JSP(JavaServer Page)

EL / JSTL

① EL 표현식

EL은 "$"와 "{}"를 사용하여 값을 표현한다. {}안에는 값으로 표현되는 것만 와야 한다.

② EL 기본객체(내장객체)

EL에는 jsp/servlet의 객체(pageContext제외)의 속성값이나 파라미터 값등을 쉽게 표현하기 위해서 기본객체를 제공한다.

EL / JSTL

기본 객체	설 명
pageContext	pageContext 객체 참조
pageScope	page 영역 참조
requestScope	request 영역 참조
sessionScope	session 영역 참조
applicationScope	application 영역 참조
param	요청 파라미터의 값을 String으로, request.getParameter()의 결과와 동일
paramValue	요청 파라미터의 값을 String[]으로, request.getParameterValues()의 결과와 동일
header	요청 header 정보, request.getHeader()의 결과와 동일
headerValues	요청 header 정보를 배열, request.getHeaders()의 결과와 동일
cookie	쿠키 참조, request.getCookies()결과와 동일
initParam	컨텍스트의 초기화 파라미터, application.getInitParameter(이름)의 결과와 동일하다.

EL / JSTL

■ EL (Expression Language)

▼ 사용목적

<%= %> , out.println()과 같은 자바코드를 더 이상 사용하지 않고 좀더 간편하게 출력을 지원하기 위한 도구.배열이나 컬렉션에서도 사용되고, JavaBean의 프로퍼티에서도 사용.

▼ 문법

Attribute형식에서는 <%= cnt + 1 %>를 쓰지 않고 ${cnt + 1}로 쓰고
Parameter형식에서는 ${param.abc}으로 쓴다.
여기서 cnt는 자바에서는 변수 이름이고, EL 식에서는 Attribute의 이름으로 해석되는데.
값을 찾을때 Attribute는 작은 Scope에서 큰 Scope로 찾는다.
(page → request → session → application)

EL / JSTL

▼ 연산자

단어 연산자	기호 연산자
+	+
-	-
*	*
/	div
%	mod
&&	and
||	or
!	not
>	lt (less than)
<	gt (greater than)
>=	le (less or equal)
<=	ge (greater or equal)
==	eq (equal)
!=	ne (not equal)

EL / JSTL

▼ 내장객체

1) pageScope → 페이지Scope에 접근
2) request Scope → 리퀘스트Scope에 접근
3) sessionScope → 세션Scope에 접근
4) applicationScope → 어플리케이션Scope에 접근
5) param → 파라미터값 얻어올때 (1개의 Key에 1개의 Value)
6) paramValues → 파라미터값 배열로 얻어올때(1개의 Key에 여러개의 Value)
7) header → 헤더값 얻어올때 (1개의 Key에 1개의 Value)
8) headerValues → 헤더값 배열로 얻어올때 (1개의 Key에 여러개의 Value)
9) cookie → ${cookie. key값. value값}으로 쿠키값 조회
10) initParam → 초기 파라미터 조회
11) pageContext → 페이지컨텍스트 객체를 참조할때

EL / JSTL 정리

표현언어(EL)는 JSTL(JSP standard tag Library)에서 사용되는 언어이다.

1. 네가지 영역 속성을 사용할수 있다.
2. 관계 논리 수치 연산 가능하다
3. 클래스 메서드 호출 가능.
4. 표현언어만의 기본객체 제공.

사용방법은 ${ expr } 이다.
JSP 태그 어디에나 추가할 수있으며
HTML 태그 안에서도 사용할수있다. => ${ sessionScope.member.id}

EL / JSTL

▼ JSTL은?

JSP는 자신만의 태그를 추가할 수 있는 기능을 제공하고 있다.
<jsp:include>나 <jsp:usebean>과 같은 커스텀 태그처럼
연산이나 조건문이나 반복문인 if문, for문, DB를 편하게
처리할 수 있는것이 JSTL이다.

EL / JSTL

설명	taglib 사용
변수, 제어문등의 자바코드를 대체	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
XML 연동 처리	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/xml" %>
국제화 표준화 형식(날짜, 화폐 등)	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/fmt" %>
데이터베이스 연동 sql 처리	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/sql" %>
함수	<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/functions" %>

EL / JSTL

▼ 태그 종류

(1) Core (prefix : c)
→ 일반 프로그래밍에서 제공하는 것과 유사한 변수선언
→ 실행 흐름의 제어 기능을 제공
→ 페이지 이동 기술 제공
URI → http://java.sun.com/jsp/jstl/core

(2) Formatting (prefix : fmt)
→ 숫자, 날짜, 시간을 포매팅하는 기능을 제공
→ 국제화, 다국어 지원 기능 제공
URI → http://java.sun.com/jsp/jstl/fmt

(3) DataBase (prefix : sql)
→ DB의 데이터를 입력 / 수정 / 삭제 / 조회 하는 기능을 제공
URI → http://java.sun.com/jsp/jstl/sql

(4) XML (prefix : x)
→ XML문서를 처리할 때 필요한 기능 제공
URI → http://java.sun.com/jsp/jstl/xml

(5) Function (prefix : fn)
→ 문자열을 제공하는 함수 제공
URI → http://java.sun.com/jsp/jstl/functions

EL / JSTL

1. <c:set>

```
<c:set var="num" value="100">
<c:set var="num" value="100" scope="page">
```

2. <c:out>

```
<c:out value="출력할 값" default="value가 null값일 경우 출력할 값">
```

3. <c:remove>

```
<c:remove var="변수명" scope="영역"/>
```

4. <c:if>

```
<c:if test="조건식" var="조건을 검사하고 return되는 bool값을 저장할 변수"
      scope="bool 변수가 사용될 범위" />
```

5. <c:choose>

```
<c:when test="조건식">
<c:otherwise>
```

6. <c:foreach>

```
<c:foreach begin="시작값" end="끝값" step="증가값" var="count값이 저장될 변수" />
<c:foreach var="변수" items="배열or컬렉션" />
```

7. <c:forTokens>

```
<c:forTokens items="배열or컬렉션" delims="구분자" var="" begin="" end="" step="" />
```

8. <c:catch>

try문에 해당하고 catch에 해당하는 코드는 따로 작성해야 됨

EL / JSTL

1. <C:set>
자바의 int num = 100; 을 <c:set var="num" value="100">으로 바꿔 쓴 코드다.

2. <c:out>
역시 자바의 system.out.println(" 안녕하세요 ");을 간단하게 <c:out value=" 안녕하세요 ">로 변경 되었다. 또한 장점은 이 태그는 특수문자를 그대로 출력한다.

3. <c:remove>
한 영역의 변수명을 지우는 코드다. 만약에 영역을 생략할 경우 모든 영역의 변수가 삭제된다.
영역에는 아까 Attribute에서 (page → request → session → application) 순서의 영역을 가진다.

EL / JSTL

4. <c:if>
자바의 if - else 문과 동일하지만 JSTL에서는 else문이 없다. 여기서 scope 값을 생략하면 기본으로 page영역이 지정된다.

5. <c:choose> / <c:when> / <c:otherwise>
자바의 switch 구문과 if-else 구문을 혼합한 형태로 다수의 조건문을 걸고 싶을때 사용한다

EL / JSTL

6. <c:forEach>
자바에서는 for문으로 불리던게 JSTL에서는 forEach로 변경되었다. 배열이나 컬렉션, Map에 저장되어 있는 값들을 순서대로 처리 할때 사용되며,
<c:forEach var=" i " begin=" 1 " end=" 10 " step=" 1 "> ${ i }
</c:forEach>로 i가 1부터 10까지 1씩 증가한다는 구문을 쉽게 만들 수 있다.

7. <c:forTokens>
자바의 StringTokenizer 를 JSTL를 사용하면 아주 간편하게 사용할 수 있다.
<c:forTokens var="abc" items="안녕/하세요/hunit블로그/입니다"
delims="/" >

EL / JSTL

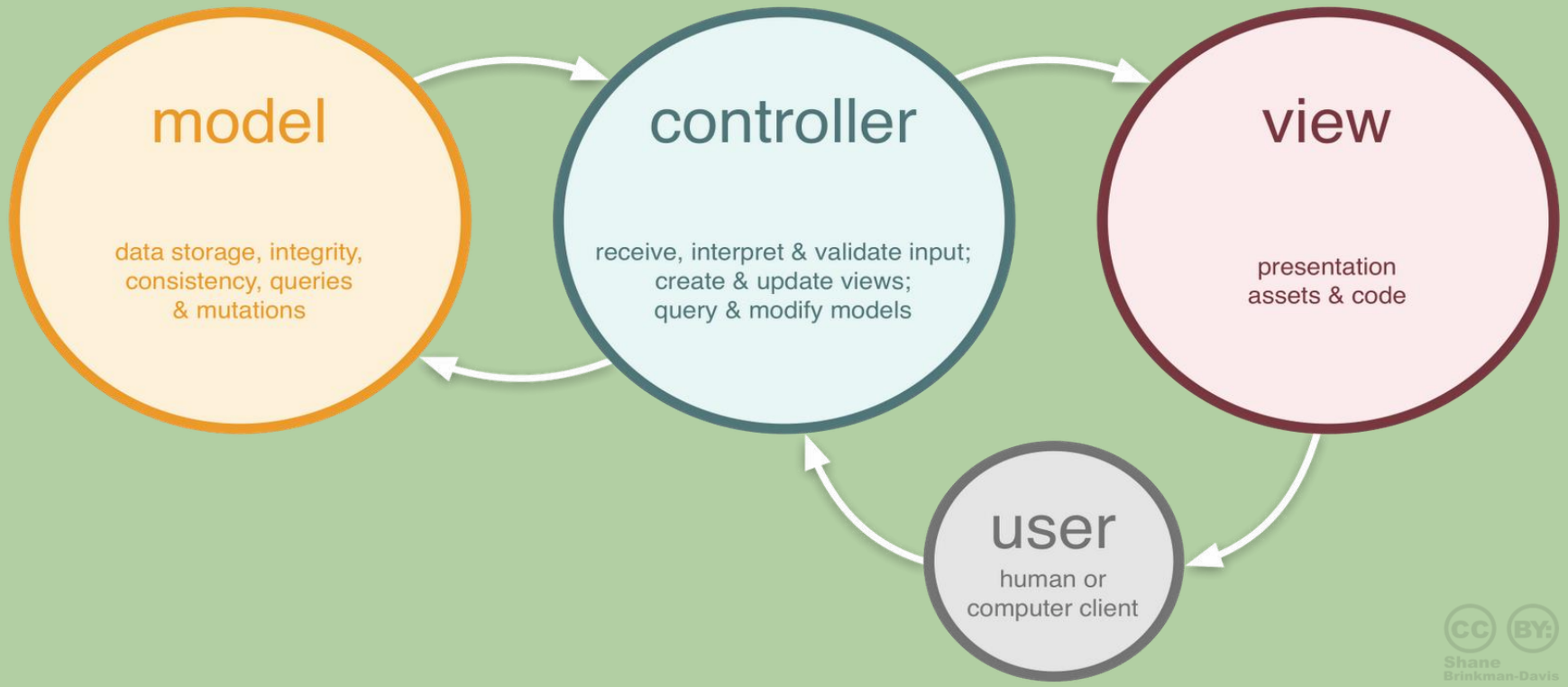
9. <c:redirect>는 아래와 같이 파라미터 값을 지정된 url로 보낸다.

```
<c:redirect url="boardList.jsp">
<c:param name="abc" value="안녕하세요" />
</c:redirect>
```

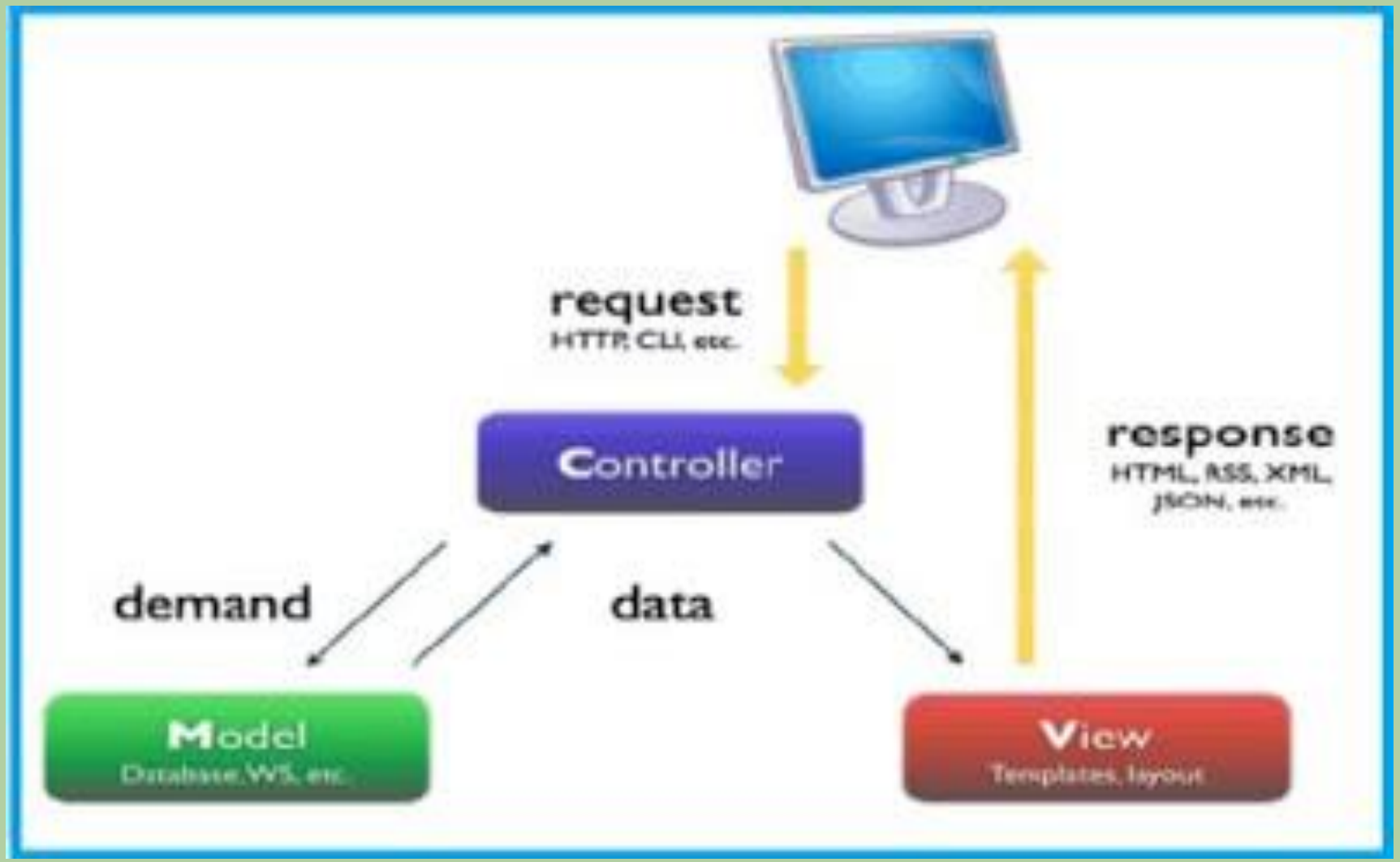
10. <c:import>는 <jsp:include>와 비슷하다.

11. <c: url>은 <c:set>과 비슷하며 GET방식으로 파라미터를 전달한다.

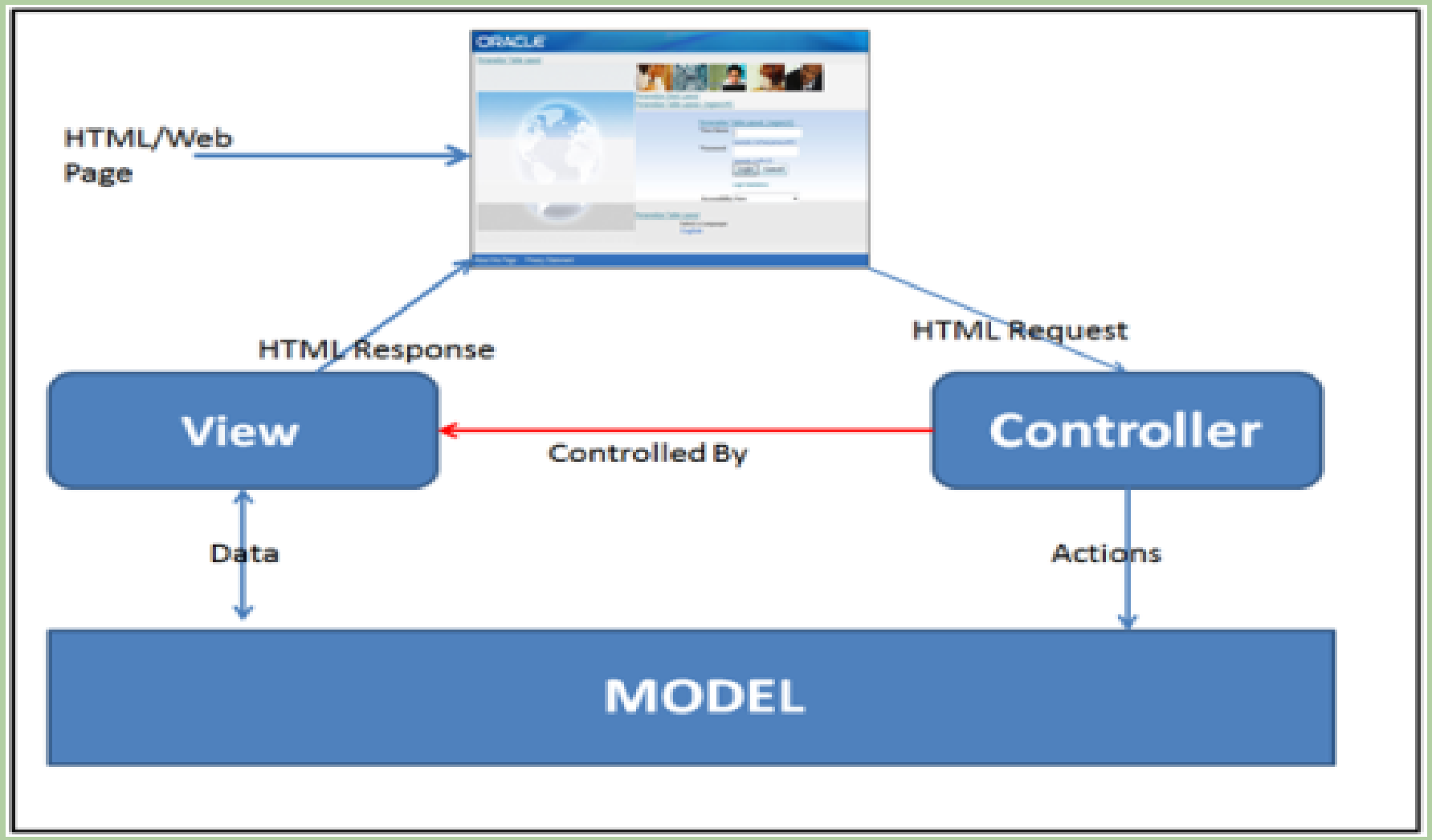
M(Model)V(View)C(Controller)



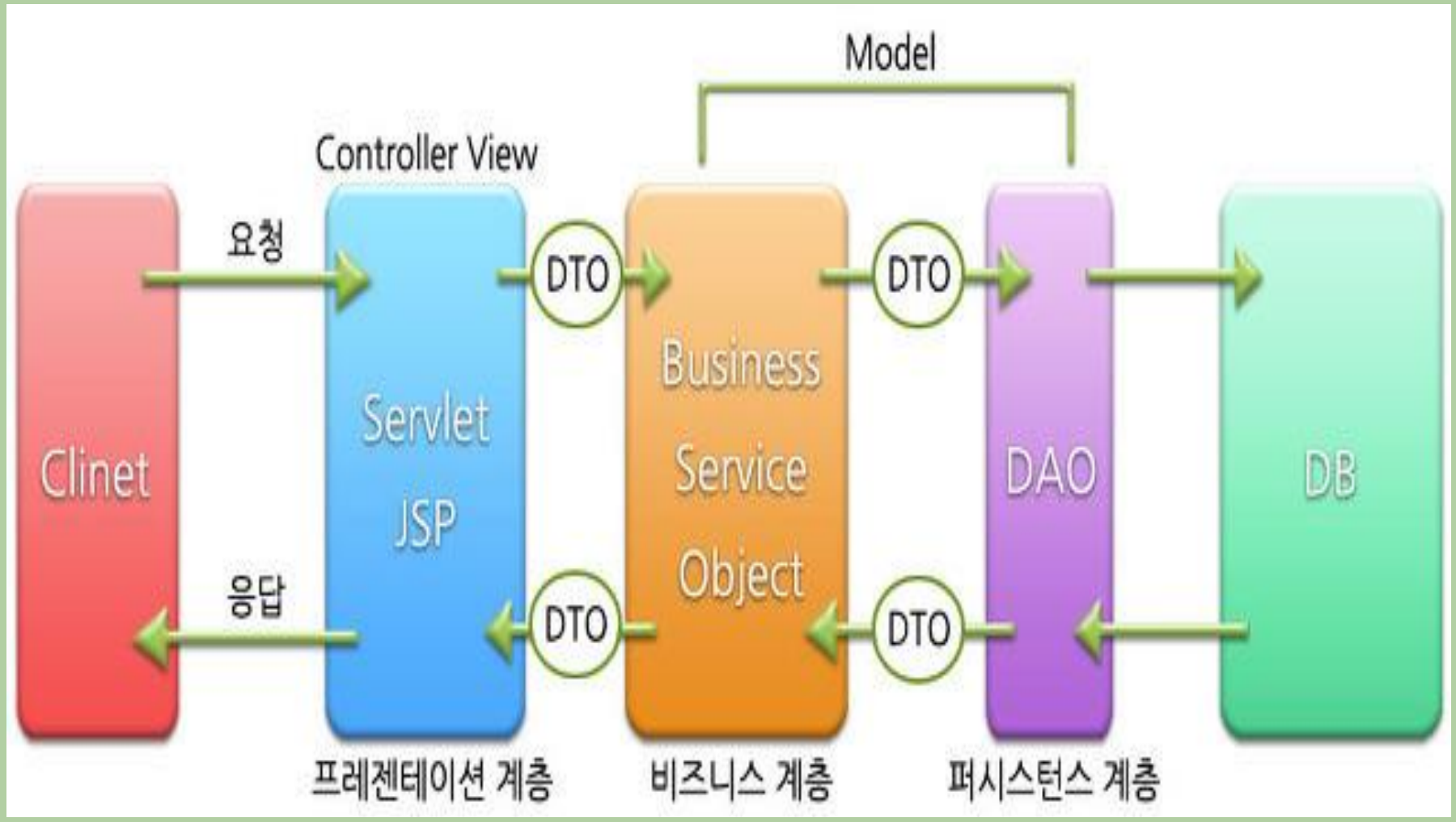
M(Model)V(View)C(Controller)



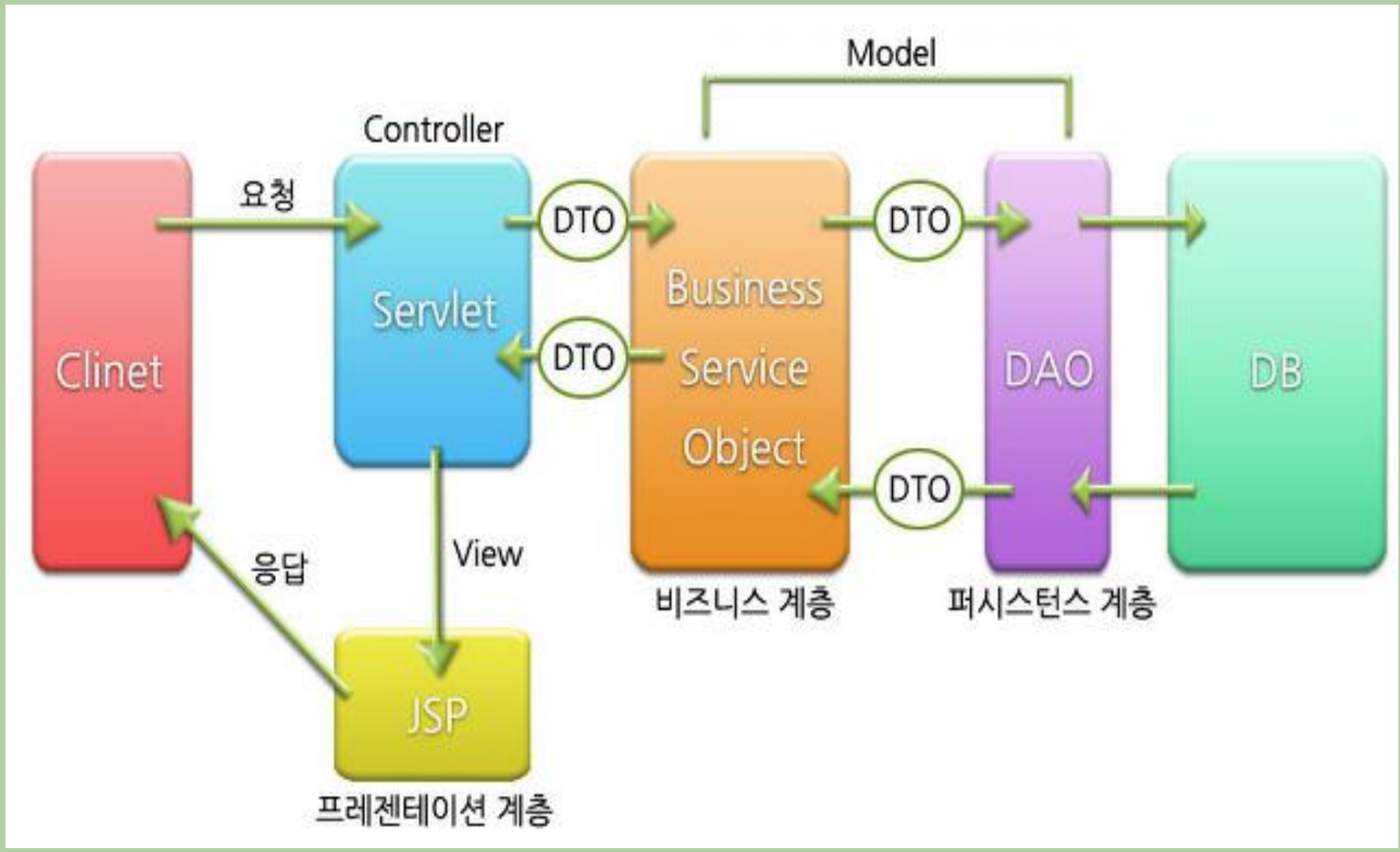
M(Model)V(View)C(Controller)



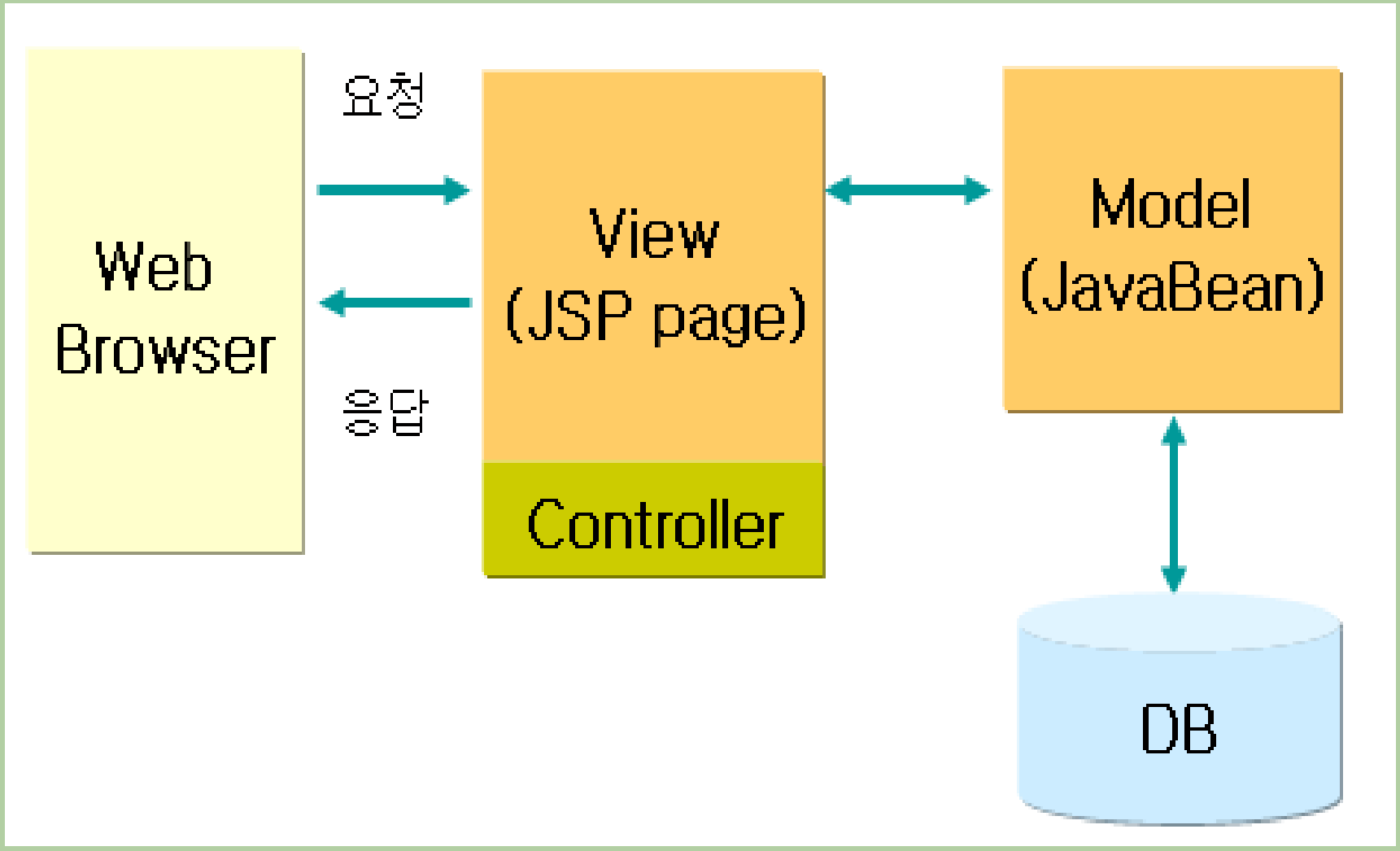
M(Model)V(View)C(Controller)



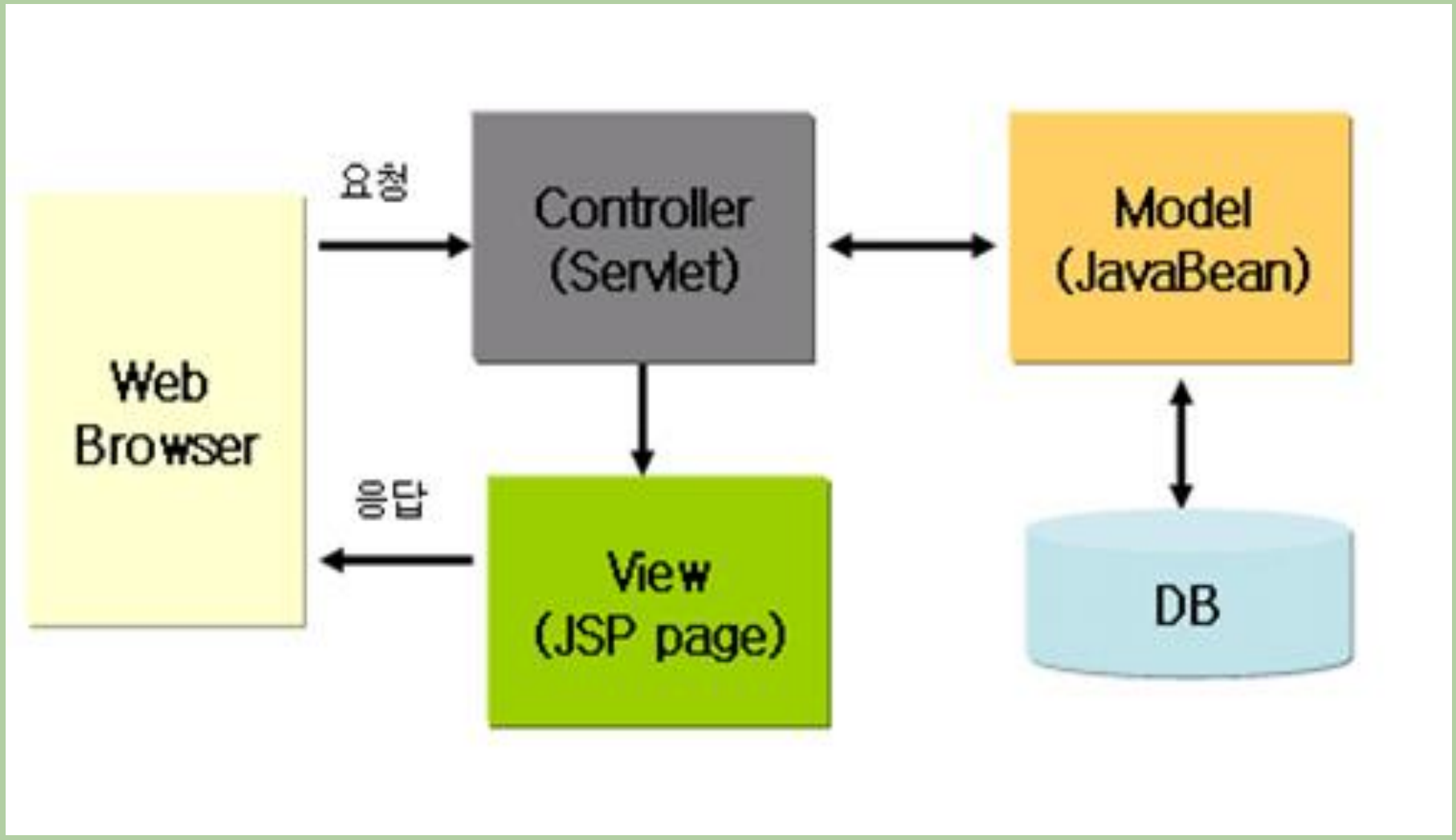
M(Model)V(View)C(Controller)



M(Model)V(View)C(Controller)



M(Model)V(View)C(Controller)



M(Model)V(View)C(Controller)