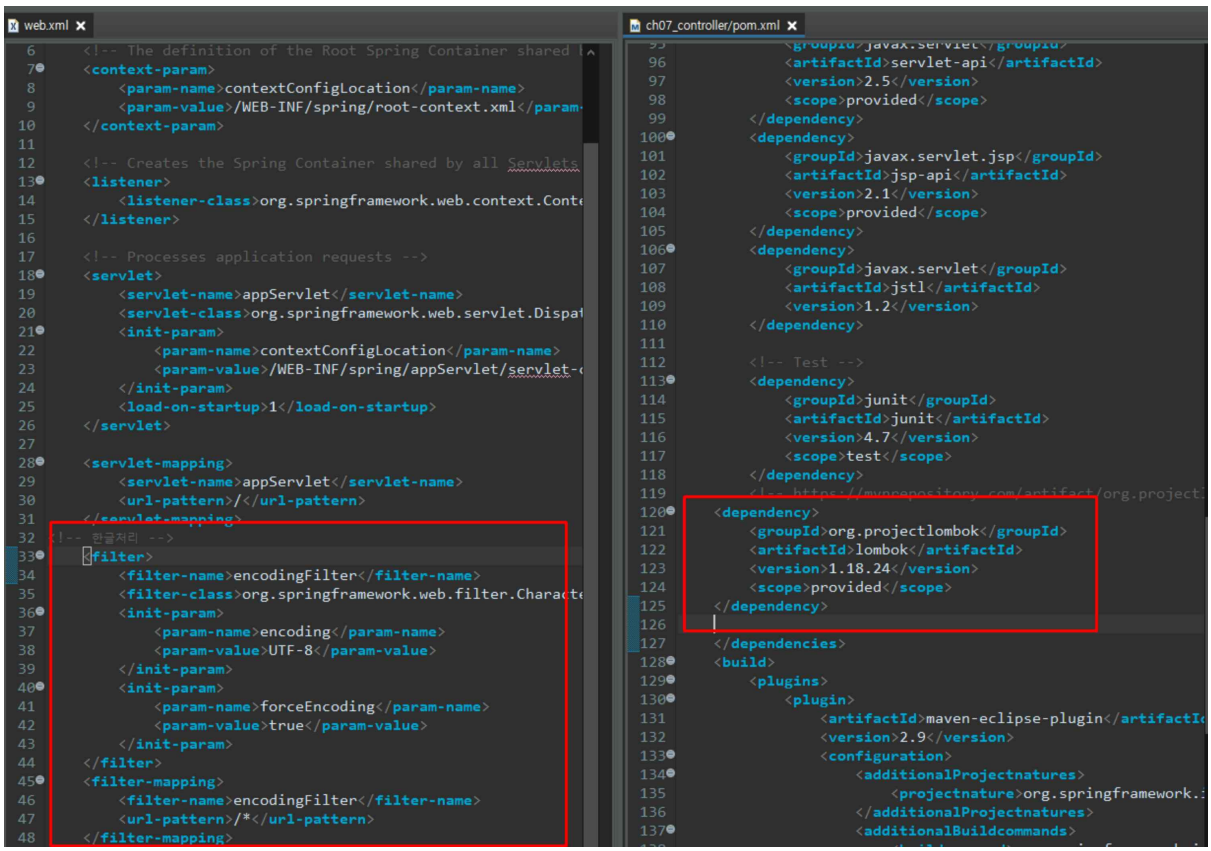
[07] 컨트롤러

1. 컨트롤러 클래스 제작

(1) 최초 클라이언트로부터 요청이 들어왔을 때, (DispatcherServlet이 매핑이 되어 있으므로, DispatcherServlet이 받고 HandlerMapping과 HandlerAdapter를 통해) 컨트롤러로 진입하게 되고, 컨트롤러는 요청에 대한 작업을 한 후 (ViewResolver를 통해 뷰를 찾아) 뷰 쪽으로 데이터를 전달한다. 그러면 뷰로 인해 화면에 응답을 하는 구조로 되어 있다.

Xml에 Lombok , 한글처리 추가



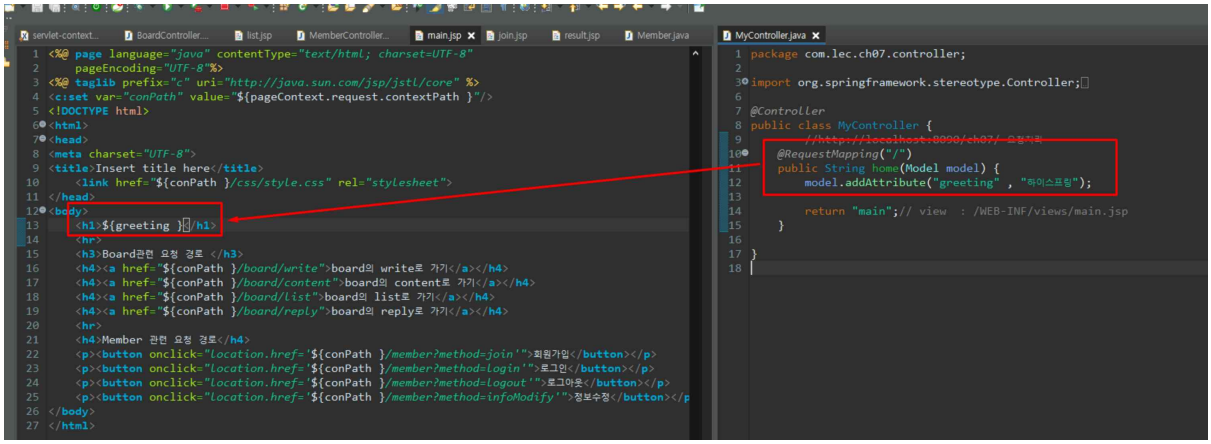
1 . webapp

-css 폴더 파일 만들기

2.spring

-appServlet 밑에 servlet-context.xml 에 css 가져오기? 추가

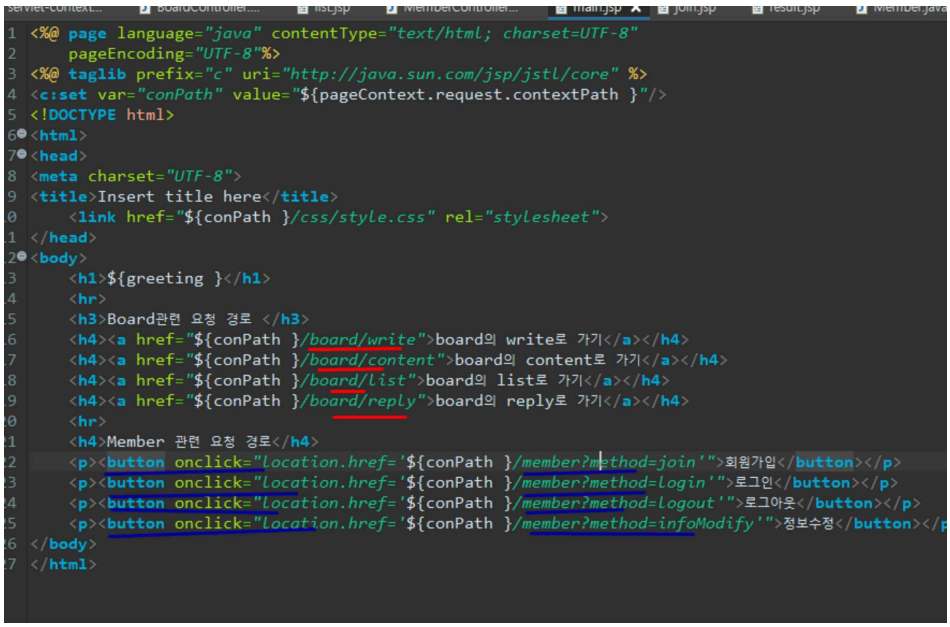
순서



```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="conPath" value="${pageContext.request.contextPath }"/>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
    <link href="${conPath }/css/style.css" rel="stylesheet">
</head>
<body>
    <h1>${greeting }</h1>
    <hr>
    <h3>Board관련 요청 경로 </h3>
    <h4><a href="${conPath }/board/write">board의 write로 가기</a></h4>
    <h4><a href="${conPath }/board/content">board의 content로 가기</a></h4>
    <h4><a href="${conPath }/board/list">board의 list로 가기</a></h4>
    <h4><a href="${conPath }/board/reply">board의 reply로 가기</a></h4>
    <hr>
    <h4>Member 관련 요청 경로</h4>
    <p><button onclick="location.href='${conPath }/member?method=join'">회원가입</button></p>
    <p><button onclick="location.href='${conPath }/member?method=login'">로그인</button></p>
    <p><button onclick="location.href='${conPath }/member?method=logout'">로그아웃</button></p>
    <p><button onclick="location.href='${conPath }/member?method=infoModify'">정보수정</button></p
</body>
</html>
```

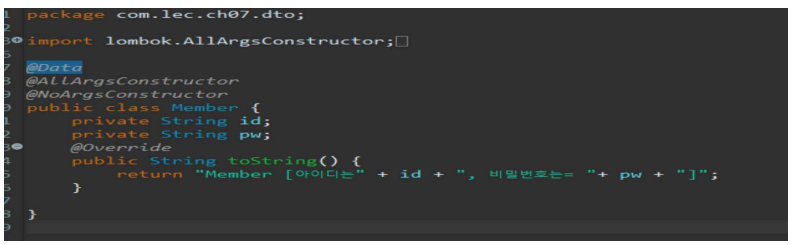
```java
package com.lec.ch07.controller;

import org.springframework.stereotype.Controller;

@Controller
public class MyController {
    //http://localhost:8090/ch07/ 요청지점
    @RequestMapping("/")
    public String home(Model model) {
        model.addAttribute("greeting" , "하이스프링");

        return "main";// view  : /WEB-INF/views/main.jsp
    }

}
```

1. MyController - > main.jsp (“greeting” “하이스프링” 출력)




```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="conPath" value="${pageContext.request.contextPath }"/>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
    <link href="${conPath }/css/style.css" rel="stylesheet">
</head>
<body>
    <h1>${greeting }</h1>
    <hr>
    <h3>Board관련 요청 경로 </h3>
    <h4><a href="${conPath }/board/write">board의 write로 가기</a></h4>
    <h4><a href="${conPath }/board/content">board의 content로 가기</a></h4>
    <h4><a href="${conPath }/board/list">board의 list로 가기</a></h4>
    <h4><a href="${conPath }/board/reply">board의 reply로 가기</a></h4>
    <hr>
    <h4>Member 관련 요청 경로</h4>
    <p><button onclick="location.href='${conPath }/member?method=join'">회원가입</button></p>
    <p><button onclick="location.href='${conPath }/member?method=login'">로그인</button></p>
    <p><button onclick="location.href='${conPath }/member?method=logout'">로그아웃</button></p>
    <p><button onclick="location.href='${conPath }/member?method=infoModify'">정보수정</button></p
</body>
</html>
```


2. 두가지 방법이 있다 .



```java
package com.lec.ch07.dto;

import lombok.AllArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Member {
    private String id;
    private String pw;
    @Override
    public String toString() {
        return "Member [아이디는" + id + ", 비밀번호는= "+ pw + "]";
    }

}
```

@Data : 게터 셋터 toStirng equls

@AllArgsConstructor : 매개변수 모두 들어가있는

@NoArgsConstructor : 매개변수 없는거 디폴트

ModelAndView

- 컨트롤러 처리 결과 후 응답할 view와 view에 전달할 값을 저장 및 전달하는 클래스

2. 개요

@MVC에서 사용하는 주요 어노테이션은 아래와 같아요.

이름	설명
@Controller	해당 클래스가 Controller임을 나타내기 위한 어노테이션
@RequestMapping	요청에 대해 어떤 Controller, 어떤 메소드가 처리할지를 맵핑하기 위한 어노테이션
@RequestParam	Controller 메소드의 파라미터와 웹요청 파라미터와 맵핑하기 위한 어노테이션
@ModelAttribute	Controller 메소드의 파라미터나 리턴값을 Model 객체와 바인딩하기 위한 어노테이션
@SessionAttributes	Model 객체를 세션에 저장하고 사용하기 위한 어노테이션
@RequestPart	Multipart 요청의 경우, 웹요청 파라미터와 맵핑가능한 어노테이션(egov 3.0, Spring 3.1.x부터 추가)
@CommandMap	Controller메소드의 파라미터를 Map형태로 받을 때 웹요청 파라미터와 맵핑하기 위한 어노테이션(egov 3.0부터 추가)
@ControllerAdvice	Controller를 보조하는 어노테이션으로 Controller에서 쓰이는 공통기능들을 모듈화하여 전역으로 쓰기 위한 어노테이션(egov 3.0, Spring 3.2.X부터 추가)

그 중에 오늘은 @RequestMapping 어노테이션에 대해 알아볼거에요.

3. @RequestMapping 어노테이션이란?

위의 표에서 설명을 한번 봤지만, @RequestMapping 어노테이션은 URL를 컨트롤러의 메서드와 매핑할 때 사용하는 스프링 프레임워크의 어노테이션이에요.

클래스나 메서드 선언부에 @RequestMapping과 함께 URL를 명시하여 사용하죠.

URL외에도 HTTP 요청 메서드나 헤더값에 따라 매핑되도록 옵션을 제공하는데요. 메서드 레벨에서 정의한 @RequestMapping은 타입 레벨에서 정의된 @RequestMapping의 옵션을 상

참고로, 메서드 내에서 viewName을 별도로 설정하지 않으면 @RequestMapping의 path로 설정한 URL이 그대로 viewName으로 설정됩니다.

간단하게 정리하면, 클라이언트는 URL로 요청을 전송하고, 요청 URL을 어떤 메서드가 처리할지 여부를 결정하는 것이 바로 "@RequestMapping" 라고 할수 있어요.

4. @RequestMapping 이 사용하는 속성

이름	타입	설명
value	String[]	URL 값으로 매핑 조건을 부여 (default)
method	RequestMethod[]	HTTP Request 메소드 값을 매핑 조건으로 부여 사용 가능한 메소드는 GET, POST, HEAD, OPTIONS, PUT, DELETE, TRACE (7개)
params	String[]	HTTP Request 파라미터를 매핑 조건으로 부여
consumes	String[]	설정과 Content-Type request 헤더가 일치할 경우에만 URL이 호출됨
produces	String[]	설정과 Accept request 헤더가 일치할 경우에만 URL이 호출됨

-

Member 요청 방식

Member의 요청 방식은 파리미터로 join,login,logout,modify 나눔

받는 방법

공통 요청 경로 ("member") 를 세팅한뒤

params="method=join" 으로 구분한다 . (params="파라미터이름=파라미터값")