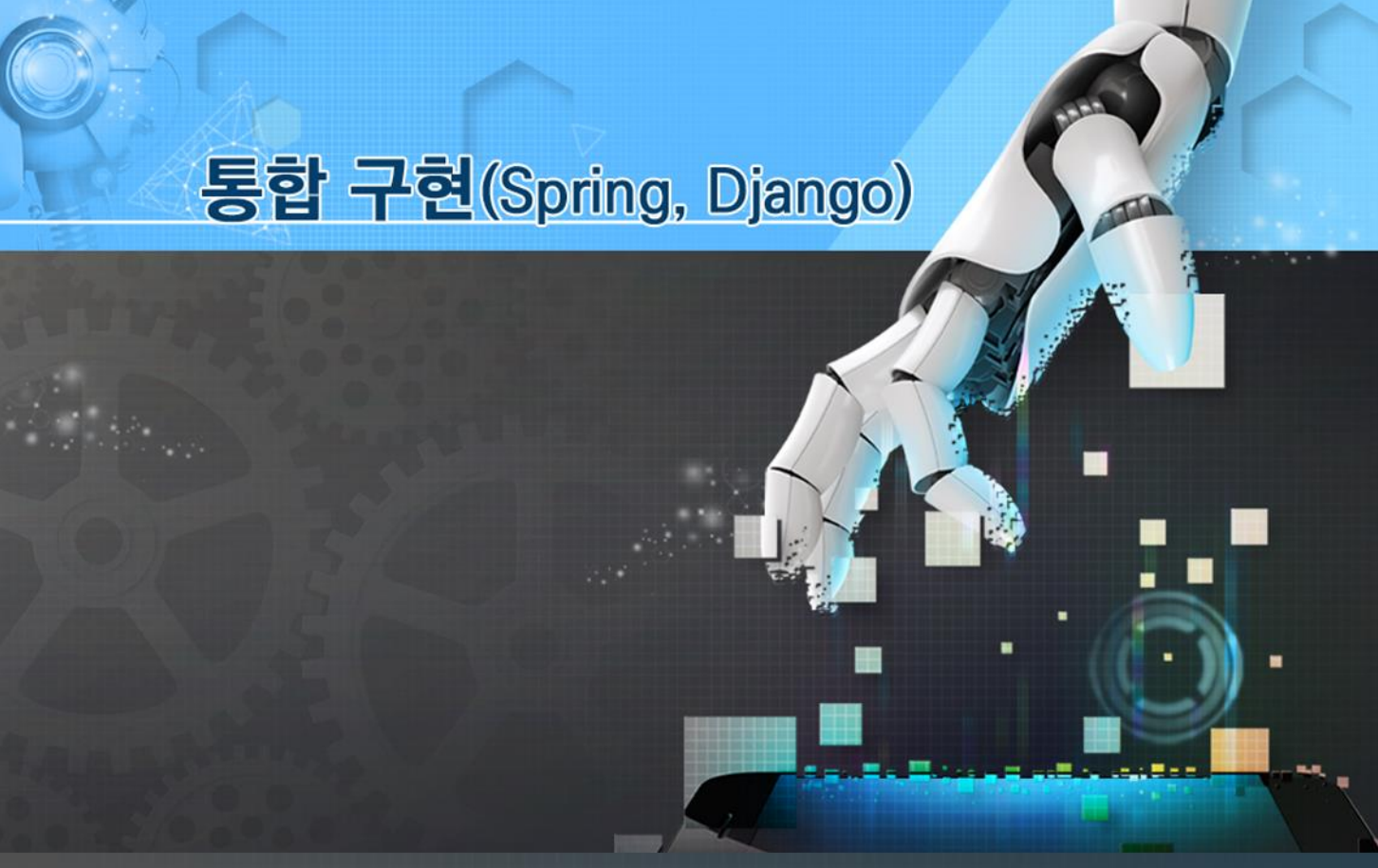

Spring MVC 구성하기

한국기술교육대학교
온라인평생교육원

학습내용

- MVC 패턴 살펴보기
- Controller
- Spring Form

학습목표

- MVC 패턴이란 무엇인지 파악할 수 있다.
- Spring의 Controller를 구성하는 방법에 대해 설명할 수 있다.
- Spring의 Form을 활용하는 방법에 대해 파악할 수 있다.

MVC 패턴 살펴보기

1 MVC 패턴이란

(1) MVC(Model View Controller)

1. 사용자의 뷰페이지(View)와 데이터(Model) 그리고 이들 상호 간의 흐름을 제어(Control)하는 비즈니스 로직 분리
2. 상호 영향 없이 모듈을 재사용, 확장 가능한 구조적 패턴
3. 웹 어플리케이션 개발 영역에까지 보편적으로 사용되기 시작하였고, 웹 개발자라면 반드시 알아야 할 패턴

(2) MVC의 구조

Model	상태 정보 처리
View	프리젠테이션 뷰 (즉, 사용자가 보게 될 결과 화면 담당)
Controller	사용자의 입력 및 흐름 제어 담당

MVC 패턴 살펴보기

1 MVC 패턴이란

《3》 MVC의 작동

1. 클라이언트가 서버로 보내는 요청을 컨트롤러가 받음
2. 모델은 상태 정보 및 관련 기능을 제공하는데, 컨트롤러는 이 모델을 통해 사용자의 요청을 처리
3. 모델을 사용하여 알맞은 로직을 수행한 후 컨트롤러는 사용자에게 보여줄 뷰를 선택하며, 선택된 뷰는 사용자에게 알맞은 결과 화면을 보여줌

《4》 MVC의 특징

- 비즈니스 로직을 처리하는 모델과 결과 화면을 보여주는 뷰 분리
- 어플리케이션의 흐름 제어나 사용자의 처리 요청은 컨트롤러에 집중
- MVC 패턴을 사용함으로써 유지 보수 작업이 간단해지고 어플리케이션을 쉽게 확장 가능

2 Spring의 MVC

《1》 DispatcherServlet

1. 클라이언트로부터 요청이 들어오면 DispatcherServlet이 가로챔
2. 가로챈 정보를 HandlerMapping에게 보내 해당 요청을 처리할 수 있는 적절한 Controller를 찾아냄
3. 컨트롤러를 찾았다면 해당 컨트롤러로 요청을 보냄(예 : HomeController)

MVC 패턴 살펴보기

2 Spring의 MVC

⟨1⟩ DispatcherServlet

4 컨트롤러는 요청에 대한 처리 후 해당 View 리턴(예: hello.jsp)

5 이 때, 해당 이름으로 ViewResolver가 찾아내서 처리결과를 View에 보내줌

6 이 결과를 다시 DispatcherServlet에게 보내주고, DispatcherServlet은 최종 결과를 다시 클라이언트에게 전송

⟨2⟩ ViewResolver 설정

컨트롤러가 “hello”를 리턴

/WEB-INF/views/hello.jsp가 리턴됨

```
<bean id="contentNegotiatingViewResolver"
class="org.springframework.web.servlet.view.ContentNegotiatingViewResolv
er">
 <property name="viewResolvers">
  <list>
   <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver"
>
    <property name="prefix" value="/WEB-INF/views/" />
    <property name="suffix" value=".jsp" />
   </bean>
  </list>
 </property>
</bean>
```

MVC 패턴 살펴보기

2 Spring의 MVC

3 Controller 예제

```
@Controller
@RequestMapping("/")
public class HomeController {

  @RequestMapping
  public String home(Model model) {
    model.addAttribute("textFromController", "World");
    return "hello";
  }
}
```

Controller

1 RequestMapping

RequestMapping이란?

클라이언트가 서버에 요청할 때, URL, HTTP 메소드, 헤더 등에 대해 정의

String[] value	URL 패턴
RequestMethod[] method	HTTP 메소드(GET, POST, HEAD, OPTIONS, PUT, PATCH, DELETE, TRACE)
String[] params	요청 Parameter
String[] headers	HTTP 헤더

Get 형식 예제

```
@RequestMapping(value = "/requestMappingGetTest",
      method = RequestMethod.GET)
  public String requestMappingGetTest(Model model) {
    model.addAttribute("textFromController", "World");
    return "hello";
  }
```

Controller

2 RequestParam

RequestParam

클라이언트가 서버로 요청을 보낼 때, 주소 외에 별도의 데이터(Parameter) 형식에 대해 정의

String value	Parameter 이름
String name	Parameter 이름(스프링 4.2.0 이후)
boolean required	Parameter가 필수인지 아닌지 여부
String defaultValue	Parameter 값이 없을 경우 기본값 지정

RequestParam 예제

```
@RequestMapping("/requestParamTest")
  public String requestParamTest(
  @RequestParam(name= "a", required=false, defaultValue = "0") int a,
  @RequestParam("b") String b,
  @RequestParam(name= "c", defaultValue = "true") boolean c) {
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    return "hello";
  }
```

Controller

2 RequestParam

RequestParam에 Map 사용

- Map을 통해 Parameter를 받으면 불특정 다수 Parameter를 한 번에 받을 수 있음

```
@RequestMapping("/requestParamMapTest")
 public String requestParamMapTest(@RequestParam Map<String, String>
map) {
   for(Map.Entry entry: map.entrySet()) {
     System.out.println(entry.getKey() + "=" + entry.getValue());
   }
   return "hello";
 }
```

3 PathVariable

PathVariable

- REST형식의 주소로 Parameter를 받을 때 사용
- 객체 형식으로 값을 전달받기 때문에 null일 경우 주의

PathVariable 예제

```
@RequestMapping("/pathVariableTest/{a}/{b}/{c}")
 public String pathVariableTest(
   @PathVariable(value="a") String a,
   @PathVariable String b,
   @PathVariable int c) {
    System.out.println("a = " + a);
    System.out.println("b = " + b);
    System.out.println("c = " + c);
    return "hello";
}
```

Spring form

1 Spring Form 처리

Spring Form 태그

1. Spring에서는 MVC 패턴에 따라 모델 데이터를 삽입하거나 컨트롤러로 Form 태그 라이브러리 제공
2. 스프링 폼 태그는 Model 데이터 객체나 참조 데이터(Reference Data)들을 화면상에서 쉽게 출력하도록 도와줌
3. 스프링 폼 태그는 spring-form.tld를 필요로 하므로 jsp파일 상단에 다음과 같이 선언해야 함

```
<%@ taglib uri="http://www.springframework.org/tags/form"
prefix="form"%>
```

Spring Form 태그 예제(signup.jsp)

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib uri="http://www.springframework.org/tags/form"
prefix="form"%>
<html>
<head>
  <title>Form Test</title>
</head>
<body>
<form:form modelAttribute="user">
  이름: <form:input path="name"/><br/>
  이메일: <form:input path="email"/><br/>
  비밀번호: <form:password path="password"/><br/>
  나이: <form:input path="age"/><br/>
  <input type="submit" value="가입"/>
</form:form>
</body>
</html>
```

Spring form

1 Spring Form 처리

Spring Form 태그 예제(User.java)

```
package com.spring. domain;

public class User {
  private int id; //DB의 primary key 역할
  private String name;
  private String email;
  private String password;
  private int age;

  //Getter method
  ...
  //Setter method
  ...
  @Override
  public String toString() {
    return "id: " + id + ", name: " + name
        + ", email: " + email + ", password: " + password + ", age: " + age;
  }
}
```

Spring form

1 Spring Form 처리

Spring Form 태그 예제((1/2) UserController.java)

```
package com.spring.controller;

import com.spring.domain.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/user")
public class UserController {

  @RequestMapping("/signup")
  public String signup(Model model) {
    User user = new User();
    model.addAttribute("user", user);
    return "signup";
  }
```

Spring Form 태그 예제((2/2) UserController.java)

```
  @RequestMapping(value="/signup", method= RequestMethod.POST)
  @ResponseBody
  public String signup(@ModelAttribute User user) {
      System.out.println("user = " + user);
      return "success";
  }
}
```

Spring form

2 Response 처리

⟨1⟩ @ResponseBody

1. Spring 컨트롤러에서 Return “hello”를 하는 것은 기본적으로 hello.jsp를 찾아서 화면에 보여주라는 뜻
2. @ResponseBody annotation을 붙인 후 return “hello”를 한다면 String 그대로의 ”hello”가 리턴됨
3. ViewResolver에 의한 View 파일(jsp)이 아닌 순수한 값 자체를 클라이언트에 응답

⟨2⟩ Redirect

Spring 컨트롤러에서 URL에 대한 처리 후 항상 ViewResolver를 통해 View 파일을 보여주지는 않음

URL에 대한 처리 후 다른 페이지로 이동할 필요가 있을 때, redirect 기능 이용

- redirect : 주소와 같이 입력하면 해당 주소로 이동

```
@RequestMapping(value="/signup", method= RequestMethod.POST)
 public String signup(@ModelAttribute User user) {
    System.out.println("user = " + user);
    return “redirect:/signup”;
 }
```

학습정리

1. MVC 패턴 살펴보기

- MVC 패턴이란
  - 사용자의 뷰페이지(View)와 데이터(Model) 그리고 이들 상호 간의 흐름을 제어(Control)하는
  - 비즈니스 로직을 분리
  - 상호 영향 없이 모듈을 재사용, 확장 가능한 구조적 패턴
  - 웹 어플리케이션 개발 영역에까지 보편적으로 사용되기 시작
  - 웹 개발자라면 반드시 알아야 할 패턴
- Spring의 정의
  - DispatcherServlet : 클라이언트로부터 요청이 들어오면 DispatcherServlet이 가로챔

학습정리

2. Controller

- RequestMapping
  - 클라이언트가 서버에 요청할 때, URL, HTTP 메소드, 헤더 등에 대해 정의
- RequestParam
  - 클라이언트가 서버로 요청을 보낼 때, 주소 외에 별도의 데이터(Parameter) 형식에 대해 정의
- PathVariable
  - REST형식의 주소로 Parameter를 받을 때 사용
  - 객체 형식으로 값을 전달받기 때문에 Null일 경우 주의

학습정리

3. Spring Form

- Spring Form 처리
  - Spring에서는 MVC 패턴에 따라 모델 데이터를 삽입하거나 컨트롤러로 Form 태그 라이브러리를 제공
  - 스프링 폼 태그는 Model 데이터 객체나 참조 데이터(Reference Data)들을 화면상에서 쉽게 출력하도록 도와줌

- Response 처리
  - Spring 컨트롤러에서 Return “hello”를 하는 것은 기본적으로 hello.jsp를 찾아서 화면에 보여주라는 뜻
  - @ResponseBody annotation을 붙인 후 Return “hello”를 한다면 String 그대로의 ”hello”가 리턴 됨
  - ViewResolver에 의한 View 파일(jsp 이 아닌 순수한 값 자체를 클라이언트에 응답