@RequestMapping() :

요청이 들어왔을 시에 컨트롤러와 매핑해주고, 그 컨트롤러를 실행시켜 응답을 받는 것, 보통 공통적인 url 경로를 괄호안에 작성한다

main 패키지에서 TestDTO 클래스를 생성후 @Data 를 설정해주면 내부 필드 값들의 get,set,toString등을 사용할 수 있게된다

**@Data**

**public class TestDTO {**

**private String id; private int age;**

**}**

컨트롤러에서 GETMapping 을 사용할 경우

url 에서 파라미터 값을 바인딩해서 줄 수 있다

@RequestParam 또는 @ModelAttribute 를 사용해서 값을 받을 수 있다

mapper.xml 에서 db update 사용

interface [mapper.java](http://mapper.java)에서

반환값은 void 또는 int 이고 메소드의 매개변수로 VO 객체를 설정한다

**public int update(BoardVO bo);**

mapper.xml 에서

update 태그를 사용해서 update 할 내용을 작성하고

where 에서 bnum으로 조건을 추가한다

**<update id="update">**

**update t\_board set content = #{content}, title = #{title}, writer = #{writer},**

**updateDate = sysdate where bnum = #{bnum}**

**</update>**

Test 클래스에서 업데이트를 위한 메소드를 만들고

메소드 내부에서 VO 객체를 새로 생성하고 변경할 값을 객체에 담아주어서

mapper 클래스의 update 메소드를 실행할 수 있게한다

@Test 를 사용해서 생성된 메소드를 테스트할 수 있게해준다

@Test

public void up() {

BoardVO bo = new BoardVO();

bo.setBnum(21);

bo.setTitle("ctitle");

bo.setContent("Changecontent");

bo.setWriter("author2");

bm.update(bo);

[log.info](http://log.info/)(bo);

}

**BServiceTests 클래스에서 @Autowired 를 사용한 BService는 어떻게**

**BServiceImpl 클래스를 사용하게 되는가**

@Autowired 는 각 상황의 타입에 맞는 IoC 컨테이너 안에 존재하는 Bean을 자동으로 주입해준다

보통의 자바에서는 BService bser = new BServiceImpl(); 로 이루어 지는 부분이 @Autowired를 통해 처리되게 된다

@Autowired BService bser; 바로 이부분이

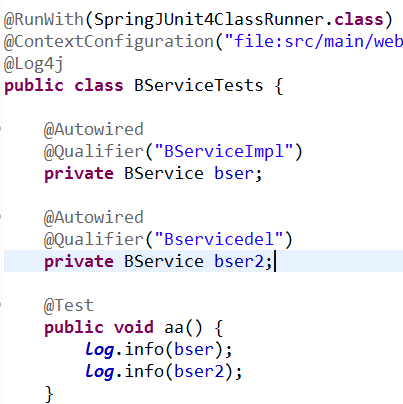
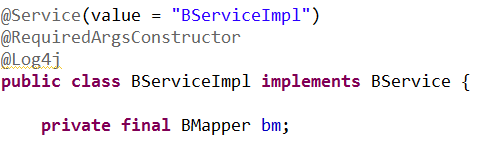
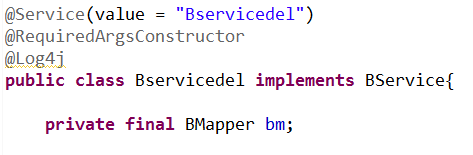
BServiceImpl 클래스에서 부여된 @Service 로 인해 해당 구현클래스를 실행하기 전까지는 알지 못하는 상태였다가 실행이 되면 인터페이스 Service를 상속받은 BServiceImpl 클래스를 자동으로 주입하게된다

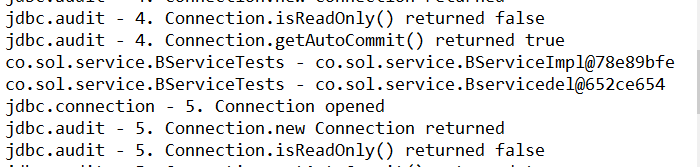
참조 글 : <https://codevang.tistory.com/312>

확인을 하기 위해 같은 인터페이스를 상속받은 클래스를 한 개 더 생성 후 각각의 클래스에서 @Service (value)값을 설정후 BServiceTests 에서 @Autowired 와 @Qualifier(”설정한 value “)로

각각의 클래스를 자동주입 되도록한뒤 호출해본 결과 같은 BService 인터페이스이지만 value값을 지정한 다른 클래스를 호출하게된다

**위의 실험결과를 이미지파일로 저장해서 다음 장에 올려둠**





**삭제나 수정 작업에서 글의 존재 유무를 먼저 확인 후 진행하는 법**

인터페이스 BService 에서 삭제나 수정 메소드를 boolean으로 만든다

**boolean update(BoardVO bo);**

해당 인터페이스를 상속받은 클래스에서 메소드를 실행시켜서

리턴 받은 값이 1(true)값일 때 만 작동하도록 한다

**public boolean update(BoardVO bo) { return bm.update(bo) == 1; }**

테스트 또는 실제 작동시 객체를 매번 새롭게 생성해서 값을 주입하는 방식대신

원하는 데이터를 가져와서 객체를 생성하고 null 값이 반환된다면 함수가 종료되고

아니라면 수정또는 삭제가 이루어지도록한다

**@Test**

**public void up() {**

**BoardVO bo = bser2.get(21);**

**// 수정하고자 하는 글을 가져온다**

**if(bo == null) {**

**// 존재 하지않는 다면 수정 작업을 취소하고**

**return;**

**}**

**bo.setContent("change content");**

**//있다면 변경내용을 set해주고**

**log.info(bser2.update(bo));**

**// update를 해준다**

**}**