1. **스프링의 특성**
   1. 경량의 프레임워크이다.
   2. MVC 프레임워크를 제공한다.
   3. DI(Dependency Injection)를 지원한다.
   4. AOP(Aspect Oriented Programming)를 지원한다.
   5. POJO(Plain Old Java Objects)를 지원한다.
   6. 처리를 위한 일괄된 방법을 제공한다.
   7. 관련된 다양한 API를 제공한다.
   8. 다양한 API에 대한 연동을 지원한다.
2. **스프링을 사용하는 목적**
   1. 큰 프로젝트에서 프로그램을 수정할 때 관련되는 모든 프로그램의 소스 코드를 변경할 필요가 없으므로 유연성이 확보된다.
3. **메이븐(Maven)**
   1. 라이브러리 관리 프로그램
   2. 자동으로 라이브러리를 다운로드 받아서 처리해주는 툴
4. **DI(Dependency Injection)**
   1. 의존성 객체를 특정한 위치에 알아서 넣어주는 기능
5. **DI(Dependency Injection)의 종류**
   1. Constructor Injection : 생성자를 이용하여 객체를 얻어와서 주입
   2. Setter Injection : Setter Method를 이용하여 객체를 매개변수로 받아서 주입
6. **빈의 범위**
   1. Singleton : 하나의 객체를 요청할 때마다 공유해서 대여
   2. Prototype : 매번 요청할 때마다 새로운 객체를 만들어서 대여
   3. Session : 로그인한 동안만 객체를 공유
   4. Request : 한 페이지 이상 객체를 공유
7. **어노테이션의 개요 및 장, 단점**
   1. 개요
      1. 환경설정 부분을 단순하게 설정
      2. 자바코드의 일부 대용
      3. 어노테이션은 Spring 2.0 부터 시작
   2. 장점
      1. 시스템 복잡성이 아니라면 어노테이션 사용은 적합하게 쓰이면 코드가 간결해지고 유지보수가 용이해짐
      2. 대형 시스템엔 계층 구조가 잘 파악되기 위해서는 xml 사용이 필수
   3. 단점
      1. 어노테이션은 메타 정보가 소스 코드에 들어가므로 파악하기 어려움
      2. 소스 코드가 같이 제공되지 않으면 사용에 제약이 따름
8. **@Required의 기능 및 선언**
   1. 기능
      1. 메서드를 호출하지 않으면 에러를 발생 시킨다.(필수 메서드)
   2. 선언
      1. 메서드 위에 선언 : 반드시 호출할 메서도 표시
9. **@Autowired의 기능 및 선언**
   1. 기능
      1. @Inject와 동일하다.
      2. 생성자, 멤버 변수, 메서드에 지정이 가능
      3. Setter Method 외에 다른 메서드에서 사용이 가능
      4. 해당하는 빈즈 객체가 없거나 두 개 이상 존재 시 에러 유발
   2. 선언
      1. 멤버 변수에 선언 : Setter Method 호출 없이 자동적으로 타입을 찾아서 저장
      2. 메서드 위에 선언 : 이 메서드를 반드시 호출해서 해당되는 타입의 객체를 찾아서 저장
   3. 형식
      1. @Autowired(속성명1=값1, 속성명2=값2, …)
10. **@Resource의 기능 및 선언**
    1. 기능
       1. byName으로 검색해서 원하는 자바빈즈 객체를 저장시키는 방법
    2. 선언
       1. 멤버 변수, 메서드 위에 선언
    3. 형식
       1. @Resource(name=”빈즈의 구분자 id”)
11. **@PostConstruct, @PreDestroy**
    1. @PostConstruct : 빈즈 객체 생성 전에 호출하는 초기화 작업
    2. @PreDestroy : 빈즈 객체 생성 후에 호출하는 메모리 해제 작업
12. **@Component**
    1. 자동적으로 그 패키지에 들어가 있는 모든 클래스를 자동적으로 빈즈 객체로 등록을 시켜주는 역할을 하는 어노테이션
    2. 형식
       1. @Component(“등록하길 원하는 id값”)
       2. @Component => 클래스 이름을 id로 인식
13. **스프링에서 빈즈 객체를 만들어서 등록하고 불러오는 방법(2가지)**
    1. xml 파일을 만들지 않고 직접 자바의 클래스에 어노테이션을 이용해서 빈즈를 만들고 등록하는 방법
    2. 빈을 xml 파일로 만들어서 환경설정을 하는 방법
    3. 보편적으로 둘 다 혼합해서 사용함.
14. **JNDI(Java Naming Directory Interface)**
    1. DB 연결의 정보를 Context.xml에 저장하고, DBConnectionMgr.java 클래스에서 uri, driver, 계정, 암호를 저장하고 이를 통해 자동으로 연결하는 역할
15. **MyBatis 개요 및 장, 단점**
    1. 개요
       1. xml 파일을 이용해서 SQL을 자바 코드와 연결해서 관리해주는 프레임워크
    2. 장점
       1. xml 파일로 SQl 문장을 관리하므로 가독성이 높아진다.
       2. JDBC 드라이버가 있다면 어떤 DB에서도 사용이 가능하다.
       3. DB연결 정보의 관리가 용이하다.
       4. JavaBean 스타일의 클래스를 지원한다.
       5. 여러 개의 DB에 접근이 쉽다.
       6. SQL 구문을 작성할 때 매개 변수와 반환형은 자바의 기본 자료형, 객체 자료형을 사용한다.
       7. 복잡한 객체 모델 등을 쉽게 생성한다.
    3. 단점
       1. 환경 설정이 어렵다.
16. **MyBatis CRUD 개요 및 속성**
    1. 개요
       1. Insert, update, delete, select할 때 사용되는 태그
    2. 속성
       1. id(필수) : SQL 문장의 종류와 상관없이 필수로 작성
       2. parameterType(선택) : SQL 구문 중에서 매개 변수를 입력 받아서 처리해주는 경우에 필요
       3. resultType(선택) : SQL 구문을 사용해서 반환 값이 있는 경우
17. **MyBatis에서 매개 변수를 받을 때 주의할 점**
    1. #{매개 변수} => 멤버 변수와 변수 명이 같아야 한다.
    2. 매개 변수가 순수 값을 입력 받는 것이 아닌 필드 명으로 입력 받거나 특수 기호가 포함되는 값은 ${매개 변수}를 써야한다.
18. **POJO 클래스의 개요 및 특징**
    1. 개요
       1. 독립적으로 사용자로부터 요청을 받아서 처리해주는 기능을 가진 메소드를 직접 사용할 수가 있기 때문에 상속받지 않고 단독적으로 사용할 수 있는 클래스(컨트롤러 역할을 하는 클래스)
    2. 특징
       1. 인터페이스 or 추상클래스를 상속받지 않아도 된다.
       2. @Controller를 부여한다.
       3. 요청을 받아서 처리해주는 메서드를 개발자 마음대로 변경이 가능하다.
       4. 클래스 내부에 여러 개의 메서드를 작성할 수 있다.
19. **@RequestMapping, @RequestParam 형식**
    1. @RequestMapping(value=”요청 경로”, method=”요청 방식”)
    2. @RequestParam(value=”매개변수 명”, defaultValue=”디폴트 값”)
20. **Dynamic Web Project에서 스프링 Maven을 만드는 방법**
    1. 다 작성된 프로젝트 오른쪽 클릭 -> configure -> convert to maven project
    2. pom.xml 생성
21. **xml 에서 <, > 연산자 인식시키는 방법**
    1. <![CDATA[ <, > 연산자가 들어가는 SQL 내용 ]]>