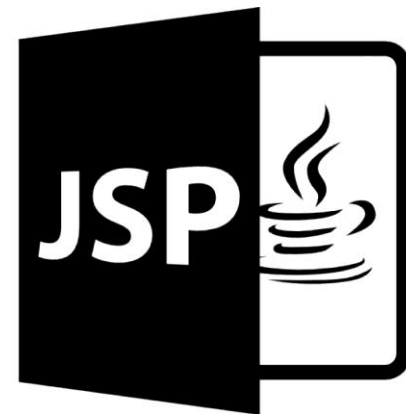


# 9장. MVC와 EL 언어



*Expression Language*



# EL(Expression Language)

## EL 언어

- MVC 패턴에 따라 뷰(view) 역할을 수행하는 JSP를 더욱 효과적으로 만들려는 목적으로 개발  
(개발 배경 : 코드와 태그가 섞여서 복잡해짐)
- JSP의 데이터를 표현할 때 스크립트릿 <% %>이나 표현식 <%= %>을 대체하기 위해 사용되는 언어이다.

```
<jsp:useBean id="test" class="TestBean" />  
<%= test.getName()>
```



```
${test.name}
```

- ① 표현 언어는 \$로 시작한다.
- ② 모든 내용은 '{표현식}'과 같이 표기한다.
- ③ 표현식에는 기본적으로 변수 이름, 혹은 '**객체\_이름**.멤버 변수\_이름' 구조로 이루어짐
- ④ 표현식에는 기본적인 연산을 할 수 있다.



## 표현 언어에서 사용할 수 있는 연산자

### 산술 연산자

연산자	기 능	연산자	기능
+	더하기	-	빼기
*	곱하기	/ or div	몫
% or mod	나머지		

### 비교 / 조건 연산자

연산자	기 능	연산자	기능
== 또는 eq	같다	!= 또는 ne	같지 않다.
< 혹은 lt	좌변이 우변보다 작다.	> 혹은 gt	좌변이 우변보다 크다
<= 혹은 le	좌변이 우변보다 같거나 작다.	>= 혹은 ge	좌변이 우변보다 크거나 같다
a ? x : y	a가 참이면 x, 거짓이면 y를 반환한다.		

# EL 언어

## 논리 연산자

연산자	기 능
&& 또는 and	AND 연산
또는 or	OR 연산
! 또는 not	NOT 연산

## Empty 연산자

연산자	기 능
empty <값>	<값>이 null 이거나 빈 문자열이면 true 를 반환 \${ <b>empty</b> param.n}

# EL 언어

## 표현 언어 실습

`<h3>문자, 숫자 데이터 표현</h3>`

`${300}<br>`

`${"감사합니다."}<br>`

`${10+1}<br>`

`${300+"10"}<br>`

`<h3>산술 연산자</h3>`

`\${7 + 4} : ${7 + 4}<br>`

`\${7 - 4} : ${7 - 4}<br>`

`\${7 * 4} : ${7 * 4}<br>`

`\${7 / 4} : ${7 / 4}<br>`

`\${7 % 4} : ${7 % 4}<br>`

문자, 숫자 데이터 표현

300

감사합니다.

11

310

산술 연산자

`${7 + 4} : 11`

`${7 - 4} : 3`

`${7 * 4} : 28`

`${7 / 4} : 1.75`

`${7 % 4} : 3`

# EL 언어

## 표현 언어 실습

<h3>비교 연산자</h3>

\\${10==10} : \\${10==10}<br>

\\${10 eq 10} : \\${10 eq 10}<br>

\\${"face"=="face"} : \\${"face"=="face"}<br>

\\${"face" eq "face"} : \\${"face" eq "face"}<br>

\\${10 < 20} : \\${10 < 20}<br>

\\${10 lt 20} : \\${10 lt 20}<br>

\\${10 > 20} : \\${10 > 20}<br>

\\${10 gt 20} : \\${10 gt 20}<br>

<h3>논리 연산자</h3>

\\${(4==4) && (7==7)} : \\${(4==4) && (7==7)}<br>

\\${(4==4) and (7!=7)} : \\${(4==4) and (7!=7)}<br>

\\${(4!=4) || (7==7)} : \\${(4!=4) || (7==7)}<br>

\\${(4==4) or (7!=7)} : \\${(4==4) or (7!=7)}<br>

\\${!(4==4)} : \\${!(4==4)}<br>

\\${not(4==4)} : \\${not(4==4)}<br>

### 비교 연산자

\\${10==10} : true

\\${10 eq 10} : true

\\${"face"=="face"} : true

\\${"face" eq "face"} : true

\\${10 < 20} : true

\\${10 lt 20} : true

\\${10 > 20} : false

\\${10 gt 20} : false

### 논리 연산자

\\${(4==4) && (7==7)} : true

\\${(4==4) and (7!=7)} : false

\\${(4!=4) || (7==7)} : true

\\${(4==4) or (7!=7)} : true

\\${!(4==4)} : false

\\${not(4==4)} : false

## 표현 언어 실습

- 예제는 첫번째 jsp 요청을 처리하는 파일과, 두번째로 Member 클래스를 만들어 <jsp:useBean>태그를 이용하여 처리 결과를 출력하는 2개의 방법으로 실습한다.

### 프로그램 소스 목록

파일 이름	역 할
Member.java	회원 정보를 제공하는 빈즈클래스로 jsp에 데이터를 공급
memberForm.jsp	회원 정보를 등록하기 위한 jsp 파일
member01, member02, member01_el, member02_el	회원 정보를 출력하기 위한 jsp 파일 Jsp 기본 문법을 사용한 파일과 EL로 구현한 파일 비교

## 회원 가입 처리 – 1. 입력 폼과 처리 페이지(param 객체 실습)

회원 가입	
아이디	<input type="text" value="chu17"/>
패스워드	<input type="password" value="...."/>
이 름	<input type="text" value="추신수"/>
<input type="button" value="가입"/> <input type="button" value="취소"/>	



회원 정보		
아이디	패스워드	이 름
chu17	1234	추신수



## 회원 가입 처리 - 입력 폼과 요청 처리 페이지(param 객체 실습)

```
<div id="container">
  <h2>회원 가입</h2>
  <hr>
  <form action="member01_process_el.jsp" method="post">
    <table>
      <tr>
        <td>아이디</td>
        <td><input type="text" name="id"></td>
      </tr>
      <tr>
        <td>비밀번호</td>
        <td><input type="password" name="passwd"></td>
      </tr>
      <tr>
        <td>이 름</td>
        <td><input type="text" name="name"></td>
      </tr>
      <tr>
        <td colspan="2">
          <input type="submit" value="가입">
          <input type="reset" value="취소">
        </td>
      </tr>
    </table>
  </form>
```

member01.jsp

## 회원 가입 처리 - 입력 폼과 요청 처리 페이지(param 객체 실습)

```
@charset "UTF-8";

#container{width: 800px; margin: 0 auto; text-align: center;}

table{width: 400px; margin: 0 auto;}
table, th, td{border: 1px solid #ccc; border-collapse: collapse;}
table th, td{height: 40px;}
table input{height: 25px;}
```

# EL 언어

## 스크립트 태그 VS EL 표기 - 비교

member01\_process.jsp

member01\_process\_el.jsp

```
<%
    request.setCharacterEncoding("utf-8");

    String id = request.getParameter("id");
    String passwd = request.getParameter("passwd");
    String name = request.getParameter("name");
%>
<body>
    <div id="container">
        <h2>회원 정보</h2>
        <hr>
        <table>
            <tr>
                <th>아이디</th>
                <th>패스워드</th>
                <th>이 름</th>
            </tr>
            <tr>
                <td><%=id %></td>
                <td><%=passwd %></td>
                <td><%=name %></td>
            </tr>
        </table>
    </div>
</body>
```

```
<body>
    <div id="container">
        <h2>회원 정보</h2>
        <hr>
        <table>
            <tr>
                <th>아이디</th>
                <th>패스워드</th>
                <th>이 름</th>
            </tr>
            <tr>
                <td>${param.id}</td>
                <td>${param.passwd}</td>
                <td>${param.name}</td>
            </tr>
        </table>
    </div>
</body>
```

## 회원 가입 처리 – 2. 입력 폼과 Bean 객체를 사용한 처리

```
public class Member {  
    private String id;  
    private String passwd;  
    private String name;  
  
    public String getId() {  
        return id;  
    }  
    public void setId(String id) {  
        this.id = id;  
    }  
    public String getPasswd() {  
        return passwd;  
    }  
    public void setPasswd(String passwd) {  
        this.passwd = passwd;  
    }  
    public String getName() {  
        return name;  
    }  
    public void setName(String name) {  
        this.name = name;  
    }  
}
```

# EL 언어

member02\_process.jsp

```
<jsp:useBean id="member" class="member.Member" />
<jsp:setProperty property="id" name="member"/>
<jsp:setProperty property="passwd" name="member"/>
<jsp:setProperty property="name" name="member"/>
<body>
    <div id="container">
        <h2>회원 정보</h2>
        <hr>
        <table>
            <tr>
                <th>아이디</th>
                <th>패스워드</th>
                <th>이름</th>
            </tr>
            <tr>
                <td><%=member.getId() %></td>
                <td><%=member.getPasswd() %></td>
                <td><%=member.getName() %></td>
            </tr>
        </table>
    </div>
</body>
```

member02\_process\_el.jsp

```
<jsp:useBean id="member" class="member.Member" />
<jsp:setProperty property="id" name="member"/>
<jsp:setProperty property="passwd" name="member"/>
<jsp:setProperty property="name" name="member"/>
<body>
    <div id="container">
        <h2>회원 정보</h2>
        <hr>
        <table>
            <tr>
                <th>아이디</th>
                <th>패스워드</th>
                <th>이름</th>
            </tr>
            <tr>
                <td>${member.id}</td>
                <td>${member.passwd}</td>
                <td>${member.name}</td>
            </tr>
        </table>
    </div>
</body>
```



## pageContext 객체 사용 실습

로그인	
아이디	<input type="text"/>
비밀번호	<input type="password"/>
<input type="button" value="로그인"/> <input type="button" value="취소"/>	

[회원 가입하기](#)

[회원 가입하기](#)

/Chapter13

[회원 가입하기](#)



회원 가입	
아이디	<input type="text"/>
비밀번호	<input type="password"/>
이름	<input type="text"/>
<input type="button" value="가입"/> <input type="button" value="취소"/>	

## pageContext 객체 사용 실습

pageContext 객체는 javax.servlet.jsp.PageContext 클래스를 상속해 웹 컨테이너가 JSP 실행 시 자동으로 생성해서 제공하는 내장 객체이다.

**<a>태그를 이용해 서블릿이나 JSP를 요청하는 방법.**

1. `<a href="http://localhost:8181/Chapter13/member02.jsp">회원가입</a>`
2. `<a href <%=request.getContextPath() %>/member02.jsp">회원가입</a>`

- 1 방법은 컨텍스트 이름(Chapter13)이 바뀌면 일일이 찾아서 수정해야 하는 단점이 있다.
2. 방법은 1방법의 단점을 해결했으나 자바코드가 사용되어 화면이 복잡해 질 수 있다.

### 3. Jstl을 사용하는 방법

`<a href =${contextPath.request.contextPath }/member02.jsp">회원가입</a>`



## pageContext 객체 사용 실습

```
<div id="container">
  <h2>로그인</h2>
  <hr>
  <form action="login_process.jsp" method="post">
    <table>
      <tr>
        <td>아이디</td>
        <td><input type="text" name="id"></td>
      </tr>
      <tr>
        <td>비밀번호</td>
        <td><input type="password" name="passwd"></td>
      </tr>
      <tr>
        <td colspan="2">
          <input type="submit" value="로그인">
          <input type="reset" value="취소">
        </td>
      </tr>
    </table>
  </form>
  <p><a href="http://localhost:8181/Chapter13/member02.jsp">회원 가입하기</a>
  <p><a href="<%=request.getContextPath() %>/member02.jsp">회원 가입하기</a>
  <p><%=request.getContextPath() %>
  <p><a href="${pageContext.request.contextPath}/member02.jsp">회원 가입하기</a>
</div>
```

member03.jsp





## 표현 언어 실습

- 예제는 Product 클래스를 만들어 JSP에서 <jsp:useBean> 액션과 표현 언어를 사용하는 구조로 구성되어 있다.

### 프로그램 소스 목록

파일 이름	역 할
Product.java	상품 정보를 제공하는 빈즈클래스로 jsp에 데이터를 공급
productList.jsp	상품 목록을 출력하기 위한 jsp 파일
selProduct.jsp	productList에서 item을 선택하고 <확인> 버튼을 누르면 호출되는 jsp로 표현언어를 이용해 데이터 출력

## 표현 언어 실습



# EL 언어

## 표현 언어 실습

```
package org.bean;

public class Product {
    private String[] productList = {"수박", "참외", "포도", "토마토"};
    private int num1 = 10;
    private int num2 = 20;

    public String[] getProductList() {
        return productList;
    }
    public int getNum1() {
        return num1;
    }
    public int getNum2() {
        return num2;
    }
}
```

Product.java



# EL 언어

```
<title>상품 목록</title>
<style>
    #container{width: 80%; margin: 0 auto; text-align: center;}
</style>
</head>
<jsp:useBean id="product" class="org.bean.Product" scope="session"/>
<body>
    <div id="container">
        <h2>상품 목록</h2>
        <hr>
        <form action="./selProduct.jsp" method="get">
            <select name="select">
                <%
                    for(String item : product.getProductList()){
                        out.println("<option>" + item + "</option>");
                    }
                %>
            </select>
            <input type="submit" value="선택">
        </form>
    </div>
</body>
```

productList.jsp



selProduct.jsp

```
<title>상품 선택</title>
<style>
    #container{width: 80%; margin: 0 auto; text-align: center;}
</style>
</head>
<body>
    <div id="container">
        <h2>상품 선택</h2>
        <hr>
        <p>1. 선택한 상품은: ${param.select}</p>
        <p>2. num1 + num2 = ${product.num1 + product.num2}</p>
    </div>
</body>
```

# MVC란?

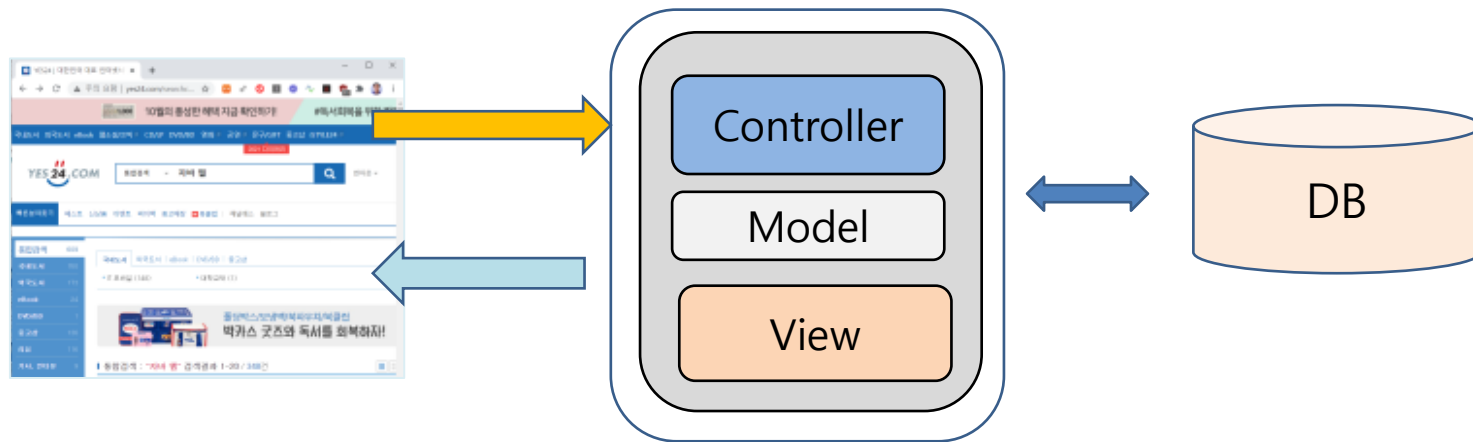
## MVC란?

Model, View, Controller의 약자로 웹 애플리케이션을 비즈니스 로직, 프레젠테이션 로직, 데이터 로 분리하는 디자인 패턴이다.

모델2 방식이라고 하며 클라이언트의 요청 처리, 응답 처리, 로직 처리 부분을 모듈화한 구조이다.

**모델 1 방식 : 컨트롤러와 뷰가 물리적으로 분리되지 않은 방식**

**일반적인 JSP로 구현하는 방식**



# MVC란?

## 모델 1 방식 : 일반적인 JSP로 구현하는 방식

```
<title>mvc 예제</title>
</head>
```

mvc.jsp

```
<%
    int num = 0;
    if(request.getParameter("num") != null){
        num = Integer.parseInt(request.getParameter("num"));
    }

    String result = null;
    if(num % 2 != 0)
        result = "홀수";
    else
        result = "짝수";
%>
```

**Controller**  
[자바코드]

```
<body>
    <%=result %>입니다.
</body>
</html>
```

**Model** - 출력데이터

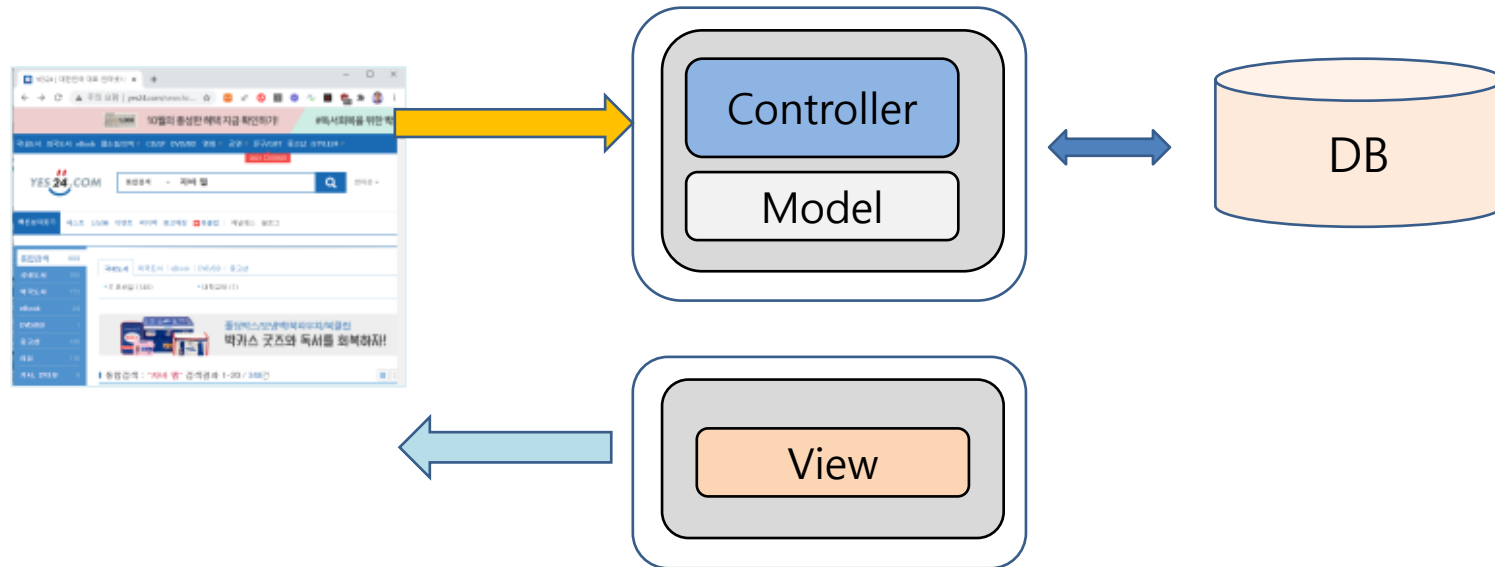
**View**  
[HTML 코드]



# MVC란?

Model 2 : 컨트롤러와 뷰가 물리적으로 분리된 방식

컨트롤러를 서블릿으로 만들고, 뷰는 JSP로 작성





# MVC 웹 애플리케이션

## 모델2 방식 - MVC 패턴 구현 방법

### 1. **web.xml** 파일에 서블릿 구성하기

<servlet>은 웹 애플리케이션에서 사용될 기본 서블릿 객체와 매개변수를 설정하는 요소로 형식은 다음과 같다.

```
<servlet>
  <servlet-name>서블릿 이름</servlet-name>
  <servlet-class>서블릿 클래스 </servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>서블릿 이름</servlet-name>
  <url-pattern>요청할 URL 패턴 </ url-pattern >
</servlet-mapping>
```



# MVC 웹 애플리케이션

## 2. 컨트롤러 생성하기

컨트롤러는 뷰와 모델 간의 인터페이스 역할을 하여 웹 브라우저의 모든 요청 URL을 받아들이고 요청 URL과 함께 전달되는 요청 파라미터를 받아 처리하는 서블릿 클래스이다.

```
public class 서블릿 이름 extends HttpServlet {  
  
    protected void doGet(HttpServletRequest request, HttpServletResponse  
        response) throws ServletException, IOException {  
        //Get 방식으로 전송되는 요청을 처리  
    }  
  
    protected void doPost(HttpServletRequest request, HttpServletResponse  
        response) throws ServletException, IOException {  
        //Post 방식으로 전송되는 요청을 처리  
    }  
}
```



# MVC 웹 애플리케이션

## 3. 모델 생성하기

모델은 웹 애플리케이션의 비즈니스 로직을 포함하는 데이터로 웹 애플리케이션의 상태를 나타낸다.

```
request.setAttribute("message", "Hello Java Server Page!!");
```

## 4. 페이지 이동하기(포워딩)

서블릿 클래스에서 웹 브라우저로부터 요청된 처리 결과를 보여줄 응답 페이지로 이동하는 형식으로 뷰 페이지가 이동해도 처음에 요청된 URL을 계속 유지하기 위해 포워딩 방식을 사용한다.

```
RequestDispatcher rd = request.getRequestDispatcher("jsp 페이지");  
rd.forward(request, response);
```



# MVC 웹 애플리케이션

## mvc 실습 - xml 등록 방식

Controller

```
public class CalcServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int num = 0;
        if(request.getParameter("num") != null) {
            num = Integer.parseInt(request.getParameter("num"));
        }

        String result = "";
        if(num % 2 == 0) {
            result = "짝수";
        } else {
            result = "홀수";
        }

        //model - data 저장
        request.setAttribute("result", result);

        //포워딩 - jsp페이지로 보내줌
        RequestDispatcher dispatcher = request.getRequestDispatcher("/mvc/calc.jsp");
        dispatcher.forward(request, response);
    }
}
```

Com.mvc.CalcServlet.java

Model



# MVC 웹 애플리케이션

## 서블릿 - xml 등록

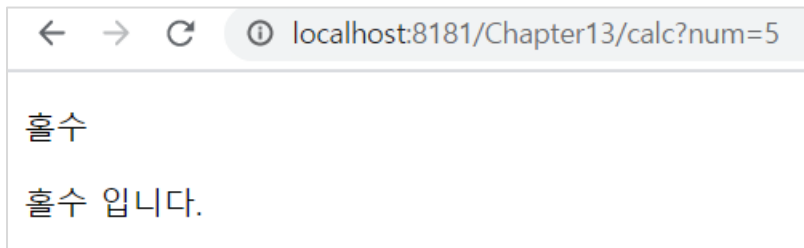
WEB-INF/web.xml

```
<servlet>
  <servlet-name>calcServlet</servlet-name>
  <servlet-class>com.mvc.CalcServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>calcServlet</servlet-name>
  <url-pattern>/mvc/calc</url-pattern>
</servlet-mapping>
```



# MVC와 EL 언어



mvc/calc.jsp

```
<title>짝수/홀수 판정</title>
</head>
<body>
    <!-- <%=request.getAttribute("result") %>입니다.<br> -->
    <!-- servlet에서 보낸 result(model) 출력 -->
    <p>결과 : ${result}입니다.
</body>
</html>
```

**VIEW**



## 표현 언어에서 사용할 수 있는 내장 객체(데이터 저장소)

내장 객체	기 능
pageScope	page 영역의 생명 주기에서 사용되는 저장소
<b>requestScope</b>	request 영역의 생명 주기에서 사용되는 저장소
sessionScope	session 영역의 생명 주기에서 사용되는 저장소
applicationScope	application 영역의 생명 주기에서 사용되는 저장소
<b>param</b>	request.getParameter("name")로 얻을 수 있는 값들이다. \${param.name}으로 사용한다.
paramValues	request.getParameterValues("name")로 얻을 수 있는 값들이다. \${paramValues.name}으로 사용한다. - 배열
<b>pageContext</b>	page 범위의 컨텍스트 저장소 \${pageContext.request.contextPath}
cookie	Cookie 정보를 저장하고 있는 저장소 – 클라이언트에 저장됨

# MVC와 EL 언어

## 컨트롤러 만들기 - 애너테이션(@) 방식

Controller

```
@WebServlet("/mvc02")
public class Mvc02 extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

        //배열 자료
        String[] name = {"김산", "이강", "정들"};

        request.setAttribute("names", name); //model : data
        //request - 서버의 저장소(서블릿과 jsp 둘 사이를 연결 공유할때)

        //ArrayList 자료
        ArrayList<Integer> lotto = new ArrayList<>();
        lotto.add(15);
        lotto.add(77);
        lotto.add(4);
        lotto.add(83);
        lotto.add(69);
        lotto.add(33);

        //model 저장
        request.setAttribute("lotto", lotto);
    }
}
```

Model





# MVC와 EL 언어

## 컨트롤러 만들기 - 애너테이션(@) 방식

```
//HashMap 자료  
Map<String, Object> car = new HashMap<>();  
car.put("brand", "sonata");  
car.put("cc", 2500);  
//model  
request.setAttribute("cars", car);
```

```
//포워딩  
RequestDispatcher dispatcher =  
    request.getRequestDispatcher("mvc02.jsp");  
dispatcher.forward(request, response);  
}
```

페이지이동



# MVC와 EL 언어

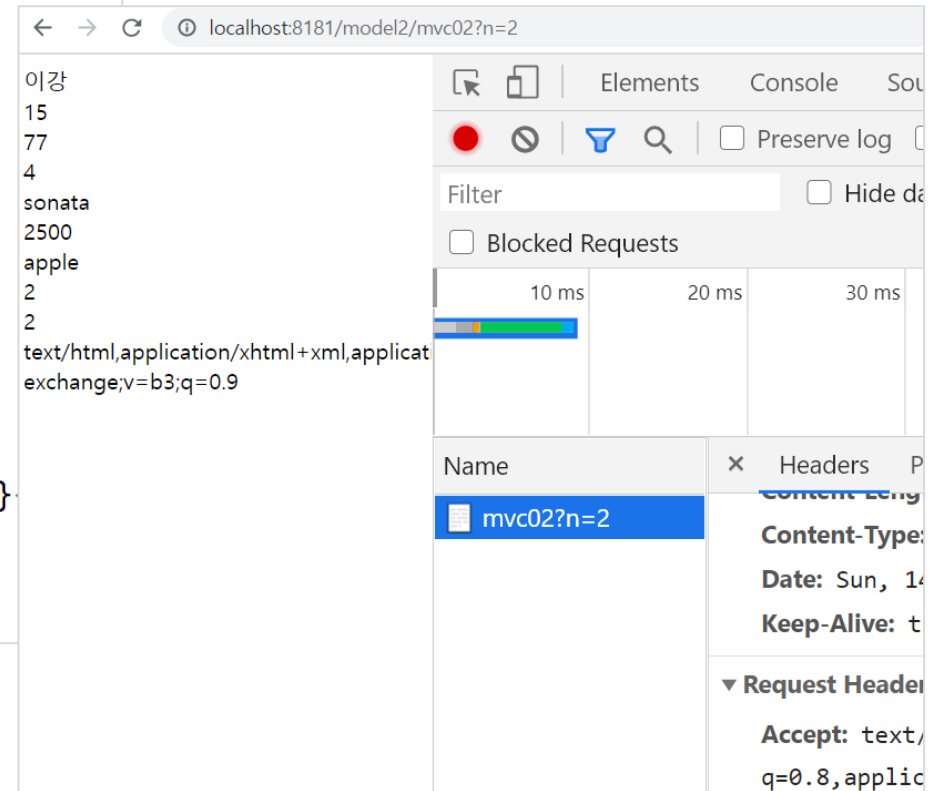
```
<title>EL 예제</title>
</head>
<%
    pageContext.setAttribute("fruit", "apple");
%>
<body>
    ${names[1]}<br>

    ${lotto[0]}<br>
    ${lotto[1]}<br>
    ${lotto[2]}<br>

    ${cars.brand}<br>
    ${cars.cc}<br>

    ${fruit}<br>
    ${param.n}<br>
    ${empty param.n ? '값이 비어있습니다.' : param.n}
    ${header.accept}
</body>
</html>
```

VIEW



# MVC와 EL 언어

```
<body>
  <!-- 특정 요소 보기 -->
  ${names[0]}<br>

  <!-- 목록 보기 -->
  <c:forEach var="name" items="${names}">
    ${name}<br>
  </c:forEach>
</body>
```