**[ 08 ] MVC 기초**

1. 스프링 MVC 기본 흐름과 주요 컴포넌트

스프링 MVC는 여러 주요 컴포넌트가 맞물려 동작하기 때문에, 스프링 MVC를 이용해서 웹 어플리케이션을 개발하려면 적어도 MVC가 어떤 식으로 동작하는지 이해하고 있어야 한다. 이에 대한 이해가 부족하면 문제를 해결하는 좋은 방법을 찾을 수 없게 된다

최초에 웹 브라우저에서 요청이 들어오면 DispatcherServlet이 요청을 받으며 아래의 그림과 같은 과정을 거쳐 웹 브라우저에 응답을 전송하게 된다

<<spring bean>>

HandlerAdapter

<<spring bean>>

HandlerMaping

**Model & View**

<<spring bean>>

**Controller**

DispatcherServlet

Client

Service

**View(.jsp)**

<<spring bean>>

ViewResolver

DAO

< 스프링 MVC의 웹 요청 처리과정 >

* 1. 웹브라우저를 통해 클라이언트의 요청이 들어오면 무조건 맨 처음 DisptcherServlet에서 요청을 받는다.
  2. 요청된 내용을 HandlerMapping에게 작업을 보내 요청 URL과 매칭되는 컨트롤러를 검색해 준다.
  3. 요청된 내용의 처리를 컨트롤러에게 요청한다
  4. 요청된 내용을 Controller에게도 보내고 요청된 내용에 따라 다른 일을 하고, 컨트롤러 샐행결과를 ModelAndView로 변환해서 리턴한다
  5. 컨트롤러의 실행결과를 보여줄 View를 검색하기 위해 ViewResolver에게도 보낸다.
  6. 요청한 결과의 View로 응답 생성을 한다.

우리가 작업을 많이 할 부분은 Controller, View, Model이고 나머지 부분은 Sprring에서 거의 자동으로 다루어준다.

요청이 들어오면 DispatcherServlet 이 먼저 받고, Controller에게 요청을 한다. Controller에서 다시 DispatcherServlet으로 뭔가 응답을 준다. 그러면 DispatcherServlet에서 ViewResolver를 통해 View에 호출이 된다. 그러면 사용자에게 응답이 이루어 진다.

1. 스프링 MVC 일단 해보기
   1. 스프링 MVC 프로젝트로 새 프로젝트 생성
   2. 스프링 MVC 프로젝트 생성시 디폴트 패키지명의 XX.XX.XX시 세번째 것이 그 프로젝트의 컨텍스트명이 된다(디폴트 패키지명은 꼭 디렉토리 3개이상. 세번째 디렉토리는 컨텍스트명이 되므로 유일한 값으로)
   3. web.xml에 한글처리 추가

<filter>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

<init-param>

<param-name>forceEncoding</param-name>

<param-value>true</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

* 1. 컨트롤러에 다음 추가

@RequestMapping(value = "/", method = RequestMethod.***GET***)

**public** String home(Model model) {

model.addAttribute("greeting", "안녕하세요");

**return** "home";

}

1. 스프링 MVC 구조 살펴보기

※ 스프링 MVC 프로젝트 템플릿의 구조

₪ src/main/java - 개발되는 Java 코드의 경로

₪ com.ch.ch08-1

* HomeController.java - **컨트롤러 : Dispatcher에서 전달된 요청을 처리**

₪ src/main/resources - 서버가 실행될 때 필요한 파일들의 경로

₪ src/test/java - 테스트 전용 경로 (각 테스트 코드 작성 경로)

₪ src/test/resources - 테스트 시에만 사용되는 파일들 경로

₪ JRE System Library

₪ Maven Dependencies

₪ src

₪ main

₪ webapp - web root

₪ resources

₪ WEB-INF

₪ classes

₪ spring - 스프링 설정 파일의 경로

₪ appServlet

>servlet-context.xml - 스프링 컨테이너 설정 파일

₪ root-context.xm - 스프링 컨테이너 설정 파일

₪ views - 뷰(.jsp)

₪ home.jsp

web.xml - Maven의 설정 파일 DispatcherServlet 서블릿 맵핑.

스프링 설정 파일(servlet-context.xml, root-context.xml) 위치 정의

* 1. 소스에서의 정리
     + Web.xml에서 /으로 요청이 들어오면 org.springframework.web.servlet.DispatcherServlet으로 보내라고 했으나, 스프링 프레임워크의 클래스이기 때문에 내가 신경쓸 일이 없다. DispatcherServlet는 작업이 끝난 후 알아서 Controller로 보낸다.
     + Servlet-context.xml의 <context:component-scan base-package="com.ch.ch08\_1" />에 명시된 com.ch.ch08\_1 패키지에 @Controller라고 명시된 클래스만 Controller로 본다.

1. Resources 폴더

우리가 웹사이트를 만들다 보면 이미지도 있을 수 있고, js이나 css도 있을 수 있다. 그런 것들은 어떻게 들어가는지 살펴보자.

우선 이미지를 예제로 본다

* 1. 이미지를 webapp아래의 resources폴더에 이미지를 copy해서 넣고, home.jsp에 “<p> <img src="/ch08\_1/resources/h.jpg"></p>”추가하고 실행하면 이미지가 보인다.
  2. webapp아래에 img라는 폴더를 만들고 이미지를 copy해서 넣고 home.jsp에 “<p> <img src="/ch08\_1/img/h.jpg"></p>” 추가하고 실행하면 이미지가 보이지 않는다. 다음과 같은 순서에 따라 이미지가 보이도록 해야 보일 수 있다.
     + web.xml에 있듯이, '/'가 들어오면 DispatcherServlet으로 보내고 그 DispatcherServlet은 컨트롤러로 보낸다.
     + 컨트롤러를 찾는 것은 servlet-context.xml에 나온 패키지 중 @Controller라는 어노테이션이 있는 곳을 컨트롤러로 본다.
     + 그안의 @requestMapping(요청을 매핑하는 어노테이션)대로 메소드를 수행하고 "home"을 뷰의 이름으로 리턴.
     + servlet-context.xml의 prefix와 surfix로 뷰의 경로로 찾아가 실행한다.
     + "/"로 시작하는 경우, 요청이 들어온 건 DispatcherServlet이 가로채라라고 했으므로, DispatcherServlet에서 컨트롤러에게 보낸다. 그러나, 제외시키는 코드가 servlet-context.xml에 명시해 놓는다.

<!-- Handles HTTP GET requests for /resources/\*\* by efficiently serving up static resources in the ${webappRoot}/resources directory -->

<resources mapping="/resources/\*\*" location="/resources/" />

<resources mapping="/img/\*\*" location="/img/" />

* + - 위의 코드에 의해 /resources/로 시작하는 것과 /img/로 시작하는 것은 경로 그대로 읽게 한다

**Controller.java**

@Controller

**public** **class** HomeController {

**private** **static** **final** Logger ***logger*** = LoggerFactory.*getLogger*(HomeController.**class**);

@RequestMapping(value = "/", method = RequestMethod.***GET***)

**public** String home(Locale locale, Model model) {

***logger***.info("Welcome home! The client locale is {}.", locale);

Date date = **new** Date();

DateFormat dateFormat = DateFormat.*getDateTimeInstance*(DateFormat.***SHORT***, DateFormat.***SHORT***, locale);

String formattedDate = dateFormat.format(date);

model.addAttribute("serverTime", formattedDate );

model.addAttribute("greeting", "Hello, Spring");

**return** "home";

}

}

**style.css**

@CHARSET *"UTF-8"*;

**body** { background-color:*lightyellow*; 굴림체; text-align: *center*; }

**home.jsp**

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>

<%@ page session=*"false"* %>

<html>

<head>

<title>Home</title>

<link href=*"css/style.css"* rel=*"stylesheet"*>

</head>

<body>

<h1>

${greeting }

</h1>

<P> The time on the server is ${serverTime}. </P>

<img alt=*"A image"* src=*"resources/a.jpg"*><br><br>

<img alt=*"A image"* src=*"img/a.jpg"*>

</body>

</html>

**Servlet-context.xml**

<resources mapping=*"/resources/\*\*"* location=*"/resources/"* />

<resources mapping=*"/img/\*\*"* location=*"/img/"*/>

<resources mapping=*"/css/\*\*"* location=*"/css/"*/>

<beans:bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<beans:property name=*"prefix"* value=*"/WEB-INF/views/"* />

<beans:property name=*"suffix"* value=*".jsp"* />

</beans:bean>

<context:component-scan base-package=*"com.tj.ch08mvc"* />