**Framework**

  ① 특정 형태의 소프트웨어 문제를 해결하기 위해, 상호 협력하는 클래스들과 인터페이스의 집합

  ② 장점 : 재사용성, 단순성, 역할구분, 확장성, 유지보수 용이

  ③ 라이브러리와 프레임워크의 차이

     •  라이브러리는 어플리케이션에서 호출할 수 있는 함수와 루틴으로 구성되어 있음

     •  프레임워크는 어플리케이션에서 특정 기능들을 제공하기 위해 확장할 수 있는 일반적이고

        상호 협력적인 컴포넌트를 제공

**DI와 AOP에 대해 설명하시오.**

  ① DI  : Dependency injection = 의존성 주입

     스프링을 적용하여 applicationContext.xml에서 설정만 해주면 외부 설정파일(xml)에서

연관관계에 있는 객체를 주입해주기 때문에 의존객체를 찾기 위한 코드가 필요하지 않게 됨.

     즉, 외부설정에서 객체를 찾아서 쓰기 때문에 코드 내에 의존관계를 맺는 코드 생성이 불필요.

  ② AOP : Aspect oriented programming = 약자 관심지향 프로그래밍

     프로그래밍 할 때 특정한 관심사를 가진 코드 부분을 별도의 모듈로 분리함으로써 기존 객체지향의 강력한 기능(상속, 위임)만으로는 처치가 곤란했던 중복을 할 수 있게 됨.

**Spring**

  ① Java Enterprise Application 개발에 사용되는 Application Flamework : 빠르고 효율적인 개발을 할 수 있도록 Application의 바탕이 되는 틀과 공통프로그래밍 모델, 기술 API를 제공.

  ② 특징

    • 스프링은 종속객체주입이라는 기술을 통해 낮은 결합도를 유지할 수 있음.

    • AOP를 이용하여 객체지향만이 아닌 관심지향 기법을 활용.

    • EJP 기능을 대체할 수 있음. 트랜젝션 처리를 위한 일관된 방법을 제공.

    • 레이어간 연결이 interface로 이어지기 때문에 interface 생성이 필요.

    • 다양한 프레임워크와의 통합

  ③  MVC 모델에서 커버하는 부분 : Controller 지원.

  ④ 실행 순서 (실행 구조)

    •  web.xml의 등록된 DispatcherServiet를 통해서 요청에 대해 진입.

    •  DispatcherServiet은 Client로부터 들어온 URL을 HandlerMapping이라는 곳으로 전송 후,

      URL을 분석하여 알맞은 Controller 이름을 다시 DispatcherServiet로 보냄.

    •  HandlerMapping이라는 것을 통해서 실행될 Controller의 이름을 입력받은 DispatcherServiet은

      전달받은 Controller를 실행함. 이렇게 실행된 Controller는 스프링에서 제공하는 ModelAndView

      객체에 View Page에 전달할 객체와 View Page 이름 정보를 담고 DispatcherServiet로 보냄.

    •  ViewResolver를 통해 보여질 View Page를 탐색한 후, View Page를 보여 줌.

 6. MyBatis(iBatis)

  ① Java에서 DataBase를 편하게 Handling 할 수 있데 해주는 Framework.

  ② 특징

    •  SQL문과 Java코드와의 분리만으로도 Java개발자는 Query문을 신경 쓰지 않아도 됨.

     SQL문이 변경되더라도 파라미터 값이 변경되지 않는다면 Java소스에서 수정부분이 없기 때문.

  ③ MVC 모델에서 커버하는 부분 : Model 지원

  ④ 실행 순서 (실행 구조)

    • 자바코드 내에서 특정 쿼리문을 실행하기 원할 때, 파라미터와 필요한 조건을 넘기기 위한 객체 생성.

    • SQLMaps를 실행하기 위해 쿼리의 객체와 이름을 넘겨 줌.

    • 쿼리가 실행되었을 때 SQLMaps는 쿼리 결과를 받기위해 정의된 클래스의 인스턴스를 생성하게 됨.

    • 인스턴스는 데이터베이스에 의해 반환된 Resultset으로부터 만들어짐.

**MVC 패턴이란?**

   ∘ Model, View, Control 의 역할을 확실하게 분리시켜 놓는 프로그래밍 기법.

**WAS (Web Application Server)**

  ① Server와 Client 사이에 있는 3-tier 방식으로서, Server가 처리하는 양이 많아지면서 Server에  생기는 부하를 해결하기 위해 개발. Client에서 요청이 들어오면 실제적인 처리는 WAS가 하고

 Server는 단지 Client에 결과 값을 뿌려주는 역할만 하게 됨.

  ② 대표적인 제품 : 톰캣, 웹스피어, 웹로직, 제우스 등.

**String, StringBuffer, StringBuilder의 차이점**

  ① String        클래스 : 상수 문자열, 한번 생성한 후 변하지 않는 문자열 용도.

  ② StringBuffer  클래스 : 프로그램 내에서 계속 변하는 문자열 용도.

  ③ StringBuilder 클래스 : Java5에 추가된 클래스로 StringBuffer와 기능이 같다.

   \* 차이점 : StringBuffer는 동기화(synchronized)되지만 StringBuilder는 그렇지 않다.

              즉, StringBuilder는 다중 thread에서는 안전하지 않으므로 동기화가 필요한 경우는

              StringBuffer를 사용하는 것이 좋다.

              StringBuilder에서 동기화 하려면 synchronized블록으로 감싸야 한다.

**트랜젝션, 롤백, 커밋**

  ① 트랜잭션      : 일련의 작업단위. 특성으로 ACID(원자성, 일관성, 고립성, 영구성)가 있다.

  ② 트랜잭션의 필요성 : 여러 작업이 한 자원에서 동시에 변경, 참조 필요시./작업공정이 긴 경우.

③ 트랜잭션의 시작

• 첫 DML(data조작어)구문이 실행될 때. (SELECT에선 제외)

                   • 사용자가 SAVEPOINT를 설정할 때.

④ 트랜잭션의 종료

• commit 수행   • rollback 수행   • 오라클 내부에서 자동 commit 수정

                  • DDL(data조작어), DCL(data제어어)이 실행될 때.

                  • 사용자가 sqlplus를 종료할 때, 시스템 크래쉬 발생.

⑤ 커밋과 롤백 전

• 롤백으로 데이터 회복 가능 • 현재 유저는 자신이 바꾼 데이터 확인 가능

                   • 다른 유제에게는 바뀐 데이터 내용이 반영되지 않음

                   • 영향받은 레코드는 잠금 상태

⑥ 커밋 후

• 데이터는 영구적으로 변경    • 복구 불가능    • 세이브 포인트 삭제

                  • 모든 유저에게 변경 내용 반영

                  • 해당 레코드 잠금 해제, 다른 유저들이 다룰 수 있음

⑦ 롤백 후

• 데이터의 변경 취소 • 데이터 이전 상태 회복  • 데이터에 대한 잠금 해제

**C와 C++의 차이점**

  ① C   : 함수 기반. 절차함수 기능만 가지고 있음.

         필요하면 함수 prototype을 만들어서 본문 추가.

  ② C++ : 클래스 기반. 컴파일러 단에서 “클래스”라는 객체지향적 방법을 지원.

         클래스에서 생성자/소멸자 등을 사용하여 자원 관리를 자동화 할 수 있음.

  연산자 재정의가 가능.

**다형성**

  ∘ 객체지향 개념에서의 다형성이란 ‘여러 가지 형태를 가질 수 있는 능력’을 의미하며

     자바에서는 한 타입의 참조변수로 여러 타입의 객체를 창조할 수 있도록 함으로써

     다형성을 프로그램적으로 구현.

**OverIoading (method 중복정의),**

**Overriding (method 재정의)**

  ① OverIoading

     - 기존의 method 인자를 이용하여 하나의 함수에 여러 가지 기능을 만드는 것.

  ② Overriding

     - 상위 클래스에 있는 method와 똑같은 method를 하위 클래스에 다시 만들기.

       즉 하위 클래스에서 method를 재정의하는 것.

**전역변수, 지역변수, 정적변수**

  ① 전역 변수

     - 프로그램에서 전반적으로 다 사용이 가능.

       함수 밖에서 정의해 놓으면 다른 함수에서도 사용할 수 있음.

  ② 지역 변수

     - 함수 안에서만 사용이 가능한 변수.

       다른 함수에서는 사용을 못함.

  ③ 정적 변수

     - 한 번 정의하면 그 값은 계속적으로 메모리에 남아 있음.

       프로그램이 종료될 때, 메모리에서 지워짐.

       함수 안에서 사용해도 함수가 끝날 때 지워지지 않음.

**프로그램 개발하는데 제일 중요한 것은?**

    \* 정답이 없으므로 주관적으로 답할 것.

    \* [참고]

       “프로젝트 진행과정”

        요구사항 분석-설계(데이타베이스,클래스,화면)-개발-테스트-이관/교육-트로젝트 종료

**자바의 특징**

   ∘ 플랫폼에 독립적인 객체지향 언어.

   ∘ 플랫폼에 독립적이라는 것은 리눅스 그리고 윈도우 등과 같은 개발환경에 제약을 받지 않고,

     어느 플랫폼에서나 코드의 호환성과 재사용이 자유롭다는 것을 의미한다.

**ERD(Entity Relationship Diagram)란?**

   ∘ 개체관계도 : 말로 되어있는 요구분석사항을 그림으로 그려 그 관계를 도출하는 것.

**Stack과 Queue  - Stack과 Queue  는 컴퓨터의 알고리즘에서 가장 흔하게 사용되는 자료 구조.**

  ① Stack : 모든 작업이 리스트의 한 쪽 끝에서만 수행되는 선형 리스트의 한 형태.

             리스트의 한 쪽 끝(TOP)에서 원소를 삽입하거나 제거하는데 사용.

             그러므로 리스트에서 Stack에 마지막으로 입력된 원소가 제일 먼저 제거의 대상이 됨.

             그래서 Stack을 LIFO (Last in First Out) 라고 함.

  ② Q방식 : 삽입은 REAR에서 이루어지고, 제거는 FRONT에서 이루어짐

             Q에 A, B, C, D, E 순으로 원소가 삽입되었다면, 제일 먼저 제거되는 원소는 A.

             Q는 FIFO (First in First Out) 라고 함.

**JSP란?**

  ① Java Servet Page : 자바를 기반으로 하는 스크립트 언어.

  ② 특징

    • 자바를 기반으로 하는 스크립트 언어로서 자바의 장점을 사용할 수 있음.

    • 자바를 기반으로 하고있으므로 플랫폼에 상관없이 사용할 수 있음.

    • 표현언어, 표현식, 스크립틀릿 등의 다양한 스크립트 요소와 액션 태그 등을 제공함으로써 보다 쉽게 웹 어플리케이션을 개발할 수 있음.

    • 서블릿/EJB 등의 엔터프라이즈 기술들과 잘 융합됨.

**JSP를 구동하기 위해 서버를 구축하는 과정**

1. 자바 인스톨

• JDK 인스톨

                     • JRE 인스톨 (JDK를 설치하면 설치가 되므로 따로 설치할 필요 없음)

  ② 환경변수 setting : 컴퓨터의 환경변수(Path)를 잡아야 함.

  ③ 미들웨어        : JSP를 웹으로 변환해 줄 수 있는 톰켓 설치.

**AJAX란?**

  ① 웹클라이언트 화면 제어를 보다 세밀하게 하기 위한 기술.

  ② 특징

    • 필요한 데이터만 서버로 보내고 해당 정보만 받음.

    • 서버로부터 정보를 받을 때는 xml or json 형태로 받고, Client에서는 그 정보들을 DOM을 통해

 화면을 재가공하여 보여주므로 서버와 교환되는 데이터의 양이 적어지므로 점유율을 낮출 수 있음.

    • Client에서의 데이터 처리로 서버에 부담이 덜 감으로 다른 응답 요청에 신속히 응대.

**XML이란?**

  ① 사용자 정의 태그를 통해 텍스트 데이터의 구조화된 문서 저장과 application간의 문서교환을 위해 1986년 ISO에 제안된 마크업 언어.

     SGML의 일부 기능과 인터넷에서 이용하기 쉬운 HTML의 장점을 도입하여 개발한 언어.

  ② 특징

     사용자 정의 태그가 가능.

     텍스트 데이터를 구조화하고 전송할 목적(텍스트 데이터의 재사용과 정보 검색 용이).

     Application  and WepApplication 모두 사용 가능.

**객체(object)**

   ∘ 효율적으로 정보를 관리하기위하여 사람들이 의미를 부여하고 분류하는 논리적인 단위.

     프로그래밍에서는 클래스에 정의된 내용대로 메모리에 생성된 것을 말함.

**클래스**

   ∘ 객체를 만드는 설계도 (객체를 생성하는 틀의 개념)

**객체와 인스턴스**

   ∘ 클래스로부터 객체를 만드는 과정을 클래스의 ‘인스턴스화’라고 하며,

     어떤 클래스로부터 만들어진 객체를 그 클래스의 ‘인스턴스’라고 함.

**객체지향프로그래밍(Objecr-Oriented Programming : OOP)**

     컴퓨터 프로그래밍의 패러다임의 하나다.

     객체지향 프로그래밍(OOP)은 컴퓨터 프로그램을 명령어의 목록으로 보는 시각에서 벗어나

     여러 개의 독립된 단위, 즉 “객체”들의 모임으로 파악하고자 하는 것.

     각각의 객체는 메시지를 주고 받고, 데이타를 처리할 수 있음.

  ① 캡슐화(Encapsulation): 하나의 문제 해결을 위한 data와 method를 한 단위로 묶는 것으로서, 클래스 내부 정의에 대해 외부에서 볼 수 없도록 함이 특징(은닉화)

  ② 추상화(Abstraction)  : 모델(Object)의 자세한 성질을 무시하고(숨기고) 그들의 일반적인 성질을 나타낸다는 것. 일반적으로 클래스는 클래스로 표현할 서브클래스

   (또는 객체)의 공통적인 성질과 행위를 일반화해 디자인되며 그로부터  생성된 객체는 자신의 고유한 성질을 갖게 됨.

  ③ 다형성(Polymorphism) : 다형성이란 같은 메시지에 대해 클래스에 따라 다른 행위를 하는 특성.일반적으로 같은 이름을 갖는 method에 대해 인자(Agument) 개수와

 Data Type에 따라 수행되는 행위가 달라짐을 의미. 다형성을 통해 사용자는 약속된 인터페이스를 따르는 서로 다른 객체들을 같은 방식으로 사용할 수 있게 됨.

  ④ 상속(inheritance)   : 기존에 있던 클래스(즉, 기존의 클래스로부터 상속받은)를 바탕으로

다른 특성을 추가해 새로운 클래스를 만들 수 있음.

  ⑤ 인스턴스(Instance)  : 인스턴스는 추상화 개념 또는 클래스 객체, 컴퓨터 프로세스 등과 같은

 템플릿(무엇인가를 만들 때 안내 역할 하는 데 사용되는 형식, 꼴,틀 또는 모형 등을 의미)이 실제로 구현된 것.

**자바언어와 기존의 언어의 다른 특징**

   ∘ 포인터를 사용하지 않는다. (포인터는 존재, 연산을 허용하지 않음)

   ∘ 자동으로 쓰레기 수집(garbage collection) 기능을 수행한다.

   ∘ 엄격한 형 검사(strict type checking)를 수행하여 에러를 조기에 발견한다.

   ∘ 실행시간에 발생하는 에러를 처리한다.

**디버깅**

   ∘ 버그(결함) 에러를 수정하는 것.

**DBMS (Database Management System)**

   ∘ 데이터베이스 관리시스템 : 응용프로그램이 데이터에 대한 액세스가 가능하도록 DB를

                               관리하는 소프트웨어 기능 및 데이터가 저장되어 있는 장소.

**RDBMS (Relational Database Management System)**

   ∘ 관계형  〃              : 일반적으로 관계형 데이터 모델에 기반한 데이터베이스를

                               관계형 데이터 베이스라고 함.

   ∘ 특징

    • 데이터를 테이블로 표현하고 테이블을 집합으로 받아들임.

    • 데이터를 조작하는 데는 집합론에 기초한 수학적인 연산을 테이블에 대해 실행.

**InnerClass 를 쓰는 이유**

   ∘ 내부 클래스를 사용하면 같은 패키지에 있는 다른 클래스한테 까지도 숨길 수 있기 때문에

     outer class를 통하지 않고서는 접근할 수 없음. 보안성이 좋아짐.

**미들웨어 (Middleware)**

   ∘ 최근에 네트워크 상호 호환과 연동상태를 관리하는 분산 환경에 관심을 갖고 데이터베이스,

       스프레드쉬트, 윈도우 OLE등 다양한 소프트웨어를 다루는 데 더 관심을 가짐.

   ∘ 이러한 공통의 인터페이스를 준수하는 컴포넌트 기반의 소프트웨어는 많은 장점을 제공함.

   ∘ 이러한 공통의 인터페이스를 미들웨어(Middleware)라고 함

**API : 자바 API (Application Programming Interface)**

   ∘ 자바개발 환경에서 제공되는 거대한 패키지.

   ∘ 자바언어는 작고 단순한 대신 많은 기능들을 제공하는 라이브러리를 API로 가지고 있음.

**예외처리의 필요성과 목적**

   ∘ 자바에서 프로그램 실행 중 예외가 발생하면 발생된 시점에서 프로그램이 바로 종료가 된다.

       때에 따라서는 예외가 발생 했을 때 프로그램을 종료시키는 것이 바른 판단일 수도 있다.

     하지만 가벼운 예외나 예상을 한 예외라면 프로그램 종료가 조금은 가혹(?)하다고 느껴진다.

   ∘ 그래서 ‘예외처리’라는 수단(mechanism)이 제안되었고

     예외 처리를 통해 우선 프로그램의 비정상적인 종료를 막고 발생한 예외에 대한 처리로

     정상적인 프로그램을 계속 진행할 수 있도록 하는 것이 예외처리의 필요성이라 할 수 있다.

   ∘ 예외의 발생으로 실행 중인 프로그램의 갑작스런 비정상 종료를 막고,

     정상적인 실행 상태를 유지 할 수 있도록 하는 것.

**Abstract 클래스**

   ∘ 추상 method를 하나 이상 가지는 클래스.

   ∘ new 객체를 생성할 수 없음.

   ∘ 상속받은 클래스는 추상 method를 구현해야만 인스턴스를 생성할 수 있음.

    (추상클래스 끼리의 상속은 method 재정의가 필요없음. 사용할 때 일반 클래스에서 재정의)

   ∘ 추상 method란 함수의 리턴타입과 파라메터만 있고 정의 부분이 없는 함수를 말함.

**추상화 (abstraction)**

   ∘ 구체적인 개념으로부터 공통된 부분들만 추려내어 일반화 할 수 있도록 하는 것을 의미.

     일반적으로 사용할 수 있는 단계가 아닌 아직 미완성적 개념.

**Interface**

   ∘ 상수 또는 추상 method만 정의 할 수 있음.

   ∘ 구현된  method는 포함 할 수가 없음.

   ∘ 모든 변수는 static(정태적)이고 final임.

   ∘ 자식클래스들이 공통된 method를 포함하도록 하는 기능만 함.

   ∘ 다중 상속이 가능함.

**추상클래스와 인터페이스의 공통점**

  ∘ 객체를 발생시킬 수 없음.

  ∘ 상속해서 하위 class를 통하여 객체를 발생시킴. (method를 재정의 해야 함.)

**Camel 규칙 (변수이름 정하는 규칙)**

  ① 클래스: 첫 문자는 대문자로 시작.   둘 이상 단어 묶어 구성시 새로 시작 단어 대문자.

  ② 메소드: 첫 문자는 소문자로 시작.   둘 이상 단어 묶어 구성시 새로 시작 단어 대문자.

  ③ 상  수: 모든 문자를 대문자로 구성. 둘 이상 단어 연결시는 \_(언더바)를 사용하여 연결 함.

**JVM의 구성**

  ① 클래스영역 : 클래스 코드를 저장하는 영역.

  ② 자바스택   : method를 호출할 때 관련 정보를 저장하는 영역.

  ③ 힙(Heap)  : new 라는 키워드를 통해 객체가 생성될 때 할당받는 영역.

  ④ 네이티브 method 스택

**Collection에서 데이터를 저장 하는 3가지와 그 특징**

  ① Set  : 순서가 없고, 동일한 데이터 허용 안 됨.

  ② List  : 배열과 같은 구조지만, 가변적 길이를 가지고 있음. (크기가 지정 되어있지 않음.)

  ③ Map : key 값과 value값의 형식으로 저장되면 key값은 절대 중복이 안 됨.

**스트림**

  ① 데이터를 목적지로 입 · 출력하기 위한 방법. 스트림에 데이터를 쓸 수 있고 읽을 수 있음.

     연결하고자 하는 Device에 따라 다양한 스트림이 존재.

  ② 특징

   • 스트림은 FIFO 구조. 읽기, 쓰기가 동시에 되지 않음.

   • 읽기, 쓰기가 필요하다면 읽는 스트림과 쓰는 스트림을 각각 하나씩 열어 사용해야함.

   • 데이터가 처리되기 전까지 스트림에 사용되는 스레드는

     데이터가 모두 전송되기 전까지 블락킹 상태에 빠짐.

**Statement 와 PreparedStatement의 차이**

  ① Statement : 정적 쿼리 시 사용. 매번 파싱과정을 거쳐야 함.(부하가 생길 수 있음)

                 SQL문 전체를 명확히 알수 가 있어서 디버깅이 쉬움.

  ② Prepared Statement : 동적 쿼리 시 사용.

                 한번 파싱하면 그 동일 SQL문장을 곧 바로 파싱 과정 없이 Execution 가능.

                 (반복적인 다량의 SQL 수행 시 성능 상 이득이 있음)

                 오류발생 시, 변수에 입력되는 값을 알 수 없어서 디버깅이 어려움.

**서블릿의 실행 과정**

   ∘ 서버가 클라이언트의 연결 요청받음. 컨테이너는 연결 요청 정보를 담고 있는 Request 객체

     와 연결 응답 정보를 담고 있는 Response 객체를 생성.

   ∘ 접수된 Url을 분석 후 해당 서블릿 객체를 생성하고, 사용자의 요청을 처리하기 위해 스레드

     를 생성 후 service() method에 인자 값을 담아 호출.

   ∘ service() method는 Request 객체를 참고하여 어떤 연결 요청 방식으로 들어왔는지 파악함.

     (GET 방식인지 POST 방식인지)

   ∘ 들어온 요청 방식에 따라 get 방식은 doGet() method를, post 방식은 doPost() method를

  호출하여 처리 함.

   ∘ service() method의 인자 값으로 넘겨받은 response 객체를 이용하여 클라이언트에게 결과

      를 보여줌. 사용자 요청을 처리하기 위해 생성한 스레드를 소멸.

**서블릿에서 데이터를 처리 하는 방식은?**

  ① GET  : 서버에 있는 정보를 가져오기 위해 설계 됨. 240바이트까지 전달 할 수 있음.

             POST 방식에 비해 속도가 빠름. 검색엔진에서 검색단어 전송에 많이 이용함.

             URL노출로 보안성이 요구되는 경우엔 사용 할 수 없음.

1. POST : 서버로 정보를 올리기 위해 설계 됨. URL에 파라미터가 표시 되지 않음.

             내부적으로 데이터가 이동함. GET 방식에 비해 속도 느림. 데이터 크기 제한 없음.

**JSP 에서 페이지 이동 방법에 대해 설명 하시오.**

  ① Forward 방식 • Url이 바뀌지 않음. 요청객체와 응답객체가 유지됨.

                  • 속도가 빠르며 요청객체에 소속 되어 있음.

         처리 구조 • 요청이 들어오면 Servlet이 받음.

                  • 요청에 알맞은 페이지를 찾음. 알맞은 페이지가 있다면 응답.

                  • 알맞은 페이지가 없다면, Forwarding 방식으로 알맞은 페이지로 넘기는데,

                    요청객체와 응답 객체를 포함해 넘김.

                  • Url이 바뀌지 않은 상태로 응답 페이지를 통해 응답.

  ② Redirect 방식 • Url이 바뀜. 요청객체와 응답객체가 유지 되지 않음.

                  • 속도가 느리며, 응답객체에 소속 되어 있음.

         처리 구조 • 요청이 들어오면 Servlet이 받음.

                  • 요청에 알맞은 페이지를 찾음. 알맞은 페이지가 있다면 응답.

                  • 알맞은 페이지가 없다면, 알맞은 페이지로 다시 요청하게끔 응답 보냄.

                  • 클라이언트는 응답을 받고, 다시 그 요청의 맞는 Url로 요청함.

**자바 빈즈**

   ∘ 자바에서 사용하는 컴포넌트.(부품)

     Bean은 자바에서 컴포넌트를 이용하기 위해 만들어 놓은 기술.

   ∘ 컴포넌트를 사용하면 좋은 이유는 필요 할 때마다 가져다가 사용 할 수 있다는 점인데,

     JSP에서 자바의 컴포넌트를 이용해서 프로그래밍 하는 것을 빈즈 프로래밍이라 함.

**Connection Pool 개념**

   ∘ 미리 생성해 놓은 커넥션을 할당하고 반납함으로써 커넥션 생성 시간을 줄인다는 개념.

   ∘ 미리 생성할 때 너무 조금 생성하면 오히려 타임 오버헤드가 늘어나고,

     반대로 너무 많이 생성하면 메모리 오버헤드가 늘어남.

**Model 1 방식 과 Model 2 방식의 특징과 차이점**

  ① Model 1 : 디자인 코드와 자바 코드(비즈니스로직)를 구분하지 않고, 하나의 JSP일 내에

               기술해서 웹 프로그램을 제작 하는 방식.

         특징 • 개발하기가 쉽고, 배우기가 쉬움.

              • 디자인코드와 비즈니스로직의 구분이 명확하지 않아, 복잡도가 높음.

              • 수정 시 디자이너와 개발자의 협업이 필요.

              • 비즈니스로직의 재 사용성이 어려우며, 유지 보수가 힘듬.

  ② Model 2 : 웹 어플리케이션을 개발할 때, MVC패턴을 적용하여, 웹 어플리케이션의 개발이

               가능하도록 구현한 것.

         특징 • 초기 설계에 많은 시간이 소요.

              • 디자인코드와 비즈니스로직이 분리되며, 비즈니스로직의 재사용성이 높아짐.

              • 비즈니스로직 계층의 확장성이 용이하며, 유지보수가 편하다.

**Singleton의 대해 설명 하시오.**

   ∘ 발전된 형태의 전역 변수.

   ∘ 프로그램 상에서 두 번째 인스턴스를 만들 수 없게 하는 기능.

   ∘ 만들어진 클래스의 객체를 단 하나만 사용하며 어디서든 그 객체를 사용가능토록 만들어 줌.

**Cookie 와 Session 의 차이**

  ① Cookie  : 클라이언트에 정보를 저장하기 때문에 서버에 부하가 없음.

               사라지는 때(시간)를 지정 할 수가 있음.

  ② Session : 서버에 정보를 저장하기 때문에 서버에 부하가 많이 생김.

               해당 클라이언트와 일정시간동안 작용이 없으면 서버메모리에서 해제 됨.

               직접적으로 해제 시킬 수 있는 method가 있음.

**DB정규화의 목적**

   ∘ 자료정규화 작업의 가장 큰 목적은 자료저장의 중복성 배제이다.

   ∘ 정규화이론에서는 릴레이션 형태가 여러 단계로 구분되며, 가장 기본적인 정규화조건도 만족

     하지 못하는 릴레이션을 비정규형, 만족하는 릴레이션을 제1정규형이라고 부름.

   ∘ 조건이 점점 엄격해짐에 따라 제2, 제3, 제4, 제5정규형으로 구분됨.

     높은 단계의 정규형으로 나아갈수록 데이타의 본질적 의미가 릴레이션 구조에 보다 정확히

     반영되고, 데이타 중복을 줄이고, 데이타 변경시 발생하는 문제점을 방지하고,

     궁극적으로 데이타 무결성(data integrity)을 제고할 수 있다고 가정함.

**DDL (Data Definition Language)**

     테이블 및 객체의 구조 생성 그리고 삭제, 변경과 관련된 명령어.

   ∘ CRATE : 새로운 테이블 생성.

   ∘ DROP : 기존 테이블의 구조 및 모든 행을 삭제.

   ∘ ALTER : 기존 테이블 변경, 필드를 추가 또는 수정 할 수 있지만 삭제 불가.

   ∘ TRUNCATE : 테이블의 구조를 남기고 모든 데이터 삭제.

   ∘ RENAME : 오브젝트의 이름을 변경.

   ∘ COMMENT : 테이블이나 컬럼에 주석문.

**DML (Data Manipulation Language)**

     데이터 조작과 관계된 명령어.

     데이터의 검색 및 출력, 정렬과 조인에 관계 됨.

   ∘ SELECT : 데이터의 조회.

   ∘ INSERT : 데이터의 입력.

   ∘ UPDATE : 데이터의 수정.

   ∘ DELETE : 데이터의 삭제.

   ∘ WHERE : 데이터의 조건.

   ∘ COMMIT : 수행한 데이터 작업들의 영구 저장.

   ∘ ROLLBACK : 수행한 데이터 작업들의 원상복구.

**DCL (Data Control Language)**

     계정의 권한과 관계된 명령어.

     DB 사용자의 권한 정의.

   ∘ COMMIT : 데이터베이스 조작을 영구적으로 반영.

   ∘ ROLLBACK : 마지막 COMMIT 상태로 복구.

   ∘ GRANT : 권한 부여.

   ∘ REVOKE : 권한 취소.