230627 시험공부정리

▼ **스프링 환경설정**

서술형

▼ **WAS(Web Application Server)정의**

DB 조회 및 다양한 로직 처리 요구시 **동적인 컨텐츠를 제공**하기 위해 만들어진 애플리케이션 서버

▼ **버전관리시스템(==형상관리시스템) 사용 장점**

👉**소프트웨어 변경사항을 체계적으로 추적하고 통제**

버전관리와 변경내역 조회 가능

1. 소스코드의 변경이력을 관리할 수 있어서 추적성이 높다.
2. 배포가 편리
3. 파일 및 프로젝트를 이전 상태로 되돌릴 수 있다.
4. 여러 사람이 동일한 소스코드를 공유해 개발할 수 있으며 공유할 때 생기는 버전 충돌 문제 등을 해결할 수 있음.

▼ **git 로컬저장소(Local Repository)와 원격저장소(Remote Repository) 차이**

로컬 저장소 (Local Repository)

- 현재 내가 사용하고 있는 **내 디바이스(PC)**에 저장되는 저장소
- 원격저장소의 파일, 혹은 폴더를 로컬 저장소로 **Pull** 할 수 있다.

원격 저장소 (Remote Repository)

- **원격 서버**에 저장되고 관리되는 저장소
- 로컬 저장소의 파일, 혹은 폴더를 원격 저장소에 **Push** 할 수 있다.

▼ **http Error(에러)상태코드**

(https://developer.mozilla.org/ko/docs/Web/HTTP/Status)

400번대 클라이언트 오류

400 Bad Request ： 잘못된 문법으로 인하여 서버가 요청을 이해할 수 없음을 의미

403 Forbidden : 서버에 요청이 전달되었지만, 권한 때문에 거절되었다는 것을 의미

404 Not Found ： 서버가 요청받은 리소스를 찾을 수 없다는 것을 의미

500번대 서버오류

500 Internal Server Error : 서버가 처리 방법을 모르는 상황이 발생

502 Bad Gateway : 서버가 요청을 처리하는 데

필요한 응답을 얻기 위해

게이트웨이로 작업하는 동안 잘못된 응답을 수신했음을 의미

▼ **git 용어**

```
git config --global user.name "[firstname lastname]"
git config --global user.email "[valid-email]"
// 사용자 이름, 이메일 설정 및 확인

git status
// 현재 상태 확인

git log
// 전체 로그 확인

git init
// git 저장소 생성

git clone [url]
// git 저장소 복제

git add .
// 디렉토리 내 모든 파일 추가

git commit -m "message"
// 커밋생성

git push origin master
// git 변경사항 원격서버 업로드(push)

git pull
```

```
git diff [브랜치이름] [다른브랜치이름]
//변경내용 merge하기 전에 바뀐 내용 비교

git rm [file]
디렉토리 내 파일 삭제

git clean
// 디렉토리 내 필요없는 파일 삭제



git branch

git branch -r
// git 브랜치 목록 조회

git branch -a
// 전체 브랜치 목록 조회

git branch [branch-name]
// git 브랜치 생성

git checkout
// 브랜치 변경 하고 디렉토리로 복사

git merge [branch명]

git reset
// 중대실수해서 코드 초기로 돌려야 될 때 사용

git revert
// 커밋 되돌리기
```

▼ 스프링 주요개념

IOC **(Inversion of Control)** **제어 반전**	컨트롤의 제어권이 개발자가 아니라 프레임워크에 있다는 뜻으로 객체의 생성부터 모든 생명주기의 관리까지 프레임워크가 주도하고 있다. 객체를 생성하고, 직접 호출하는 프로그램이 아니라, 만들어둔 자원을 호출해서 사용한다.
DI **(Dependency Injection)** **의존성 주입**	설정 파일이나 어노테이션을 통해 객체간의 의존 관계를 설정하여 개발자가 직접 의존하는 객체를 생성할 필요가 없다.
POJO 기반 프레임워크 **(Plain Old Java Object)**	J2EE, EJB와 같은 특정 기술이나 라이브러리의 내용을 상속받아 클래스를 구현하지 않고 일반적인 기본 기능만을 가진 순수한 자바 객체를 의미한다. 특정 클래스에 종속되지 않으므로 자바의 객체지향적 설계가 쉬워지고, 코드길이 감소, 유지보수성 증가, 기존 Java API, 라이브러리 지원에 용이하다. * J2EE(Java2 Enterprise Edition) : Servlet, JSP 레벨의 서버 프로그래밍 인터페이스 * EJB(Enterprise Java Bean) : 쉽게 웹 개발이 가능한 기술, 객체지향 장점을 포기해야 하는 문제점 발생
Spring AOP **(Aspect Oriented Programming)** **관점 지향 프로그래밍**	트랜잭션, 로깅, 보안 등 여러 모듈, 여러 계층에서 공통으로 필요로 하는 기능의 경우 해당 기능들을 분리하여 관리한다.

Spring JDBC	Mybatis나 Hibernate 등의 데이터베이스를 처리하는 영속성 프레임워크와 연결할 수 있는 인터페이스를 제공한다.
Spring MVC	MVC 디자인 패턴을 통해 웹 어플리케이션의 Model, View, Controller 사이의 의존 관계를 DI 컨테이너에서 관리하여 개발자가 아닌 서버가 객체들을 관리하는 웹 애플리케이션을 구축 할 수 있다.
PSA **(Portable Service Abstraction)**	스프링은 다른 여러 모듈을 사용함에 있어 별도의 추상화 레이어를 제공한다. 예를 들어 JPA를 사용할 때에서 Spring JPA를 사용하여 추상화하므로 실제 구현에 있어서 Hibernate를 사용하든 EclipseLink를 사용하든 개발자는 이 모듈의 의존 없이 프로그램에 집중할 수 있다.

▼ 스프링 각 어노테이션 역할

Bean등록시 사용

@Component	- 객체(컴포넌트)를 나타내는 일반적인 타입으로 <bean> 태그와 동일한 역할
@Repository	- 퍼시스턴스(persistence) 레이어, 영속성을 가지는 속성(파일, 데이터베이스)를 가진 클래스 ex) Data Access Object Class
@Service	- 서비스 레이어, 비즈니스 로직을 가진 클래스 ex) Service Class
@Controller	- 프리젠테이션 레이어, 웹 애플리케이션에서 View에서 전달된 웹 요청과 응답을 처리하는 클래스 ex) Controller Class

@Repository, @Service, @controller 는 특정 객체의 역할에 대한 @Component의 구체화 형태다.

@Autowired	- 정밀한 의존 관계 주입(DI)이 필요한 경우에 유용하다. - @Autowired는 필드 변수, setter 메소드, 생성자, 일반 메소드에 적용 가능하다. - 의존하는 객체를 주입할 때 주로 Type을 이용하게 된다. - @Autowired 는 <property>, <constructor-arg> 태그와 동일한 역할을 한다.	
	@Qualifier	@Autowired와 함께 쓰이며, 한 프로젝트 내에 @Autowired로 의존성을 주입하고자 하는 객체가 여러개 있을 경우, @Qualifier("name")를 통해 원하는 객체를 지정하여 주입할 수 있다.
@Resource	- 애플리케이션에서 필요로 하는 자원을 자동 연결할 때 사용된다. - @Resource는 프로퍼티, setter 메소드에 적용 가능하다. - 의존하는 객체를 주입 할 때 주로 Name을 이용하게 된다.	
@Value	- 단순한 값을 주입할 때 사용되는 어노테이션이다. - @Value("Sprign")은 <property .. value="Spring"/>와 동일한 역할을 한다.	

▼ **JAR파일과 WAR파일**

JAR (Java Archive)

- JAVA 어플리케이션이 동작할 수 있도록 **자바 프로젝트를 압축한 파일**
- Class와 같은 Java 리소스와 속성 파일, 라이브러리 및 액세서리 파일이 포함

WAR (Web Application Archive)

- servlet / jsp 컨테이너에 배치 할 수 있는 **웹 어플리케이션(Web Application) 압축한 파일**
- **WAR파일을 실행**하려면 Tomcat같은 **웹 서버 (WEB)또는 웹 컨테이너(WAS)가 필요**

▼ **jsp안에 scope 외우기**

> 💡 내장객체 우선순위 : page > request > session > application

1. **page : 1페이지**

현재 페이지만 , 현재 Servlet 또는 JSP에서만 사용 가능

2. **request : 최소2페이지 이상**

👉요청받은 페이지(Servlet/JSP)와 요청을 위임받은 페이지(Servlet/JSP)에서 사용 가능

3. **session**

현재 사이트에 접속한 브라우저 1개식 생성
브라우저가 종료되거나, session이 만료될 때 까지 유지
(세션에 로그인 정보를 기록해둠

👉 브라우저가 종료되거나 로그아웃 되기 전까지 계속 로그인 상태 유지됨)

4. **application**

하나의 웹 애플리케이션 당 1개만 생성되는 객체
👉 서버 시작시 생성되며, 종료 시 까지 유지
👉 누구든지 사용 가능

▼ mybatis에다가 어떻게 적으면 sql돌아가는지(mapper내에서의 상황)

👉 mapper 파일 생성

```
<!-- SQL이 작성되는 mapper 파일 위치를 등록 -->
  <mappers>
    <!--
       <mapper resource="mapper 파일 경로" />
        경로를 작성하는 기준(시작지점)은 src/main/resources 폴더
     -->

     <mapper resource="/mappers/member-mapper.xml" />
     <mapper resource="/mappers/myPage-mapper.xml" />
     <mapper resource="/mappers/board-mapper.xml" />
     <mapper resource="/mappers/reply-mapper.xml" />
     <mapper resource="/mappers/chatting-mapper.xml"/>
  </mappers>
```

mapper생성시 반드시 해줘야 되는거 2개

1. cache-ref 태그 삭제
2. mapper 태그에 namespace 속성 추가

namespace : 해당 파일(공간)을 쉽게 부르는 이름 (DAO에서 사용)

```
<mapper namespace="boardMapper">

  <!-- 이미지 정보 조회용 resultMap -->
    <resultMap type="boardImage" id="boardImage_rm">
      <id property="imageNo" column="IMG_NO" />
      <result property="imageReName" column="IMG_RENAME" />
      <result property="imageOriginal" column="IMG_ORIGINAL" />
      <result property="imageLevel" column="IMG_LEVEL" />
      <result property="boardNo" column="BOARD_NO" />
    </resultMap>


  <!-- 특정 게시글 이미지 목록 조회 -->
    <select id="selectImageList" resultMap="boardImage_rm">
      SELECT * FROM BOARD_IMG
      WHERE BOARD_NO = #{boardNo}
      ORDER BY IMG_LEVEL
    </select>



  <!-- 게시판 코드, 이름 조회 -->
    <select id="selectBoardType" resultMap="boardType_rm">
```

```
      SELECT * FROM BOARD_TYPE
      ORDER BY BOARD_CD
    </select>



  <!-- 게시글 이미지 1개 삽입 -->
    <insert id="insertBoardImage">
      INSERT INTO BOARD_IMG VALUES(

        SEQ_IMG_NO.NEXTVAL,
        #{imageReName},
        #{imageOriginal},
        #{imageLevel},
        #{boardNo}
      )
    </insert>


  <!-- 게시글 이미지 1개 수정 -->
    <update id="updateBoardImage">
      UPDATE BOARD_IMG SET
      IMG_RENAME = #{imageReName},
      IMG_ORIGINAL = #{imageOriginal}

      WHERE BOARD_NO = #{boardNo}
      AND IMG_LEVEL = #{imageLevel}
    </update>

    <!-- 게시글 이미지 삭제( IN 구문 작성 시 삭제되는 deleteList에 '' 없도록 주의) -->
    <delete id="deleteBoardImage">
      DELETE FROM BOARD_IMG
          WHERE BOARD_NO = ${boardNo}
          AND IMG_LEVEL IN ( ${deleteList} )
    </delete>

</mapper>
```

▼ **session관리하는 클래스**

httpSession 세션정보 기록 지우는거 .....…

session.invalidate()

▼ **aop할 때 advice에서 5가지 나열 및 관점 서술**

@Before @Around @After @AfterReturning @AfterThrowing

@Before(“pointcut”)	- 타겟 객체의 메소드가 실행 되기 전에 호출되는 어드바이스 - JoinPoint를 통해 파라미터 정보를 참조할 수 있다.
@Around (“pointcut”)	- 타겟 객체의 메소드 호출 전과 후에 실행 될 코드를 구현할 어드바이스
@After(“pointcut”)	- 타겟 객체의 메소드가 정상 종료 됐을 때와 예외가 발생했을 때 모두 호출되는 어드바이스로, 반환 값을 받을 수 없다.
@AfterReturning(Pointcut=“”, Returning=“”)	- 타겟 객체의 메소드가 정상적으로 실행을 마친 후에 호출되는 어드바이스 - 리턴 값을 참조할 때는 returning 속성에 리턴 값을 저장할 변수 이름을 지정해야 된다.
@AfterThrowing(Pointcut=“”, throwing=“”)	- 타겟 객체의 메소드에서 예외가 발생하면 호출되는 어드바이스 - 발생된 예외를 참조할 때는 throwing 속성에 발생한 예외를 저장할 변수 이름을 지정해야 한다.

▼ CI/CD (Continuous Integration/Continuous Delivery)

애플리케이션 개발 단계를 자동화하여 애플리케이션을 더욱 짧은 주기로 고객에게 제공하는 방법

문제해결시나리오

▼ jstl 때문에 나는 오류

라이브러리 3개 따로 넣어줬음 못찾아서 나는 오류임

메이븐에서 못찾아서 lib파일 따로 만들어서 jstl관련 lib넣어주면 오류 안뜸

▼ pom.xml에서 스프링 버전? 사용할 거 하는거

스프링프레임워크는 한가지 버전으로 모두 통일 시켜줘야 오류가 안뜸.
버전이 안맞아서 나는 오류임

properties 안에 <org.springframework-version>5.3.14</org.springframework-version> 부분을 아래 dependency에서 변수로 사용해서 버전을 불러옴

```
<!-- properties : 메이븐이 적용된 프로젝트에서 공통적으로 사용할 버전
                  또는 설정값 정보를 작성하는 태그 -->
  <properties>
    <java-version>11</java-version>
    <org.springframework-version>5.3.14</org.springframework-version>
```

```
    <org.aspectj-version>1.9.4</org.aspectj-version>
    <org.slf4j-version>1.7.25</org.slf4j-version>
  </properties>


<!-- 스프링에서 JDBC를 사용할 수 있게 하는 라이브러리 -->
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>${org.springframework-version}</version>
      <!-- 위 properties의 지정한 Spring 버전을 따라감 -->
    </dependency>
```

▼ pom.xml 안에 gson 라이브러리 넣어놨는데 dependency에서 빨간줄 뜨는 이유

```
<!--gson lib (틀린문구) -->
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>com.google.code.Gson</artifactId>
  <version>2.8.6</version>
</dependency>
```

❗오류문구 : Missing aartifact com.google.code.gson:com.google.code.Gson:jar.2.8.6

👉오류나는 이유

GSon 해당 라이브러리 버전을 못찾아서 or `artifactId`나 `version`이 없어서 뜨는거임

👉 해결방법

`artifactId` 은 메이븐에서 이미 정해준 코드(?)가 있어서 위에 처럼 적는게 아니라 아래에 `artifactId` `version` 아래처럼 기록하기

```
// pom에 적는 gson 라이브러리
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.9.0 </version>
</dependency>
```