스프링 MVC
- 데이터 바인딩 -

@InitBinder 어노테이션과 커스텀 데이터 타입 변환처리

❖ 문자열을 지정한 타입으로 변환

- PropertyEditor를 사용
- HTTP 요청 파라미터를 객체의 프로퍼티 값으로 저장할 때
  - WebDataBinder가 내부적으로 PropertyEditor를 사용

@InitBinder 어노테이션과 커스텀 데이터 타입 변환처리

❖ @InitBinder 어노테이션을 이용한 Date 타입 변환 처리

- java.util.Date 타입의 프로퍼티 저장하는 경우
  - 타입 변환을 처리할 커스텀 PropertyEditor를 WebDataBinder에 등록
  - java.util.Date 타입에 대한 PropertyEditor로 CustomDateEditor 제공
  - **new CustomDateEditor(dateFormat, true)**
    - - 두번째 파라미터 : true인 경우 null 또는 빈 문자열 허용
      false인경우 검증에러(typeMissmatch) 발생

```
@Controller
public class QueryLogController {
   …

   @InitBinder
   protected void initBinder(WebDataBinder binder) {
      DateFormat dateFormat = new SimpleDateFormat('yyyy-MM-dd');
      binder.registerCustomEditor(Date.class,
                  new CustomDateEditor(dateFormat, true));
   }
}
```

@InitBinder 어노테이션과 커스텀 데이터 타입 변환처리

❖ @WebBindingInitializer를 이용한 공통 PropertyEditor 등록

- @InitBinder 어노테이션 메서드는 컨트롤러 단위로 동작
- @WebBindingInitializer는 전체 컨트롤러에 공통으로 적용
  - @WebBindingInitializer 인터페이스 정의

```
public interface WebBindingInitializer {
   vod initBinder(WebDataBinder binder, WebRequest request);
}
```

@InitBinder 어노테이션과 커스텀 데이터 타입 변환처리

❖ @WebBindingInitializer를 이용한 공통 PropertyEditor 등록

- @WebBindingInitializer 구현

```
public class CustomWebBindingInitializer implements WebBindingInitializer
{
   @Override
   public void initBinder(WebDataBinder binder, WebRequest request) {
      DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
      binder.registerCustomEditor(Date.class,
                  new CustomDateEditor(dateFormat, true));
   }
}
```

@InitBinder 어노테이션과 커스텀 데이터 타입 변환처리

❖ @WebBindingInitializer를 이용한 공통 PropertyEditor 등록

- 스프링 설정 등록

```
<beans:bean
class="org.springframework.web.servlet.mvc.annotation.AnnotationMethodHan
dlerAdapter">
   <beans:property name="webBindingInitializer">
      <beans:bean
         class="com.lecture.spring.binding.CustomWebBindingInitializer"/>
   </beans:property>
<beans:bean>
```

@Valid 어노테이션

❖ @Valid 어노테이션

- 유효성 검사 어노테이션
  - VO 객체의 필드에 어노테이션 연결

```
<dependency>
   <groupId>org.hibernate</groupId>
   <artifactId>hibernate-validator</artifactId>
   <version>4.2.0.Final</version>
</dependency>
```

@Valid 어노테이션

❖ @Valid 어노테이션

- JSR-303 Validator

어노테이션	설명
@AssertFalse	false 값만 통과 가능
@AssertTrue	true 값만 통과 가능
@DecimalMax(value=)	지정된 값 이하의 실수만 통과 가능
@DecimalMin(value=)	지정된 값 이상의 실수만 통과 가능
@Digits(integer=,fraction=)	지정된 정수와 소수 자리수보다 적을 경우 통과 가능
@Future	현재보다 미래일 경우만 통과 가능
@Past	현재보다 과거일 경우만 통과 가능
@Max(value)	지정된 값보다 아래일 경우만 통과 가능
@Min(value)	지정된 값보다 이상일 경우만 통과 가능
@NotNull	**null 값이 아닐 경우만 통과 가능**
@Null	null일 경우만 통과 가능
@Pattern(regex=, flag=)	해당 정규식을 만족할 경우만 통과 가능
@Size(min=, max=)	**문자열 또는 배열의 길이가 지정된 값 사이일 경우 통과**

@Valid 어노테이션

❖ @Valid 어노테이션

- Hibernate

어노테이션	설명
@NotEmpty	**Empty값이 아닌가?**
@Email	**Email 형식**
@URL	URL 형식
@Length(min=,max=)	**문자열 길이 min과 max 사이인지**
@Range(min=,max=)	숫자범위 체크

@Valid 어노테이션

❖ @DateTimeFormat

- 문자열을 Date 타입으로 바인딩

```
@DateTimeFormat(pattern="yyyy-MM-dd")
Date regDate
```

@Valid 어노테이션

```
public class Member implements Serializable{
    @NotEmpty
    @Length(min=4)
    private String userId;

    @NotEmpty
    private String name;

    @NotEmpty
    @Length(min=4)
    private String password;

    @NotEmpty
    @Email
    private String email;

    @DateTimeFormat(pattern="yyyy/MM/dd")
    private Date date;
    :
}
```

@Valid 어노테이션

❖ @Valid 유효성 검사

```
@RequestMapping(value="/create", method=RequestMethod.POST)
public String createPost(@Valid Member member, BindingResult result) {
   if(result.hasErrors()) return "member/create";

   if(service.insert(member))
      return "redirect:list";
   else
      return "member/create";
}
```