

5. 스프링 MVC

스프링 MVC 설계

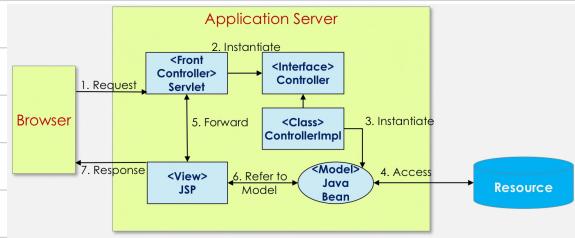
- 프론트 컨트롤러 패턴에 기반한 MVC 구조화된 설계

- 자바 웹 애플리케이션(모듈 1, 2, 프론트 컨트롤러)
- 모델, 보, 컨트롤러 각각 인터페이스 정의, 서로 협력에 의존성 X → 액션 경로로 구조화된 유연하고 확장성이 사용
- 다양한 서비스 패티 구조화된 예제 지원
 - Jackson, Google Gson 등
 - JasperReports, Apache POI 등
- 데이터베이스로 스프링 MVC 활용 예제됨.
 - ① Controller, ② RequestMapping 등으로 요청 간단

~~프론트 컨트롤러 패턴~~

프론트 컨트롤러 패턴 (스프링 MVC)

- 클라이언트 요청을 별도의 프론트 컨트롤러에 합동
- 모든 요청의 공통부분을 별도(프론트 컨트롤러)로 처리.



프론트 컨트롤러 패턴 구조

DispatcherServlet

실행 프로세스

- ① DispatcherServlet이 HTTP 요청 처리
- ② DS는 서블 컨트롤러로 HTTP 위임 요청
- ③ 서블 컨트롤러는 클라이언트 요청 처리를 위해 DAO 객체 호출.
- ④ DAO 객체는 DAO를 액세스하여 Model 객체를 생성 후 요청 결과 리턴.
- ⑤ DS는 처리 결과에 저장하고 뷰에 환경변수 요청
- ⑥ 선택된 뷰는 환경에 모델 객체를 가져와 처리
- ⑦ HTTP 응답

~~스프링~~ MVC 구조 구현요소

~~프론트~~ 컨트롤러 패턴
~~프론트~~ DispatcherServlet

HandlerMapping

컨트롤러 담당 클라이언트 요청 대신 Controller에게

클라이언트 요청 처리, 각각 결과값을 View에게 전달하기 때문에 응답 생성.

URL과 요청정보를 기준으로 어떤 컨트롤러를 실행하기 결정하는 객체

DSL은 하나 이상의 핸들러 디렉션을 가질 수 있고 어떤 템플릿 이용할 때는

MVN: annotation-driven 디자인

Controller

클라이언트의 요청 처리한 뒤 결과를 DSL에게 알려줄

Model

컨트롤러가 뷰에게 넘겨줄 데이터 저장하기 위한 객체

ViewResolver

Controller가 처리한 뷰 이름 기반으로 Controller 처리 결과 생성하는 설정

View

Controller의 처리 결과 HTML을 생성함.

스프링 스토리오 타입 어노테이션 (Spring Stereotype Annotation)

- ① Component, ② Service, ③ Controller, ④ Repository 는 스프링이 관리하는
컴포넌트를 나타내는 일반적인 타입

- MVC 아키텍처에서 Service, Presentation, Persistence 계층이 분리한 컴포넌트는 ② Service, ③ Controller,
④ Repository 대신은 보통 이름으로 구현됨
- ① Service, ② Controller, ④ Repository는 ① Component의 확장형 타입임.

① Component 일반적인 컴포넌트

② Repository Persistence 계층 컴포넌트

③ Service Business (Service) 계층 컴포넌트

④ Controller Presentation 계층 컴포넌트

⑤ RestController Controller + ④ ResponseBody (스프링 4.0 이후)

2) 이전의 예제.

- Spring-webmvc 설정하면 스프링 웹(Spring - web)과 기타 스프링 프레임워크
으로 보는 Spring-context)에 대한 의존 관계도 추가 처리.
- 스프링 MVC Bean Validation 설정(hibernate-validator)을 이용해 입력값 유효성 검증.

웹 앱 컨테이너 등록 예제. (web.xml)

- 1) ContextLoadListener 클래스 root-context.xml
 - 서버로 커스터마이징(Tomcat)으로 ContextLoadListener 클래스를
 - 서비스 계층 어노테이션(@Service, @Repository)을 적용하기 위한 클래스
- 2) DispatcherServlet 클래스 servlet-context.xml
 - 서버로 커스터마이징 커스터마이징(DS 설정) 등록
 - 컨트롤러(@Controller or @Component) 를 통제하기 위한 클래스

CharacterEncodingFilter 예제.

- 한글과 사용 가능한 문자 인코딩 설정 예제.



컨트롤러 기본 구조

- 모든 컨트롤러 클래스에는 @Controller 어노테이션
- 매서드별 @RequestMapping 사용하여 URL 매핑 처리.
- @RequestParam은 HTTP 요청 파라미터를 Bind 처리하여 결과물을 사용하는 View 이용.

컨트롤러 흐름

- servlet-context.xml에서 컨트롤러 등록.

DispatcherServlet 클래스 / 흐름.

- <annotation-driven />

→ 스프링 MVC 아키텍처 패턴의 컨포넌트 빈 정의가 자동으로 추가

→ 퍼미션 설정에서 차운 @Controller 라는 URI 맵핑.

- <context: component-scan base-package = "org.kpu.web.controller" />

→ base-package 내부의 클래스에서 @Controller 지정된 컨트롤러를 검색하여 빈을 등록.

<annotation-driven />
// 정적 리소스 파일(CSS, 이미지, 자바스크립트) 액세스 설정
resources mapping="/resources/**" location="/resources/" />

<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
 <beans:property name="prefix" value="/WEB-INF/views/" /> 경로
 <beans:property name="suffix" value=".jsp" /> 경로
</beans:bean>

<context:component-scan base-package="org.kpu.web.controller" />

뷰의 파일 위치 설정 : /WEB-INF/view/뷰이름.jsp

@RequestMapping → 비밀 헤더의 주거 가능

- URL과 컨트롤러 매핑의 매핑을 설정하는 어노테이션.

- 경로 값 Value - URL (요청 경로)

method - HTTP 메소드

params - 요청 파라미터 유무와 파라미터 값

컨트롤러 매핑 DMSR

주의점!!

- Model 속성

- ① Model Attribute 의 값을 가져온다

→ http 요청 파라미터 적용 범위로 값이 넘침.
Query String.

→ 해당 값은 View 기록에 저장해서 전송됨.

- Matrix Variable

- Request Body.

- ② Path Variable

→ URL에서 Query String 값이 아닌 Path 헤더의 변수값이 적용 범위로 값이 넘침.

→ 해당 값은 View 기록에 저장해서 전송됨.

- ③ Request Param

→ http 요청 파라미터의 값을 적용 범위로 값이 넘침

→ 해당 값은 View 기록에 저장해서 전송됨.

Local 헤더에서 사용 가능

| REST 아키텍처 | 리소스 표시 → JSON (JavaScript Object Notation) | XML Jackson - databind 사용하는 JSON과 자바와 서로 간의 자료형 |
|--|--|---|
| (Representational State Transfer) | | |
| - 클라이언트가 서버 사이에 데이터 전송 양을 위한 아키텍처 표준 | | |
| - 웹상의 정보를 리소스로 표기하고 그 상태와 URI를 통해 고유하게 통장함 | | |
| - HTTP 프로토콜을 사용해 리소스에 접근 | | |

HTTP Method

| | |
|--------|---|
| GET | URL에 지정된 리소스 가져옴 ex) 사용 경로 REST |
| POST | 리소스를 생성하고 생성된 리소스에 접근하는 URI 생성 ex) 사용자 등록 REST. |
| PUT | URI에 지정된 리소스 생성하거나 갱신. |
| DELETE | URI에 지정된 리소스 삭제. |

REST 구조화에 사용되는 애플리케이션

- ① Rest Controller
 - REST API를 제공하는 컨트롤러
 - ① Controller 와 ② ResponseBody 같이 쓰임.
- ② Request Body
 - 컨트롤러가 Directly 받아들이 ② Request Body가 어노테이션된 경우

스프링은 요청한 HTTP request body를 처리하는 데에 넘겨짐.
- ③ ResponseBody
 - 컨트롤러가 ③ ResponseBody로 어노테이션된 경우,

스프링은 뒷한 값을 넣는 HTTP response body에 넘김.
 - 스프링은 요청한 MIME 타입의 HTTP 헤더에 Orae Content-Type 기반으로 HTTP Message converter를 사용하여 뒷한 값을 HTTP response body로 변환함.
- ④ Response Entity
 - 전체 응답
 - statusCode, headers, body 3개가 속성을 가지게됨.

XML 뿐만 REST 커트롤러

- JAXB2RootElementHttpMessageConverter.

→ XML 형태의 HTTP 데이터를 JSON로 변환

Spring MVC에서의 예외처리.

- 커트롤러에서도 예외 처리.

→ 커트롤러의 ModelAndView에서 예외가 발생했을 때의 처리를 정의.

→ 컨트롤러 예외 처리는 딱히 정의하고 아니면 `@ExceptionHandler` 설정.

- 브라우저의 웹 애플리케이션은 공통된 예외 처리.

→ 브라우저 컨트롤러에서 사용할 수 있는 공통된 예외 처리 클래스 정의.

→ 공통된 예외 처리 클래스를 정의하고 그 클래스에 `@ControllerAdvice` 설정.

MemberControllerAdvice.java

```
@Component
@ControllerAdvice
public class MemberControllerAdvice {
```

```
@ExceptionHandler(DataNotFoundException.class)
public String handleDataException(DataNotFoundException e) {
    return "member/not_found";
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>DataNotFoundException</title>
</head>
<body>
    <h1>DataNotFoundException</h1>
    <c:url value="/member/list" var="url"/>
    <a href="${url}">학생목록 화면가기</a>
</body>
</html>
```

Not_found.jsp

Not_found.jsp

File Tree:

- src/main/java
- src/main/resources
- src/main/webapp
- src/main/webapp/WEB-INF
- src/main/webapp/WEB-INF/web.xml
- src/main/webapp/WEB-INF/spring
- src/main/webapp/WEB-INF/spring/service-context.xml
- src/main/webapp/WEB-INF/spring/spring-beans.xml
- src/main/webapp/WEB-INF/spring/spring-mvc.xml
- src/main/webapp/WEB-INF/web.xml
- src/main/webapp/member
- src/main/webapp/member/not_found.jsp
- src/main/webapp/member/list.jsp
- src/main/webapp/member/login.jsp
- src/main/webapp/member/logout.jsp
- src/main/webapp/member/register.jsp
- src/main/webapp/member/search.jsp

ControllerAdvice 예제

`<context:component-scan base-package = "org.kpu.web.exception"/>`

- `@Component` 설정은 component-scan은 기본적으로 풀고

- `@ControllerAdvice`는 예외처리 설정.

- `@ExceptionHandler`은 특정 예외 처리 설정.