Spring Web MVC 시작하기

2021-10-04

이승진

**목차**

[1. Spring Web MVC 구현 2](#_Toc84216029)

[1) 개요 2](#_Toc84216030)

[2. 프로젝트 생성 3](#_Toc84216031)

[1) hello 프로젝트 생성 3](#_Toc84216032)

[2) pom.xml 파일 수정 7](#_Toc84216033)

[3) application.properties 파일 수정 8](#_Toc84216034)

[3. home/index 페이지 구현 9](#_Toc84216035)

[1) 개요 9](#_Toc84216036)

[2) HomeController 클래스 구현 9](#_Toc84216037)

[3) home/index.jsp 뷰 파일 구현 11](#_Toc84216038)

[4) 실행 16](#_Toc84216039)

[4. 프로젝트 수정 18](#_Toc84216040)

[1) HelloController.java 수정 18](#_Toc84216041)

[2) index.jsp 수정 18](#_Toc84216042)

[3) 실행 19](#_Toc84216043)

[5. Spring Web MVC 설명 20](#_Toc84216044)

[1) 컨트롤러 클래스 20](#_Toc84216045)

[2) 뷰(view) 22](#_Toc84216046)

[3) HelloController.java 수정 24](#_Toc84216047)

[4) index.jsp 수정 24](#_Toc84216048)

[5) 실행 절차 25](#_Toc84216049)

[6. 연습 문제 26](#_Toc84216050)

[1) test1 액션 메소드 26](#_Toc84216051)

[2) test1.jsp 뷰 파일 26](#_Toc84216052)

# Spring Web MVC 구현

## 개요

프로젝트 규모가 커질 수록, 유지보수성과 재사용성이 중요해진다.

유지보수성과 재사용성이 좋은 구조로 만들기 위한 대표적인 기술이 객체지향 설계, 객체지향 프로그래밍이다.

객체지향 구조를 만들기 위한 중요한 원칙 중 하나는,

기능별로 역할별로 소스 코드를 분리해서 구현한다는 것이다.

그래서 웹서버 애플리케이션의 기능을, Model View Controller로 분리하는 것이 객체지향적인 구조이다.

Model은 데이터를 채워서 전달하기 위한 객체이다.

View는 화면 출력을 담당한다.

Controller는 전체적인 실행 흐름을 제어한다.

화면 출력과 관련된 코드는 View에만 구현되어야 한다.

실행 흐름 제어와 관련된 코드는 Controller에만 구현되어야 한다.

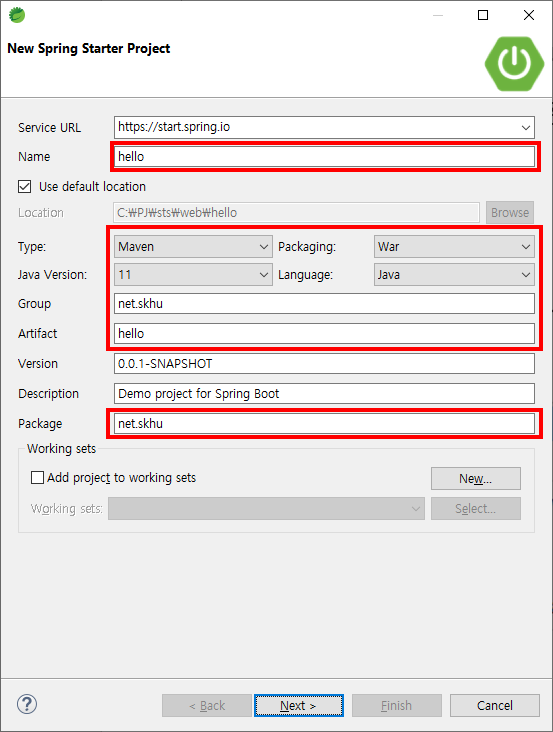
View를 구현하는 기술 중 하나는 JSP 이다.

Controller와 Model은 Java 클래스로 구현되어야 한다.

# 프로젝트 생성

## hello 프로젝트 생성

메뉴: File - New - Spring Starter Project



위와 같이 입력하고, Next 버튼 클릭

Graphical user interface, text, application, email

Description automatically generated

Spring Boot Version 은 위 화면의 값과 가장 가까운 번호를 선택한다.

Developer Tools 그룹 아래에서

Spring Boot DevTools 항목을 체크한다.

Web 그룹 아래에서

Srping Web 항목을 체크한다.

위와 같이 선택하고

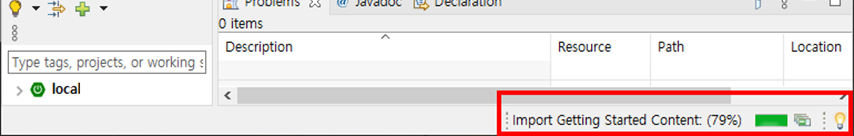
Finish 버튼 클릭

### 프로젝트 빌드

프로젝트가 빌드 완료될 때까지 기다려야 한다.

Spring Tools Suite 화면 오른쪽 아래에 녹색 막대기로 빌드 작업 진척도가 보인다.

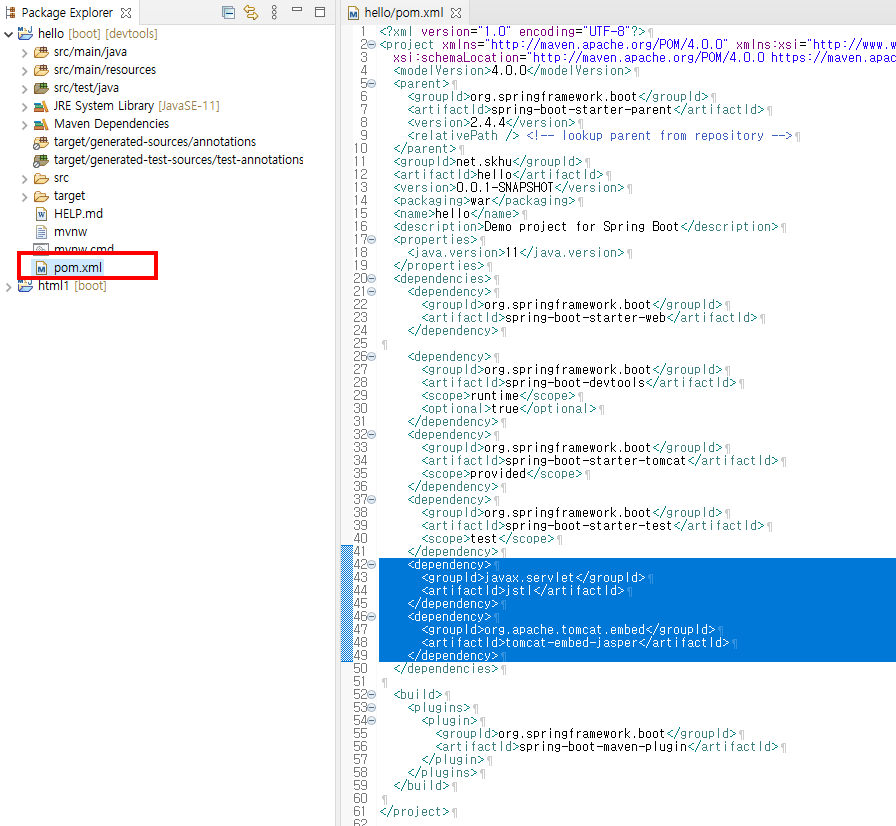
빌드 작업이 완료되면 녹색 막대가 사라진다.



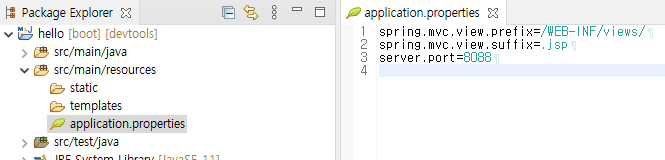
## pom.xml 파일 수정

|  |
| --- |
| <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  </dependency>  <dependency>  <groupId>org.apache.tomcat.embed</groupId>  <artifactId>tomcat-embed-jasper</artifactId>  </dependency> |

위 내용을 <dependencies> 태그 내부에 추가한다.



## application.properties 파일 수정



### src/main/resources/application.properties

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp  server.port=8088 |

spring.mvc.view.prefix=/WEB-INF/views/

spring.mvc.view.suffix=.jsp

뷰(view) 파일의 위치와 확장자를 지정한다.

/WEB-INF/views/뷰이름.jsp

서버가 사용할 포트 번호를 8088로 설정한다.

위 설정을 하지 않으면, 디폴트 포트 번호는 8080 이다.

8080 포트를 사용하는 앱들이 많아서 포트 충돌 에러가 발생할 확률이 높으니,

8088로 수정하는 것이 바람직하다.

# home/index 페이지 구현

## 개요

웹브라우저가 http://localhost:8088/home/index URL을 서버에 요청했을 때,

서버에서 실행될 웹 페이지를 구현하자.

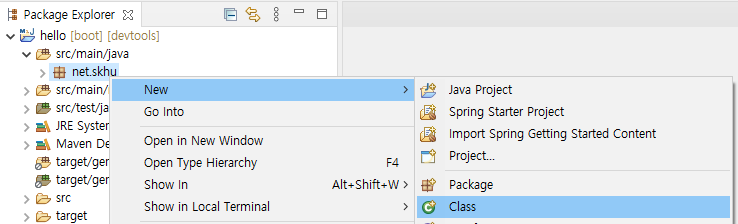
### controller 클래스 구현

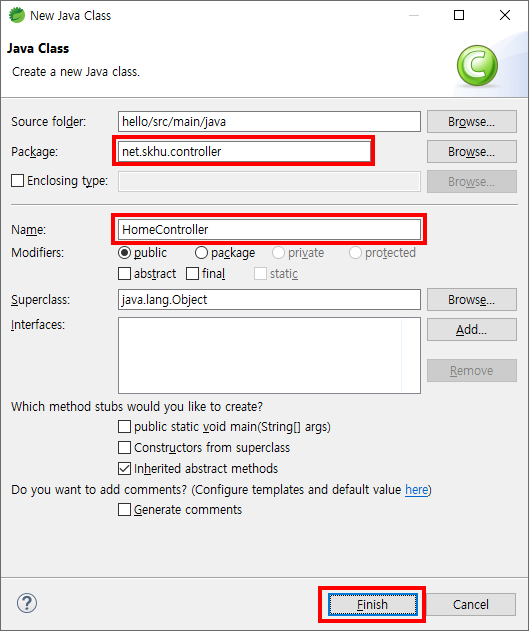
URL 요청에 의해서 서버에서 실행될 controller 클래스와 action method를 구현해야 한다.

### view 파일 구현

home/index 페이지의 내용을 출력하기 뷰 파일을 구현해야 한다.

## HomeController 클래스 구현



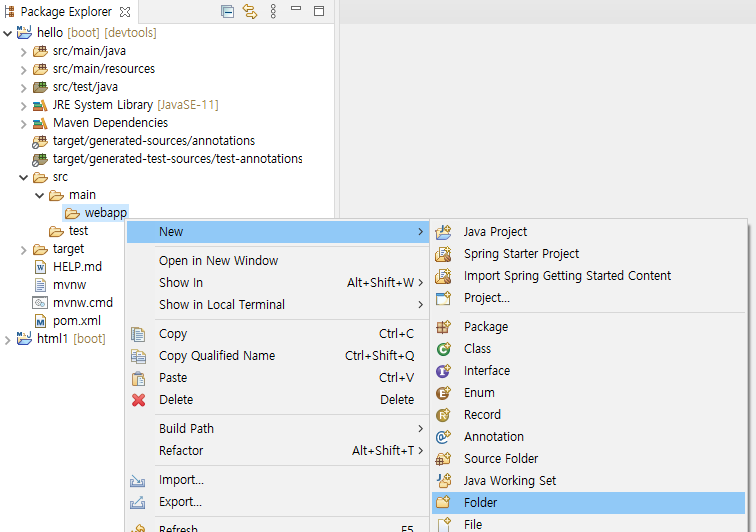


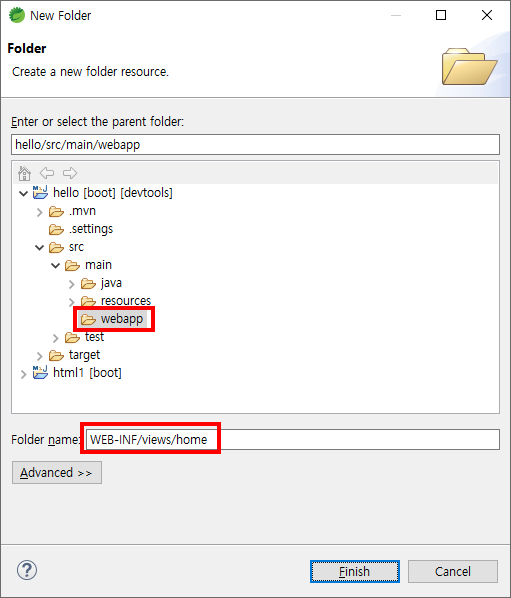
|  |  |
| --- | --- |
| Package | net.skhu.controller |
| Name | HomeController |

### src/main/java/net/skhu/HomeController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("home/index")  public String index(Model model) {  model.addAttribute("message", "좋은 아침입니다");  return "home/index";  }  } |

## home/index.jsp 뷰 파일 구현





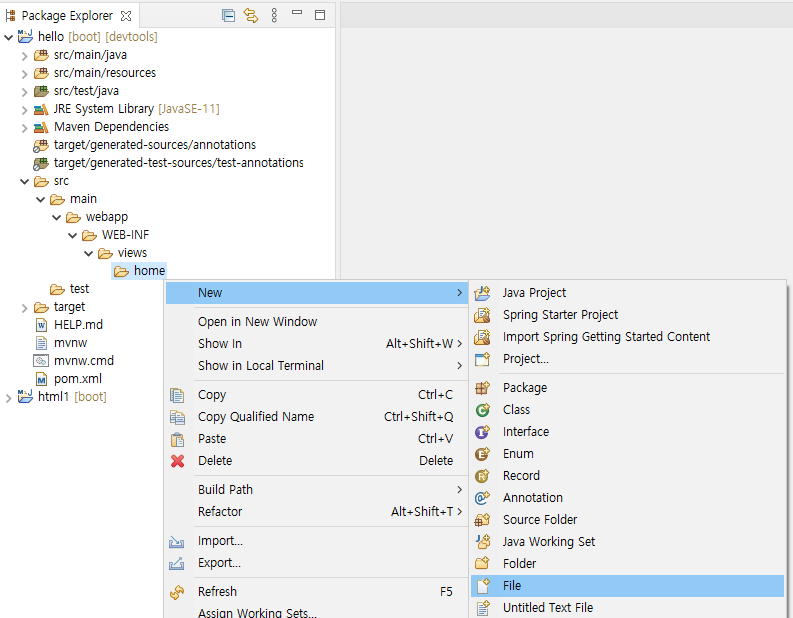
|  |  |
| --- | --- |
| Folder name: | WEB-INF/views/home |

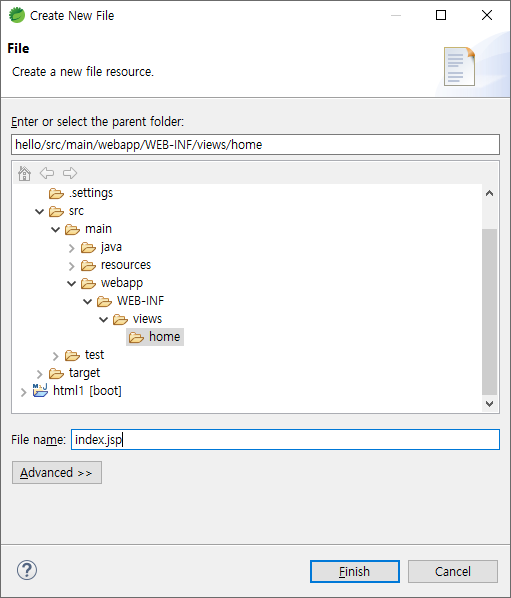
### 주의

위 폴더 경로명에서 **WEB-INF/views/** 부분은

application.properties 파일의 spring.mvc.view.prefix 항목과 일치해야 한다.

일치하지 않을 경우에 404 Not Found 에러가 발생한다.





|  |  |
| --- | --- |
| File name: | index.jsp |

### src/main/webapp/WEB-INF/views/index.jsp

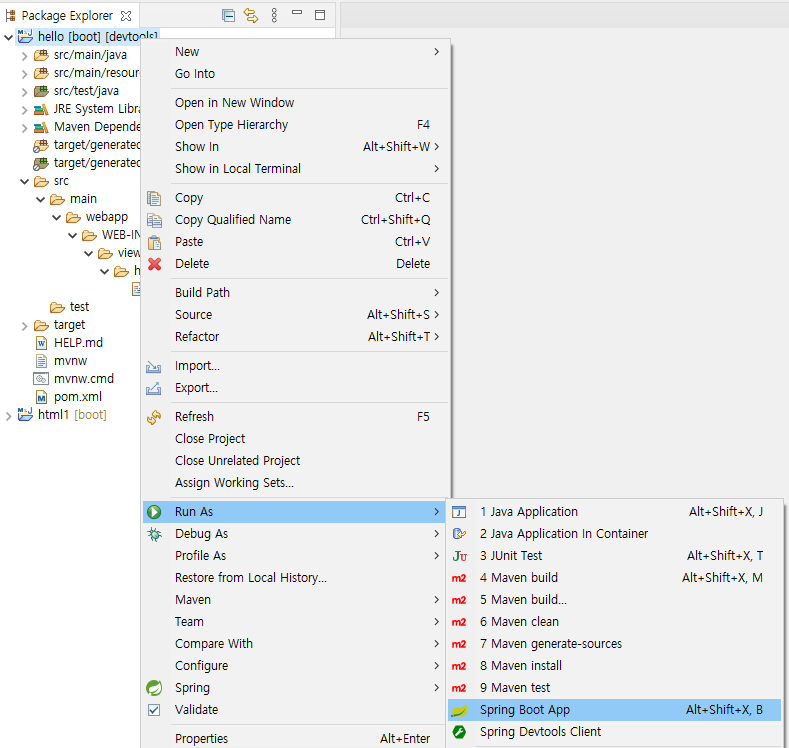
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  </body>  </html> |

## 실행

### Spring Boot App 실행

STS 화면 왼쪽의 Project Explorer에서 hello1 프로젝트 명을 선택하고

Run - Run As - Spring Boot App 메뉴 클릭

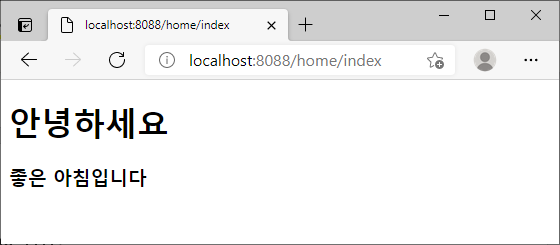




콘솔 창의 출력이 위와 같다면, Spring boot app이 잘 실행된 것이다.

웹브라우저를 열어서 http://localhost:8088 URL을 실행하자.

### http://localhost:8088/home/index



서버에서 웹브라우저로 출력된 내용은 아래와 같다.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>좋은 아침입니다.</h3>  </body>  </html> |

# 프로젝트 수정

수정된 소스코드가 얼마나 빨리 실행되는지 확인해 보자.

## HelloController.java 수정

### src/main/java/net/skhu/controller/HomeController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package net.skhu.controller;  import java.util.Date;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("home/index")  public String index(Model model) {  model.addAttribute("message", "좋은 아침입니다");  model.addAttribute("now", new Date());  return "home/index";  }  } |

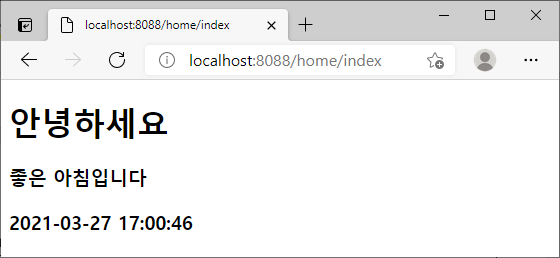
## index.jsp 수정

### src/main/webapp/WEB-INF/views/home/index.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  <h3><fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" /></h3>  </body>  </html> |

## 실행

웹브라우저를 새로고침하자.



서버에서 웹브라우저로 출력된 내용은 아래와 같다.

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>좋은 아침입니다.</h3>  <h3>2021-03-27 17:00:46</h3>  </body>  </html> |

# Spring Web MVC 설명

## 컨트롤러 클래스

### src/main/java/net/skhu/controller/HomeController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | package net.skhu.controller;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("home/index")  public String index(Model model) {  model.addAttribute("message", "좋은 아침입니다");  return "home/index";  }  } |

컨트롤러 클래스는 웹브라우저의 URL 요청을 받아서, 웹서버에서 실행되는 자바 클래스이다.

웹브라우저가 웹서버에 어떤 URL을 요청을 하면, 그 URL에 해당하는 컨트롤러의 메소드가 자동으로 호출되어 실행된다.

### @Controller

컨트롤러 클래스에 이 어노테이션(annotation)을 붙여주어야 한다.

이것을 붙여주지 않으면 컨트롤러 클래스가 실행되지 않는다.

### 액션 메소드

웹브라우저가 어떤 URL을 웹서버에 요청하면, 그 요청된 URL에 해당하는 컨트롤러의 어떤 메소드가 자동으로 호출된다. 이렇게 웹브라우저의 요청에 의해서 자동으로 호출되는 컨트롤러의 메소드를 액션 메소드라고 부른다.

### @RequestMapping 어노테이션과 액션 메소드 URL

액션 메소드에 붙은 @RequestMapping("home/index") 어노테이션은

이 액션 메소드를 호출하기 위한 URL 이다.

**http://localhost:8088/**home/indexURL을 웹브라우저가 웹서버에 요청하면,

웹서버에서 HomeController 컨트롤러 클래스의 index 액션 메소드가 자동으로 호출되어 실행된다.

자동으로 호출할 액션 메소드를 찾을 때 컨트롤러 클래스 이름이나 액션 메소드의 이름은 상관 없고,

@RequestMapping("home/index") 어노테이션에 등록된 URL만 일치하면 된다.

### 컨트롤러 액션 메소드의 리턴 값

컨트롤러의 액션 메소드는 문자열을 리턴한다.

컨트롤러의 액션 메소드가 리턴하는 문자열은, 뷰(View) 파일의 이름이다.

컨트롤러의 액션 메소드가 리턴된 후, 뷰 파일이 실행된다.

액션 메소드가 리턴한 이름의 뷰 파일이 실행된다.

### Model 객체

|  |
| --- |
| public String index(Model model) { |

컨트롤러의 액션 메소드는 Spring Web MVC 엔진에 의해서 호출된다.

대부분의 액션 메소드의 파라미터에 Model model 객체가 포함된다.

컨트롤러의 액션 메소드가 뷰 파일에 전달할 데이터를, Model 객체에 넣어서 전달한다.

즉 Model 객체는, 데이터 전달 상자라고 보면 된다.

|  |
| --- |
| model.addAttribute("message", "좋은 아침입니다"); |

뷰 파일에 전달할 데이터를 Model 객체에 넣는 코드이다.

전달되는 데이터의 이름은 "message" 이고, 값은 "좋은 아침" 문자열이다.

이렇게 전달된 문자열을 뷰 파일에서 출력하는 코드는 아래와 같다.

|  |
| --- |
| ${ message } |

## 뷰(view)

### src/main/webapp/WEB-INF/views/home/index.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  </body>  </html> |

웹브라우저의 요청(http request)에 대한 서버의 응답(http response)으로,

웹서버에서 웹브라우저로 전송되는 것은 대부분 html 태그이다.

우리가 웹서핑하면서 보는 웹페이지들의 내용이 html 태그이다.

### 실행 순서

웹브라우저에서 웹서버에 요청(http request)이 전달되면,

요청된 URL과 일치하는, 컨트롤러의 액션 메소드가 실행된다.

그리고 액션 메소드의 뒤를 이어서 뷰(view) 파일이 실행된다.

뷰 파일의 실행 결과 출력된 html 태그들이 웹브라우저로 전송된다.

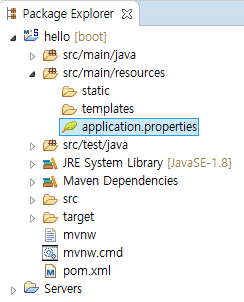
### 뷰 파일 위치 설정

Spring Starter Project 메뉴를 클릭하여 생성한 프로젝트의 설정 파일은

**src/main/resources/application.properties** 파일이다.

이 파일에 뷰(view) 파일들의 위치를 지정해야 한다.

### application.properties 파일



### src/main/resources/application.properties

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/  spring.mvc.view.suffix=.jsp |

뷰 파일들은 **src/main/webapp/WEB-INF/views/** 폴더 아래에 \*.jsp 파일이어야 한다.

|  |
| --- |
| spring.mvc.view.prefix=/WEB-INF/views/ |

뷰 파일들이 위치할 폴더를 지정한다.

|  |
| --- |
| spring.mvc.view.suffix=.jsp |

뷰 파일들의 확장자를 지정한다.

예를 들어 액션 메소드가 리턴한 뷰의 이름이 다음과 같으면

|  |
| --- |
| return "home/index"; |

이 뷰 파일의 위치는 **/WEB-INF/views/home/index.jsp** 이어야 한다.

만약 폴더 명이나 파일명에 오타가 있다면, 뷰 파일을 찾지 못할 테니 에러가 발생할 것이다.

이 경우 웹브라우저에 "404 Not Found" 메시지가 보일 것이다.

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> |

JSP 파일의 첫 줄은 언제나 위와 같아야 한다. 그래야 한글이 깨지지 않는다.

그리고 개발 도구의 JSP 파일의 인코딩도 "UTF-8" 이어야 한다. (spring tool suite 설치 강의노트 참고)

|  |
| --- |
| ${ message } |

index.jsp 뷰 파일에 전달된 데이터 중에서, 이름이 "message"인 데이터를 여기에 출력하라는 명령이다.

## HelloController.java 수정

### src/main/java/net/skhu/controller/HomeController.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | package net.skhu.controller;  import java.util.Date;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class HomeController {  @RequestMapping("home/index")  public String index(Model model) {  model.addAttribute("message", "좋은 아침입니다");  model.addAttribute("now", new Date());  return "home/index";  }  } |

|  |
| --- |
| model.addAttribute("now", new Date()); |

뷰 파일에 전달할 데이터를 Model 객체에 넣는 코드이다.

전달되는 데이터의 이름은 "now" 이고, 값은 현재 시각 Date 객체이다.

## index.jsp 수정

### src/main/webapp/WEB-INF/views/home/index.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding="utf-8"%>  <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %>  <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8">  </head>  <body>  <h1>안녕하세요</h1>  <h3>${ message }</h3>  <h3><fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" /></h3>  </body>  </html> |

|  |
| --- |
| <fmt:formatDate pattern = "yyyy-MM-dd HH:mm:ss" value = "${ now }" /> |

<fmt:formatDate /> 태그는 날짜와 시각을 포멧에 맞춰서 출력하기 위한 JSTL 확장 태그이다.

JSTL 확장 태그를 사용하기 위해 pom.xml 파일에 JSTL dependency를 이미 추가했다.

value = "${ now }"

뷰 파일에 전달된 데이터 중에서, 이름이 "now"인 데이터가 출력된다.

pattern = "yyyy-MM-dd HH:mm:ss"

날짜와 시각이 이 포멧에 맞춰서 출력된다. 이 포멧 문자열은 Java의 SimpleDateFormat 클래스의 그것이다.

|  |
| --- |
| <%@ taglib prefix = "fmt" uri = "http://java.sun.com/jsp/jstl/fmt" %> |

<fmt:formatDate /> 확장 태그를 사용하려면, 먼저 이 선언이 있어야 한다.

## 실행 절차



|  |  |
| --- | --- |
| 1 | 웹브라우저가 웹서버에 **요청(HTTP Request)**을 전달한다.  이 요청에는, 요청 대상을 가르키는 **URL**이 담겨 있다. |
| 2 | 웹브라우저로부터 웹서버에 전달된 요청을 Spring Web MVC 엔진이 받는다.  스프링 엔진은 요청된 URL과 일치하는 **컨트롤러 액션 메소드**를 찾아서 호출한다. |
| 3 | 컨트롤러의 액션 메소드는 데이터를 Model 객체에 넣는다. |
| 4 | 컨트롤러의 액션 메소드는 뷰의 이름을 리턴한다. |
| 5 | 그 이름의 뷰가 실행된다. |
| 6 | 뷰는 Model 객체에서 데이터를 꺼내서 출력하고, HTML 태그들도 출력한다. |
| 7 | 뷰가 출력한 HTML 태그들이 웹브라우저에 전송된다.  이 전송은 최초 웹브라우저의 요청(http request)에 대한 응답(http response)이다. |
| 8 | 웹 서버로부터 전송된 HTML 태그들이 웹브라우저에 표시된다. |

# 연습 문제

## test1 액션 메소드

HomeController 클래스에 test1 액션 메소드를 구현하자.

이 액션 메소드를 호출하기 위한 URL은 "home/test1" 이어야 한다.

자신의 이름을 Model 객체에 넣는 코드를 추가하자.

이 데이터의 이름을 "myName" 이라고 하자.

액션 메소드는 뷰 파일의 이름을 리턴해야 한다.

## test1.jsp 뷰 파일

myName 이름의 데이터를 출력하는 뷰 파일을 구현하자.