07장. 스프링 폼 태그 라이브러리
_@modelAttribute를 이용한 데이터 바인딩

☞ **@ModelAttribute**

정의

- 요청 파라미터를 객체에 바인딩하고 모델에 자동 등록해주는 어노테이션
- 컨트롤러 메서드의 매개변수 또는 반환값을 모델에 자동 바인딩하여 뷰에서 사용 가능하게 함

역할

- 메서드의 매개변수에 사용: 요청 데이터를 해당 객체에 바인딩 (form → 객체)
- 메서드에 자체에 사용: 반환 객체를 공통 모델 속성으로 등록 (모든 뷰에서 사용 가능)

사용법 (뷰에서 ${모델속성이름}으로 사용)

방법1. 컨트롤러의 요청처리메서드의 매개변수에 사용 (폼 데이터 바인딩)

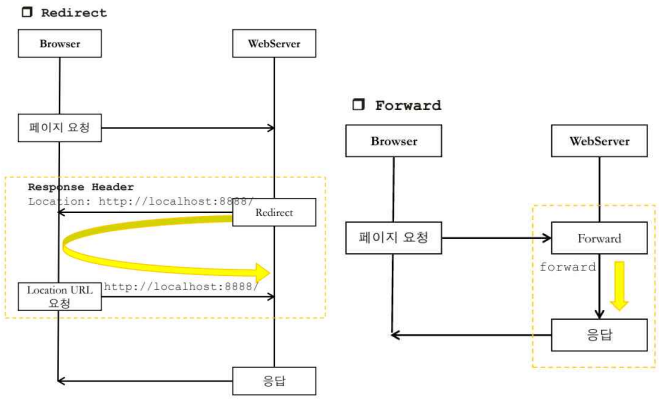
```java
public String method_name(@ModelAttribute 매개변수, Model model) {
 // model.addAttribute(...);
 return "뷰이름";
}
```

방법2. 메서드에 사용 (공통 데이터 제공)

```java
// 1. @ModelAttribute("이름") + 리턴값 사용
@ ModelAttribute("모델_속성_이름")
public String 메소드_이름() {
   return 메소드 반환값;
}

// 2.@ModelAttribute + Model 파라미터로 수동 등록
@ModelAttribute
public void 메소드_이름(Model model) {
    model.addAttribute("모델 속성 이름", "모델 속석 값");
}
```

☞ **Redirect와 Forward**

	Redirect	Forward
특징	HTTP 응답 헤더로 브라우저가 새로운 URL 이동	서버 내에서 제어권 이동
목적	페이지이동(외부포함)	서버 내 로직 분기, 화면 전환
변화	URL 변경	URL 변경 X
데이터 전달	제한적 url query string	쉽게 전달 Model, Request.
이동 범위	타 서버/도메인 가능	같은 서버 내
Spring 리턴값	redirect:/경로	forward:/경로



07장. 스프링 폼 태그 라이브러리
_@InitBinder를 이용한 커스텀 데이터 바인딩

☞ **@InitBinder**

정의

- 폼 데이터가 빈 객체에 바인딩되기 전에 개입해서 WebDataBinder를 초기화해 바인딩 설정을 조정하는 어노테이션

역할

- 데이터 바인딩 제어: 특정 필드만 허용/제외
- 바인딩 전처리: 잘못된 값 바인딩 안되게 사전 필터링

사용법

방법1. 전역 바인딩

```java
@InitBinder
public void 메소드이름(WebDataBinder binder) {
    ...
}
```

방법2. 특정 객체에만 적용

```java
@InitBinder("Java Bean 객체 이름")  // @ModelAttribute("모델이름")
public void initBookBinder(WebDataBinder binder) {
    ...
}
```

☞ **폼 파라미터의 커스텀 데이터 바인딩**

원하지 않는 폼 파라미터와 객체의 바인딩을 차단

- setAllowedFields(): 허용할 필드만 지정
- setDisallowedFields(): 차단할 필드만 지정

매개변수의 기본값은 none

```java
@InitBinder
public void initBinder(WebDataBinder binder) {
    binder.setAllowedFields("id", "name", "price");  // 이 필드만 허용
    // 또는
    binder.setDisallowedFields("admin", "role");     // 이 필드는 거부
}
```

09장. 스프링 시큐리티_스프링 시큐리티 개요

☞ **스프링시큐리티**

정의

- 사용자 정의 가능한 인증(Authentication) 및 접근 제어(Authorization) 보안 프레임워크

역할

- 사용자인증(authentication)
- 권한부여(authorization)
- 접근제어(Access Control)

☞ **스프링 시큐리티 설정 태그(security-context.xml)**

접근 권한/ 사용자 권한 제어를 위한 태그로 분류

접근 권한태그

- 허가된 사용자만 특정 페이지 접근 가능
- 로그인/로그아웃 페이지 호출 및 처리하도록 설정

사용자 권한태그

- 인증 처리를 위해 사용자 정보를 가져오는데 사용

태그명	설명
<http>	Spring Security 설정의 시작과 끝 (보안 정책 정의 영역)
<intercept-url>	보안이 적용될 URL 경로와 접근 권한(역할)을 정의
<form-login>	로그인 페이지, 성공/실패 시 이동 경로 등 폼 로그인 설정
<logout>	로그아웃 처리 및 로그아웃 성공 시 이동 경로 설정
<authentication-manager>	인증 관련 설정의 시작과 끝 (인증 관리자)
<authentication-provider>	실제 인증 처리를 담당 (사용자 정보 어떻게 검증할지 정의)
<user-service>	사용자 정보를 로딩하는 서비스 지정
<user>	정적인 사용자 정보를 정의 (아이디, 비밀번호, 권한 등)

09장. 스프링 시큐리티_접근 권한과 사용자 권한 설정

☞ **접근 권한을 설정하는 시큐리티 태그**

<http>태그

- 스프링 시큐리티 설정의 핵심인 최상위태그

<intercept-url> 태그: 접근 권한

- 접근 권한에 대한 URL 패턴 설정
- <http> 태그 안에 여러 개 설정 가능
- 선언된 순서대로 접근 권한 적용

속성	설명
pattern	ant 경로 패턴으로 접근 경로 설정 (?: 문자 1개, *: 0개 이상 문자, **: 0개 이상 디렉토리와 매핑)
access	pattern에 설정된 경로에 접근할 수 있는 사용자 권한을 설정
requires-channel	정의된 패턴 URL로 접근하면 설정된 옵션 URL로 리다이렉션 (옵션: http, https, any)

<authentication-manager> 태그: 접근 권한

- 허가된 사용자의 ID/PW 등 사용자 정보를 직접 설정

태그명	설명
<authentication-manager>	사용자 권한 인증 설정을 위한 최상위 태그
<authentication-provider>	사용자 정보 인증 요청 시 사용
<user-service>	사용자 정보 로드 시 사용 (ID, 비밀번호, 권한 등)
<user>	name, password, authorities 속성으로 사용자 정보를 표현

☞ **교차 사이트 요청 위조**
(Cross-Site Request Forgery, CSRF)

- 신뢰할 수 있는 사용자를 사칭해 웹 사이트에 원하지 않는 명령을 보내는 공격
- CSRF Token: 서버에서 발급한 고유한 인증 코드로 요청의 진위 여부를 확인
- CSRF 방지: 요청 시 CSRF Token을 포함하도록 강제

09장. 스프링 시큐리티_로그인과 로그아웃 처리

☞ **<form-login> 태그**

- 인증되지 않은 사용자가 특정 경로에 접근하거나 사용자 인증이 필요 할 때 로그인 페이지 보여줌

속성명	설명
login-page	로그인 페이지의 경로
login-processing-url	로그인 요청을 처리할 경로 (<form> 태그의 action 속성)
default-target-url	로그인에 성공하면 이동할 기본 경로
always-use-default-target	로그인 성공 후 항상 default-target-url로 이동할지
authentication-failure-url	로그인 실패 시 이동할 경로 (기본값: /login?error)
username-parameter	로그인 ID 파라미터 이름
password-parameter	로그인 비밀번호 파라미터 이름

☞ **<logout> 태그**

속성명	설명
delete-cookies	로그아웃 시 삭제할 쿠키 이름 지정 (여러 개일 경우 쉼표로 구분)
invalidate-session	로그아웃 시 세션을 제거할지 여부 (기본값: true)
logout-success-url	로그아웃 성공 시 이동할 경로 지정 (기본값: /login?logout)
logout-url	로그아웃 요청을 처리할 경로 지정
success-handler-ref	로그아웃 성공 후 처리 제어를 위한 LogoutSuccessHandler Bean 지정

10장. 파일 업로드 처리_파일 업로드의 개요

☞ **파일 업로드**

- 파일을 브라우저(클라이언트)에서 서버로 전송하고 서버에서 파일 처리
- HTTP는 multipart/form-data 프로토콜 사용

JEE Servlet 3.0 이전

- Servlet API에서 파일 업로드 지원 X
- 외부 라이브러리를 이용해 업로드 처리 (Apache Commons Fileupload)

JEE Servlet 3.0 이후

- Servlet API 파일업로드 지원
- Servlet API의 Fileupload를 이용

Sppringframework6.0 → Servlet6.0 지원

- 스프링 6.0: Third-party Fileupload 기능 지원 X
- Servlet API의 Fileupload 기능 사용

☞ **파일 업로드 설정 (web.xml에 Servlet API 이용)**

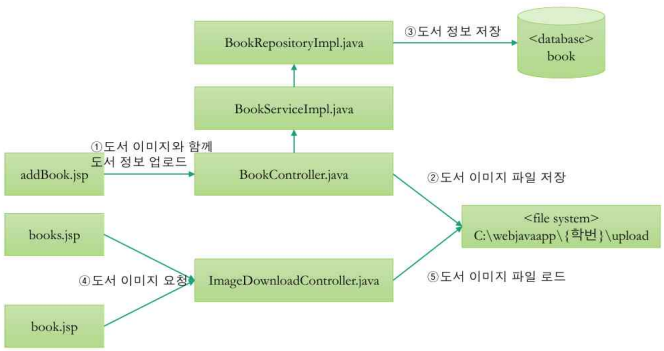
```
<servlet>
    <multipart-config>
        <!-- 업로드 파일 저장 경로 -->
        <location>C:\\webjavaapp\\{학번}\\upload</location>
        <!-- 파일의 최대 크기 (20MB) -->
        <max-file-size>20971520</max-file-size>
        <!-- multipart/form-data 전체 요청의 최대 크기 (40MB) -->
        <max-request-size>41943040</max-request-size>
        <!-- 파일이 디스크에 기록되는 크기 임계값 (20MB) -->
        <file-size-threshold>20971520</file-size-threshold>
    </multipart-config>
</servlet>

```

10장. 파일 업로드 처리 _MultipartFile을 사용한 파일 업로드

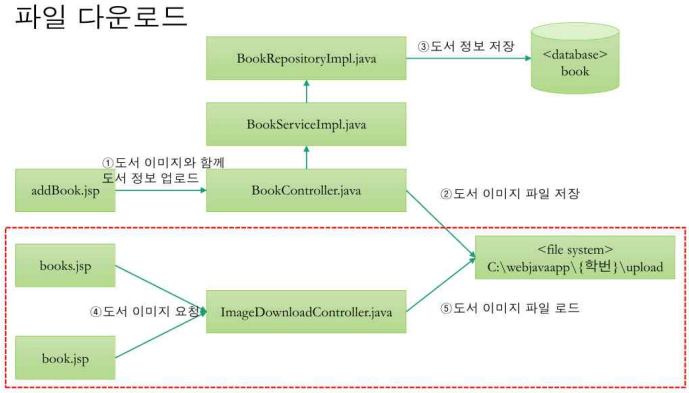
☞ **파일 업로드 경로(★★★)**

웹 애플리케이션의 경로와 무관한 경로에 저장

1) 파일 업로드가 웹 서비스에 영향을 주지 않도록함
2) 업로드한 파일에 대한 접근 제어
3) 서버 분산 환경에서도 파일 공유 가능



파일 다운로드



11장. 예외처리_예외 처리의 개요

☞ **예외 처리 방법의 종류**

어노테이션	설명
@ResponseStatus	예외 처리를 위한 가장 간단한 방법 발생한 예외를 HTTP 상태코드로 매핑시켜 응답하는 어노테이션
@ExceptionHandler	컨트롤러 메서드에서 발생하는 예외를 직접 처리할 메서드에 선언하는 어노테이션
@ControllerAdvice	여러 컨트롤러에서 발생하는 예외를 공통적으로 처리하기 위한 클래스에 선언하는 어노테이션

☞ **우선순위(★★★)**

1)Controller에적용된@ExceptionHandler
2)공통적으로적용된@ControllerAdvice
3)Exception에적용된@ResponseStatus

11장. 예외처리_ResponseStatus를 이용한 HTTP 상태 코드 기반 예외 처리

☞ **@ResponseStatus를 이용한 예외 처리**

- 함께 사용해야함

```
// @ResponseStatus + 사용자 정의 예외 클래스
@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "찾을 수 없습니다")
public class Example02Exception extends Exception {
    public Example02Exception(String message) {
        super(message);
        System.out.print(message);
    }
}
// 컨트롤러에서 예외 발생
@Controller
public class Example02Controller {
    @GetMapping("/exam02")
    public void handleRequest() throws Exception {
        throw new Example02Exception("Example02Exception 메시지입니다");
    }
}
```

11장. 예외처리_@ExceptionHandler를 이용한 컨트롤러 기반 예외처리

☞ **@ExceptionHandler를 이용한 예외처리**

- 웹 요청에 따라 컨트롤러의 요청 처리 메소드를 실행하는 동안 예외가 발생하면 이를 처리하기 위해 예외 처리 메소드에 사용

11장. 예외처리

_@ControllerAdvice를 이용한 전역 예외처리

☞ 전역 예외처리를 위한 @ControllerAdvice

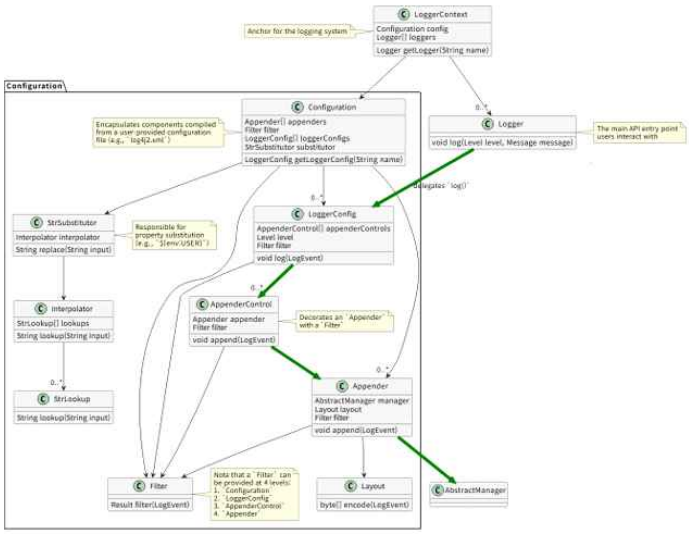
- 여러 컨트롤러에서 발생하는 예외를 공통으로 처리

- 산업용 Java 로깅 프레임워크
- 다양한 사용 사례에 대한 배포를 지원하는 API, 구현체로 구성
- Apache에서 관리하는 프로젝트
- 자원봉사팀과 대형커뮤니티의 지원으로 유지관리

☞ Log4j 1.x

- 관리 위원회 2015년 8월5일 Log4j1.x 버전 지원 중단
- 원인: 여러 가지 보안 위협이 발견 됨

☞ Log4j2 구조



요소	설명
Logger	로깅을 위한 기본 사용자 진입점. 로그를 생성하는 핵심 인터페이스.
Filter	로그 이벤트의 매개변수를 평가하여 로깅 호출을 허용할지 결정하는 필터. 필요 시 파이프라인에서 추가 처리를 수행.
Appender	로그 이벤트를 특정 대상 에 전달 (콘솔, 파일, DB 등)
Layout	로그 이벤트를 지정된 형식으로 변환해 Appender가 처리할 수 있도록 함. (텍스트, JSON 등)

☞ Log4j2 설정

src/main/resources/log4j2.xml

```
<Loggers>
    <Root level="INFO" additivity="false">
        <AppenderRef ref="console" />
    </Root>
    <Logger name="com.springmvc" level="DEBUG" />
    <Logger name="org.springframework.web" level="DEBUG" />
</Loggers>
```

☞ Slf4j

- 다양한로깅프레임워크(logback, log4j등)에 대한 간단한 추상화 역할을 제공
- 배포 시점에 원하는 로깅 프레임워크를 플러그인
- 스프링에서 기본 로깅 API로 사용

☞ 로깅레벨

레벨	설명
ERROR	기타 런타임 오류 또는 예기치 않은 상태
WARN	바람직하지 않거나 예기치 않은 런타임 상황의 경고성 메시지
INFO	시작, 종료 같은 런타임 이벤트 메시지
DEBUG	디버그 용도로 시스템 흐름에 대한 자세한 정보
TRACE	처리 흐름을 추적할 수 있는 메시지

12장. 인터셉터_인터셉터 개요

☞ 인터셉터

- 사용자가 URL 요청 시, 컨트롤러의 메소드 호출 전/후에 흐름을 가로채 특정 작업을 할 수 있는 기능

목적

- 인증, 권한 확인, 로깅, 공통 처리 로직 삽입 등

기능

- Spooling, 중복 제출 방지, 파일 업로드 처리, 요청 로깅(Logging), 유효성 검사(Validation)

12장. 인터셉터

_인터셉터를 이용한 로그 기록 만들기

☞ HandlerInterceptor 인터페이스

HttpServletRequest(요청)&HttpServletResponse(응답)을 가로채 로직을 처리하도록 지원하는 인터페이스

1. preHandle(): 컨트롤러 호출 전

- HandlerAdapter가 컨트롤러로 호출하기 전
- 로깅, 인증 체크 등을 수행
- true 반환 시 다음 단계로 진행

2. postHandle(): 컨트롤러 호출 후, 뷰 렌더링 전

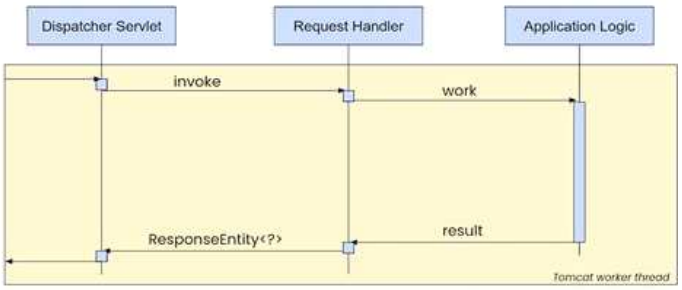
- 핸들러 메서드가 실행된 후 호출, 뷰 렌더링 전
- ModelAndView 객체를 조작 => 추가 데이터 뷰에 전달 가능

3. afterCompletion(): 전체 요청 완료 후

- 뷰 렌더링 후, 전체 요청 처리 완료 후
- 리소스 정리, 로그 기록, 예외 처리 수행
- ex: Handler 실행 중 예외가 발생했다면 예외 객체

☞ Asynchronous Processing

1. 동기화 처리



- 서버 스레드가 요청 처리 전체 과정을 끝까지 점유
- Dispatcher Servlet → Request Handler → Application Logic 순 흐름
- 모든 작업이 끝난 후 응답이 클라이언트에 반환

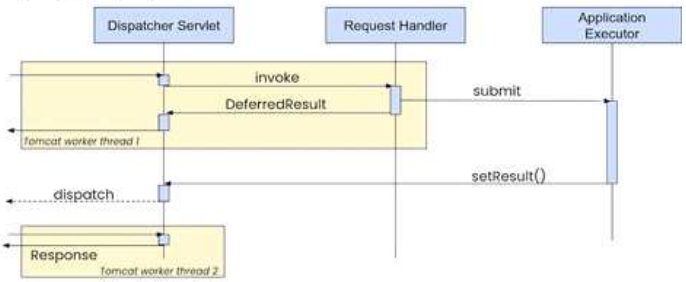
특징

- 서버 스레드는 처리 완료될 때까지 계속 대기
- 작업이 오래 걸리면 다른 요청도 대기 상태

예: SQL 실행, 외부 API 요청, 파일 업로드

구분	동기 처리
스레드 점유	요청 → 응답까지 계속 점유
응답 시점	작업 끝난 후 응답
효율성	비효율적 (스레드 낭비)

2. 비동기적 처리



- 스레드1이 요청을 받으면, DeferredResult를 통해 비동기적 처리 후 바로 반환 (스레드 반납)
- 실제 작업은 Application Executor가 별도 처리
- 작업 완료 시 setResult() → 응답 준비
- 새로운 스레드2가 응답을 클라이언트에 전송

특징

- 요청-응답 사이의 처리 시간 동안 스레드 점유 X
- 서버 스레드 효율적으로 사용, 다른 요청 동시 처리

구분	비동기 처리
스레드 점유	요청 처리 맡기고 → 스레드 반환
응답 시점	작업 완료 후 스레드 다시 할당해 응답
효율성	효율적 (동시 요청 처리 가능)

☞ java.lang.ThreadLocal

항목	설명
의미	스레드 별로 고유한 로컬 변수 제공 (스레드전용변수)
용도	멀티스레드 환경에서 공유되지 않아야 할 값을 스레드마다 안전하게 보관할 때 사용
접근 방식	set(), get() 메서드로 값 저장 및 조회 (스레드마다 독립적으로 작동)
귀속 대상	일반 변수: 객체 인스턴스에 귀속 ThreadLocal변수: 스레드에 귀속
사용 예	사용자 인증 정보, 트랜잭션 ID, 로깅 컨텍스트 등 스레드별로 달라야 하는 값 저장

특징

- 스레드별 할당된 변수 공간
- 스레드 안전성 보장
- 객체가 아닌 스레드에 귀속
- 공유 방지, 동기화 불필요
- private static final로 선언

13장. 다국어 처리_다국어 처리의 개요

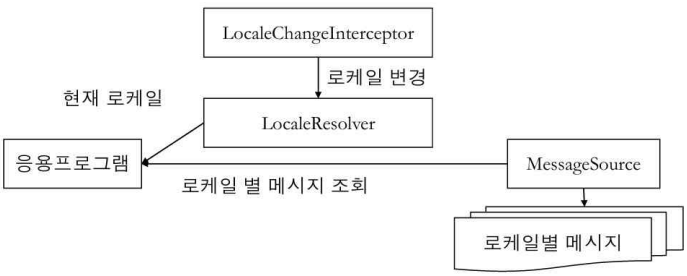
☞ **Spring에서 다국어 처리 방법**

MessageSource

- 화면에 출력할 메시지를 쉽게 다국어로 표현 가능
- 언어별 메시지 코드와 메시지를 정의

LocaleResolver & LocaleChangeInterceptor

- 사용자가 웹에서 원하는 언어를 자유롭게 선택,변경



13장. 다국어 처리

_MessageSource를 이용한 다국어 처리

☞ **메시지리소스(ResourceBundle) 파일작성**

***[언어코드[_국가코드]].properties**

- key=value 쌍
- key: 뷰 페이지에서 메시지를 참조하는데 사용

파일 이름 형식	설명
파일이름.properties	리소스 파일 없을 때 기본으로 사용
파일이름_ko.properties	언어 코드: 한국어
파일이름_en.properties	언어 코드: 영어
파일이름_en_UK.properties	언어 코드: 영어 국가 코드: 영국
파일이름_ja.properties	언어 코드: 일본어

☞ **메시지리소스(ResourceBundle) 파일 위치**

- **src/main/resources** 폴더에 작성
- ISO-8859-1 인코딩을 지원 UTF-8도 지원 가능
- ISO형식: 한글은 유니코드로 변환되어 저장

☞ **뷰 페이지에 메시지 출력**

- 뷰 페이지 상단 스프링태그 라이브러리 선언

```
<%@ taglibprefix="spring" uri="http://www.springframework.org/tags" %>
```

- <spring:message>태그 사용해 출력

속성명	설명
arguments	메시지 출력을 위한 인자들을 전달
argumentSeparator	인자들을 구분할 구분자 (기본값: 쉼표 `,`)
code	출력할 메시지의 키 (지정안하면 text 속성 값 출력)
htmlEscape	HTML 특수문자를 이스케이프 처리 (기본값: false)
javaScriptEscape	JS 특수문자를 이스케이프 처리 (기본값: false)
message	유효성 검사 후 오류 메시지 간단하게 표시
scope	var로 저장할 때 변수의 범위 (page, request, session, application 중 하나)
text	지정한 code의 메시지를 못 찾았을 때 출력할 기본 텍스트
var	메시지를 변수에 저장할 경우 사용할 변수 이름 (scope와 함께 사용)

☞ **메시지 출력에 인자 사용하기**

메시지정의

- 메시지에서 인자가 삽입될 부분을 {n}형식으로 선언
- n: 0부터 시작하는 인자의 순번
- 언어에따라 출력되는 파라미터 순서가 달라질 수도
- addBook.form.param=파라미터 {0} 와 {1}

JSP에서메시지인자전달및출력

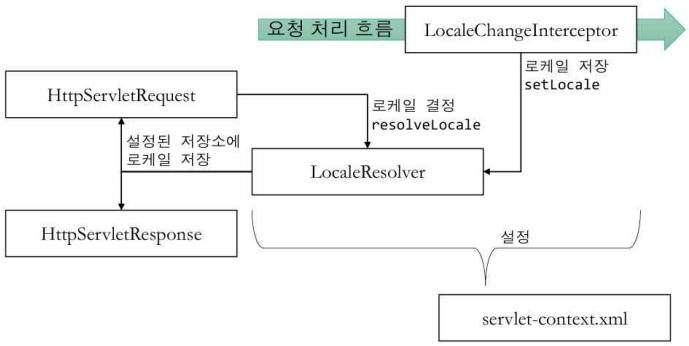
```
<spring:message code="addBook.form.param"
                arguments="Banana,Apple"/>
```

출력되는 메시지

파라미터 Banana 와Apple

13장. 다국어 처리 _LocaleResolver & LocaleChangeInterceptor를 이용한 다국어 변경

☞ **Locale 결정과 변경**



☞ **LocaleResolver 환경설정**

- 환경정보를 이용해 로케일을 결정,저장하는 인터페이스
- **class="org.springframework.web.servlet.i18n.구현체**

```
1 <beans:bean id="localeResolver"
2 class="org.springframework.web.servlet.i18n.LocaleResolver 구현체">
3     <beans:property name="defaultLocale" value="언어코드" />
4 .....
```

유형(구현체)	설명
AcceptHeader LocaleResolver	브라우저의 Accept-Language 헤더 값을 기준으로 로케일을 결정함 (기본값)
Cookie LocaleResolver	쿠키로 로케일 정보 사용 사용자 지정 로케일,표준 시간대
Session LocaleResolver	세션에 로케일을 저장하여 유지 세션 locale 속성, 요청 accept-header
Fixed LocaleResolver	특정 로케일 지정 항상 고정된 기본 로케일 선택적으로 시간대 반환