrayList**
  + 배열을 이용해 리스트를 구현한 것
  + **접근이 빠름**(순차 x) 하지만 데이터 **추가와 삭제가 느림**
  + 동적으로 사용하기 힘듬(자바의 경우 자동으로 사이즈를 키워서 관리한다. → Dynamic Array)
* **LinkedList**
  + 연결로 구현한 리스트
  + 한 원소에서 값과 다음 원소의 주소를 알고 연결하는 방식
  + **순차적으로 접근**함 W(n)
  + 삽입, 삭제는 O(1)이지만 해당 지점까지 접근해야하므로 W(n)일 수 있음
  + → 배열과 다르게 논리적 저장 순서와 물리적 저장 순서가 일치하지 않는다!

 배열은 Compile time에 할당되는 정적 메모리 할당, 리스트는 새로운 Node가 추가되는 runtime에 할당되는 동적 메모리 할당

* 런타임 : 컴파일 과정을 마친 응용 프로그램이 사용자에 의해 실행될 때
* 컴파일 타임 : 소스 코드가 컴파일을 통해 기계어 코드로 변환되어 실행 가능한 프로그램이 되는 편집 과정

스프링 동작원리

