

Model 데이터 생성

- View에서 모델에 저장된 정보 사용 시 이용하는 Model

ModelAndView	Model 객체	Map 객체
	ModelMap 객체	Command 객체
Model	설명	
ModelAndView 객체 활용	@RequestMapping이 적용된 메소드가 ModelAndView객체를 리턴하면 View에서 ModelAndView에 저장된 Model 정보를 사용할 수 있음	
Model 객체 활용	Controller 메소드의 매개변수로 선언된 Model 객체에 저장된 정보도 View에서 사용할 수 있음	
Map 객체 활용	java.util.Map 타입의 Collection에 저장된 정보도 View 페이지에서 사용할 수 있음	
ModelMap 객체 활용	org.springframework.ui.ModelMap에 저장된 정보도 View 페이지에서 사용할 수 있음	
@ModelAttribute 활용	@ModelAttribute가 적용된 메소드에서 리턴한 객체도 View 페이지에서 사용할 수 있음	


유효성 체크

- 유효성 체크란?
 - 클라이언트의 입력정보에 문제가 있으며, 입력정보에 어떤 문제가 있는지를 다시 클라이언트에게 전달하기 위해서 수행함

- 유효성 체크 방법

1

org.springframework.validation.Validator 인터페이스를 사용하여 Command 객체의 유효성을 검증함



2

Controller에서 매개변수로 받아들인 Command 객체를 Validator의 validate() 메소드를 이용해서 유효성을 검증함

- ValidationUtils를 활용한 유효성 체크
 - org.springframework.validation.ValidationUtils 클래스를 이용하면 쉽게 유효성 체크를 처리할 수 있음
- Validator 적용
 - Controller 클래스의 메소드에 org.springframework.validation.BindingResult 객체를 매개변수로 받아들여야 함

에러 메시지 처리

에러 메시지 설정

- 에러 메시지를 설정을 위해 Errors 인터페이스의 reject(), rejectValue() 메소드를 사용함

Errors 메소드가 reject()인 경우

- 1 에러코드 + "." + Command 객체명
- 2 에러코드

Errors 메소드가 rejectValue()인 경우

- 1 에러코드 + "." + Command 객체명 + "." + 필드명
- 2 에러코드 + "." + 필드명
- 3 에러코드 + "." + 필드타입
- 4 에러코드

에러 메시지 출력

- <spring:hasBindErrors> 커스텀 태그 : 에러 정보를 **설정**함
- <form:errors> 커스텀 태그 : 에러 정보를 **출력**함

파일 업로드

Step 1

<form> 태그에 method 속성과 enctype 속성을 추가해야 함

Step 2

Spring에서 지원하는 MultipartResolver 인터페이스를 설정해야 함

Step 3

Controller에서는 @RequestParam Annotation과 MultipartFile을 이용해서 업로드된 파일 정보를 추출함

예외 처리

HandlerExceptionResolver 인터페이스

- Spring에서는 이런 예외 처리와 관련하여 HandlerExceptionResolver를 제공함

@ExceptionHandler 사용

- AnnotationMethodHandlerExceptionResolver 클래스는 @Controller이 적용된 클래스에서 @ExceptionHandler가 적용된 메소드를 이용해서 예외를 처리함

SimpleMappingExceptionHandler 사용

- SimpleMappingExceptionHandler를 사용하기 위해서는 Bean으로 등록해야 함

Controller 요청 가로채기

- HandlerExceptionResolver 인터페이스
 - Controller에 전달되는 클라이언트의 요청을 가로채기 위해 HandlerInterceptor를 사용함
 - 클라이언트의 요청이 Controller에 전달되기 전에 **사전 처리 작업을 하거나 Controller가 수행된 후에 리턴되는 ModelAndView를 조작할 수 있음**

HandlerInterceptor 구현

- HandlerInterceptor **모든 추상메소드를 Overrrding**해야 함
 - **HandlerInterceptorAdaptor** 클래스를 사용하여 필요한 메소드만 적절히 구현해서 사용하면 됨

HandlerInterceptor 설정

- 구현한 HandlerInterceptor는 **HandlerMapping** 클래스를 빈으로 등록할 때, **Spring 설정 파일에도 등록해야 함**