Spring MVC

9 관심사의 분리, MVC패턴 - 이론

관심사의 분리, MVC 패턴 - 이론

1. 관심사의 분리 Separation of Concerns

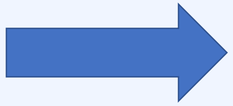
```java
@Controller
public class YoilTeller {
    @RequestMapping("/getYoil")
    public void main(HttpServletRequest request, HttpServletResponse response) throws IOException {
        // 1. 입력
        String year = request.getParameter("year");
        String month = request.getParameter("month");
        String day = request.getParameter("day");

        int yyyy = Integer.parseInt(year);
        int mm = Integer.parseInt(month);
        int dd = Integer.parseInt(day);

        // 2. 처리
        Calendar cal = Calendar.getInstance();
        cal.set(yyyy, mm - 1, dd);

        int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
        char yoil = " 일월화수목금토".charAt(dayOfWeek);

        // 3. 출력
        response.setContentType("text/html");    // 응답의 형식을 html로 지정
        response.setCharacterEncoding("utf-8");  // 응답의 인코딩을 utf-8로 지정
        PrintWriter out = response.getWriter();  // 브라우저로의 출력 스트림(out)을 얻는다.
        out.println("<html>");
        out.println("<head>");
        out.println("</head>");
        out.println("<body>");
        out.println(year + "년 " + month + "월 " + day + "일은 ");
        out.println(yoil + "요일입니다.");
        out.println("<body>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }
```

관심사의 분리, MVC 패턴 - 이론
2. 공통 코드의 분리 – 입력의 분리

1. 입력
2. 처리
3. 출력

1. 입력
2. 처리
3. 출력

1. 입력
2. 처리
3. 출력



1. 입력

2. 처리
3. 출력

2. 처리
3. 출력

2. 처리
3. 출력

관심사의 분리, MVC 패턴 - 이론
2. 공통 코드의 분리 – 입력의 분리

```java
@RequestMapping("/getYoil")
public void main(HttpServletRequest request,
                 HttpServletResponse response) throws IOException {
    // 1. 입력
    String year = request.getParameter("year");
    String month = request.getParameter("month");
    String day = request.getParameter("day");

    int yyyy = Integer.parseInt(year);
    int mm = Integer.parseInt(month);
    int dd = Integer.parseInt(day);
```

```java
@RequestMapping("/getYoil")
public void main(int year, int month, int day,
    HttpServletResponse response) throws IOException {

    // 2. 처리
    Calendar cal = Calendar.getInstance();
    cal.set(year, month - 1, day);
```

```java
@RequestMapping("/getYoil")
public void main(String year, String month, String day,
        HttpServletResponse response) throws IOException {

    int yyyy = Integer.parseInt(year);
    int mm = Integer.parseInt(month);
    int dd = Integer.parseInt(day);
```

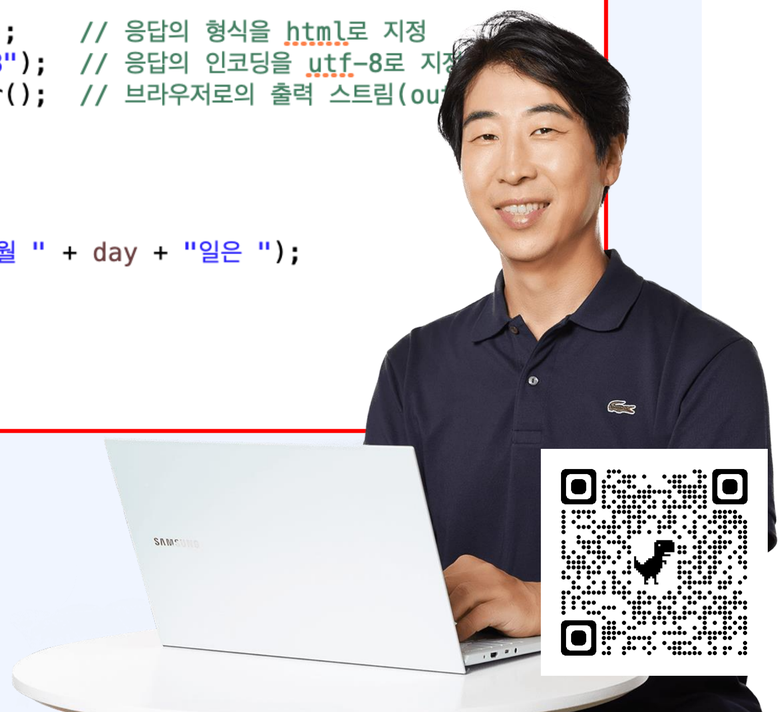
관심사의 분리, MVC 패턴 - 이론

3. 출력(view)의 분리 - 변하는 것과 변하지 않는 것의 분리

```java
@RequestMapping("/getYoil")
public void main(int year, int month, int day, HttpServl
        throws IOException {
    // 2. 처리
    Calendar cal = Calendar.getInstance();
    cal.set(year, month - 1, day);

    int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
    char yoil = " 일월화수목금토".charAt(dayOfWeek);

    // 3. 출력
    response.setContentType("text/html");     // 응답의 형
    response.setCharacterEncoding("utf-8");   // 응답의 인
    PrintWriter out = response.getWriter();   // 브라우저
    out.println("<html>");
    out.println("<head>");
    out.println("</head>");
    out.println("<body>");
    out.println(year + "년 " + month + "월 " + day + "일
    out.println(yoil + "요일입니다.");
    out.println("<body>");
    out.println("</body>");
    out.println("</html>");
    out.close();
}
```

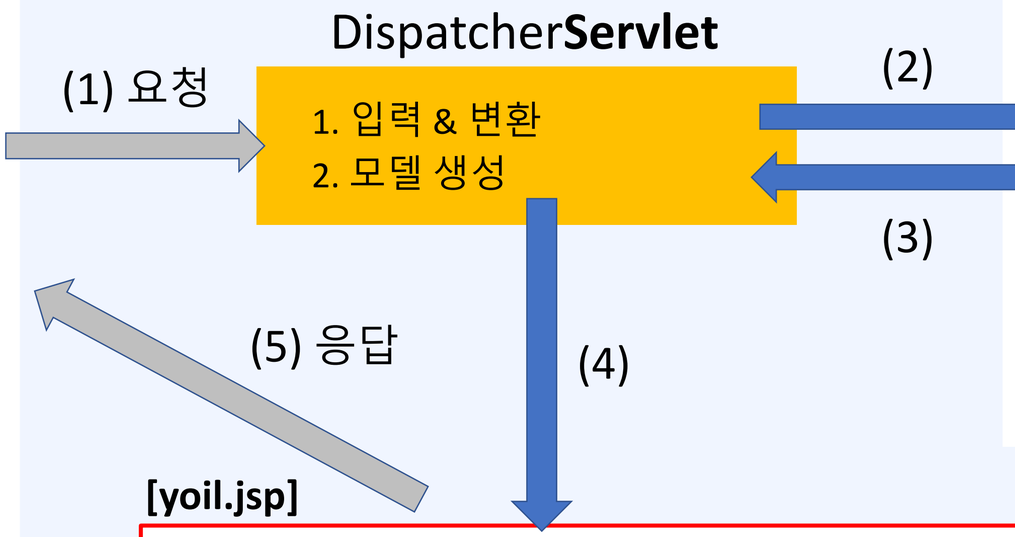
```java
// 2. 처리
Calendar cal = Calendar.getInstance();
cal.set(year, month - 1, day);

int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
char yoil = " 일월화수목금토".charAt(dayOfWeek);
```

```java
// 3. 출력
response.setContentType("text/html");     // 응답의 형식을 html로 지정
response.setCharacterEncoding("utf-8");   // 응답의 인코딩을 utf-8로 지정
PrintWriter out = response.getWriter();   // 브라우저로의 출력 스트림(ou
out.println("<html>");
out.println("<head>");
out.println("</head>");
out.println("<body>");
out.println(year + "년 " + month + "월 " + day + "일은 ");
out.println(yoil + "요일입니다.");
out.println("<body>");
out.println("</body>");
out.println("</html>");
out.close();
```



관심사의 분리, MVC 패턴 - 이론

4. MVC 패턴




```java
@RequestMapping("/getYoil")
public String main(int year, int month, int day, Model model) {
    // 1. 유효성 검사
    if(!isValid(year, month, day))
        return "yoilError";  // 유효하지 않으면, /WEB-INF/views/yoilError.jsp로 이동

    // 2. 처리
    char yoil = getYoil(year, month, day);

    // 3. Model에 작업 결과 저장
    model.addAttribute("year", year);
    model.addAttribute("month", month);
    model.addAttribute("day", day);
    model.addAttribute("yoil", yoil);

    // 4. 작업 결과를 보여줄 View의 이름을 반환
    return "yoil"; // /WEB-INF/views/yoil.jsp
}
```

[yoil.jsp]

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
    <title>YoilTellerMVC</title>
</head>
<body>
<h1>${year}년 ${month}월 ${day}일은 ${yoil}요일입니다.</h1>
</body>
</html>
```

[yoilError.jsp]

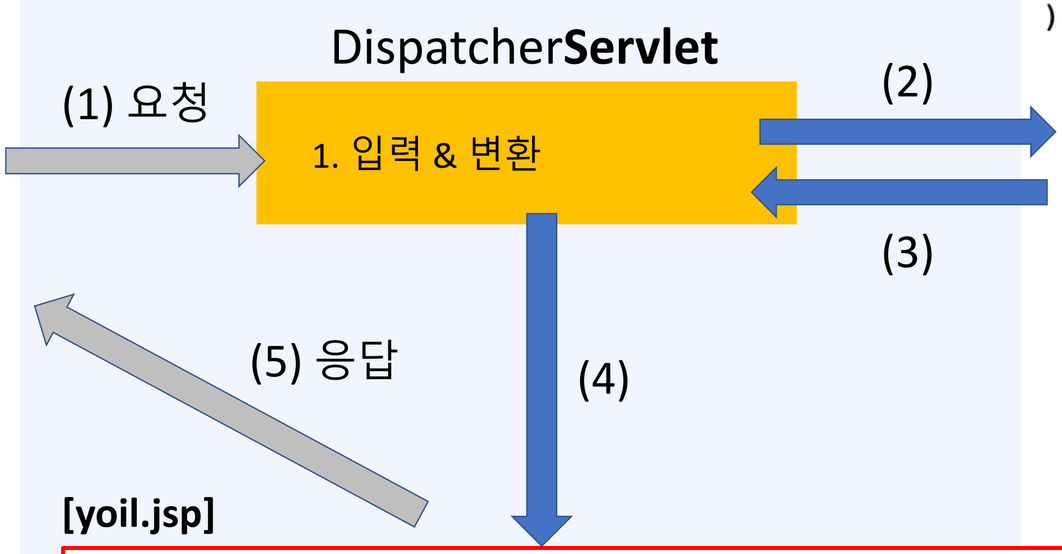
```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
<%@ page session="false" %>
<html>
<head>
    <title>YoilTellerMVC</title>
</head>
<body>
<h1>잘못된 요청입니다. 다시 요청해 주세요.</h1
<h1>year, month, day를 모두 올바른 값으로 입
</body>
</html>
```

Spring MVC

10 관심사의 분리, MVC패턴 - 실습

관심사의 분리, MVC 패턴 - 이론

5. ModelAndView



[yoil.jsp]

```jsp
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
    <title>YoilTellerMVC</title>
</head>
<body>
<h1>${year}년 ${month}월 ${day}일은 ${yoil}요일입니다.</h1>
</body>
</html>
```

```java
@RequestMapping("/getYoilMVC3") // http://localhost/ch2/getYoilMVC3
public ModelAndView main(  // 반환 타입이 ModelAndView
        int year, int month, int day //, Model   // Model을 받지 않음.
) throws IOException {
    // 1. ModelAndView를 생성하고, 기본 뷰를 지정
    ModelAndView mv = new ModelAndView();
    mv.setViewName("yoilError"); // 뷰의 이름을 지정

    // 2. 유효성 검사
    if(!isValid(year, month, day))
        return mv;

    // 3. 작업
    char yoil = getYoil(year, month, day);

    // 4. ModelAndView에 작업한 결과를 저장
    mv.addObject("year",  year);
    mv.addObject("month", month);
    mv.addObject("day",   day);
    mv.addObject("yoil", yoil);

    // 5. 작업 결과를 보여줄 뷰의 이름을 지정
    mv.setViewName("yoil");

    // 6. ModelAndView를 반환
    return mv;
```

관심사의 분리, MVC 패턴 - 이론
6. 컨트롤러 메서드의 반환타입

[String] 뷰 이름을 반환

```
@RequestMapping("/getYoil")
public String main(int year, int month, int day) {
    // 2. 처리
    Calendar cal = Calendar.getInstance();
    cal.set(year, month - 1, day);

    int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
    char yoil = " 일월화수목금토".charAt(dayOfWeek);

    return "yoil"; // /WEB-INF/views/yoil.jsp
}
```

[void] 맵핑된 url의 끝단어가 뷰 이름

```
@RequestMapping("/yoil")  // /WEB-INF/views/yoil.jsp
public void main(int year, int month, int day, Model m) {
    // 2. 처리
    Calendar cal = Calendar.getInstance();
    cal.set(year, month - 1, day);

    int dayOfWeek = cal.get(Calendar.DAY_OF_WEEK);
    char yoil = " 일월화수목금토".charAt(dayOfWeek);

    m.addAttribute("year",  year);
    m.addAttribute("month", month);
    m.addAttribute("day",   day);
    m.addAttribute("yoil",  yoil);
//  return "yoil";
}
```

[ModelAndView] Model과 뷰 이름을 반환

```
@RequestMapping("/getYoil")
public ModelAndView main(int year, int month, int day) {
    // 1. ModelAndView를 생성
    ModelAndView mv = new ModelAndView();

    // 2. 처리
    char yoil = getYoil(year, month, day);

    // 3. ModelAndView에 작업한 결과를 저장
    mv.addObject("year",  year);
    mv.addObject("month", month);
    mv.addObject("day",   day);
    mv.addObject("yoil", yoil);

    // 4. 작업 결과를 보여줄 View의 이름을 지정
    mv.setViewName("yoil");

    // 5. ModelAndView를 반환
    return mv;
}
```