

JSP MVC

일반적인 JSP 프로그래머가 구현하는 방식의 코드

/*--- 입력 코드 ---*/	/*--- 입력 코드 ---*/
<% String num_ = request.getParameter("n"); %>	<% String num_ = request.getParameter("n"); ...
/*--- 출력 코드 ---*/	if(num%2 != 0) model = "홀수"; else model = "짝수"; %>
.....	/*--- 출력 코드 ---*/
.....
<% if(num%2 != 0) { %> 홀수입니다. <% } else { %> 짝수입니다. <%}%>	<%=model%>입니다.

출력을 가깝게 만들기 위한 코드작성 방식

/*--- 입력 코드 ---*/	입력과 제어를 담당 : Controller [자바 코드]
<% String num_ = request.getParameter("n"); ... if(num%2 != 0) model = "홀수"; else model = "짝수"; %>	<div style="border: 1px solid black; padding: 5px; display: inline-block;">출력 데이터 : Model</div> <div style="text-align: center; margin: 10px 0;">↓ 분리 ↑</div>
/*--- 출력 코드 ---*/	
.....	[HTML 코드] 출력 담당 : View
<%=model%>입니다.	

```

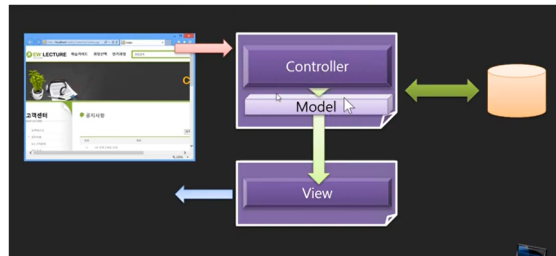
1  <%@ page language="java" contentType="text/html; charset=UTF-8
2  pageEncoding="UTF-8"%>
3  <%
4      int num = 0;
5      String num_ = request.getParameter("n");
6      if(num_ != null && !num_.equals(""))
7          num = Integer.parseInt(num_);
8
9      String result;
10     if(num%2 != 0){
11         result = "홀수입니다";
12     }
13     else{
14         result = "짝수입니다";
15     }
16 %>
17 <!DOCTYPE html>
18 <html>
19 <head>
20 <meta charset="UTF-8">
21 <title>Insert title here</title>
22 </head>
23
24 <body>
25     <%=result %>입니다.
26 </body>
27 </html>

```

MVC Model 1 vs Model 2

- 모델 1 : 컨트롤러와 뷰가 물리적으로 분리되지 않은 방식
- 모델 2 : 컨트롤러와 뷰가 물리적으로 분리된 방식
 - 사용자 요청이 들어오게되면 디스패처 서블릿이 적절한 Controller를 찾아서 수행하게하는 방식

model 2 : 컨트롤러와 뷰가 물리적으로 분리된 방식



•

```
@WebServlet("/spag")
public class Spag extends HttpServlet{

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) {
        int num = 0;
        String num_ = request.getParameter("n");
        if(num_ != null && !num_.equals(""))
            num = Integer.parseInt(num_);

        String result;
        if(num%2 != 0)
            result = "홀수입니다";
        else
            result = "짝수입니다";

        request.setAttribute("result", result);

        //redirect
        //forward
        RequestDispatcher dispatcher =
            request.getRequestDispatcher("spag.jsp");
        dispatcher.forward(request, response);
    }
}
```

- dispatcher를 통해 포워딩을 하여 request와 response를 공유할 수 있다. (현재 작업했던 내용들을 담고있는 것을 jsp로 이어주는것)
- request.setAttribute를 통해 result 값을 저장하여 dispatcher를 통해 포워딩하여 spag.jsp로 넘겨준다.

```

1 <%@ page language="java" contentType="text/html; charset=
2   pageEncoding="UTF-8"%>
3
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8   <title>Insert title here</title>
9 </head>
10
11 <body>
12   <%=request.getAttribute("result") %>
13 </body>
14 </html>

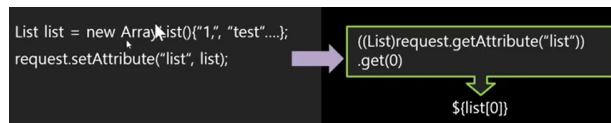
```

- 넘겨받은 spag.jsp 는 request.getAttribute를 통해 result 값을 출력한다.

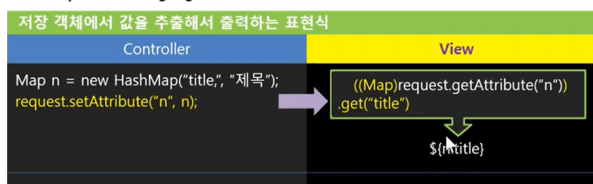
EL 표기법

- View에서 자바코드를 사용하지 않는 것이 MVC 패턴에서 바람직하다
-

EL(Expression Language)



EL(Expression Language)



```

@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
    int num = 0;
    String num_ = request.getParameter("n");
    if(num_ != null && !num_.equals(""))
        num = Integer.parseInt(num_);

    String result;
    if(num%2 != 0)
        result = "홀수입니다";
    else
        result = "짝수입니다";

    request.setAttribute("result", result);

    String[] names = {"taein", "dragon"};
    request.setAttribute("names", names);

    Map<String, Object> notice = new HashMap<String, Object>();
    notice.put("id", 1);
    notice.put("title", "EL은 좋아요");
    request.setAttribute("notice", notice);

    //redirect
    //forward
    RequestDispatcher dispatcher =
        request.getRequestDispatcher("spag.jsp");
    dispatcher.forward(request, response);
}

```

```

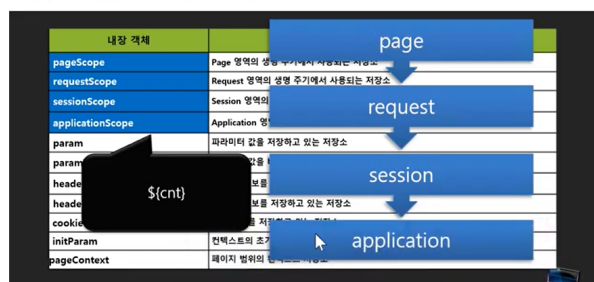
<!-- pageEncoding="UTF-8"%>
-->
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<%=request.getAttribute("result") %>
${result}<br>
${names[1]}<br>
${notice.title}<br>
</body>
</html>

```

EL의 데이터 저장소

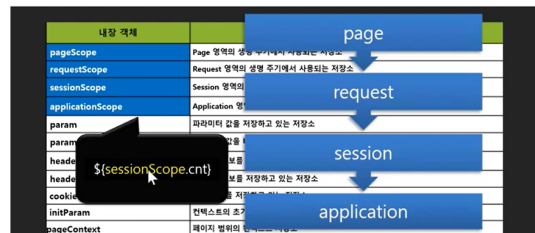
•

저장 객체에서 값을 추출하는 순서



- EL를 이용해서 page 객체, request 객체, session 객체, application 객체를 뽑아낼 수 있다.
- `${}` 이 찾는 첫번째 저장소 공간은 page를 먼저 찾음, 찾게되면 가져오고 다른곳은 안 찾음 , 이후 request , session, application
- requestScope 를 쓰면 request 저장소에서만 찾아서 꺼내온다
 -

저장 객체에서 값을 추출하는 순서



클라이언트의 입력 값 추출

내장 객체	설명
pageScope	Page 영역의 생명 주기에서 사용되는 저장소
requestScope	Request 영역의 생명 주기에서 사용되는 저장소
sessionScope	Session 영역의 생명 주기에서 사용되는 저장소
applicationScope	Application 영역의 생명 주기에서 사용되는 저장소
param	파라미터 값을 저장하고 있는 저장소
paramValues	파라미터의 값을 저장하고 있는 저장소
header	Header 정보를 저장하고 있는 저장소
headerValues	Header 정보를 저장하고 있는 저장소
cookie	쿠키 정보를 저장하고 있는 저장소
initParam	컨텍스트의 초기화 파라미터를 저장하고 있는 저장소
pageContext	페이지 범위의 컨텍스트 저장소

EL 연산자

- 사용하고있는 html 자체가<>기호를 사용하고 있기 때문에 EL 연산자를 사용하는 것이 바람직하다.
-

EL 연산자

```

[] .
()
not ! empty
* / div % mod
+ -
< > <= >= lt gt le ge
== != eq ne
&& and
|| or
?:

```

EL Expression	Result
<code>\${1 > (4/2)}</code>	false
<code>\${4.0 >= 3}</code>	true
<code>\${100.0 == 100}</code>	true
<code>\${(10*10) ne 100}</code>	false
<code>`\${a} < `b`</code>	true
<code>`\${hip} gt `hit`</code>	false
<code>`\${4} > 3`</code>	true
<code>`\${1.2E4 + 1.4}`</code>	12001.4
<code>`\${3} div 4`</code>	0.75
<code>`\${10} mod 4`</code>	2

•

```

1  <%@ page language="java" contentType="text/html; charset=U
2  pageEncoding="UTF-8"%>
3
4  <!DOCTYPE html>
5  <html>
6  <head>
7  <meta charset="UTF-8">
8  <title>Insert title here</title>
9  </head>
10 <%
11   pageContext.setAttribute("result", "hello");//페이지 개
12   %>
13 <body>
14   <%=request.getAttribute("result") %>
15   ${requestScope.result}<br>
16   ${names[1]}<br>
17   ${notice.title}<br>
18   ${result}<br>
19   ${empty param.n? '값이 비어 있습니다.' : param.n}<br>
20   ${param.n == null || param.n == ''}<br>
21   ${param.n/2}<br>
22   ${header.accept}
23 </body>
24 </html>

```