Sec03

Model 에서 @ 사용하기

작성자: 강태경



# Exam05

주요 어노테이션

* @Slf4j : 로깅 라이브러리 어노테이션 (롬북) , 인터페이스 <- Logback / log4j2 (구현 클래스 로깅)
* 사용법 ex) 1) @Slf4j 클래스 위에 달고
* 2) 로깅을 하고 싶은 라인에서 log.debug(변수.toString());
* 설치된 로깅 라이브러리 : Logback 설치파일 2개 복사, build.gradle 라이브러리 추가 (4개)
* @Service : 클래스 위에 달고, @Service 달린 클래스는 스프링부트에서   
  서버 기동시 자동 객체를 생성함 생성된 객체는 스프링 컨테이너에서 관리됨(생성된 객체 : 싱글톤(공유객체))  
  유사 어노테이션 : @Repository, @Component 등
* @Autowired : 위에서 생성된 객체가 필요하면 @Audowired 를 변수위에 달면 생성된 객체를 1개 받을 수 있음
* @GetMapping : Get 방식으로 오는 요청에 대해 실행되는 어노테이션, 주로 함수위에 달며,  
   Get + URL 일치여부를 확인후 @달린 해당 함수가 실행됨 ( Select 문 실행 )  
  클라이언트 : Get 요청 -> 스프링서버 : @GetMapping -> DB : select 문 호출
* @PostMapping : Post 방식으로 오는 요청에 대해 실행되는 어노테이션, 주로 함수위에 달며,Post + URL 일치여부를 확인후 @달린 함수가 실행됨 ( insert 문 실행 )  
  클라이언트 : Post 요청 -> 스프링서버 : @PostMapping -> DB : insert 문 호출
* @PutMapping : Put 방식으로 오는 요청에 대해 실행되는 어노테이션, 주로 함수위에 달며,Put + URL 일치여부를 확인후 @달린 함수가 실행됨 ( update 문 실행 )  
  클라이언트 : Put 요청 -> 스프링서버 : @PutMapping -> DB : update 문 호출
* @DeleteMapping : Delete 방식으로 오는 요청에 대해 실행되는 어노테이션, 주로 함수위에 달며,Delete + URL 일치여부를 확인후 @달린 함수가 실행됨 ( delete 문 실행 )  
  클라이언트 : Delete 요청 -> 스프링서버 : @DeleteMapping -> DB : delete 문 호출
* editDept() : 컨트롤러 함수, updateDept 서비스 함수를 호출
* updateDept() : update 문 실행, DB 수정됨

# Exam06

부서 정보 게시판 만들기   
부서 컨트롤러deleteDept() 함수 추가, removeById () :

* @DeleteMapping : delete 요청시 사용되는 어노테이션,   
  delete + URL 일치여부 확인 후 @ 부착된 함수가 실행됨
* deleteDept() : 서비스의 removeById() 함수 실행
* removeById () : DB에 부서번호(id)에 해당하는 데이터 삭제

# Exam07

@RestController, ResponseEntity, getDeptAll() 사용 예제

* ResponseEntity : 객체 + 상태메세지(.OK 등) 를 같이 리턴하는 객체
* @RestController : 리턴값이 json 데이터 형태로 클라이언트에 전송되는 어노테이션  
  주로 Vue + Springboot 연동 프로그램을 제작할 때 사용됨  
  URL 은 Rest API ( GET, POST, PUT, DELETE )가 사용됨
* 컨트롤러 getDeptAll() : ResponseEntity 사용한 전체 부서조회 함수  
  ( Get 방식 : select 문 호출 )  
  Rest Client URL 테스트 : http://localhost:8000/exam07/dept

# Exam08

@RestController, ResponseEntity, getDeptId () 사용 예제

* ResponseEntity : 객체 + 상태메세지(.OK 등) 를 같이 리턴하는 객체
* 컨트롤러 getDeptId () : ResponseEntity 사용한 부서번호로 부서정보를 조회하는 함수( Get 방식 : select 문 호출 )  
  Rest Client URL 테스트 : http://localhost:8000/exam08/dept/10

# Exam09

@RestController, ResponseEntity, createDept () 사용 예제

* ResponseEntity : 객체 + 상태메세지(.OK 등) 를 같이 리턴하는 객체
* @PostMapping : 클라이언트에서 insert 문 요청 시 사용  
  POST + URL 일치 여부 확인 후 일치하는 함수가 실행됨
* 컨트롤러 createDept () : 새로운 부서정보를 생성하는 함수  
  ( Post 방식 : insert 문 호출 )  
  서비스의 .save() 함수를 호출함  
  Rest Client URL 테스트 : <http://localhost:8000/exam09/dept>
* 서비스 .save(객체) 함수 : 매개변수 객체의 부서번호(키값)가 없으면 insert 가 실행되고, 있으면 update 가 실행되는 함수  
  현재는 부서 객체에 부서번호가 없으므로 insert 문이 실행됨

# Exam10

@RestController, ResponseEntity, updateDept () 사용 예제

* ResponseEntity : 객체 + 상태메세지(.OK 등) 를 같이 리턴하는 객체
* @PutMapping : 클라이언트에서 update 문 요청 시 사용  
  PUT + URL 일치 여부 확인 후 일치하는 함수가 실행됨
* 컨트롤러 updateDept () : 부서번호에 해당하는 객체정보를 수정하는 함수  
  ( Put 방식 : update 문 호출 )  
  서비스의 .save() 함수를 호출함  
  Rest Client URL 테스트 : <http://localhost:8000/exam10/dept>/10
* 서비스 .save(객체) 함수 : 매개변수 객체의 부서번호(키값)가 없으면 insert 가 실행되고, 있으면 update 가 실행되는 함수  
  현재는 부서번호가 있으므로 update 가 실행됨

# Exam11

@RestController, ResponseEntity, deleteDept () 사용 예제

* ResponseEntity : 객체 + 상태메세지(.OK 등) 를 같이 리턴하는 객체
* @DeleteMapping : 클라이언트에서 delete 문 요청 시 사용  
  Delete + URL 일치 여부 확인 후 일치하는 함수가 실행됨
* 컨트롤러 deleteDept () : 부서번호에 해당하는 객체정보를 삭제하는 함수  
  ( Delete 방식 : delete 문 호출 )  
  서비스의 .removeById() 함수를 호출함  
  Rest Client URL 테스트 : <http://localhost:8000/exam11/dept>/10
* 서비스 .save(객체) 함수 : 매개변수 객체의 부서번호(키값)가 없으면 insert 가 실행되고, 있으면 update 가 실행되는 함수  
  현재는 부서번호가 있으므로 update 가 실행됨