Sec01

Controller 에서 @ 사용하기

작성자: 강태경



# Exam01

실무에서는 MVC 디자인 패턴 속해서 스프링부트를 사용해 코딩을 진행함. 파일관리 시 파일을 용도에 맞게 정리하듯이 MVC 디자인 패턴도 패키지 및 클래스를 용도에 맞게 분류해서 코딩을 진행함

* M(Mode) : Model, Service, DAO, DTO 등을 사용하는 영역  
  Model 역할 : 정보의 저장/검색이 목적인 순수 클래스, 속성(필드)/생성자/Setter/Getter 만 사용함  
  Service 역할 : 업무 로직이 정의된 함수들이 있는 클래스
* V(View) : Frontend(클라이언트)의 화면을 의미하면 주로 Vue / React / Thymeleaf 등을 사용함
* C(Controller) : View 와 Service 를 연결해주는 역할을 담당하는 클래스  
  View의 URL 를 통해서 화면과 연결되고 Service 의 결과를 다시 View 에게 전달함
* 어노테이션(@) 정의 : 주로 특별한 기능을 부여하기 위해 사용  
   클래스나 함수(메소드), 변수 위나 앞에 추가하여 사용함  
   주요 기능 : 자동 객체 생성, 자동 Setter/Getter 생성 등  
   장점 : 코드량이 감소하고, 유지보수가 쉬우며, 생산성이 증가함
* @Controller : View 가 SSR 이라면(Thymeleaf, JSP) 사용되는 컨트롤러
* @GetMapping(요청URL) : frontend 에서 Get 방식의 조회 요청에 대해 URL 이 일치할 경우 그 아래 함수가 실행됨  
  유사 어노테이션 : @PostMapping, @PutMapping, @DeleteMapping
* Model : 컨트롤러의 함수 매개변수로 사용되며 데이터를 저장해 View 로 정보를 전달하는 역할을 담당.  
  구조는 HashMap 의 구조와 유사 : (키, 값) 으로 구성됨

# Exam02

@RequestMapping(공통URL), @RequestParam(변수) 을 사용하는 예제

* @RequestMapping(공통URL) : 공통 URL 은 클래스 앞에 정의하여 세부적인 함수의 URL 과 결합하여 사용됨, 불필요한 코드 생략(생산성 증대)  
  예) @RequestMapping("/exam02") + @GetMapping("/hello-name") == @GetMapping("/exam02/hello-name")
* 쿼리스트링(QueryString) : URL 에서 매개변수를 통해 값을 전달하는 방식  
  예) http://localhost:8000/hello-name?name=kang
* @RequestParam(변수) : form의 Get/Post 방식의 쿼리스트링으로 전달되는 데 이때 사용되는 어노테이션  
   쿼리스트링의(URL 에 ?변수=값 형태) 각 변수의 값이 함수의 매개변수의 값으로 전달함 ( URL 매개변수명 == 함수 매개변수명 일치해야함 )

# Exam03

*@RequestMapping, @RequestParam 2개 이상 처리*하는 예제

* *@RequestParam 은 함수의 매개변수로 2개 이상 사용 할 수 있음  
  2개 이상 사용 시 쿼리스트링의 각 값과 연결됨  
  쿼리스트링의 각 변수=값 은 & 사용해서 여러개 전달할 수 있음*URL 테스트 : http://localhost:8000/exam03/hello-name-id?name=kang&id=forbob
* Model 의 각 키 / 값은 일반 자료형 변수( int / double / Boolean / String ) 뿐만 아니라 자료구조(List / Set / Map ) 도 담아서 View에 정보를 전달할 수 있음
* Resource / application.properties : 스프링부트의 환경을 설정하는 파일  
  포트 설정 : server.port=8000  
  타임리프 설정 : spring.thymeleaf.prefix=classpath:/templates/
* 타임리프 문법  
  텍스트 출력 : [[${변수}]]  
  반복문 : html 태그 내부에 th:each=”반복문” 사용하거나 th:block 사용함  
   <th:block …> … </th:block> : 반복 실행할 블럭에 정의  
   th:each=”요소 : ${배열}” : 배열의 값을 하나 빼서 요소에 저장하며 반복  
  예)  
   <th:block th:each="element : ${list}">  
   <p>[[${element}]]</p>  
   </th:block>  
  조건문 :   
  th:if =”${조건식}” : 조건식 이 true 이면 화면에 표시  
  th:unless=”${조건식}” : 조건식이 false 이면 화면에 표시  
    
  예)   
  <div th:if="${element == 'forbob'}">forbob 이 맞습니다.</div>  
  <div th:unless="${element == 'forbob'}">forbob 가 틀립니다..</div>