**자바 단골 질문**

**Q1. 자바의 특징**

1. 객체 지향 프로그래밍(oop) 언어입니다.
2. JVM 위에서 실행되기 때문에, 플랫폼에 의존하지 않고 실행이 가능합니다.
3. 멀티 스레딩을 지원합니다.
4. 바이트코드로 변환되어 실행되기 때문에 고성능입니다.

**Q2. 객체지향 언어의 장점과 정의**

1. 코드의 재사용성이 높아서 (라이브러리만 임포트하면 필요한 기능만 뽑아서 사용가능)
2. 생산성이 높아지고 버그 발생률이 낮습니다.
3. 객체 단의로 코드를 작성하므로 협업에 최적화되있고
4. 유지보수에 용이합니다.

**객체지향 언어의특징**

1. **상속**- 이미 정의된 상위클래스의 모든 속성과 연산을 하위클래스가 물려받는 것입니다..
2. **캡슐화**- 내부의 코드를 캡슐처럼 보호하는 것, 변수를 private로 명시하고 setter,getter로만 접근가능하게 한다. 캡슐화된 객체들은 재사용이 용이합니다.
3. **다형성**- 하나의 객체를 여러가지 타입으로 해석할 수 있는 기술입니다.
4. **추상화**- 불필요한 부분을 생략하고 객체의 속성 중 가장 중요한 것에만 중점을 두어 개략화하는 것, 즉 모델화 하는 것입니다.

**Q3. 객체란?**

1. 물리적으로 존재하거나 추상적으로 생각할 수 있는것중에서 자신과 다른 것을 식별가능한 것입니다.
2. 세상 모든 것이 객체라고 생각하면 됩니다.

**Q4. 생성자란?**

1. 생성자의 목적은 객체(필드)초기화에 있습니다.
2. 클래스와 같은 이름의 메소드입니다.
3. 오버로드가 가능합니다.
4. 명시적으로 생성자를 만들지 않아도 default로 만들어집니다.

**Q5. 지역변수와 인스턴스변수(전역변수)**

1. **지역변수** - 메소드 안에서 정의되는 변수
2. **인스턴스(전역)변수** – 클래스에서 정의되는 변수

**Q6. 오버라이딩과 오버로딩의 차이**

1. **오버로딩(Overloading)** - 다양한 타입과 개수의 파라미터를 가진 메소드를 여러 개 생성하여 다양한 유형의 호출에 응답하는 기술입니다.

메서드이름(동일), 매개변수타입(다름), 리턴타입(상관없음)

1. **오버라이딩(Overriding**) – 자식 클래스가 부모클래스의 메서드를 받아 재정의하는 것입니다. 메서드이름,매개변수타입,리턴타입이 전부 동일해야 됩니다.

**Q7. 접근제한자에 대해 설명하시오.**

1. Private – 자신(this) 클래스에서 사용가능합니다.
2. Public – 모든 클래스에서 사용가능합니다.
3. Default – 같은 패키지안에 있는 클래스에만 사용가능합니다.
4. Protected - 같은 패키지와 상속받은클래스에서 사용가능합니다.

**Q8. 추상클래스와 인터페이스의 차이**

|  |  |
| --- | --- |
| **추상클래스** | **인터페이스** |
| Default 생성자를 가진다 (서브클래스가 인스턴스화될 때 호출됨) | 생성자를 가지지 않는다 |
| 추상 메소드, 비 추상(일반) 메소드를 포함할 수 있다 | 추상 메소드만 선언할 수 있다 |
| 추상 클래스를 상속한 클래스는 꼭 모든 메소드를 구현할 필요는 없으며, 추상 메소드만 서브 클래스 안에서 구현되면 된다. | 인터페이스를 구현하는 클래스는 인터페이스에서 선언된 모든 메소드를 다 구현해야 한다. |
| 객체 생성 불가하므로 extend (상속)로 구현 | 객체 생성 불가하므로 implement 로 구현 |

**Q9. 배열과 리스트(Array and List)**

|  |  |
| --- | --- |
| **Array** | **List** |
| 선언시 배열 크기 지정이 필요하다. 예) String list = new String[7] | 사이즈 지정을 하지 않아도 동적으로 변화한다. |
| 객체를 넣을 때 인덱스 번호를 정의해야 한다. | 그냥 넣어도 된다. 예) list.add('book') |
| 다차원 저장이 가능하다 (Multidimensional) | 다차원 저장이 불가능하다 |
| 선언시 메모리가 할당된다 |  |
| 기본 데이터 타입(Primitive Type), 오브젝트 할당 가능하다 | 오브젝트만 할당 가능하다 (\* 기본 데이터 타입도 할당 가능한 것처럼 보이지만 실제로는 JVM에 의해 오브젝트와 되어 할당되는 것이다. 따라서 기본 데이터 타입만 들어가는 자료구조가 요구될 때에는 Array가 Arraylist보다 더 효율적이다.) |
| 접근이 빠르다 (인덱스 번호를 이용해 Random access를 하기 때문) | 추가, 삭제가 빠르다 (포인터로 Sequential access를 하기 때문) |

**Q10. 스트링, 스트링버퍼, 스트링 빌터의 차이**

1. **스트링**은 Heap영역에 새로운 메모리를 사용/ 메모리 주소만 바꿔줍니다.
2. **스트링 버퍼와 빌더**는 Stack영역에 값을 저장하므로 값이 바뀝니다**.**
3. **스트링버퍼**는 멀티스레드환경에 안전, **빌더**는 버퍼보다 빠릅니다..

**Q11. 해쉬맵과 해쉬테이블**

|  |  |
| --- | --- |
| HashMap | HashTable |
| 메소드 동기화를 지원하지 않는다 | 메소드 동기화를 지원한다 |
| Thread-safe 가 아니다 | Thread-safe 이다 |
| Iterator를 사용한다 | Enumerator를 사용한다 |
| 하나의 Null key와 여러개의 Null Value를 허용한다. | Null을 허용하지 않는다 |
| 성능이 비교적 빠르다 | 성능이 비교적 느리다 |

**Q12. 해쉬셋과 트리셋**

|  |  |
| --- | --- |
| HashSet | TreeSet |
| 넣은 순서대로 저장 순서를 유지하지 않는다 | 넣은 순서대로 저장 순서를 유지하지 않는다 오름차순 정렬된다 |
| Null 객체를 저장할 수 있다 | Null 객체를 저장할 수 없다 |
| 검색 성능이 비교적 빠르다: O(1) | 검색 성능이 비교적 느리다: O(log n) |
| 동일 타입 오브젝트 저장 허용 | 동일한 타입의 오브젝트 저장 불가 (ClassCastException 발생) |
| 비교 오퍼레이션시 equals 메소드 사용 | 비교 오퍼레이션시 comparedTo 메소드 사용 |

**중복, 순서에 관계없는 데이터 집합이 필요하다면 HashSet**

**정렬이 필요한 데이터 집합이 필요하다면 TreeSet**

**순서를 보장하는 데이터 집합이 필요하다면 HashLinkedSet**

**Q13. CollectionFramework에 대해 설명하시오**

1. 객체를 저장하도록 설계된 프레임워크입니다.
2. 컬렉션은 검색,조작,삽입,삭제 등의 기능을 수행할 수 있는 객체집합입니다.
3. 분류

|  |  |  |
| --- | --- | --- |
| 리스트 LIST | 세트 SET | 맵 MAP |
| * 삽입 순서가 의미를 가진다 * 중복을 허용한다 * ITERATION이 빠르다 | * 고유한 값을 가지는 자료구조로, 중복을 허용하지 않는다. * 인덱스를 사용하지 않으므로 저장순서가 보장되지 않음 (검색시 iterator메소드로 데이터를 불러와야함) | * 분별자(KEY)가 고유한 값을 가지는 자료구조로, KEY는 중복되어서는 안 된다. |

**Q14. 예외**

1. 실행 중에 발생할 수 있는, 개발자가 구현한 로직에서 발생하는 문제를 말합니다.
2. 런타임 중에 문제가 발생하면 메소드는 예외를 던져 문제 발생을 알립니다.
3. 예외 처리를 하면 실행을 멈추지 않고 진행 할 수 있으면 문제확인도 가능합니다.(이점)

**Q15. 예외의 종류**

1. 체크예외 - 컴파일 단계에서 확인되는 예외들, 주로 JVM 외부와 통신 시에 발생하는 예외들
2. 언체크예외 - 실행 단계에서 확인 되는 예외들

**Q16. 예외 처리 방법**

1. Try/Catch 블록
2. Throw 키워드 선언
3. catch에서 다른 예외를 던지는 것

**Q17. 파이널 키워드(final)**

|  |  |
| --- | --- |
| 타입 | 특성 |
| 변수 | 변수가 Final로 일단 선언되면, 변수의 값은 덮어 씌워질 수 없다. |
| 메소드 | Final 메소드는 오버라이드 될 수 없다. |
| 클래스 | 클래스가 Final로 선언되면, 상속 될 수 없다. |
| 생성자 | 생성자는 Final이 될 수 없다. |

**Q18. Framework란?**

* 1. Software를 만들 때 뼈대가 되는 부분을 미리 구현해준 것
  2. 장점: 뼈대가 있어서 생산성이 좋아짐, 품질보장
  3. 단점: 익숙해져야됌, 내부 커스터마이징이 힘듦

**Q20. 자바의 메모리 영역**

0. JVM은 OS로부터 메모리( Runtime Data Area )를 할당받는다.

1. **Class Area or Method Area or Static Area**

- 모두 같은 영역을 칭함.

- 전역변수와 정적 멤버변수(static 변수)는 이 영역에 저장된다. ★

- static area는 프로그램의 시작부터 종료가 될 때까지 메모리에 남는다.

- 이는, 전역변수가 프로그램 종료될 때까지 어디서든 사용이 가능한 이유.

2. **Stack Area**

- 지역변수, 인자값, 리턴값이 저장이 되는 영역, 즉 메소드 안에서 사용되어지는 값들이 저장 ★

- 정수타입(byte, char, short, int), 실수타입(double, float), 논리타입(boolean)

- 해당 메소드가 호출될 때 메모리에 할당되고, 종료되면 메모리가 해제.

- 각 스레드당 1개의 스택을 가진다.

3. **Heap Area**

- 인스턴스 생성 : 클래스명 변수명 = new 클래스명();

- 참조형 데이터 타입을 갖는 객체(인스턴스), 배열 등은 이 영역에 저장.

- 즉, new 명령으로 만드는 메모리는 모조리 힙 영역에 보관.

- 이 메모리는 가비지 컬렉터가 회수하기 전까지 유지된다.

**Q21. 제네릭(Generic)이란?**

* 1. 다양한 타입의 객체에 재사용을 높일 수 있는 기법으로 클래스에서 사용할 타입을 외부에서 설정하는 것을 말합니다.
  2. 제네릭이 없다면 빈번한 타입 변환이 발생 할 수 있으며 이는 프로그램의 성능을 저하시킵니다.

**Q22. JVM,JRE,JDK에 대해 설명하시오.**

1. JVM – 자바 가상 머신의 약자입니다. 어느 플랫폼에서나 동작하게 해줍니다.
2. JRE – 자바 실행환경의 약자입니다. JVM+핵심라이브러리로 구성되있습니다.
3. JDK – 자바 개발도구의 약자 입니다. JRE+개발툴로 구성되있습니다.

**Spring 단골 질문**

**Q1. 웹 서버와 WAS의 차이**

* 1. **웹서버** – 정적 컨텐츠를 제공하는 서버/ Apache (단순 HTML문서, CSS 등)
  2. **WAS** – 데이터베이스의 조회나 다양한 로직 처리가 필요한 동적 컨텐츠를 제공/ Tomcat(최근엔 tomcat이 Apache의 기능까지 포함했음)

**Q2. Sevlet이란?**

1. 클라이언트의 요청을 처리하고, 그 결과를 반환하는 Servlet클래스의 구현 규칙을 지킨 자바 웹 프로그래밍 기술
2. 웹에서 동작해야하는 로직을 담고있는 클래스(코드)
3. Ex) 사용자가 아이디와 비번을 입력후 로그인 버튼을 누름 – 서버는 정보를 확인후 다음 페이지를 띄워줌

**Q3. JSP와 javascript의 차이**

1. **JSP(Java Server Page)** – 서버에서 실행, 그 결과를 웹브라우저에 띄워줌
2. **Javascript** – 웹브라우저에서 실행

**Q4. Scope에 대해 설명하시오**

1. 빈이 관리되는 범위
2. Page>request>session>application순으로 scope가 커짐
3. Scope가 커질수록, 어떤 객체에서든 접근할 수 있음

**Q5. Maven와 gradle의 차이는?**

1. Maven – 빌드 및 라이브러리 의존성 관리 등을 해주는 도구, 상속 구조
2. Gradle - JVM기반의 빌드도구로 기존의 Ant와 Maven을 보완, 구성 주입 방식

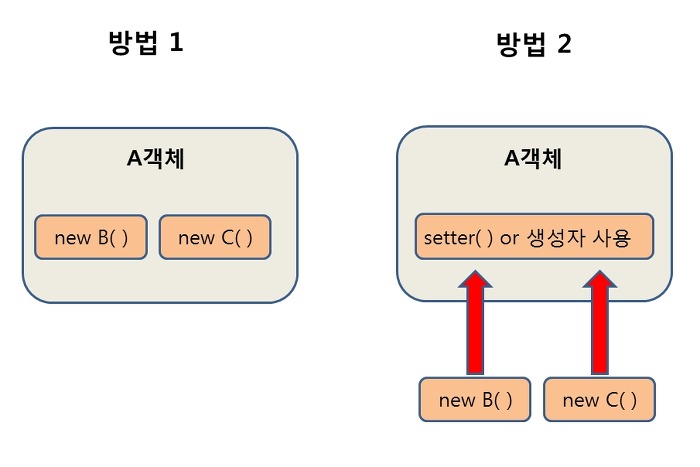
**Q6. JDBC 란?**

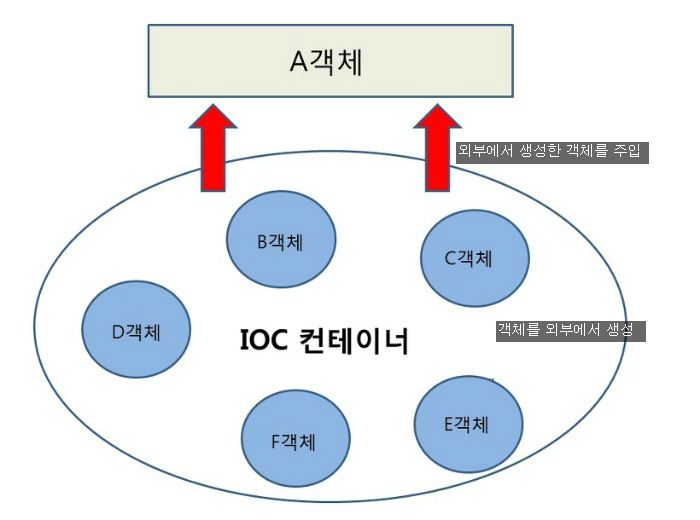
1. Java 기반의 웹 어플리케이션과 외부 DB를 연동시켜주는 라이브러리
2. java코드로 SQL쿼리문을 작성하여 실행 가능

**Q7. 컨테이너, IoC(Inversion of Control), DI(Dependency Injection),AOP?**

1. **컨테이너** –빈의 생성과 관계 사용, 생명 주기등을 관장, Bean을 담는 통이라 생각하셈
2. **IoC** – 제어의 역전, 메소드나 객체의 호출작업을 개발자가 결정하는 것이 아니라, 외부에서 결정되는 것, 객체의 의존성을 역전시켜 객체 간의 결합도를 줄이고 유연한 코드를 작성/ 가독성 및 코드중복, 유지보수 가능
3. **DI** – 의존성 주입, 객체를 직접 생성하는게 아니라 외부에서 생선한 후 주입 시켜주는 방식
4. **AOP** – 관점 지향 프로그램, 어떤 로직을 기준으로 핵심과 부가적인 관점을 나누는 것

**그림 자료**





**Q8. Bean이란?**

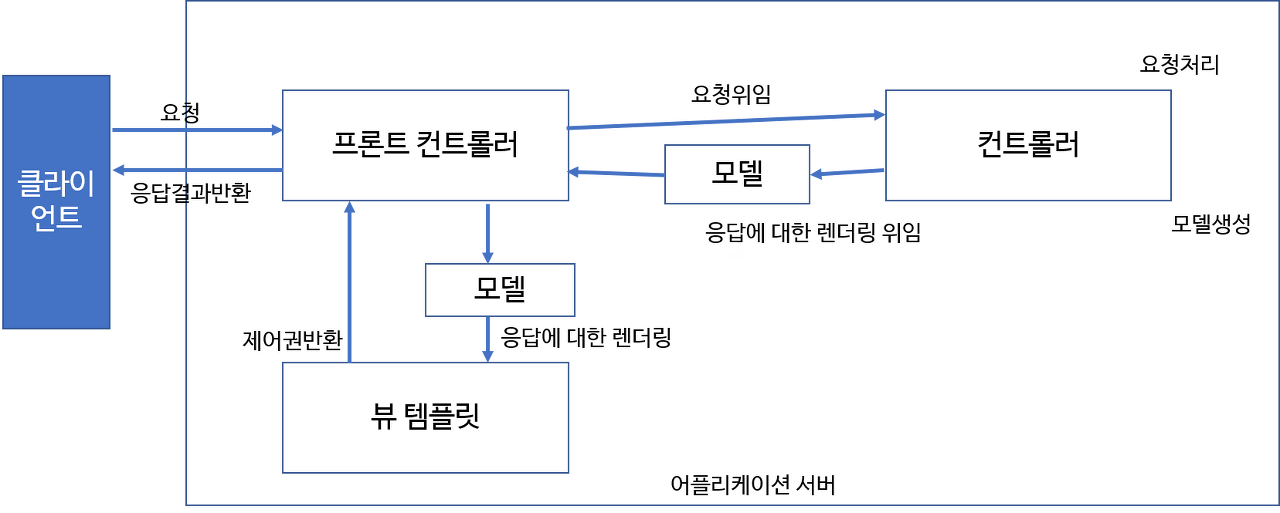
1. Spring IoC컨테이너가 관리하는 자바객체
2. ApplicationContext.getBean()으로 얻을질 수 있는 객체

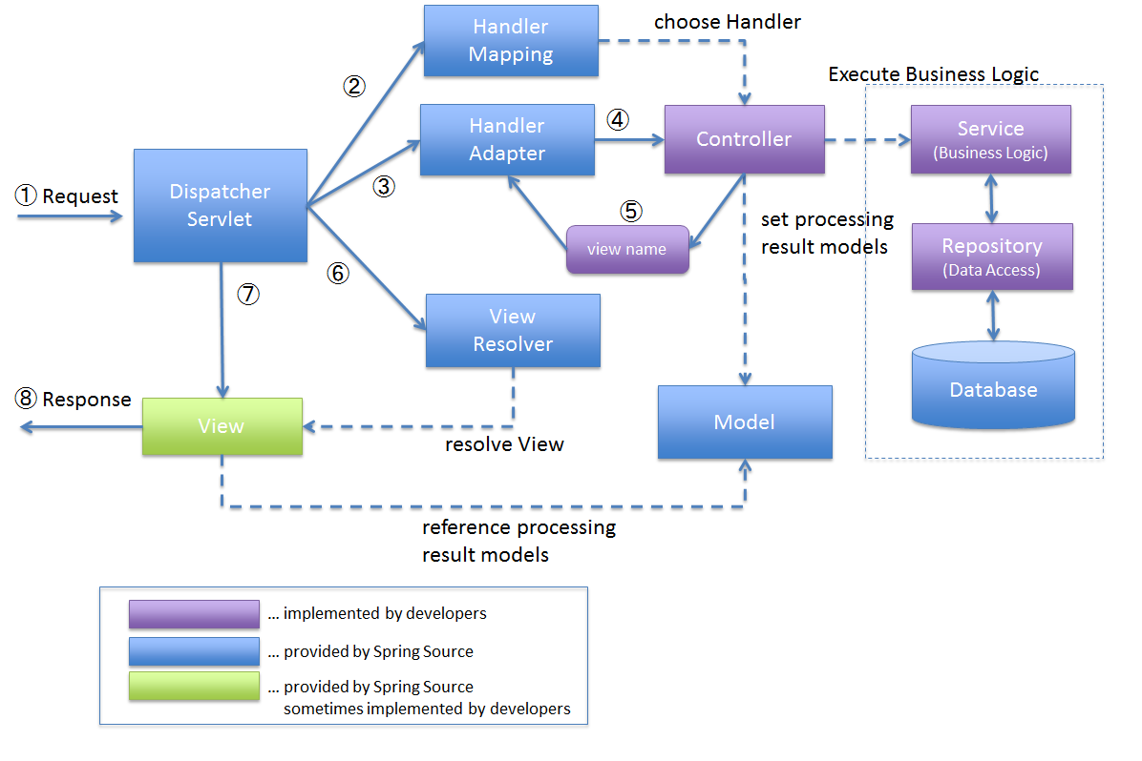
**Q9. DTO,DAO,ConnectionPool,DataSource?**

1. DTO - 계층간 데이터 교환을 위한 자바빈즈, 데이터만을 담고 있는 객체
2. DAO – 데이터를 조회하는 기능을 하는 객체, 보통 데이터베이스에 접근하는 객체임
3. ConnectionPool – DB와 항시 연결되어있는 객체들, DataSource에 의해 제어
4. DataSource – ConnectionPool을 관리하는 객체

**Q10. MVC란?**

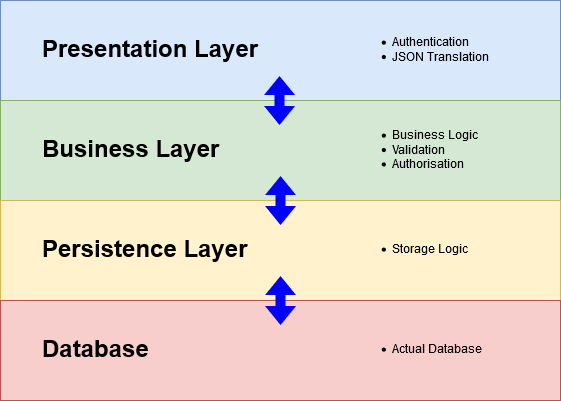
1. Model-View-Controller의 약자
2. **Model** 은 ‘데이터’ 디자인을 담당
3. **View** 는 ‘실제로 렌더링되어 보이는 페이지’를 담당
4. **Controller**는 사용자의 요청을 받고, 응답을 주는 로직을 담당





**Q11. 레이어드(Layered) 아키텍처?**

1. 여러 모듈들을 일종의 레이어 단위로 나누는 설계 방식
2. Presentation Layer – 화면 조작 또는 사용자 입력을 처리하는 레이어
3. Service Layer – 비즈니스와 관련된 도메인 로직을 처리하는 레이어
4. Repository Layer – 도메인에서 필요로 하는 데이터를 조작하기 위한 레이어



**Q12. Cookie와 Session의 차이점?**

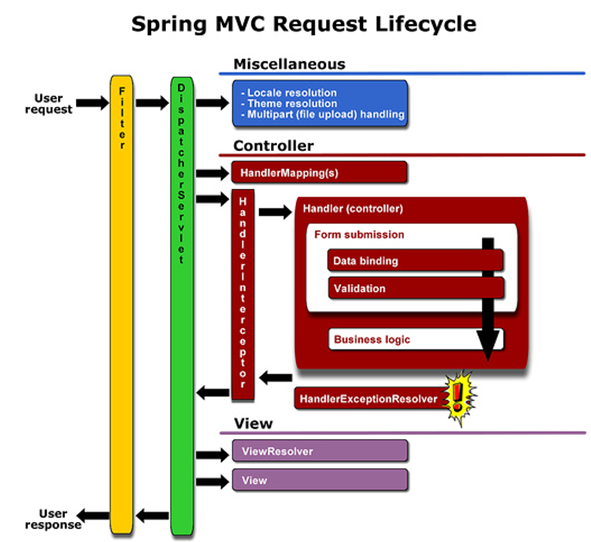
1. **쿠키** – 사용자 컴퓨터에 저장, 저장된 정보를 다른사람이나 시스템이 볼 수 있음, 유효기간이 지나면 사라짐
2. **세션** – 서버에 저장, 서버가 종료되거나 유효기간이 지나면 사라짐

**Q13. Request 전송 방식의 종류는?**

1. **GET방식** - 주소에 데이터를 추가해서 전달하는 방식이다, POST방식에 비해 전송속도가 빠르다.
2. **POST방식** - 데이터를 별도로 첨부하여 전달하는 방식이다, 데이터는 header 뒤에 숨어 전송된다.  길이의 제한이 없고, GET방식보다 보안성이 높다.
3. **그 외** PUT, DELETE, TRACE 등이 있음

**Q14. 인터셉터(Interceptor)란?**

1. 컨트롤러에 들어오는 요청 HttpRequest와 응답하는 HttpResponse를 가로채는 역할
2. Ex)관리자 페이지에 접근하기 전에 관리자 인증을 하는 용도로 활용가능.
3. **Filter**는 DispatcherServlet이 실행되기 전, **인터셉터**는 후에 실행
4. **Filter**는 web.xml, **인터셉터**는 spring-servlet.xml에 설정
5. **Filter**는 web.xml에서 설정을 하면 구현이 가능하지만, **Interceptor**는 설정은 물론 메서드 구현이 필요합니다.



**Q15. 로깅이란?**

1. 로그를 남기는 작업으로, 프로그램 실행 동작을 일련의 기록으로 남겨놓는 일이다.
2. 관련라이브러리 - java.util.logging, Apache Commons logging, Log4j, Logback

**Q16. Spring과 Spring boot의 차이점**

1. Embed Tomcat을 사용하기 때문에, (Spring Boot 내부에 Tomcat이 포함되어있다.) 따로 Tomcat을 설치하거나 매번 버전을 관리해 주어야 하는 수고로움을 덜어준다.
2. starter을 통한 dependency 자동화 :  
   과거 Spring framework에서는 각각의 dependency들의 호환되는 버전을 일일이 맞추어 주어야 했고, 때문에 하나의 버전을 올리고자 하면 다른 dependeny에 까지 영향을 미쳐 version관리에 어려움이 많았다. 하지만, 이제 starter가 대부분의 dependency를 관리해주기 때문에 이러한 걱정을 많이 덜게 되었다.
3. XML설정을 하지 않아도 된다.
4. jar file을 이용해 자바 옵션만으로 손쉽게 배포가 가능하다.  
   Spring Actuaor를 이용한 애플리케이션의 모니터링과 관리를 제공한다.

**Q17. RDS는 무엇을 사용했는가? / IDE는 무엇을 사용했는가?**

1. Mysql 5, eclipse <- 사용한거 알아서 말하면됌

**Q18. 어노테이션에 대해 설명하시오./사용한 어노테이션 설명하시오**

1. @을 이용한 주석을 달아 특별한 의미를 부여한 것 (모든 요소에 사용가능)
2. **@Controller**

Spring의 Controller를 의미한다. Spring MVC에서 Controller클래스에 쓰인다.

1. **@RestController**

Spring에서 Controller 중 View로 응답하지 않는, 컨트롤러를 의미한다.

이 어노테이션이 적혀있는 컨트롤러의 메서드는 HttpResponse로 바로 응답이 가능하다. @ResponseBody 역할을 자동적으로 해주는 어노테션이다.

1. **@RequestMapping**

Spring의 컨트롤러 혹은 그 메서드의 URI를 정의하는데 쓰인다.

요청을 받는 형식인 GET, POST, PATCH, PUT, DELETE 를 정의하기도 한다.

요청 받는 형식을 정의하지 않는다면, 자동적으로 GET으로 설정된다.

1. **@PathVariable**

URI에서 / 다음으로 넘어오는 값들을 파싱하는 어노테이션이다.

# @RequestBody

POST나 PUT, PATCH로 요청을 받을때에, 요청에서 넘어온 body 값들을 자바 타입으로 파싱해준다.

# @Service

Service Class 에서 쓰인다. 비즈니스 로직을 수행하는 클래스라는 것을 나타내는 용도이다.

# @Repository

DAO class 에서 쓰인다. 데이터베이스에 접근하는 메서드를 가지고 있는 클래스에서 쓰인다고 볼 수 있다.

**Q19. JSTL이란?**

1. JSP 페이지를 작성할 때 유용하게 사용할 수 있는 여러 가지 커스텀 액션과 함수가 포함되어 있는 라이브러리이다. (간단한 프로그램 로직의 구사, 다른 jsp페이지 호출 등)

**DB 단골 질문**

**Q1. index에 대해 설명하시오.**

1. DB 검색속도를 빠르게 도와주는 역할.
2. B-Tree방식 구조를 사용하여 0과 1로만 탐색하기 때문에 속도가 빠름
3. Select문의 join,where에서 좋은 성능을 발휘함 <-> insert,update,delete문에서는 떨어짐

**Q2. JOIN?**

1. 복수의 테이블을 결합, 하나의 테이블인 것처럼 결과를 출력 (교집합을 생각하면 편함)
2. **INNER JOIN**: 두개의 테이블에 모두 존재하는 데이터행을 가져옴
3. **OUTER JOIN**: 매챙되는 행이 없어도 결과를 가져오고 매칭되는 행이 없는 경우 NULL로 표시한다.

**Q3. 트랜잭션이란?**

1. 데이터베이스의 상태를 변화시키기 위하여, 논리적인 기능을 수행하는 하나의 작업단위 또는 한꺼번에 모두 수행되어야 하는 연산들 (all ro nothing)
2. 데이터베이스의 완전성을 보장하기 위한 것
3. 원자성, 독립성, 일관성, 지속성을 가짐
4. **Commit**: 정상적으로 처리되어서 트랜잭션이 반영된 상태
5. **Rollback**: 취소하거나 비정상적으로 처리 됬을 때 이전상태로 돌아가는 것

**Q4. 관계형 DB와 비관계형DB의 차이는?**

1. **관계형**: 테이블형태로 key, value관계를 가지고 있는 데이터베이스
2. **비관계형**: 테이블형식의 스키마를 사용하지 않음, nosql (mongoDB)
3. (비관계형의 장점: 가벼적으로 데이터 구조를 바꿀 수 있음)

**Q5. PK/ FK?**

1. **PK(Primary Key)** : 테이블을 구분할 수 있는 고유한 컬럼, 프라이머리키(부모키)
2. **FK(Foreign Key)** : 해당 컬럼의 의미를 파악하기 어려울 때 부모테이블의 pk를 참조하는 컬럼(외래키)

**Q7. http와 https의 차이**

1. **http**: 웹브라우저가 웹 서버에 요청하는 프로토콜(규칙,규격)
2. **https**: 모든 통신을 암호화 하기 때문에 http보다 보안이 좋다.

**Q8. Mapper?**

1. **parameterType** = 입력될 데이터
2. **resultType** = 쿼리 후 결과를 위한 데이터

**Q9. 서브쿼리란?**

1. SQL문을 실행하는데 필요한 데이터를 추가로 조회하기 위해 SQL문 내부에서 사용하는 SELECT문을 의미
2. Ex) SELECT name FROM member WHERE sal ( SELECT sal FROM emp WHERE name=’ 홍’ );