

JSP & Spring Framework

지도교수 : 박인상



JSP 개발 환경 설정

1. JDK 설치
2. 이클립스 개발 툴 설치
3. Tomcat 서버 설치 및 이클립스 연동
4. 이클립스에서 스프링 기반의 웹 프로젝트 생성

Tomcat의 버전별 개발 환경

Apache Tomcat 버전	동적 웹 모듈 버전
8.0, 8.5	3.1
7.0	3.0

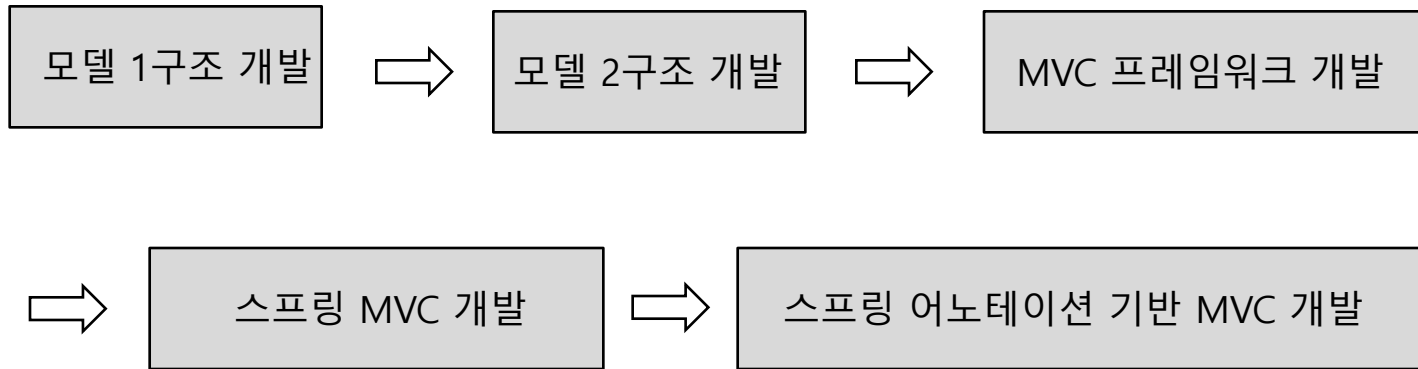
■ 즉 Apache Tomcat v8.0 의

➡ Dynamic web module version은 3.1로 설정한다

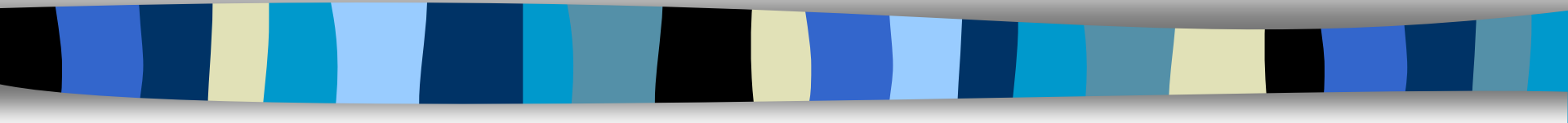
웹 프로그래밍 학습 로드맵(전문 웹 프로그래머)



게시판 프로그램 구현하기



JSP Basic





JSP(Java Server Pages)

- JSP는 웹 브라우저에 보여 줄 HTML 문서를 생성하는 것이다.
- HTML 문서를 생성하는 JSP 페이지는 크게 두 부분으로 구성된다.
 - **설정 부분**: JSP 페이지에 대한 설정 정보
 - **생성 부분**: HTML 코드 및 JSP 스크립트
- JSP는 실행시에는 자바 서블릿으로 변환된 후 실행되므로 서블릿과 거의 유사하다고 볼 수 있다.



JSP 페이지의 구성 요소

- 지시어(Directive): page, taglib, include
- 스크립트: 선언부, 스크립트릿, 표현식
- 기본 객체(Implicit Object)
- 표현언어(Expression Language)
- 표준 태그 라이브러리(JSTL)

지시어(Directive)

- 지시어는 해당하는 **JSP** 파일의 속성을 기술하는 곳으로 **JSP** 컨테이너에게 해당 페이지를 어떻게 처리해야 하는지 전달하기 위한 내용을 담고 있다.
- 즉 웹 브라우저가 요청한 **JSP** 페이지가 실행될 때 필요한 설정 정보를 지정하기 위해 사용

`<%@ 디렉티브이름 속성1="속성 값1" 속성2="속성 값2"... %>`

page 지시어 속성

page 지시어- 현재의 페이지를 컨테이너에서 처리하는 데 필요한 각종 속성 기술

(예) <%@ **page** contentType="text/html;charset=euc-kr" %>

속성	기본값	설 명
language	java	스크립트 언어를 지정
import		JSP 페이지에서 사용할 외부 자바 패키지나 클래스를 지정
session	true	JSP 페이지가 세션을 사용할 지의 여부를 지정
errorPage		오류가 발생할 때 호출할 페이지를 지정
isErrorPage	false	오류만 처리하는 페이지로 지정
contentType	text/html	JSP가 생성할 문서의 타입을 지정

taglib 지시어

- 표준 커스텀 태그인 JSTL이 제공하는 여러 가지 태그 라이브러리를 사용하려면 JSP 페이지에 taglib 지시어를 추가해 주어야 한다.

<사용 예>

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

include 지시어

- 현재 JSP 파일에 다른 HTML이나 JSP 문서를 포함하기 위한 기능을 제공한다
- include는 여러 페이지에 공통으로 들어가는 내용 즉 **공용변수**를 관리할 때 매우 유용하게 사용된다

<사용 예>

```
<%@ include file = "포함할 파일 이름" %>
```

JSP 페이지의 스크립트 요소

- 선언문(Declaration) <%! %> : 전역변수 선언 및 메소드 선언에 사용
- 스크립트릿(Scriptlet) <% %> : 프로그래밍 코드(자바코드) 기술에 사용
- 표현식(Expression) <%= %> : 화면에 출력할 내용 기술에 사용

표현식 (Expression)

- 표현식은 `<%= %>`를 사용해서 간단한 데이터 출력이나 메소드 호출 등에 이용한다.
- 표현식은 결국 `out.println()`으로 변환되는 것과 마찬가지로 사용할 수 있는 구문은 `out.println()` 인자로서 적합한 형태를 유지해야 한다.
- 따라서 산술식이 가능하며 + "문자열"을 이용한 형태도 가능하다.
- 사용 (예)
오늘의 날짜와 시간은 : `<%= new java.util.Date() %>`



JSP 기본 객체(내장 객체)

- **기본 객체란 JSP 내에서 선언하지 않고 사용하는 객체**라는 의미에서 붙여진 이름이다.
- 간단하게 JSP 컨테이너에 의해 미리 선언되어 있는 클래스 인스턴스 이름
- 기본 객체는 결국 클래스이므로 각각의 내장 객체가 어떤 클래스 인스턴스며 어떤 메시지를 가지고 있는지 알아두면 고급 JSP 프로그래밍을 하는데 도움이 된다.

JSP가 제공하는 내장 객체

내장 객체 (참조변수 이름)	자바 패키지의 인터페이스	주요 역할
request	javax.servlet.http.HttpServletRequest	HTML Form 요소 선택 값과 같은 사용자 입력 정보를 읽어올 때 사용
response	javax.servlet.http.HttpServletResponse	사용자 요청에 대한 응답을 처리할 때 사용
session	javax.servlet.http.HttpSession	클라이언트 세션 정보를 처리하기 위해 사용
application	javax.servlet.ServletContext	웹 어플리케이션에 대한 정보를 저장한다.
out	javax.servlet.jsp.JspWriter	사용자에게 전달하기 위한 output 스트림을 처리하기 위해 사용
exception	Java.lang.Throwable	에외처리를 위해 사용
config	javax.servlet.ServletConfig	현재 JSP에 대한 초기화 환경을 처리하기 위해 사용



request 내장객체의 주요 메소드

■ **getMethod()**

: 현재 요청이 GET, POST인지 확인하여 가지고 온다.

■ **getParameter(name)**

: 문자열 name과 같은 이름을 가진 인자 값을 가지고 온다.

request 내장객체의 주요 메소드

■ `getParameterValues(name)`

: 문자열 `name`과 같은 이름을 가진 인자 값을 배열 형태로 가지고 온다. **Checkbox**, multiple list 등에 주로 사용

[예]

```
String favorite[] = request.getParameterValues("favorite");
```

■ `getRemoteAddr()`

: 클라이언트 IP 주소를 알려주는 메소드

request 내장객체의 주요 메소드

- **getRequestURI()**
: 웹 브라우저가 요청한 URL에서 "**경로**"를 구한다. 리턴타입은 String
- **getSession()**
: 현재 요청과 관련된 session 객체를 리턴한다.
즉 session이 생성되어 있는 경우 생성된 session을 리턴하고
생성되어 있지 않은 경우 새롭게 session을 생성해서 리턴한다.
리턴타입은 HttpSession

request 내장객체의 주요 메소드

- **getCookies()**

: 모든 쿠키 값을 javax.servlet.http.Cookie의 '배열 형태'로 가지고 온다.

- **getProtocol()**

: 현재 서버의 프로토콜을 문자열 형태로 알려준다

- **setChracterEncoding()**

: 현재 JSP로 전달되는 내용을 지정한 캐릭터셋으로 변환한다. HTML form에서 한글을 입력할 때 정상적으로 처리하려면 반드시 필요하다



response 내장객체의 주요 메소드

- **setContentType(type)**

: 문자열 형태의 type에 지정된 MIME Type으로 ContentType을 설정

- **sendRedirect(url)**

: 클라이언트 요청을 다른 페이지로 보낸다.
즉, 현재 페이지를 다른 페이지로 전달한다.

폼 요소

- 웹에서 사용자로부터 입력을 받기 위해서는 HTML 폼이 필요하다
- 데이터 하나만 넘기는 경우
`<input type=text>`
- 여러 항목을 동시에 선택하는 경우
`<input type=checkbox>`

쿠키(cookie)

■ 쿠키는 일반적으로

1. 사용자가 방문한 시간
2. 사용자 ID
3. 패스워드
4. 쇼핑몰 등에서 사용자가 주문한 내역
5. 홈페이지에서 주로 어떤 정보를 이용하였는가 등의 정보를 저장하고 사용자가 다시 그 서버를 방문하면 그 정보를 읽어서 ‘클라이언트’를 식별하고 그에 따른 처리를 목적으로 사용된다.
6. HTTP 프로토콜에서 지원하는 것으로 서버가 브라우저에 저장하는 작은 정보조각을 의미한다.



Cookie와 Session의 차이점

■ Cookie

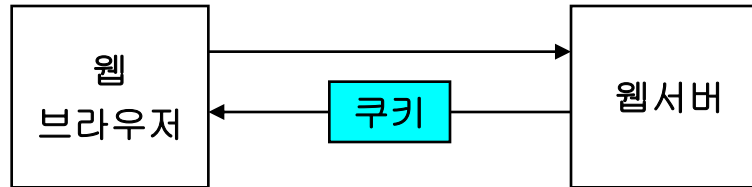
- Server와의 연결을 지속하거나 상태 정보를 유지하기 위해 **Client가 가지고 있는 자신의 정보**
즉 사용자와 관련된 정보를 PC에 보관하는 방식

■ Session

- Client와 연결을 지속하거나 상태 정보를 유지하기 위해 **Server단에서 가지고 있는 Client의 정보**
즉 사용자와 관련된 정보를 서버에 보관하는 방식

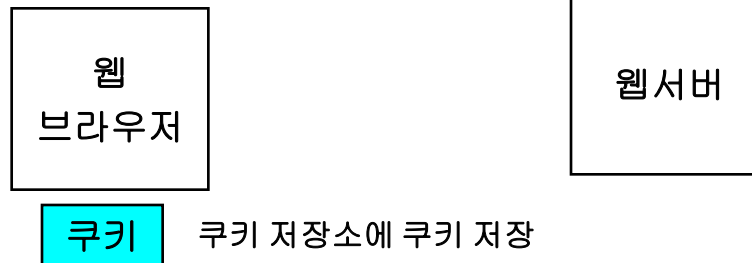
쿠키의 동작방식

(1)



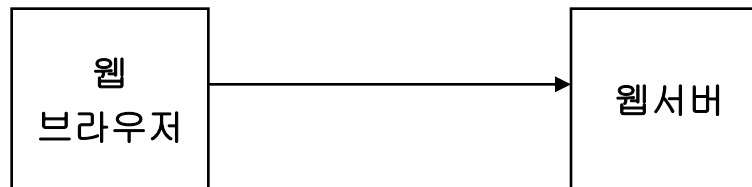
- Cookie 클래스 사용하여 쿠키 생성
- 쿠키 추가 : **response.addCookie(name)**

(2)



쿠키 저장소에 쿠키 저장

(3)



- 웹 브라우저의 요청과 함께 쿠키를 읽어 올 때는 **request.getCookies()** 메소드 사용한다

이후 같은 사이트에 접속시 저장된 쿠키가 요청 정보에 실려감

쿠키(cookie) 이용시 장점

- 쿠키를 이용한다는 것은
쿠키에 어떤 정보를 저장해 두고 사용자가 다시 사이트를 방문했을 때 ‘**그 정보를 이용**’한다는 것이다

[예]

```
document.cookie = “변수=값”
```

```
document.cookie = “test=OK”;
```

위 문장에서 test는 정보를 저장할 변수이다

이 이름을 이용해 서버는 사용자의 정보에 접근할 수 있다

쿠키(cookie)

```
document.cookie = “변수=값”
```

- 쿠키는 클라이언트와 서버간에 주고 받는 정보의 꾸러미이다
- 자바스크립트는 쿠키라고 불리는 ‘임시 텍스트’ 파일에 정보를 저장하거나 그 파일에 있는 정보를 읽어들이 수 있다
- 쿠키 파일에 들어 있거나 들어가는 정보의 구성은 기본적으로 “변수=값”의 형태이다

쿠키 클래스

- JSP에서 **javax.servlet.http.Cookie**라는 클래스를 이용하여 쿠키를 적용할 수 있다.
- 이 클래스는 쿠키를 생성하는데 이용되고, 작은 양의 정보를 서버에서 웹 브라우저로 전송을 담당하며, 브라우저는 이러한 정보를 저장한다. 그리고 이후에 서버로 다시 전송하게 된다.
- 쿠키의 값은 클라이언트마다 유일한 값을 가질 수 있으며, 쿠키는 일반적으로 세션을 관리한다. 쿠키는 이름과 단일 값을 가지고 있으며, 추가적으로 코멘트, 패스, 도메인 구분자, 최대 생명주기, 버전넘버와 같은 속성을 가질 수 있다.
- **getCookies()**메소드는 쿠키 객체의 '배열'을 반환하는 메소드이다

Cookie 클래스가 제공하는 주요 메소드

1. getName()

- 쿠키의 이름을 구한다.

2. getValue()

- 쿠키의 값을 구한다.

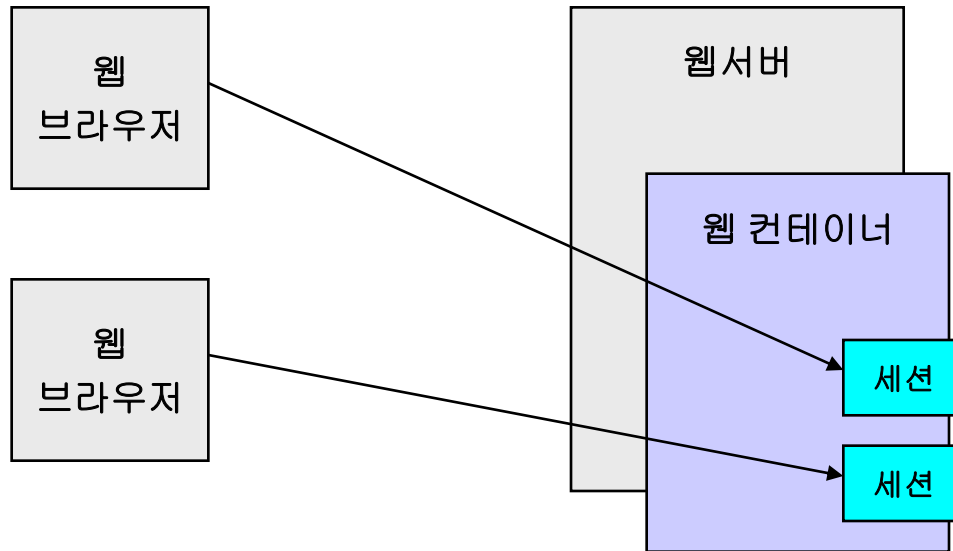
[예]

<%

```
Cookie cookie[] = request.getCookies();
```

%>

세션과 웹 브라우저의 관계



- 안정적, 보안상의 문제점 해결
- 세션은 웹 브라우저당 1개씩 생성되어 웹 컨테이너에 저장됨

session 내장객체의 주요 메소드

- `setAttribute(String name, Object value)`
 - name 값에 값을 저장한다
- `getAttribute(String name)`
 - name 값에 대한 세팅된 값을 반환한다
 - 이 메소드는 리턴타입이 Object 타입이므로
사용시 실제 할당된 객체 타입으로 형변환(casting)을 해야한다



session 내장객체의 주요 메소드

- `getId()`
 - 각 접속에 대한 세션 고유의 ID를 문자열 형태로 리턴한다.
(즉 16진수 32자리)
- `invalidate()`
 - 현재 세션을 종료한다. 세션과 관련된 값들은 모두 지워진다.

세션 생성 방법

(1) 방법

page 지시어의 session 속성을 "true"로 지정해준다.

```
<%@ page session = "true" %>
```

(2) 방법

request.getSession()을 이용한 세션 생성

<사용 예>

```
<%  
    HttpSession session = request.getSession();  
    session.setAttribute("boardList", boardList);  
    response.sendRedirect("getBoardList.jsp");  
%>
```

exception 내장 객체

- exception 내장 객체는 page 지시어에서 오류 페이지로 지정된 JSP 페이지에서 예외가 발생할 때 전달되는 `Java.lang.Throwable`의 인스턴스에 대한 참조 변수이다
- 예외 관련 메소드

메소드	설명
<code>toString()</code>	예외 클래스 이름과 함께 오류 메시지를 반환
<code>getMessage()</code>	문자열로 된 오류 메시지를 반환
<code>printStackTrace()</code>	표준 출력 스트림으로 스택 추적 정보를 출력



빈즈(Beans)

- 빈즈 – 자바의 '컴포넌트' 모델
- 컴포넌트 – 특정한 기능을 모듈화해서 '재사용'이 가능하도록 만든 소프트웨어
- 자바는 자바 빈즈, JSP 빈즈, EJB(Enterprise Java Beans) 세 분야
- 이들은 모두 컴포넌트를 기본으로 하고 있으며 소프트웨어 '재사용'이라는 목적은 동일하다.
- 빈즈는 순수 자바 소스기 때문에 JSP와는 달리 컴파일 과정이 필요

빈즈를 이용하지 않은 구현 - 계산기

calc1.jsp 파일



빈즈를 이용한 구현 - 계산기

JSP
(calc2.jsp 파일)



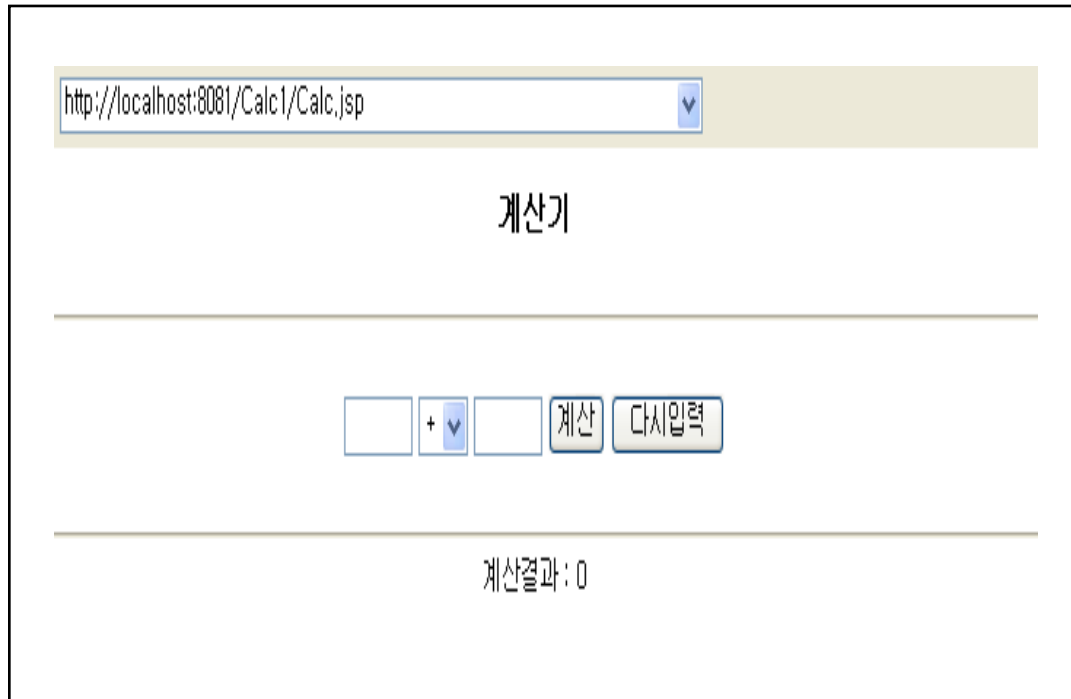
분리

빈즈
(CalcBean.java 파일)



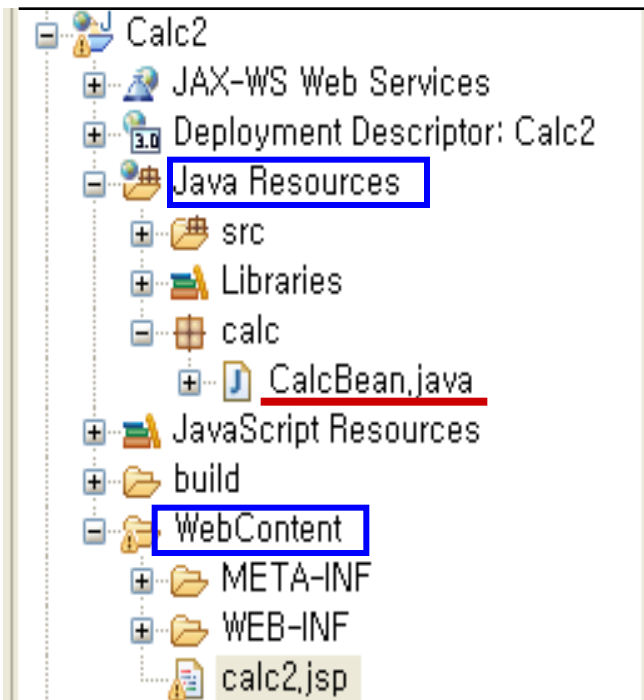
- 참고 : 빈즈를 다른 JSP에서 계산기 기능이 필요할 때 언제든지 '재활용'이 가능하게 된다.

계산기 프로젝트 실행화면



The screenshot shows a web browser window with the address bar displaying `http://localhost:8081/Calc1/Calc.jsp`. The page title is "계산기" (Calculator). Below the title is a horizontal line. Underneath the line is a calculator interface consisting of two input fields, a "+" button, a dropdown menu with a downward arrow, and two buttons labeled "계산" (Calculate) and "다시입력" (Re-input). Below the calculator interface is another horizontal line, and at the bottom, the text "계산결과: 0" (Calculation result: 0) is displayed.

Calc2 프로젝트 이클립스에서 구현시 디렉토리 구조



빈즈 클래스 구성

```
class XxxBean {  
    // 멤버 변수(프로퍼티), DB 테이블의 컬럼 이름과 매칭된다.  
    private String xxx;  
    private int xxx;  
    // 생성자, 보통 멤버 변수들을 모두 설정하는 생성자를 이용한다.  
    public XxxBean(String xxx, int xxx) {  
    }  
    // set, get 메소드, 멤버 변수와 매칭된다.  
    public void setXxx(String xxx) {  
        this.xxx = xxx;  
    }  
    public <data_type> getXxx() {  
        return xxx;  
    }  
}
```


액션(Action)

- 액션은 JSP 주요 구성 요소 중 하나로 다음과 같은 기능을 지원한다
 - JSP 페이지간 흐름 제어
 - 자바 빈즈 컴포넌트와 JSP 상호작용 지원

<액션의 종류>

include → 다른 페이지를 현재 페이지에 포함시킨다

forward → 현재 페이지의 제어를 다른 페이지로 전달한다

useBean

include 지시어와 include 액션의 차이점

◆ include 지시어

- 해당 파일을 포함시킨 다음 컴파일한다
- 파일 내용 두 개를 하나로 컴파일 하기 때문에 잘 바뀌지 않는 **'정적인' 페이지를 포함시킬 때** 사용하는 것이 좋다

◆ include 액션

- 실행 시점에서 해당 파일을 호출하여 그 결과를 포함한다
- 파일 두 개를 각각 컴파일해서 관리하기 때문에 **'동적인' 페이지를 포함시킬 경우에** 사용하는 것이 좋다
- 이 액션은 제어권을 다른 JSP로 보냈다가 다시 가져온다
- include 액션의 특징 중 하나가 변수를 전달할 수 있다는 점
- include 액션에서 JSP 파일을 포함할 때 param 태그를 이용해 속성 값을 전달할 수 있다

[형식]

```
<jsp:include page="포함할 파일이름"/>
```



forward 액션

- include 액션은 제어권을 다른 JSP로 보냈다가 다시 가져오지만 forward는 제어권을 완전히 넘겨버린다.
- forward도 include 액션처럼 param을 이용해서 인자를 전달 할 수 있다.

[형식]

```
<jsp:forward page="포워딩할 파일이름"/>
```

useBean 액션

- useBean 태그는 빈즈를 사용하겠다는 것을 의미
- `<jsp:useBean id="calc" scope="page" class="calc.CalcBean" />`

위 문장의 의미는

calc 패키지의 calcBean 클래스를 calc 라는 이름으로
page 범위에서 사용할 것을 선언

setProperty

- setProperty는 JSP에서 빈즈 값을 설정할 때 사용
- `<jsp:setProperty name="calc" property="*" />`

위 문장의 의미는
useBean 으로 선언된 빈즈 클래스의 모든 setXxx() 메소드를 호출
한다

getProperty

- getProperty는 빈즈에서 값을 가져올 때 사용
- `<jsp:getProperty name="calc" property="result" />`

위 문장의 의미는

useBean 으로 선언된 빈즈 클래스의 **getResult()** 메소드를 호출한다

JSP에서 빈즈 선언

- `<jsp:useBean id="calc" scope="page" class="calc.CalcBean"/>`
- **useBean** 액션 속성 목록

액션	속성	설명
useBean	id	빈즈 클래스 인스턴스 이름으로 사용할 변수
	class	패키지 이름까지 모두 기술한 클래스 이름
	scope	Page, request, session, application이 올 수 있다.

JSP에서 빈즈값 설정

- `<jsp:setProperty name="calc" property="*" />`
- `setProperty` 액션 속성 목록

액션	속성	설명
setProperty	name	<jsp:useBean>에서 정의된 빈의 속성 이름
	property	속성 값으로 빈즈 클래스의 setXxx(Xxx는 속성 이름) 메소드와 대응할 속성 값이다. “*”를 지정하면 모든 setXxx에 자동으로 대응된다.

빈즈에서 JSP로 값 가져오기

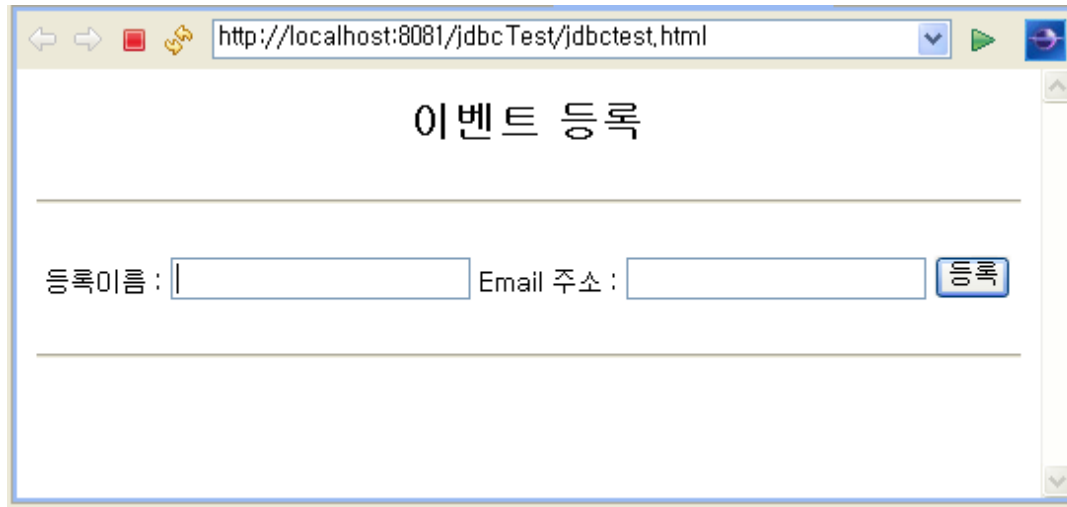
- `<jsp:getProperty name="calc" property="result" />`
- getProperty 액션 속성 목록

액션	속성	설명
getProperty	name	<jsp:useBean>의 id를 기술한다.
	property	값을 얻어올 property를 기술한다.

HTTP 에러 코드(주로 많이 발생하는 에러)

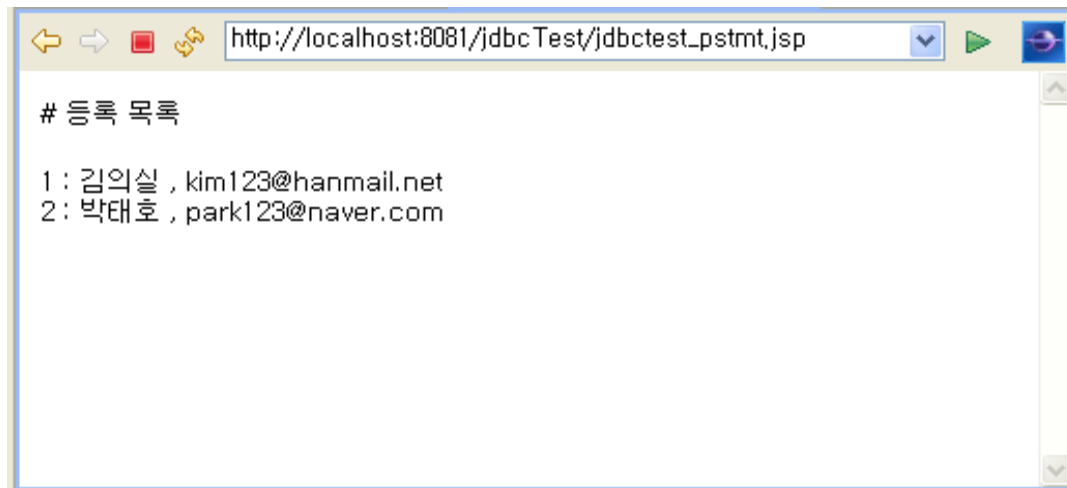
HTTP 에러 코드	에러 메시지
400	Bad Request, 요청 실패 - 문법상 오류가 있어서 서버가 요청 사항을 이해하지 못함
404	Not Found, 문서를 찾을 수 없음 - 이 에러는 클라이언트가 요청한 문서를 찾지 못한 경우에 발생함. URL을 다시 잘 보고 주소가 올바르게 입력되었는지를 확인함
500	Internal Server Error, 서버 내부 오류 - 이 에러는 웹 서버가 요청사항을 수행할 수 없을 경우에 발생함

등록이름과 이메일주소 테이블에 insert 하기 프로젝트



이벤트 등록

등록이름 : Email 주소 :



등록 목록

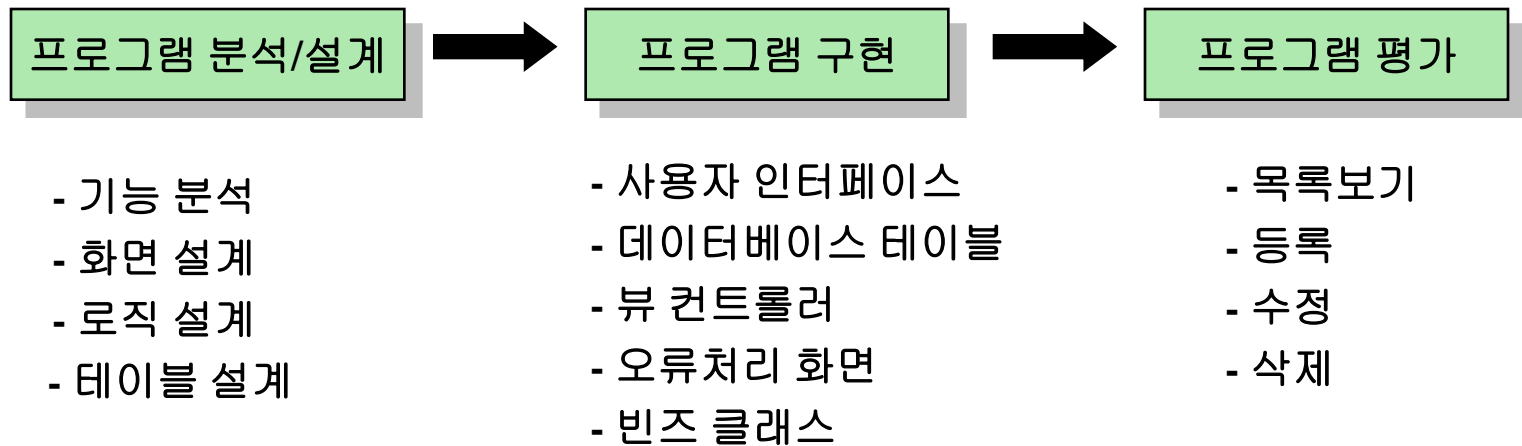
1 : 김의실 , kim123@hanmail.net
2 : 박태호 , park123@naver.com



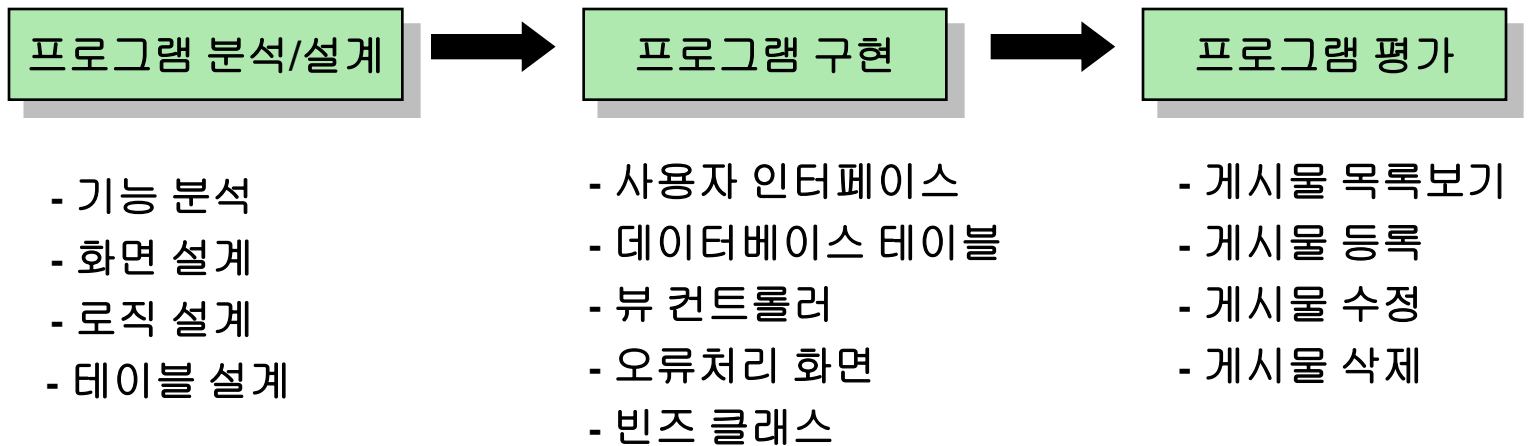
테이블 생성

```
SQL> create table event_test  
      ( username varchar2(12), -- 등록이름  
        email  varchar2(30)    -- email 주소  
      );
```

웹 애플리케이션 프로그램 제작 단계



방명록 프로그램 제작



방명록 프로그램 기능 목록

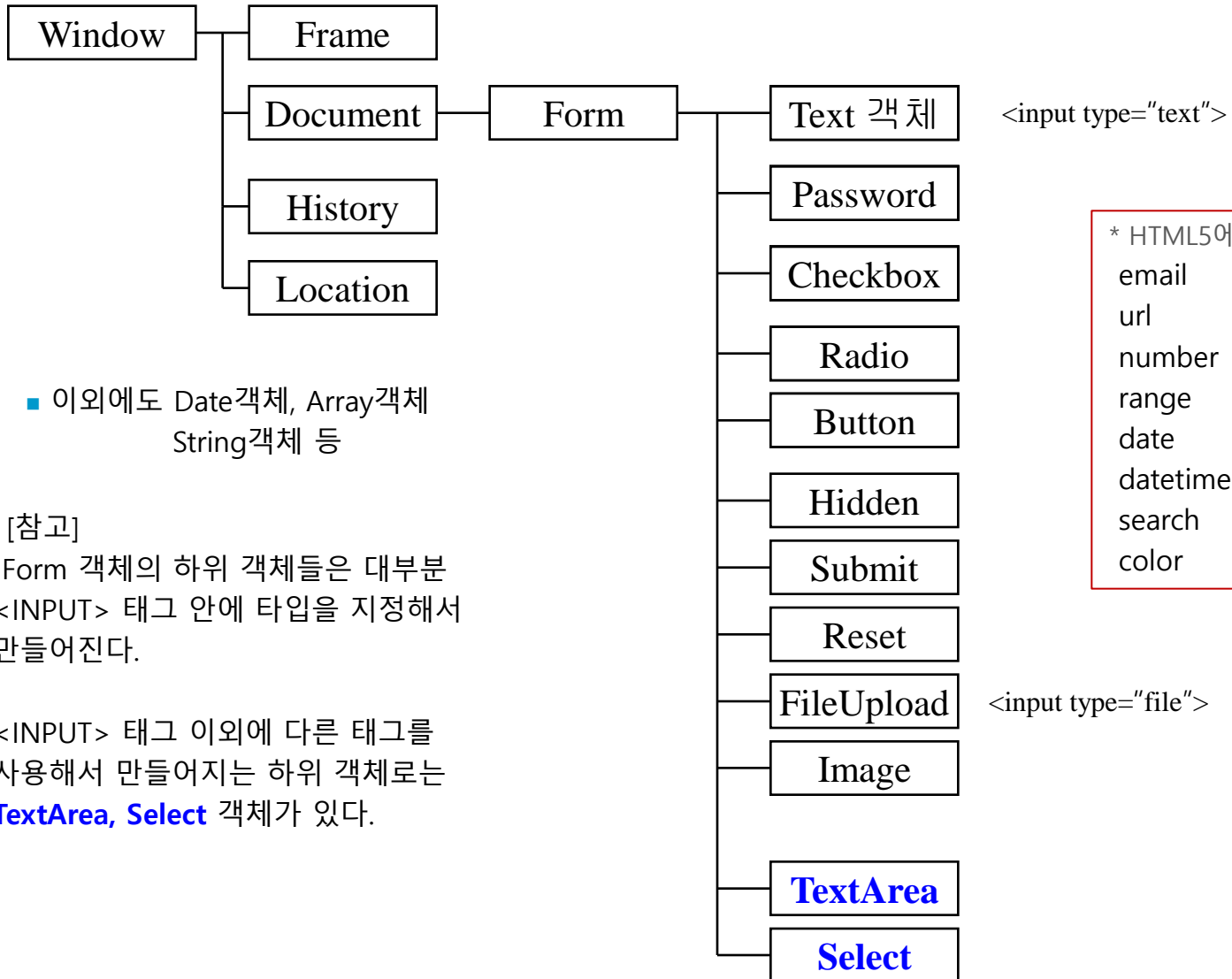
번호	기능	주요 내용
1	방명록 목록보기	페이지 구분없이 최신 게시물을 맨 위에 보여준다
2	방명록 입력	작성자, 이메일, 비밀번호, 내용을 입력한다
3	방명록 수정	방명록을 작성할 때 입력한 비밀번호로 수정한다
4	방명록 삭제	수정 화면으로 삭제하도록 한다



자바스크립트

- 브라우저에서 서버 연결 없이 사용자와 동적인 상호작용을 가능하게 하는 스크립트 기술
- Ajax 등장과 함께 새로운 생명을 얻었다.

자바스크립트 객체들





history 객체

- 브라우저가 로드한 페이지들에 대한 정보를 갖고 있다
- **go(n) 메소드**
: n 단계 만큼 페이지 이동

n : 0 이면 현재 페이지

n : 양수이면 다음 페이지

n : 음수이면 이전단계 페이지로 이동

<script> 태그

- src 속성 – 자바스크립트가 따로 파일로존재하는 경우 해당 URL을 명시하는 속성

[사용 예]

```
<script language="JavaScript" src="script.js"></script>
```

<link> 태그

- 현재문서와 다른문서와의 관계를 나타내는 element로 <head>태그 안에 들어가며 시작태그로만 이루어진다

```
<link href=" URL" rel="문서의관계" type="MIME타입">
```

[참고]

URL은 stylesheet의 위치

[사용 예]

```
<link href="style.css" rel="stylesheet" type="text/css">
```

이벤트 핸들러(Event Handler)

- 이벤트 핸들러는 'on이벤트타입'과 같이 사용하여 이벤트를 다루는 것을 의미한다
- '어떠 어떠한 이벤트가 발생하면...' 이라는 뜻

이벤트핸들러 = "함수" or "명령줄"

이벤트 타입	이벤트 핸들러	설 명
Load	onLoad	문서가 브라우저에 의해 읽혀지는 순간 함수를 실행하고자 할 때 사용
Click	onClick	버튼이나 링크 등을 마우스로 클릭할 때
KeyUp	onKeyUp	키보드의 키를 눌렀다 떼는 순간
Change	onChange	텍스트, 텍스트 필드 등에서 포커스를 잃거나 내용이 수정될 때
Submit	onSubmit	Submit 버튼을 눌러 Form의 내용을 제출할 때
Focus	onFocus	창이나 텍스트 박스 등을 선택하여 포커스가 주어질 때
Blur	onBlur	요소에 포커스가 벗어났을 때 이벤트 발생
MouseOver	onMouseOver	마우스 포인터를 특정 객체에 올렸을 때 발생하는 이벤트
MouseOut	onMouseOut	마우스 포인터를 특정 객체에서 다른 곳으로 옮겼을 때 발생하는 이벤트

Form 객체의 submit() 메소드

- 폼을 전송하는 메소드
- 서버로 데이터 전송(submit 버튼을 누른 효과)

[사용 예]

```
document.regForm.submit();
```

window 객체의 open() 메소드

- 새로운 창을 오픈하는 함수

```
open("새창파일명", "윈도우이름", "옵션");
```

[사용 예]

//ID중복체크시 검사

```
function idCheck(id)
```

```
{
```

```
    if(id == ""){
```

```
        alert("아이디를 입력해 주세요.");
```

```
    }else{
```

```
        url="IdCheck.jsp?mem_id=" + id;
```

```
        window.open(url, "post", "width=300, height=150");
```

```
    }
```

```
}
```

open() 메소드의 옵션

옵션	값	설 명
menubar	yes/no	새로운 창에 메뉴바를 보일지를 결정
toolbar	yes/no	툴바를 보일지를 결정
directories	yes/no	디렉토리 버튼을 보일지를 결정
status	yes/no	상태표시줄을 보일지를 결정
scrollbars	yes/no	스크롤바를 보일지를 결정
width	pixel	창의 너비값을 지정
height	pixel	창의 높이값을 지정



window 객체 메서드

■ alert()

- 경고용 대화상자를 보여줌

■ confirm()

- 확인, 취소를 선택할 수 있는 대화상자를 보여줌



document 객체의 메서드

- `document.getElementById(id);`
 - id명으로 엘리먼트 찾기
- `document.getElementsByName(name);`
 - name 속성으로 엘리먼트 찾기

window 객체의 속성

- opener 속성
 - 열린 창에서 상위 창을 가리킬 때 사용

[사용 형식]

opener.객체이름.속성 혹은 메소드

opener는 Window 객체를 지칭하는 대명사로 생각할 수 있다
따라서 Window 객체가 사용할 수 있는 모든 속성이나 메소드를 사용할 수 있다
따라서 Document나 하위 객체들을 모두에 접근하여 통제할 수 있다
즉 상위창의 모든 내용을 통제할 수 있다

방명록 화면 목록

번호	기능	파일 이름
1	방명록 리스트 화면	guestbook_list.jsp
2	방명록 입력 양식	guestbook_form.jsp
3	방명록 수정/취소 화면	guestbook_edit_form.jsp

방명록 프로그램 구성

번호	기능	파일 이름
1	데이터베이스 insert, update, delete, select 등을 수행하는 클래스	GuestBean.java
2	뷰 컨트롤러	guestbook_control.jsp actioncode = insert, update, delete
3	오류 처리를 위한 페이지	guestbook_error.jsp

방명록 테이블 구조 - guestbook

번호	컬럼 이름	자료형(크기)	기능	비고
1	gb_id	number	각각의 게시물을 구분하는 고유 번호	- Primary Key - Sequence 생성
2	gb_name	varchar2(15)	작성자 이름	
3	gb_email	varchar2(20)	작성자 이메일	
4	gb_date	date	작성일자	
5	gb_passwd	varchar2(6)	수정과 삭제를 위한 비밀번호	
6	gb_contents	varchar2(500)	게시물 내용	



gb_id를 관리하기 위한 guestbook_seq 시퀀스 생성

```
SQL> CREATE SEQUENCE guestbook_SEQ  
        INCREMENT BY 1 START WITH 1;
```

guestbook 테이블 생성

```
SQL> create table guestbook
      ( gb_id    number PRIMARY KEY,
        gb_name  varchar2(15),
        gb_email varchar2(30),
        gb_date  date,
        gb_passwd varchar2(10),
        gb_contents varchar2(500)
      );
```

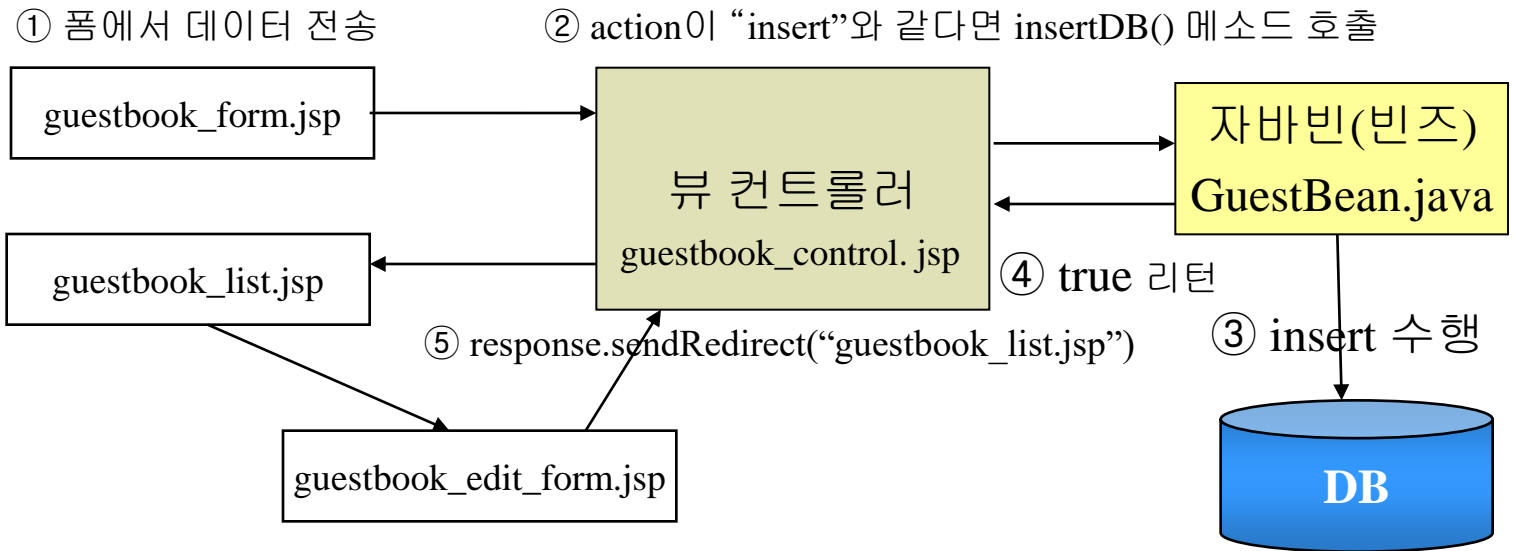

MVC 패턴(디자인 패턴)

- JSP 모델-2 개발 방법
- MVC(Model-View-Controller) 패턴
- 현재 대표적인 웹 애플리케이션 개발 모델
- MVC 패턴의 기본 개념
 - View – 사용자에게 보여주는 화면(html 파일)
 - Model – 데이터 처리(Beans 즉 java 파일)
 - Controller – 프로그램 로직(JSP, Servlet 파일)
- 웹 애플리케이션의 확장성과 유지보수 용이

MVC 패턴 구성 요소

영역	구현방식
View	JSP를 기본으로 표현 언어
Controller	MVC 패턴의 중심이 되는 부분으로 직접 구현하거나 구현된 솔루션을 이용할 수 있다. 대표적으로 스트러츠 프레임워크가 있다.
Model	데이터 영역으로 DO, DAO 등으로 구분해 구현하기도 한다.

뷰 컨트롤러 동작 과정



■ 참고

```
<form name=form1 method=post action=guestbook_control.jsp>
```

```
<input type=hidden name="action" value="insert">
```

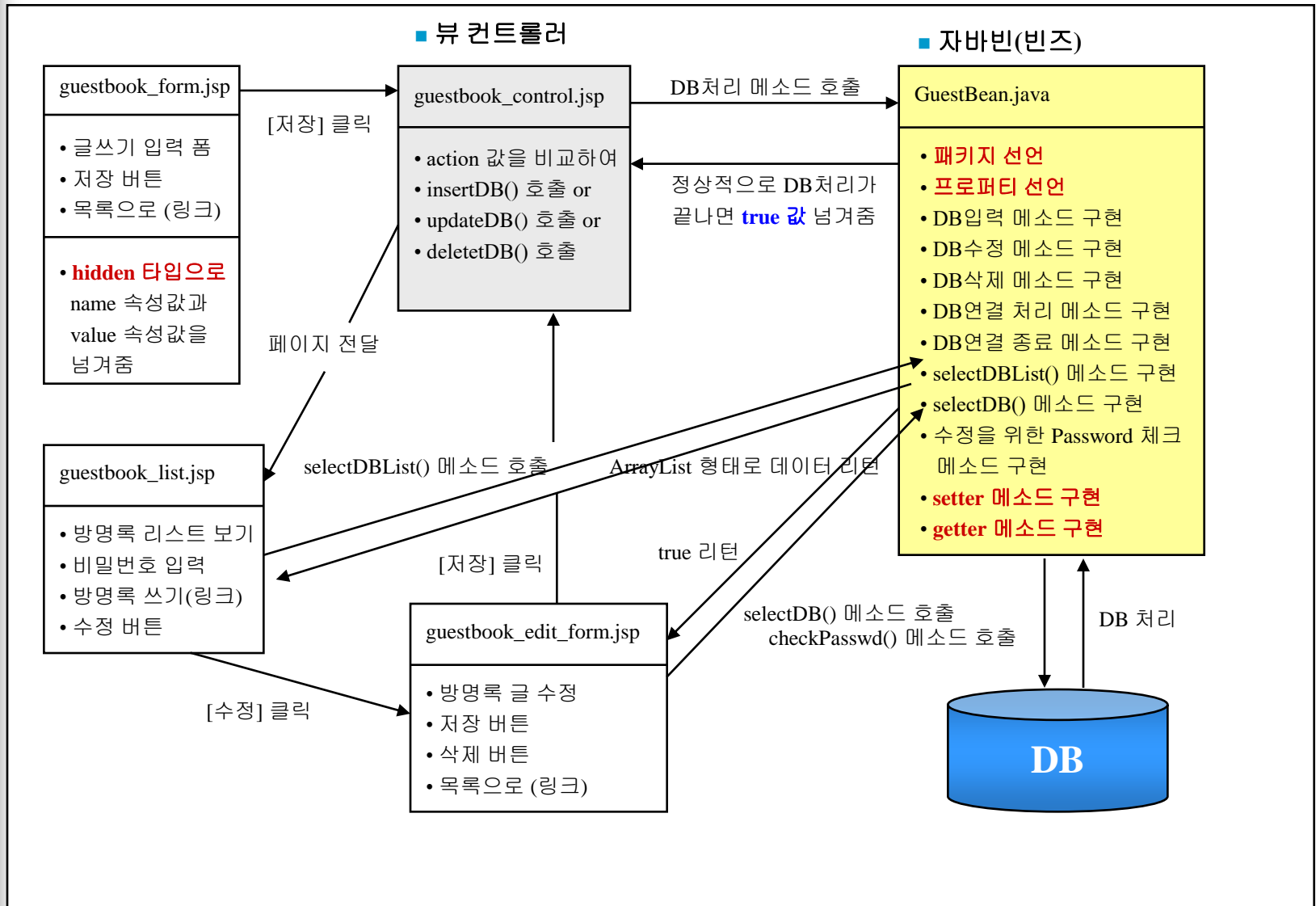
.....

```
<input type=submit value="저장">
```

: 뷰 컨트롤러로 사용할 guestbook_control.jsp는 실제로 화면에 보여주는 내용은 없다.

다만 사용자 인터페이스를 통해 들어오는 요청을 판단하고 빈즈 클래스 처리를 거쳐 적절한 JSP로 연결하는 역할을 한다.

방명록 흐름도



guestbook_form.jsp 파일에서

- submit 버튼을 클릭하면 hidden 타입으로 설정된 name 속성값인 “action”과 value 속성값인 “insert”가

쌍으로 구성되어 <form> 태그의 action 속성에서 지정한 URL인 guestbook_control.jsp 프로그램으로 전송된다

■ guestbook_control.jsp 파일 소스

```
<jsp:useBean id="gb" scope="page" class="gb1.GuestBean" />
<jsp:setProperty name="gb" property="*" />
<%
    String action = request.getParameter("action");
    if(action.equals("insert")) {
        if(gb.insertDB()) {
            response.sendRedirect("guestbook_list.jsp");
        }
        else
            throw new Exception("DB 입력오류");
    }
    else if(action.equals("update")) {
        if(gb.updateDB()) {
            response.sendRedirect("guestbook_list.jsp");
        }
        else
            throw new Exception("DB 갱신오류");
    }
}
```



숨겨진 입력 양식 다루기 – Hidden 객체

- 웹 프로그래밍을 작성하다 보면 사용자의 브라우저에는 나타낼 필요없이 **내부적으로만 처리해서** 웹 서버의 JSP, ASP, PHP 등에 전송해야 할 자료를 다루어야 할 경우가 발생한다
- 이 때 브라우저에 나타나지 않는 텍스트 박스 형태의 **숨겨진 입력 양식**을 사용한다

Hidden 객체

- 숨김 글상자의 정보를 제공
- 목록을 선택하거나 글을 입력하는 등의 작업을 할 수 없는 숨김 글상자 입력 양식은 입력 사항에 무관하게 특별한 내용을 전달할 경우에 사용된다.
- < hidden 객체의 속성 >

name	숨김 글상자의 name 속성 값을 가져온다.
value	숨김 글상자의 value 속성 값을 가져온다.

스크립트 요소

- jsp에서 실시간으로 문서의 내용을 생성하기 위해 사용되는 것이 스크립트 요소.
 - jsp를 스크립트 언어라고 부르는 이유가 바로 막강한 스크립트 코드를 제공해 주기 때문이다.
 - 표현식(Expression!) : 값을 출력한다.
 - 스크립트릿(Scriptlet) : 자바 코드를 실행한다.
 - 선언부 (Declaration) : 자바 메소드(함수)를 만든다.
-

- 스크립트릿

<%

자바코드 1;

자바코드 2;

자바코드 3;

%>

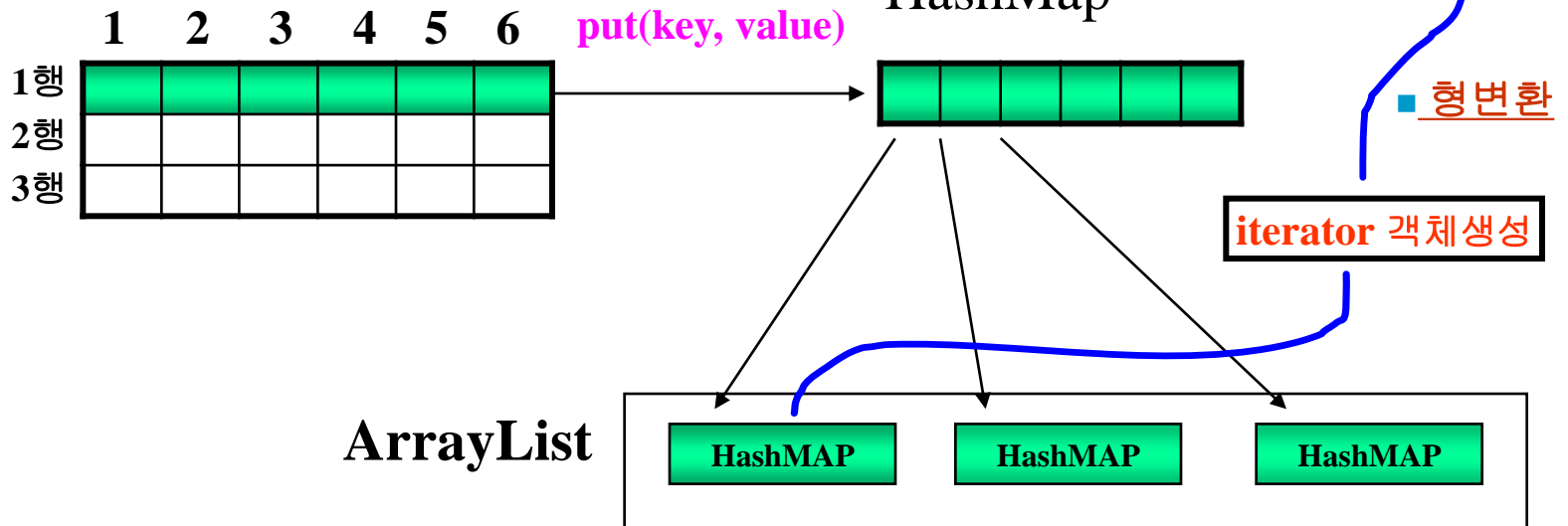
- 스크립트의 코드블록은 <%로 시작해서 %>로 끝나며 <%와 %>사이에는 실행할 자바코드가 위치한다.

- 각각의 데이터를 **HashMap**에 넣고 이를 다시 **ArrayList**에 추가하는 구조
- 방명록 리스트를 보여주기 위해 데이터 가져오기
- **ArrayList**를 좀더 쉽게 처리하기 위해 **iterator**(반복자) 생성

- 빈즈에서 가져온 **ArrayList**에서 **HashMap**을 가져와 반복해서 출력한다.

방명록 Table에서

`select * from guestbook order by gb_date desc;` 의 결과





ArrayList 객체에 iterator() 메소드

- 반복자에 의해서 컬렉션에 접근하기 전에 먼저 **반복자 객체 참조를 생성해야** 하며 각각의 컬렉션들은 컬렉션의 처음에서 시작하는 반복자를 반환하는 iterator() 메소드를 제공한다
- 이 반복자 객체를 사용하면 컬렉션에 있는 각 요소에 한번에 하나씩 접근할 수 있다

Iterator를 통한 컬렉션 접근

- 컬렉션의 요소들을 얻거나 삭제, 출력하면서 반복하는 것이 필요한데 iterator는 이것을 쉽게 구현할 수 있게 해준다

< **Iterator** 인터페이스의 주요 메소드 >

boolean **hasNext()**

: 다음 요소가 있으면 true를 리턴

Returns true if the iteration has more elements.

Object **next()**

- ArrayList의 다음 값을 Object형으로 반환하는 메소드로 적절한 형태로 형을 변환해서 받으면 된다.

- 다음 요소를 리턴, 없다면 NoSuchElementException 예외 발생

Returns the next element in the iteration.

HashTable 클래스

- HashTable 클래스는 java.util의 일부분으로 Dictionary의 실제 구현이다.
- **HashMap과 같이 '키'와 '값'의 쌍을 저장하기 위한** HashTable 저장 메커니즘을 지원한다.
- 또한 해시테이블은 저장된 정보를 키를 이용해서 빠른 속도로 저장하고 읽어오기 위해서 디자인되었다.

▶ 주요 생성자 및 메소드

HashTable(int) : 지정된 능력을 가진 비어있는 HashTable을 생성한다.

put(key, value) : HashTable에 지정된 '키'와 '값'의 쌍을 저장

get(key) : 지정된 키에 연관된 값(value)을 리턴



자바 컬렉션 프레임워크와 컬렉션 뷰

- 자바 컬렉션 프레임워크에서 컬렉션 뷰 메소드 `iterator()`

컬렉션 프레임워크

- 자바 2 플랫폼에서는 객체를 저장하고 정리하기 위한 방법으로 자바 컬렉션 프레임워크를 사용한다.
- 컬렉션 프레임워크는 `java.util` 패키지에 포함되어 있으며, 컬렉션 계열, 맵 계열, 컬렉션 뷰, 컬렉션 유틸리티 등을 포함한다.
- 컬렉션 계열에는 `Collection`, `Set`, `SortedSet`, `List` 등의 인터페이스와 `ArrayList`, `LinkedList`, `HashSet`, `TreeSet` 등의 클래스 등이 포함되어 있다. 맵 계열에는 `Map`, `SortedMap` 등의 인터페이스와 `HashMap`, `TreeMap` 등의 클래스가 포함되어 있다. 컬렉션 뷰에는 `Iterator`, `ListIterator` 인터페이스 등이 포함되어 있으며, 컬렉션 유틸리티에는 `Collections` 클래스 등이 포함되어 있다.
- 자바 2 이전에 같은 목적으로 사용되어 오던 `Vector`, `Hashtable` 등의 클래스도 `Collection`, `Map` 인터페이스를 상속함으로써 자바 컬렉션 프레임워크에 통합되었고, `Enumeration` 인터페이스 대신 향상된 `Iterator` 인터페이스를 사용할 것을 권장하고 있다.



HashMap과 ArrayList

■ HashMap

- Collections Framework의 Map 계열에 속한다
- Map 계열은 Object 형태의 데이터를 Key값으로 관리하는 메모리 구조

■ ArrayList

- Collections Framework의 List 계열에 속한다
- List 계열은 Object 형태의 데이터를 Index 값으로 관리하는 메모리 구조

컬렉션 뷰

- 컬렉션 프레임워크의 클래스들은 객체를 저장하고 정리하기 위해 사용된다.
- 컬렉션 클래스들 내에 **저장된 객체들을 차례로 접근하기** 위한 방법을 **컬렉션 뷰** 라고 한다. 자바 2 이전 버전에서 사용되던 Vector, Hashtable의 뷰 객체는 Enumeration 객체이며, 자바 2의 컬렉션 프레임워크에서 **컬렉션 뷰는 Iterator**와 ListIterator 객체이다.
- 컬렉션 뷰는 컬렉션 객체에 저장된 객체들에 대한 순차적 접근을 제공한다.



효율적인 컬렉션 객체들의 사용

- **Vector – Hashtable** : 동기화가 되었다.
동시에 여러 스레드가 접근할 수 없다.(은행의 현금 입출금의 경우)
- **ArrayList – HashMap** : 동기화가 되어 있지 않다.
동시에 여러 스레드가 접근할 수 있다.



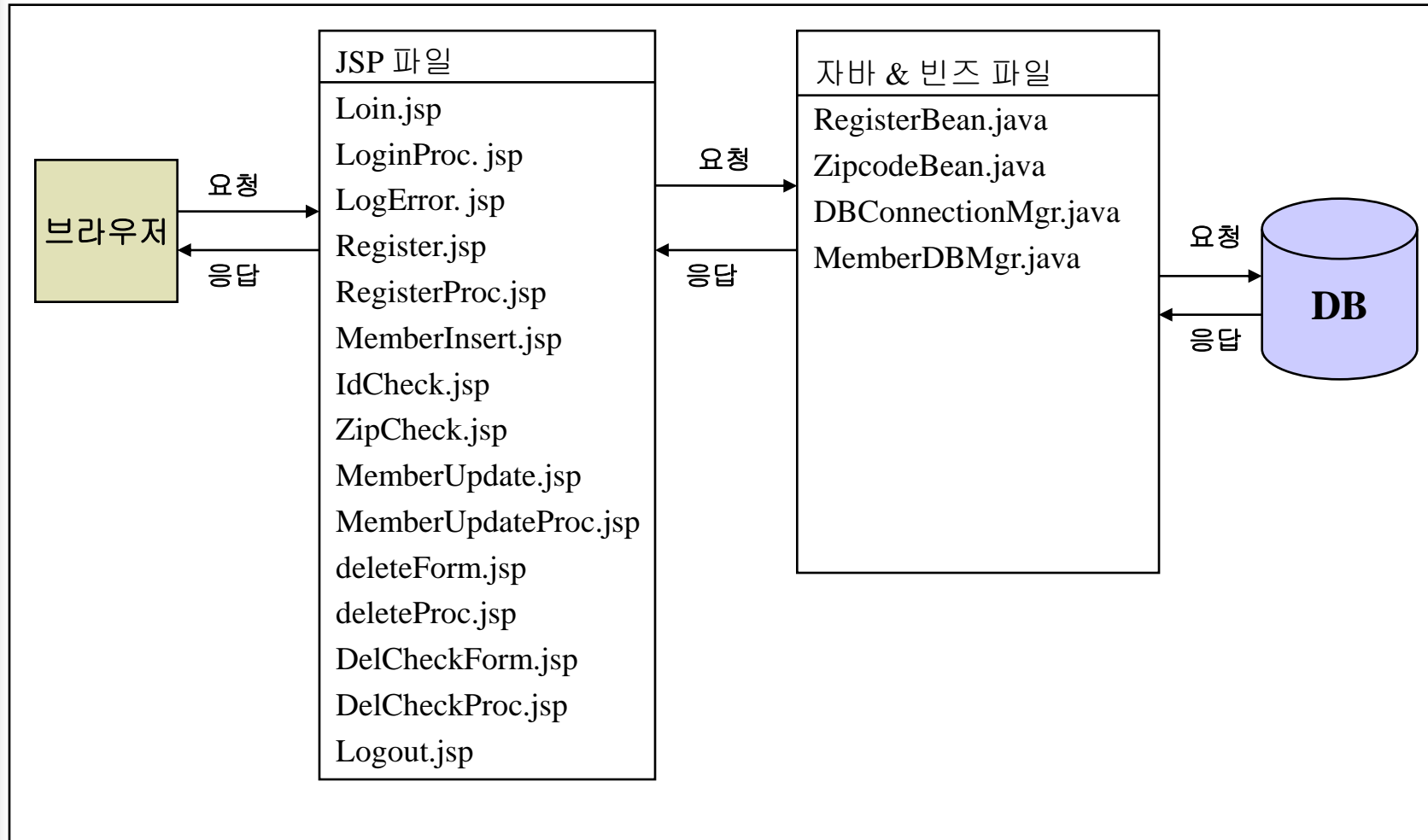
회원가입 및 인증 프로그램 제작

- 기능 -

- 회원가입
- ID 중복 체크
- 우편번호 검색
- 회원인증

회원가입 및 인증 프로그램 제작

- 전체 구조도 -



소스 목록

JSP

- Loin.jsp(로그인 페이지)
- LoginProc.jsp(로그인 처리 페이지)
- LogError.jsp(로그인 에러 페이지)
- Register.jsp(회원가입 페이지)
- RegisterProc.jsp(회원가입 처리 페이지)
- MemberInsert.jsp(회원가입 입력 페이지)
- IdCheck.jsp(중복 아이디 체크 페이지)
- ZipCheck.jsp(우편번호 검색 페이지)
- 등 등

기타

- script.css(스타일 시트 파일)
- script.js(자바스크립트 처리 파일)

자바 & 빈즈

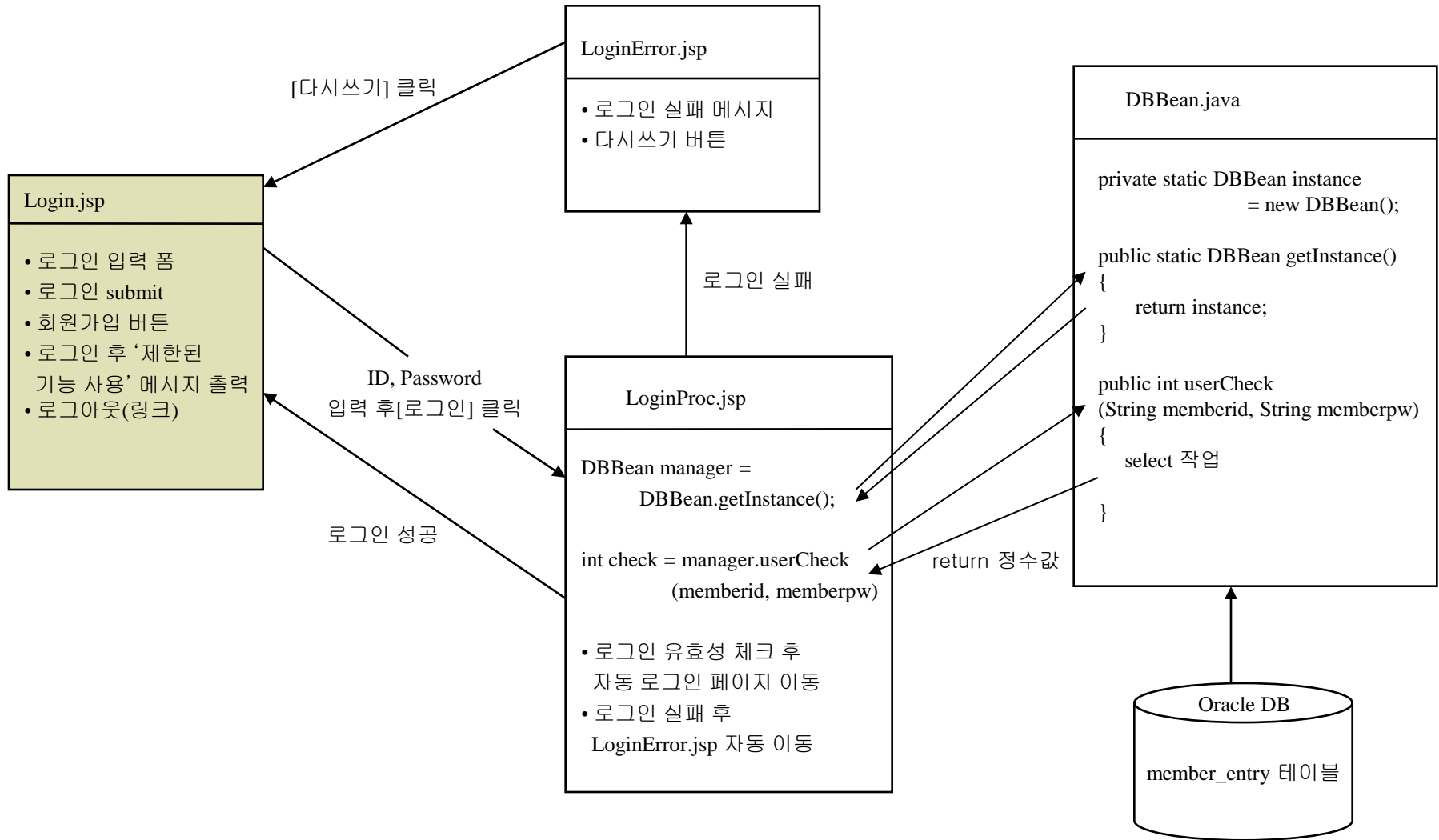
- RegisterBean.java(회원가입 자바빈즈)
- ZipcodeBean.java(우편번호 및 주소 자바빈즈)
- DBConnectionMgr.java(데이터베이스 연결 connectionpool 자바 파일)
- MemberDBMgr.java(회원인증 및 회원가입과 우편번호 처리 자바 파일)

데이터베이스 설계

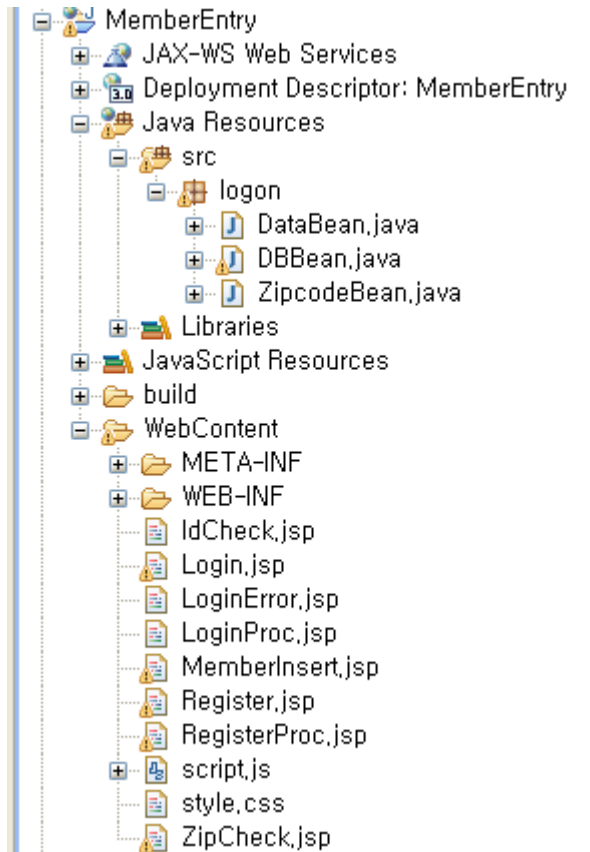
- 회원 테이블 -

컬럼명	데이터 타입	설명
id	varchar2(20)	회원 ID - primary key로 지정
passwd	varchar2(20)	회원 비밀번호
name	varchar2(20)	회원 이름
mem_num1	char(6)	주민등록번호 앞자리 6자리
mem_num2	char(7)	주민등록번호 뒤자리 7자리
e_mail	varchar2(30)	회원 이메일
phone	varchar2(30)	회원 전화번호
zipcode	char(7)	회원 우편번호
address	varchar2(60)	회원 주소
job	varchar2(30)	회원 직업

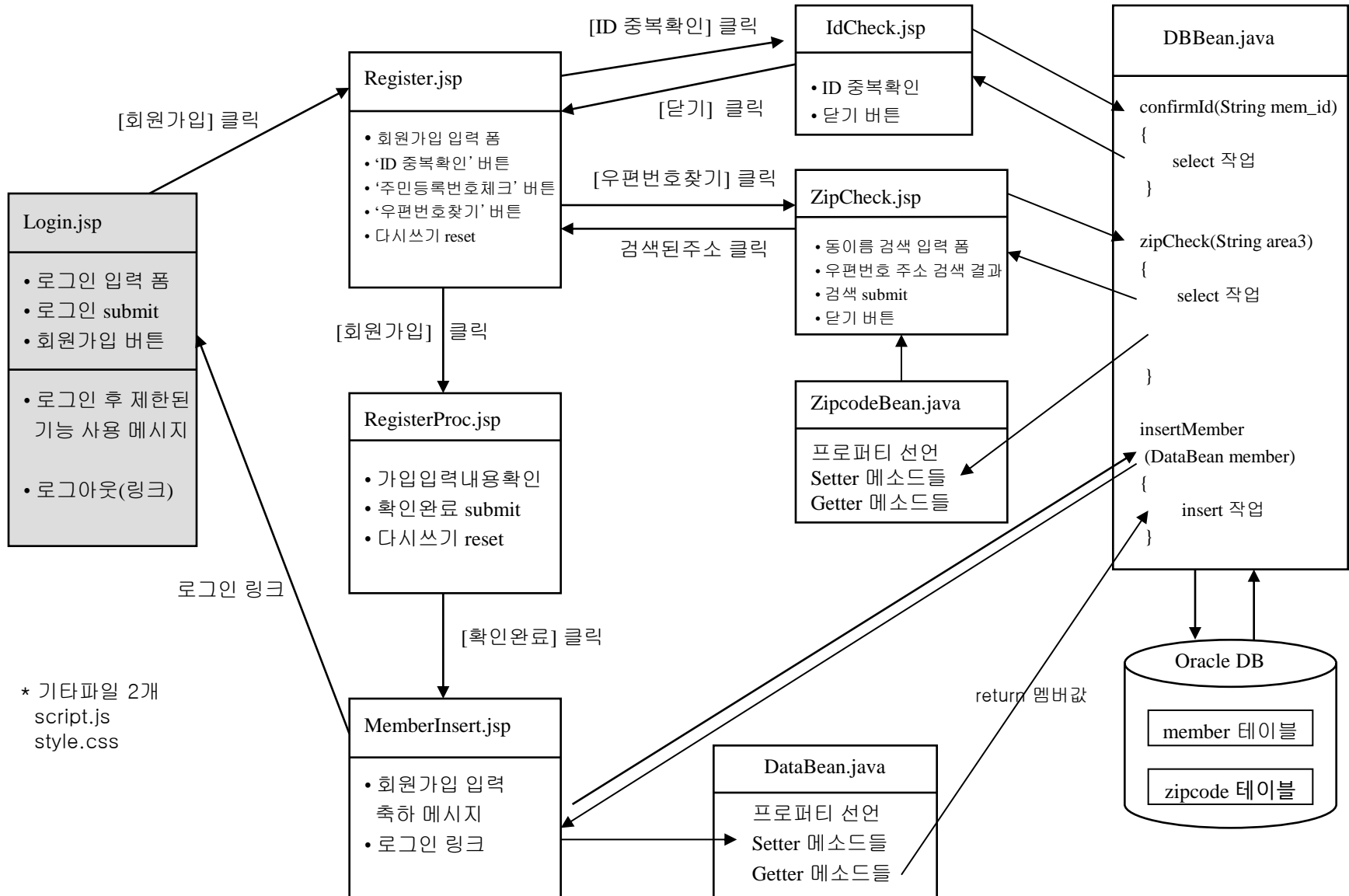
로그인(회원인증) 흐름도



■ 회원가입 프로젝트 디렉토리 구조



회원가입 흐름도



회원 테이블

```
SQL> create table member (  
    id varchar2(20) not null primary key,  
    passwd varchar2(20) null,  
    name varchar2(20) null,  
    mem_num1 char(6) null,  
    mem_num2 char(7) null,  
    e_mail varchar2(30) null,  
    phone varchar2(30) null,  
    zipcode char(7) null,  
    address varchar2(60) null,  
    job varchar2(30) null);
```

--회원의 Id
-- 암호
-- 회원이름
-- 주민등록번호 앞자리
-- 주민등록번호 뒤자리
-- 이메일
-- 회원의 전화번호
-- 우편번호
-- 회원의 주소
-- 회원의 직업

```
SQL> desc member;
```

```
SQL> insert into member values('test','1234',  
    '홍길동','123456','789000',  
    'nup49rok1@empal.com',  
    '02-123-0987','132-098','서울시 강북구','컨설턴트');
```

```
SQL> select * from member;
```

데이터베이스 커넥션 풀(DB Connection Pool)

● 데이터베이스 연동 프로그램의 문제점

- ▶ 많은 클라이언트의 동시 다발적인 데이터베이스 접속 및 접속종료 문제
- ▶ 중요한 데이터베이스 처리시 트랜잭션 처리 문제

● 데이터베이스 커넥션 풀이란

- ▶ 애플리케이션과 데이터베이스와의 연결을 의미하는 것
- ▶ 애플리케이션에서 데이터베이스 연결이 필요할 때 매번 연결하는 것은 비효율적임
- ▶ 미리 어느 정도의 연결을 만들어 두고(pool) 필요할때 사용하는 개념을 말함
- ▶ 보통 커넥션 풀 이라고 하면 단순히 미리 연결을 만들어 두는것 이외 적절한 연결을 관리할 수 있는 시스템 적인 개념을 포함

Connection Pool 구조

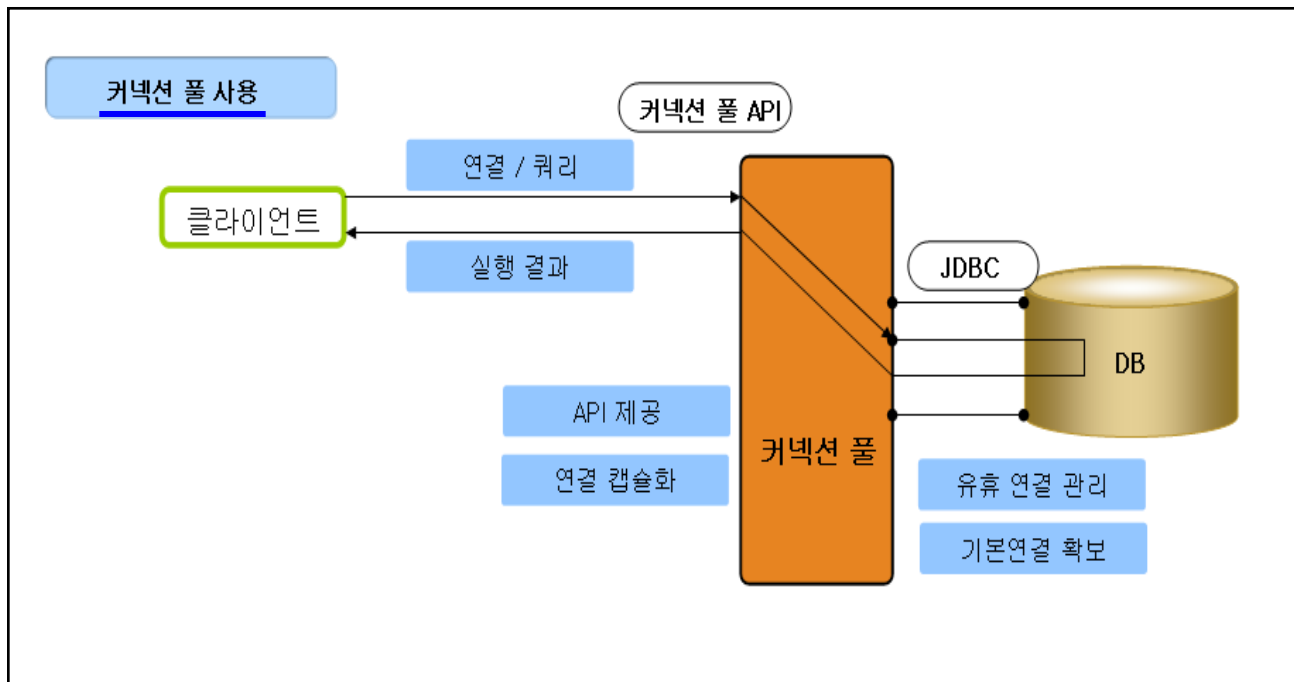
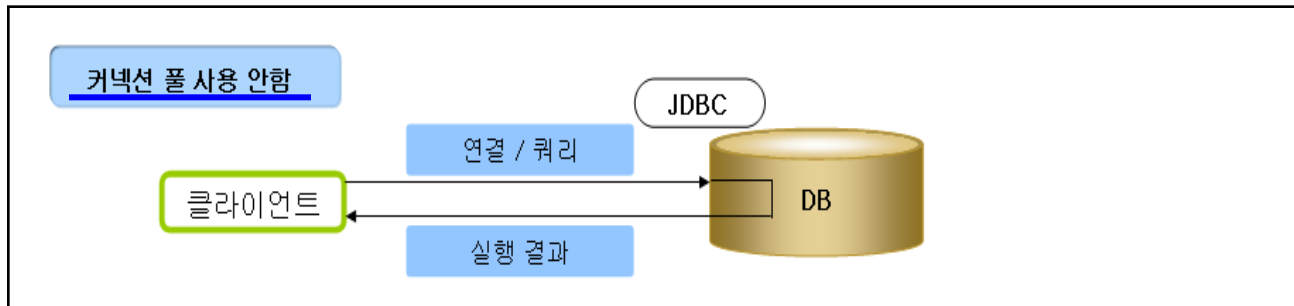
● 커넥션 풀 구조

▶ 데이터베이스 커넥션 풀은 다양한 형태로 구현이 가능

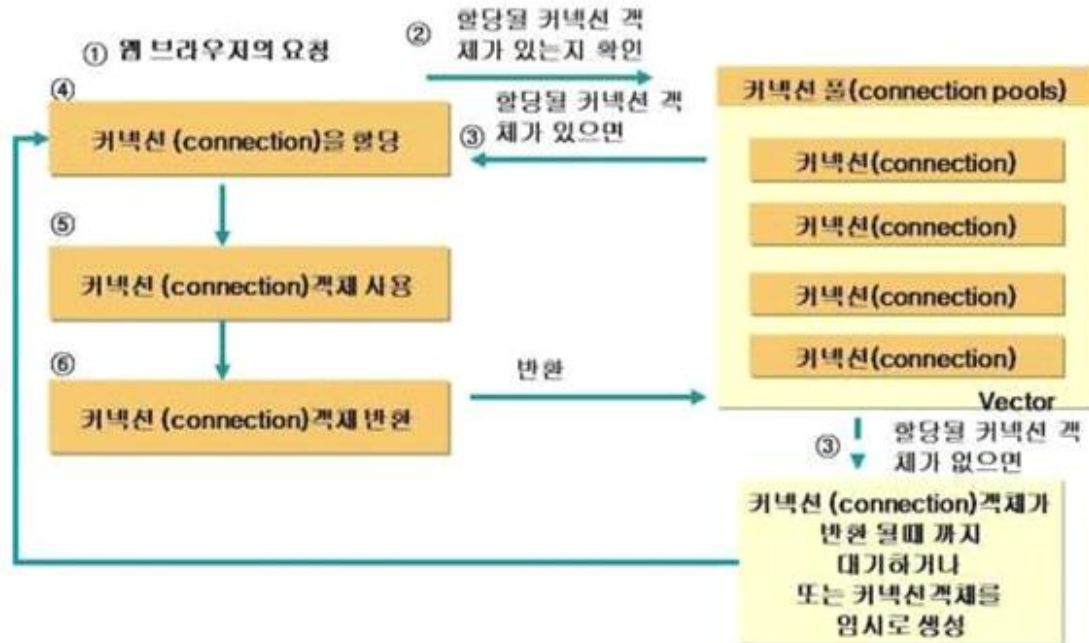
▶ 일반적인 동작 형태

- 웹 애플리케이션 서버가 시작될 때 일정 수의 커넥션을 미리 생성한다.
- 웹 애플리케이션 요청에 따라 생성된 커넥션 객체를 전달한다.
- 일정 수 이상의 커넥션이 사용되면 새로운 커넥션을 만든다.
- 사용하지 않은 커넥션은 종료하고 최소한의 기본 커넥션을 유지한다.

커넥션 풀 동작 구조



커넥션 풀 동작 구조



(1) 커넥션 풀을 사용하지 않는 경우

클라이언트에서 직접 데이터베이스에 연결하며, 사용 후 연결을 종료하는 구조이다.

(2) 커넥션 풀을 사용하는 경우

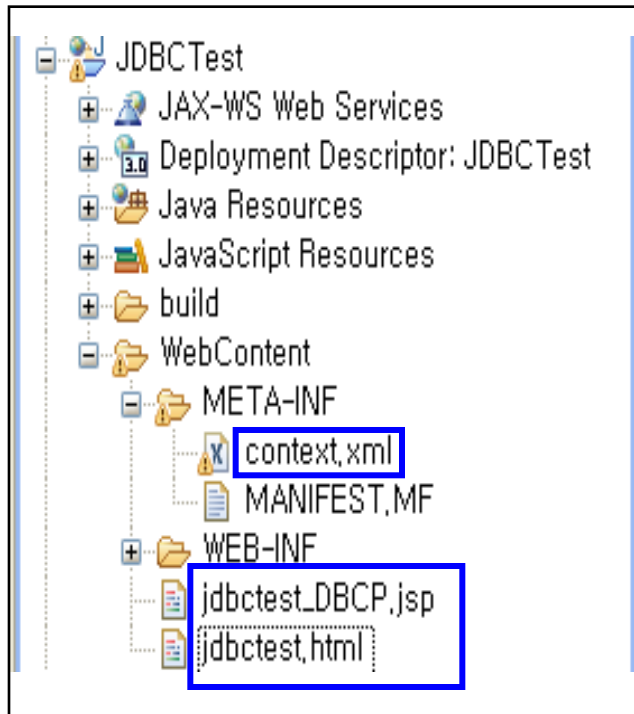
일정 수의 커넥션이 미리 만들어져 있고, 클라이언트 요청 시 커넥션 풀 관리자에 의해 새로운 커넥션이 만들어지는 것이 아니라 이미 만들어진 커넥션 중 사용 가능한 커넥션을 이용하는 구조이다. 이는 잦은 데이터베이스 접속이나 동시 다발적인 사용자 접속에도 보다 빠른 속도로 처리가 가능한 구조이다

[참고]

▶ DataSource

- 자바에서 다양한 데이터베이스 연결을 위해 캡슐화 해 놓은 객체로 데이터베이스 커넥션 풀의 사용은 DataSource 객체 참조로 부터 시작한다.
- 애플리케이션에서 DataSource 객체 참조는 JNDI 라고 하는 네이밍 서비스를 통해 이루어 진다.

커넥션 풀을 사용하는 경우 디렉토리 구조



JdbcTest 프로젝트

- WebContent - META-INF 폴더에 context.xml 파일 새로 생성하여

```
<?xml version="1.0" encoding="UTF-8"?>
<Context docBase="JdbcTest" path="/JdbcTest" reloadable="true"
    source="org.eclipse.jst.jee.server:JdbcTest">
    <Resource name = "jdbc/oracle"
        auth = "Container"
        driverClassName = "oracle.jdbc.driver.OracleDriver"
        type = "javax.sql.DataSource"
        url = "jdbc:oracle:thin:@localhost:1521:xe"
        username = "testdb"
        password = "testdb123"
        maxActive = "20"
        maxIdle = "10" />
</Context>
```

- 참고

<Context> 태그는 현재 애플리케이션에 대한 설정 태그이며

<Resource> 태그는 사용할 커넥션 풀 자원을 등록하는 태그이다

context.xml 파일 생성시 주의할 점

```
<Context docBase="JdbcT" path="/JdbcTest" reloadable="true"  
        source="org.eclipse.jst.jee.server:JdbcTest">
```

위 부분은 Project Explorer 창에 - Servers

- Tomcat v0.0 Server at localhost-config

- server.xml 파일의 아래 부분에서 따온다

<Context> 태그 부분은 현재 애플리케이션에 대한 설정으로,
이클립스에서 자동으로 생성한 경우에는 태그 바디가 없이 끝나는 구조이다
(예를 들어 <Context />)

그러므로 <Context> 태그 부분은 태그 바디를 가질 수 있도록 수정해야 한다

트랜잭션(Transaction)

개요

- ▶ 데이터베이스에서 일련의 작업을 하나로 묶어 처리하는 것을 의미함
- ▶ 트랜잭션과 관련된 대표적인 사례

한 고객이 A통장에서 B통장으로 계좌를 이체하려 한다. 실제 이러한 과정은 A통장에서 돈을 인출한 후, B통장으로 입금하는 과정을 거친다. 그런데 만약 A통장에서는 정상적으로 돈이 인출 되었지만 B통장으로 입금을 처리하는 부분에 문제가 생길 경우, A통장에서 인출된 돈은 어떻게 처리해야 할까?

- A통장이 있는 테이블에서 update 문을 이용해 이체 금액만큼 차감
- B통장이 있는 테이블에서 update 문을 이용해 이체 금액만큼 추가
- 트랜잭션은 개별적인 update 문을 데이터베이스 처리의 완료로 보는 것이 아니라 두 update 문이 모두 성공적으로 마치는 것을 하나의 처리로 간주함

commit과 rollback 명령

● commit과 rollback

- ▶ commit : DB에 트랜잭션이 완료되었음을 알리는 명령이다.
- ▶ rollback : 트랜잭션을 취소하는 명령으로, 현재 연결에서 수행한 결과를 원래대로 되돌리는 역할을 한다.

● JDBC에서 트랜잭션 처리

▶ 애플리케이션 에서 개별적으로 트랜잭션을 처리 할때 이용하는 방법

▶ auto commit 모드 해제

- 기본적으로 JDBC 에서는 자동 커밋 모드로 설정되어 있음
- 따라서 SQL 문을 실행하면 모든 처리가 바로 바로 이루어짐
- 트랜잭션 관리를 위해서는 자동 커밋 모드 해제가 필요함

```
conn.setAutoCommit(false);
```

▶ JDBC에서 commit과 rollback

- Connection 클래스가 제공하는 메서드를 이용한다

```
conn.commit();  
conn.rollback();
```

표현 언어(EL)

- 표현 언어(Expression Language)는 처음 JSTL(JSP Standard Tag Library)이 소개되었을 때 나온 것으로, SPEL(Simplest Possible Expression Language)에 기본을 두고 있다
- 따라서 처음에는 JSTL의 부분처럼 사용되었으나 지금은 JSP 스펙에 포함됨으로써 JSTL과 상관없이 JSP 페이지 내에서 **표현식 등을 대체하는 용도**로 사용될 수 있다
- 표현 언어는 자바 빈즈 속성 값을 보다 쉽고 제약을 덜 받는 방법으로 사용하기 위해 나온 것으로 JSP 파일이 useBean 액션 태그나 표현식 등으로 복잡해지는 것을 막고 일종의 템플릿 코드처럼 사용할 수 있도록 해준다



표현 언어 기본 문법

- 표현 언어는 '\$'로 시작한다
- 모든 내용은 '{표현식}' 과 같이 구성된다
- 표현식에는 기본적으로 변수명, 혹은 '속성명.메소드명' 구조로 이루어진다
- 표현식에는 부가적으로 숫자, 문자열, boolean, null과 같은 상수 값도 올 수 있다
- 표현식에는 기본적인 연산이 가능하다

내장 객체

- 표현 언어에서는 효율적으로 JSP와 상호작용하도록 다음과 같은 내장 객체를 지원한다

내장 객체	기능
pageScope	page 범위에 포함된 속성 값에 접근할 수 있는 객체
requestScope	request 범위에 포함된 속성 값에 접근할 수 있는 객체
sessionScope	session 범위에 포함된 속성 값에 접근할 수 있는 객체
applicationScope	application 범위에 포함된 속성 값에 접근할 수 있는 객체
param	request.getParameter("xxx")로 얻을 수 있는 값들 \${param.xxx} 처럼 사용
paramValues	request.getParameterValues("xxx") 와 동일 기능 수행 \${paramValues.xxx} 처럼 사용

JSTL

- JSTL(JSP Standard Tag Library)은 커스텀 태그 라이브러리 기술을 이용해 일반적으로 필요한 기능들을 표준화한 것으로 크게 코어(CORE), XML, I18N(국제화), 데이터베이스(SQL), 함수(Function) 라이브러리로 나뉜다
- jsp 파일에 JSTL을 적용하려면 **jstl.jar와 standard.jar 파일**을 해당 프로젝트의 '**WEB-INF\lib**' 폴더로 복사한다

JSTL 라이브러리별 uri 및 prefix

라이브러리	uri	prefix
코어	http://java.sun.com/jsp/jstl/core	c
XML	http://java.sun.com/jsp/jstl/xml	x
I18N(국제화)	http://java.sun.com/jsp/jstl/fmt	fmt
데이터베이스	http://java.sun.com/jsp/jstl/sql	sql
함수	http://java.sun.com/jsp/jstl/functions	fn

- JSTL은 표현언어에서 사용할 수 있는 함수를 제공한다
- **fn:replace(string, before, after)**
 - ▶ string 내에 있는 before 문자열을 after 문자열로 모두 변경해서 반환하는 함수이다
(예) `${fn:replace(list.gb_contents, CRLF, "
")}`

JSTL 라이브러리

라이브러리	기능	태그	접두어
코어	General Purpose Actions (일반적인 것)	catch out remove set	c
	Conditional Actions (조건)	if choose when otherwise	
	Iterator Actions (반복)	forEach forTokens	
	URL Related Actions (URL 관련 있는)	import redirect url param	

[참고]

코어 라이브러리는 표현식, 반복처리, URL 관리 기능 등을 제공하므로 웹 애플리케이션을 개발할 때 유용하게 사용할 수 있다

JSTL 라이브러리

라이브러리	기능	태그	접두어
I18N(국제화)	Locale	setLocale	fmt
	Number and DateFormatting	formatNumber formatDate	

■ 국제화(Internationalization)

<c:if> 태그

- 해당 조건에 맞으면 태그의 body 내용을 처리한다.
- 자바의 if문과 유사하지만, else는 지원하지 않는다.

[문법]

```
<c:if test="조건" var="변수명">  
    body 내용  
</c:if>
```

- test 속성 – 검사할 조건
- var 속성 – test 조건의 결과를 저장할 변수(결과는 true 혹은 false)

<c:choose>, <c:when>, <c:otherwise> 태그

- 이들 태그는 함께 사용되며 자바의 switch문과 유사한 기능을 한다
- <c:choose> 태그 내에는 <c:when> 태그 여러 개가 올 수 있다

[문법]

```
<c:choose>
  body 내용 (<when> and <otherwise> subtags)
  <c:when test="testCondition">
    body content
  </c:when>
  <c:otherwise>
    conditional block
  </c:otherwise>
</c:choose>
```

<c:forEach> 태그

- 반복문과 관련된 태그로 forEach 태그는 자바 for문과 유사하다.
가장 유용한 JSTL 태그 중 하나이다

[문법]

① 컬렉션 객체의 크기만큼 반복하는 경우

```
<c:forEach var="변수" items="컬렉션객체">  
    반복 문장  
</c:forEach>
```

② 지정된 횟수 만큼 반복하는 경우

```
<c:forEach begin="반복시작번호" end="반복끝번호" var="변수">  
    반복 문장  
</c:forEach>
```

■ JSTL 응용

[예시] **게시판 구현 시 페이징 처리**

```
<c:forEach begin="${pageMaker.startPage}" end="${pageMaker.endPage}" var="pageNum">
  <a href='<c:url value="/admin/user_list.do?search=${search}&page=${pageNum}"/>'>
    <i class="fa"> <input value="${pageNum}" type="button" class="button"> </i>
  </a>
</c:forEach>
```

<c:url> 태그

- URL을 생성해주는 기능을 제공한다.

[문법]

```
<c:url value="URL" [var="varName"] [scope="영역"]>
  <c:param name="이름" value="값" />
</c:url>
```

- var 속성과 scope 속성은 생략 가능하다.
- var 속성을 지정하지 않으면 현재 위치에 생성한 URL을 출력하며, var 속성을 지정하면 해당 변수에 생성한 URL을 저장한다.
- <c:param> 태그를 이용해서 파라미터를 URL에 추가할 수 있다.

[예시] 게시판 구현 시 페이징 처리

```
<c:if test="${pageMaker.prev}">
  <a href='<c:url value="/admin/user_list.do?search=${search}&page=${pageMaker.startPage-1 }"/>'>
    <input value="이전" type="button" class="button">
  </a>
</c:if>
```




jQuery

- jQuery는 javascript 함수들로 이루어진 라이브러리이다.
- jQuery는 Javascript 프로그래밍을 매우 단순하게 해준다.
- jQuery 라이브러리는 javascript 파일로 저장되어 있고, 모든 jQuery 메서드를 담고 있다.
- jQuery 라이브러리는 다음과 같은 특징이 있다.
 - HTML 엘리먼트 선택하고
 - HTML 엘리먼트 조작(즉, 그 엘리먼트에 “액션” 수행)
 - CSS 조작
 - HTML 이벤트 함수
 - javascript 효과 및 애니메이션

jQuery 문법

- jQuery 문법은 HTML 엘리먼트를 원하는대로 선택하고 그 엘리먼트에 어떤 "액션"을 수행하게 만든다.
- 기본 문법은

`$(selector).action();`

- \$표시는 jQuery라는 것을 정의하고
- (selector)는 "질의하거나 찾을" HTML 엘리먼트를 의미하며
- jQuery action()은 이 엘리먼트가 수행할 액션이다.

Document Ready Function

- 모든 jQuery 메소드가 document.ready() 함수 안에 존재한다.

```
$(document).ready(function() {  
    // jQuery 함수들은 여기에  
    ...  
    ...  
});
```

- jQuery 메소드가 document.ready() 함수안에 넣는 이유는 => jQuery 코드가 문서가 로딩이 끝나기 전 (혹은 문서가 준비되기 전)에 실행되는 것을 막기 위한 것이다.

jQuery Selectors

- jQuery selectors는 여러개 또는 한 개의 HTML 엘리먼트를 선택하고 조작할 수 있게 한다.
키포인트는 jQuery를 이용하여 효과를 주기 원하는 엘리먼트를 정확하게 선택하는 방법을 배우는 것이다.
- jQuery selector는 엘리먼트 이름, 특성이름, 또는 content로 HTML 엘리먼트를 선택할 수 있게 한다.

[jQuery Element Selectors 예]

`$("p")`는 모든 `<p>` 엘리먼트를 선택한다.

`$(".intro")`는 `class="intro"`인 모든 `<p>` 엘리먼트를 선택한다.

`$("#demo")`는 `id="demo"`인 모든 `<p>` 엘리먼트를 선택한다.

Ajax

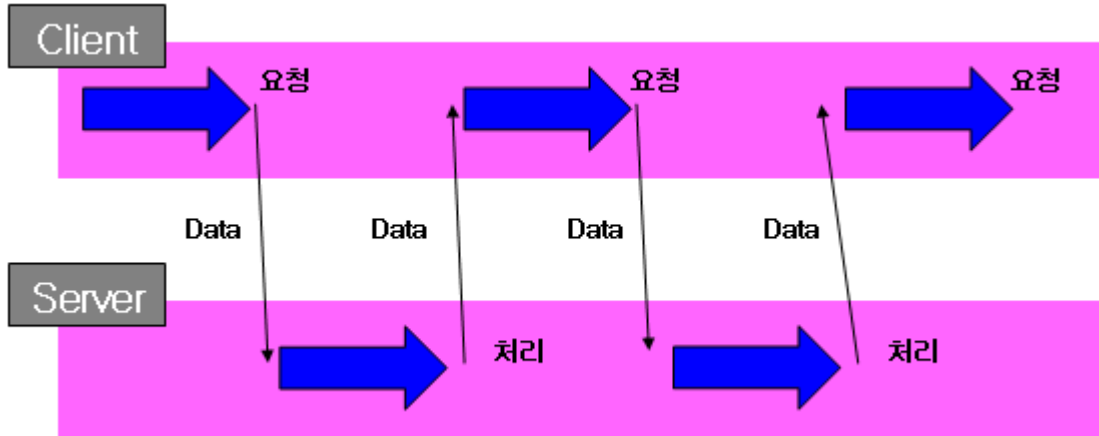
- Asynchronous Javascript and XML의 약자
- 비동기로 Javascript의 비동기 통신 기술을 이용해서 XML 데이터를 주고 받는 기술이다
- 대화식 웹 애플리케이션의 제작을 위해 아래와 같은 조합을 이용하는 웹 개발 기법이다.
 - 표현 정보를 위한 HTML (또는 XHTML) 과 CSS
 - 동적인 화면 출력 및 표시 정보와의 상호작용을 위한 DOM, Javascript
 - 웹 서버와 비동기적으로 데이터를 교환하고 조작하기 위한 XML, XSLT, XMLHttpRequest



Ajax의 장점

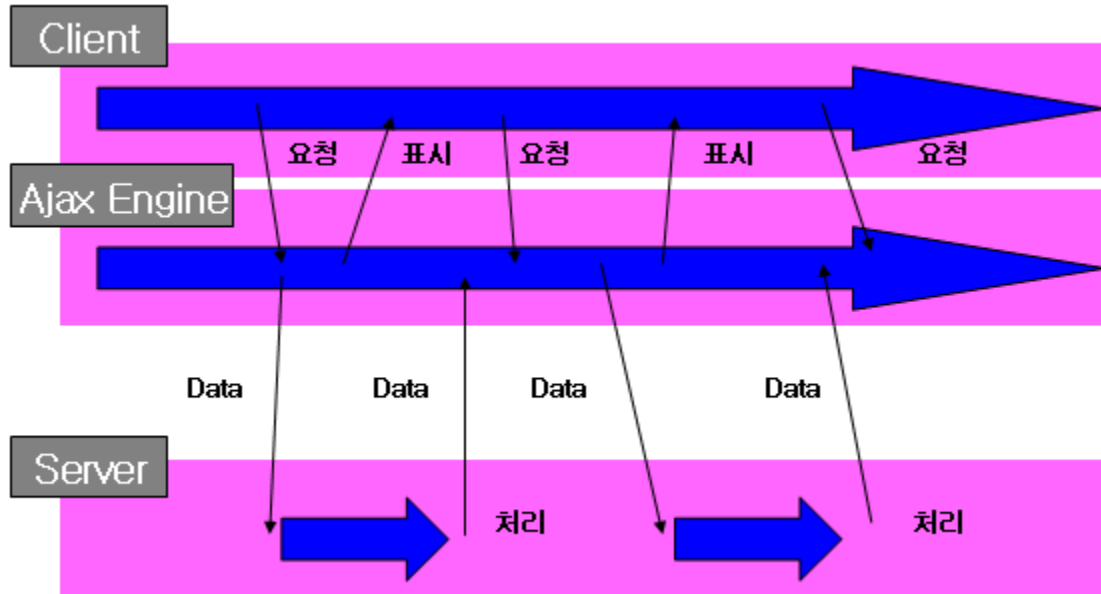
- 페이지 이동없이 고속으로 화면을 전환할 수 있다.
- 서버 처리를 기다리지 않고, 비동기 요청이 가능하다.
- 수신하는 데이터 양을 줄일 수 있고, 클라이언트에게 처리를 위임할 수도 있다.

■ 동기 통신 개념



- 클라이언트에서 사용자가 행동함에 따라 데이터가 서버로 전송되고
- 서버가 이를 받아 처리한 후 클라이언트로 보내면, 클라이언트가 이를
- 받아서 다시 처리하는 순서로 통신하게 된다.
- **서버의 처리를 기다리는 순간 동안에 사용자는 동작하지 못한다.**

■ 비동기 통신 개념

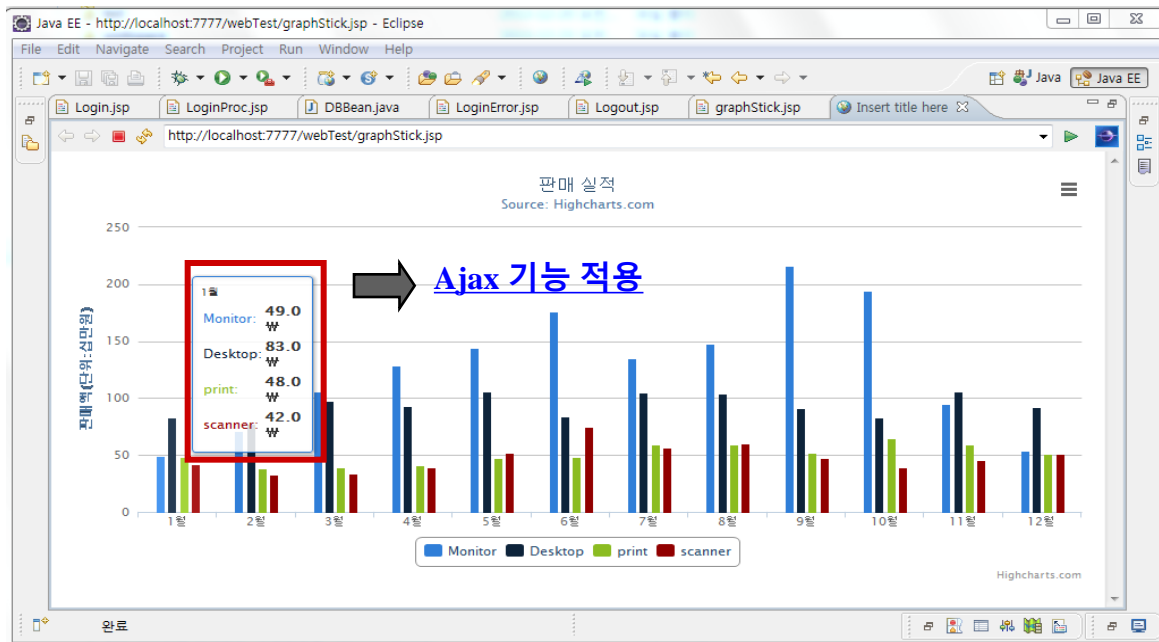
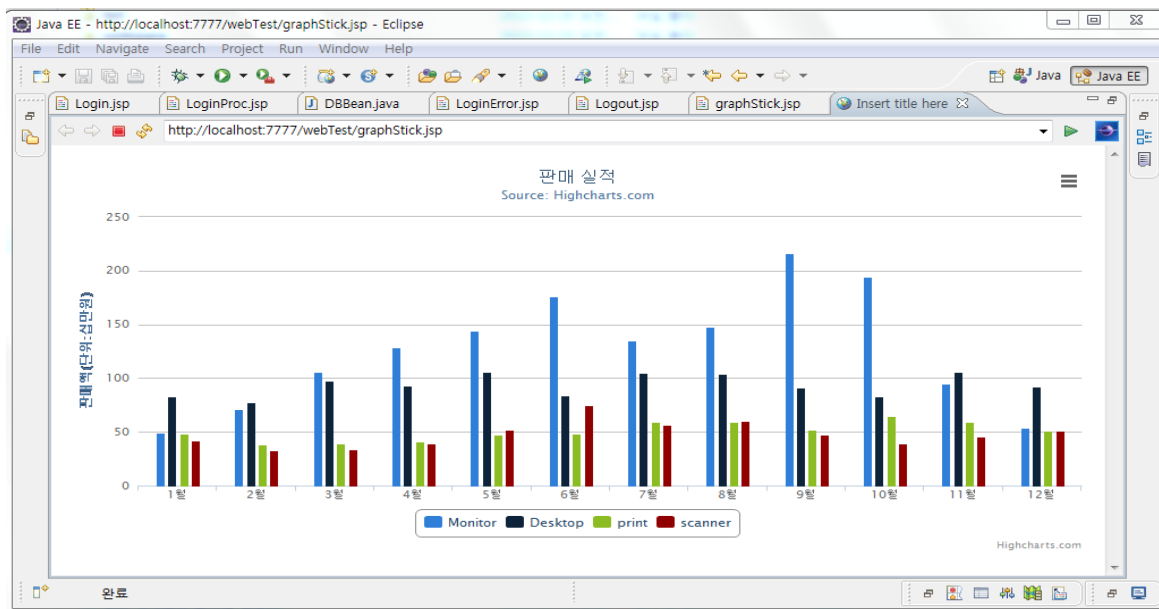


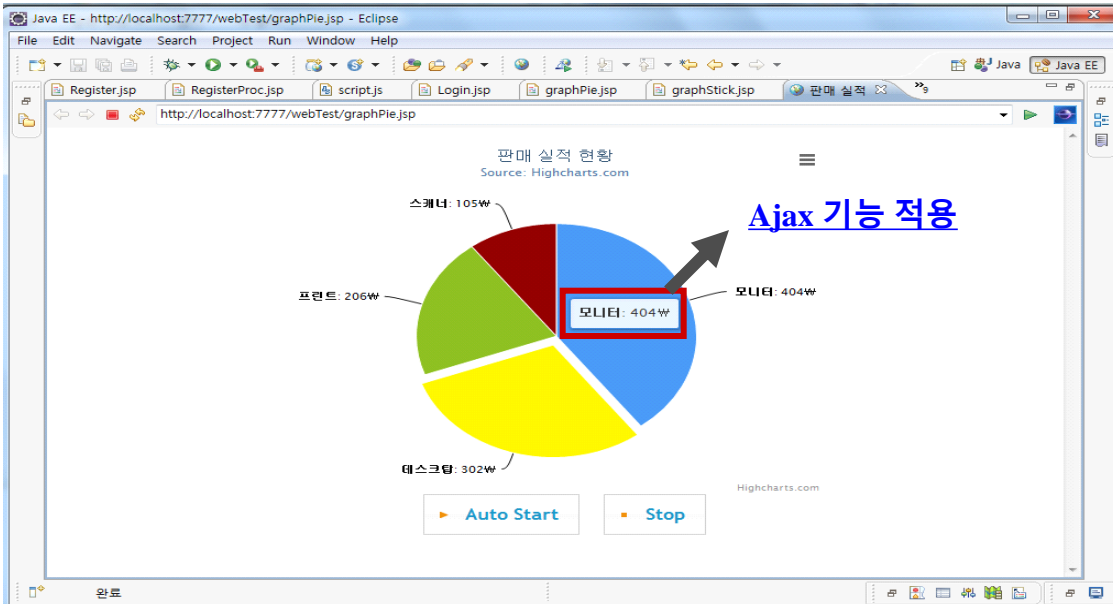
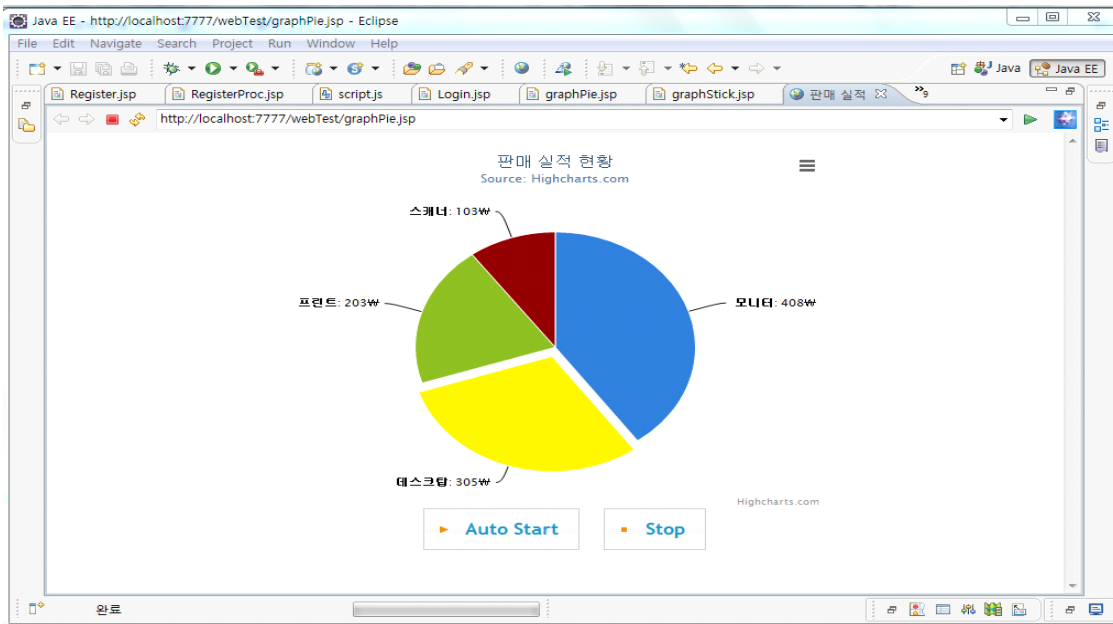
- 동기 통신과의 차이점은 동기 통신은 사용자의 화살표가 중간 중간에 끊기는 반면 비동기 통신은 끊기지 않고 계속해서 이어져 있다는 것이다.
위의 그림이 의미하는 것은 사용자가 일일이 새로운 페이지를 로드하는 것이 아니라 **중간에 Ajax 엔진이 백그라운드에서 새로운 데이터를 요청하고 받는 역할을 하게 된다는 것이다.**
결론은 XMLHttpRequest를 이용하면 페이지가 모두 로드된 이후에도 데이터를 요청할 수 있는 기능을 구현 할 수 있다는 것이다.



jquery 라이브러리

```
<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.2/jquery.min.js"></script>  
<script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.18/jquery-ui.min.js"></script>  
<script src="http://code.highcharts.com/highcharts.js"></script>  
<script src="http://code.highcharts.com/modules/exporting.js"></script>
```





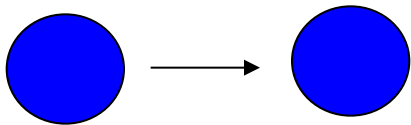
MVC패턴 구현 방안

MVC패턴 구성요소

영역	구현 방식
모델(Model)	데이터 영역으로 DAO(Data Access Object), DO(Data Object) 등으로 구분해 구현하기도 한다. iBatis, Hibernate와 같은 퍼시스턴스 프레임워크를 사용하기도 한다. EJB와 연동할 수도 있으며 EJB3.0의 POJO 기반의 Persistence API도 있다.
뷰(View)	JSP를 기본으로 표현 언어, JSTL, 커스텀 태그라이브러리 등을 함께 사용하며, 모듈화된 사용자 인터페이스 모델인 JSF(Java Server Faces)도 이용할 수 있다.
컨트롤러(Controller)	MVC 패턴의 중심이 되는 부분으로, 직접 구현하거나 구현된 솔루션을 이용할 수 있다. 대표적으로 스트러츠 프레임워크가 있다.

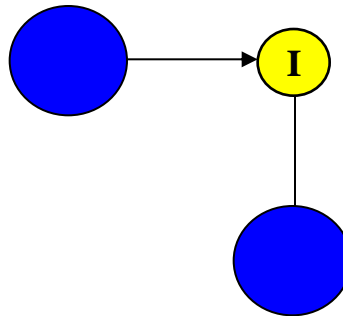
자바에서 애플리케이션 설계 방식

Sample1 방식

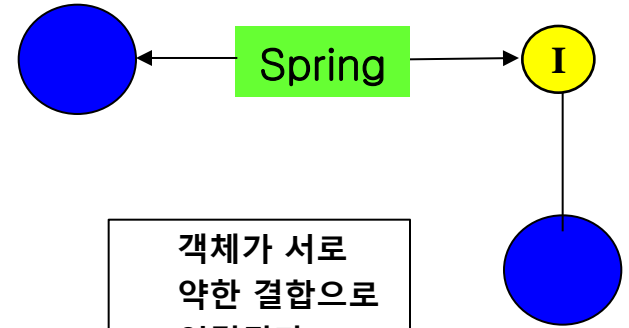


객체가 서로
강한 결합으로
묶여 있다

Sample2 방식

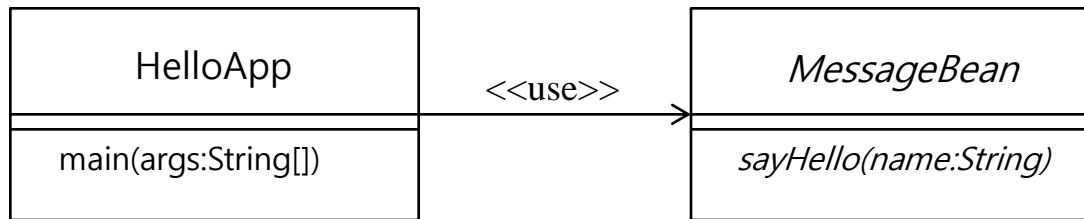


Sample3 방식

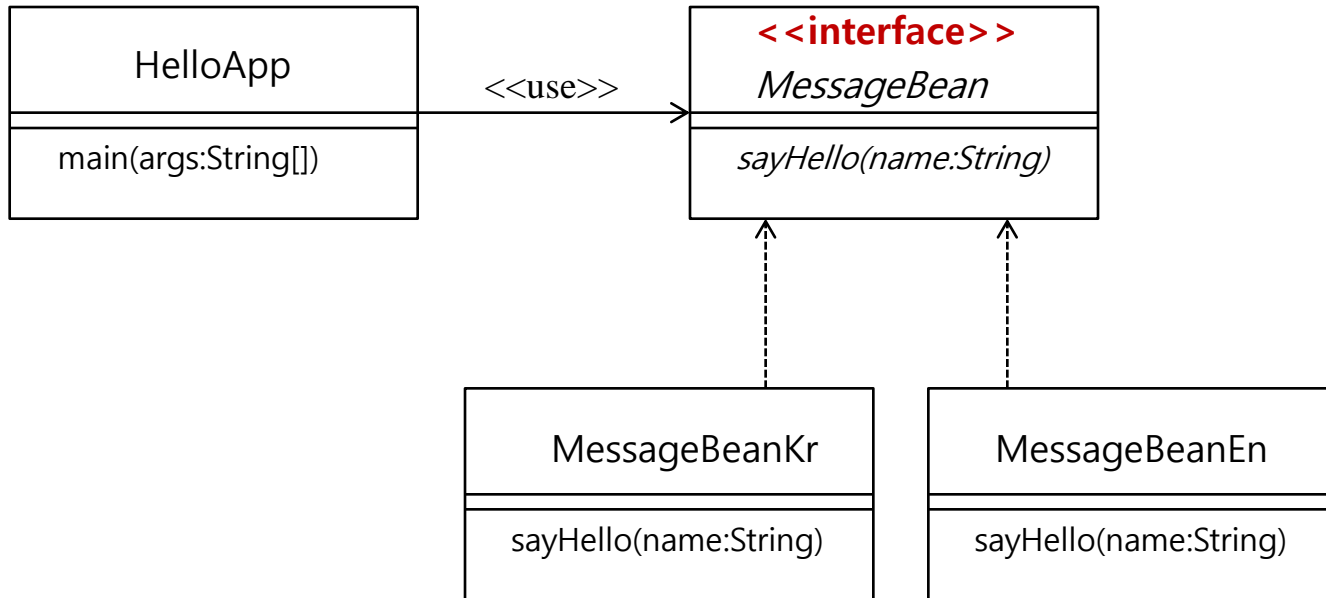


객체가 서로
약한 결합으로
연결된다

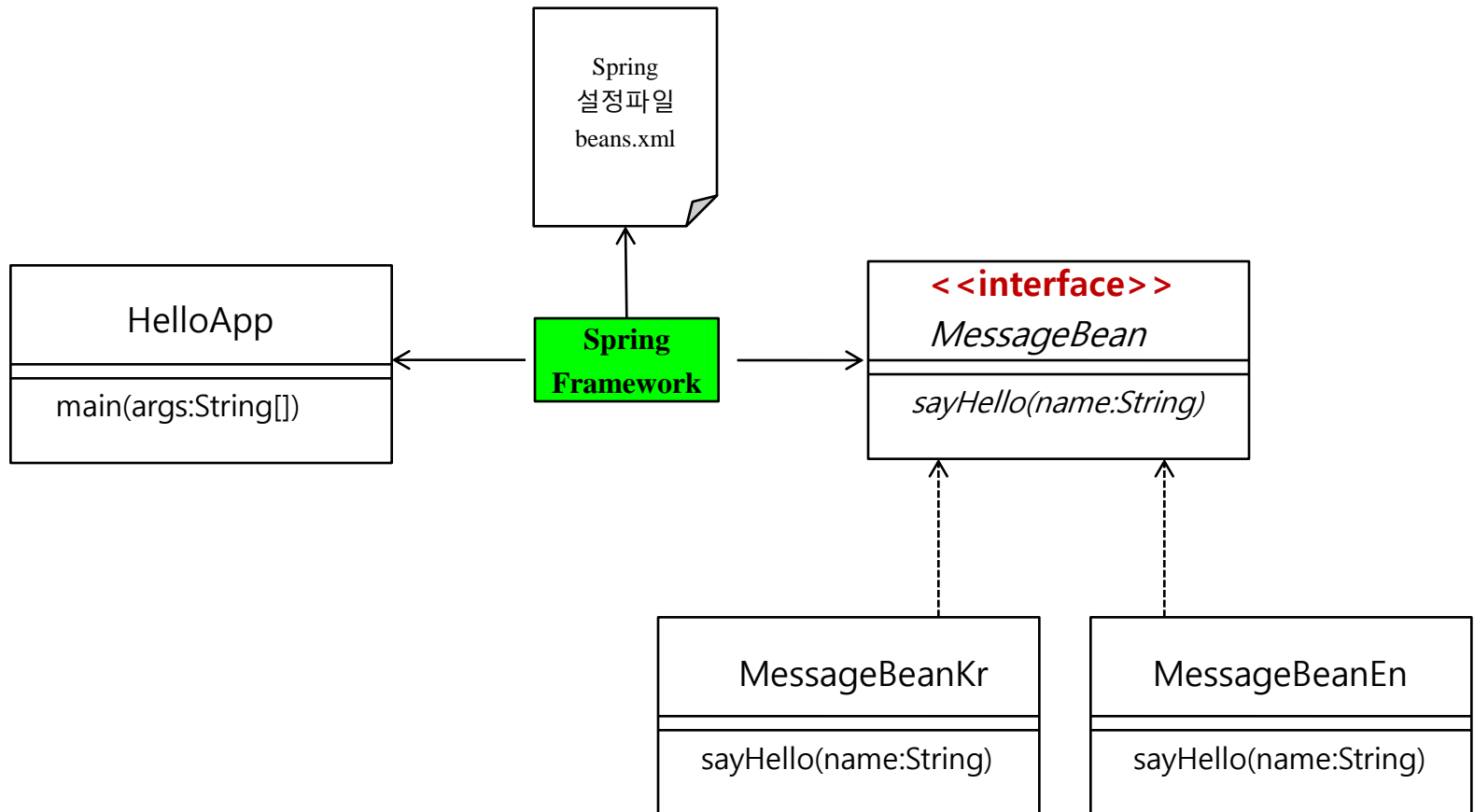
Sample1 클래스 다이어그램



Sample2 클래스 다이어그램(인터페이스 사용)

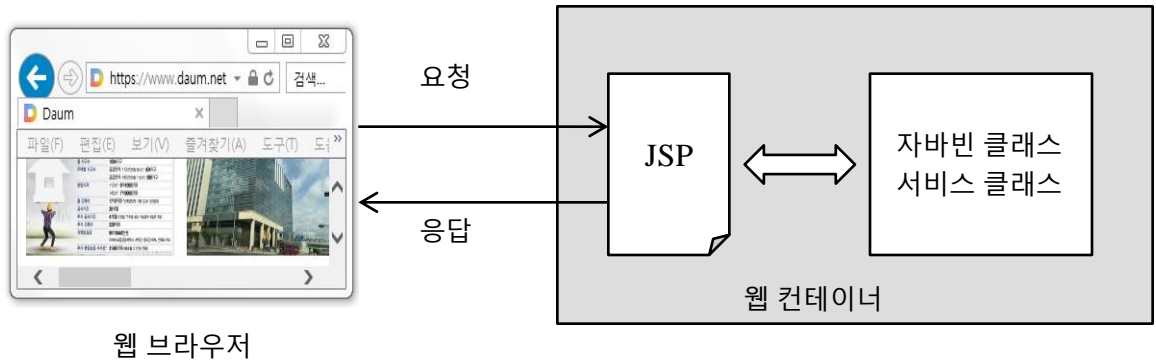


Sample3 클래스 다이어그램(스프링 사용)

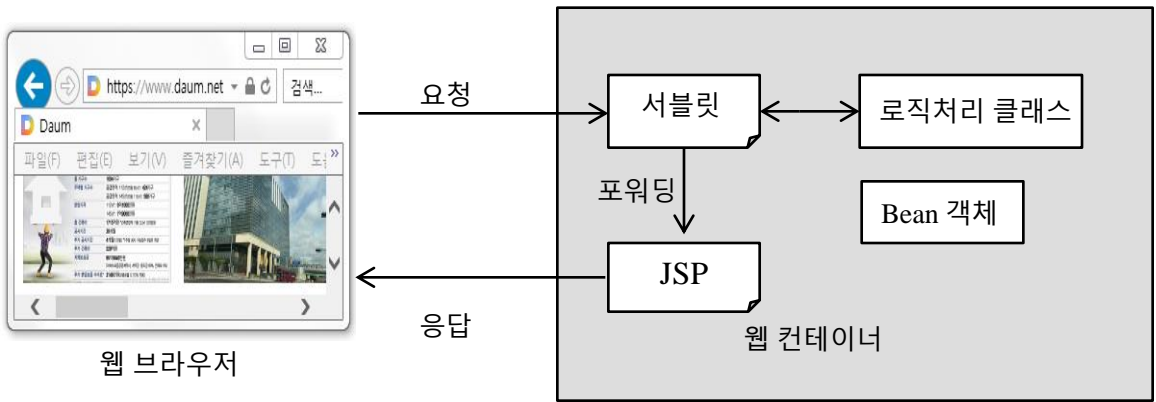


<JSP 기반 웹 어플리케이션의 구조>

모델 1 구조

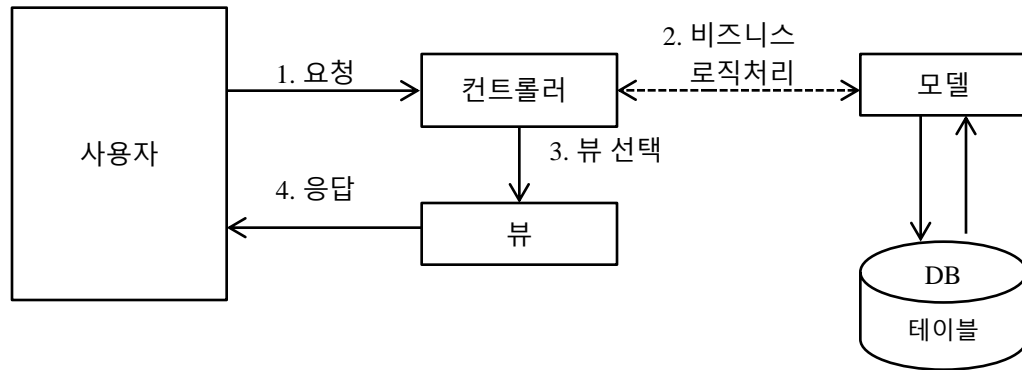


모델 2 구조



<JSP 기반 웹 어플리케이션의 구조>

MVC 패턴의 구조





스프링을 활용하기 위한 설계

- 스프링을 활용하려면 **인터페이스 기반 설계**를 이해하는 것이 중요하다
- **의존 관계**는 구상 클래스가 아닌 인터페이스를 통해 이루어진다
- 어떤 클래스를 사용할 때는 구상 클래스가 아닌 인터페이스를 통해 사용하려는 클래스의 메소드를 호출한다.

스프링의 핵심인 의존관계주입(DI)이란?

- 프로그래밍에서 **의존성(Dependency)**이란, 어떤 클래스가 자신의 임무를 다하기 위해 필요한 값(필드값)이나 사용할 다른 클래스와의 관계를 말한다.
- **주입(Injection)**이란, 어떤 클래스의 인스턴스에 대해 외부로부터 '의존성'을 설정하는 것을 말한다.
- **의존 관계 주입 컨테이너가 하는 역할은** 어떤 클래스가 필요하는 값이나 인스턴스를 생성, 취득하고, 그 클래스의 인스턴스에 대해 설정하는 것이다. 이렇게 하면 필요한 인스턴스를 생성, 취득하는 코드를 직접 만들지 않아도 된다. 그 결과 클래스간 관계가 느슨한 결합이 되어 의존성은 약해진다.



Spring 프레임워크

1. 애플리케이션에서 사용할 수 있는 프레임워크
2. 경량(Lightweight) 컨테이너
 - 필요할 때 필요한 모듈만 이용할 수 있기 때문에,
모든 요소가 포함된 EJB와 비교하여 스프링은 '경량 컨테이너'라 불린다
3. Dependency Injection(의존성 삽입이 지원되고)
4. Aspect – Oriented container(관점 지향 컨테이너)
5. JDBC, iBatis, MyBatis 와 같은 DB 처리와 관련하여 널리 사용되는 라이브러리와의 연동을 지원하고 있다



스프링의 DI

- DI(Dependency Injection: 의존 관계 주입)
- 스프링의 DI는 **부품의 연결을 특기**로 하는 기술이다.
- DI는 인터페이스를 이용해 컴포넌트화를 실현하는 것이다.
- 오브젝트 사이의 의존 관계를 만드는 것이다.
- 스프링의 대표적인 기능은 DI와 AOP다.
DI가 부품의 연결을 특기로 한다면 AOP는 각 부품이 해야 할 일에만 전념하게 한다.

스프링의 AOP

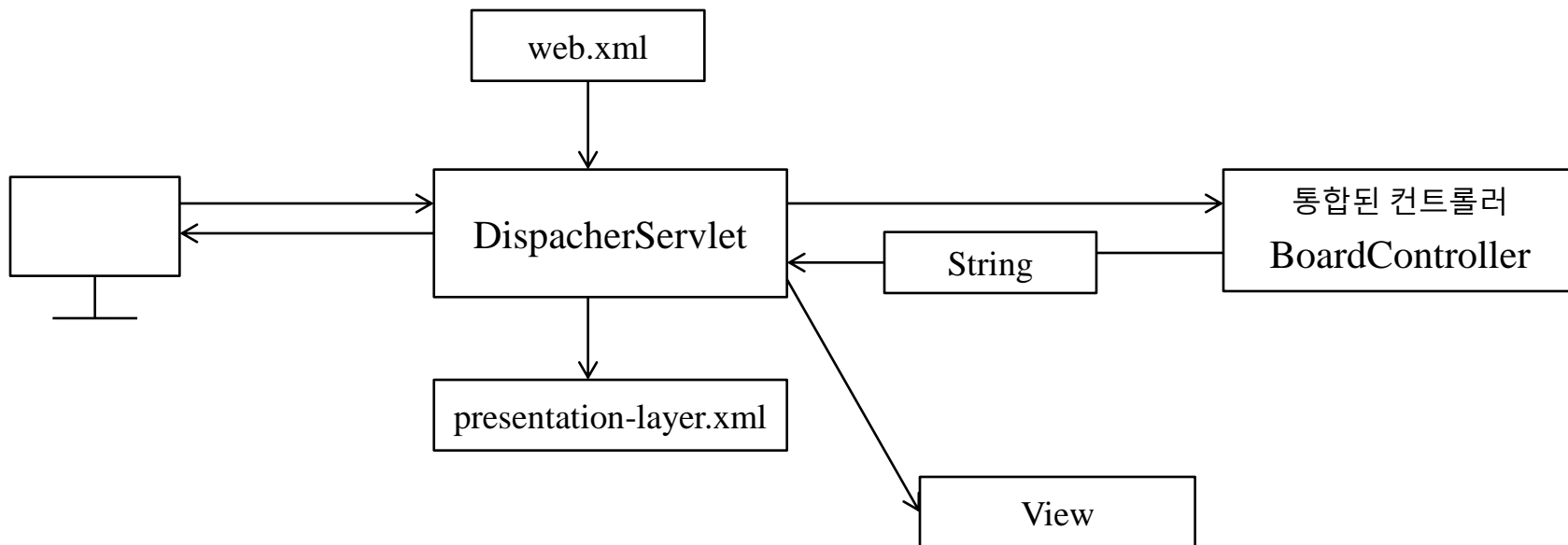
- AOP(Aspect Oriented Programming: **관점 지향 프로그래밍**)
- AOP란 업무 등 특정 책임이 있는 클래스(예를 들어 주문 클래스나 계좌 클래스) 안에 **본질적인 처리만 기술**하고, 본질적이지 않은 처리는 밖으로 꺼내는 기술이다.
- 구체적으로는 **로그 출력(로깅), 예외처리, 보안인증, 트랜잭션 처리** 등 **공통화**할 수 있는 처리를 **Aspect**라는 **하나의 단위로** 모아서 어떤 오브젝트가 원래 해야 할 일만을 하게 만드는 기술이다.
즉 기능을 핵심 비즈니스 로직과 공통모듈로 구분하고, 핵심로직에 미치지 않고 사이사이에 공통 모듈을 효과적으로 잘 끼워넣도록 하는 개발 방법을 말한다.

스프링 AOP에서 사용하는 용어

- **advice** : 언제 어떤 기능을 적용할 지에 대한 정의.
관점으로서 분리되고 실행시 위빙(weaving)된 구체적인 처리를 의미한다. advice가 위빙되는 인스턴스를 ‘대상객체’라 한다.
- **aspect** : 여러 객체에 공통으로 적용되는 공통 관심 사항
- **weaving** : 핵심로직에 공통로직을 삽입하는 것

스프링 AOP에서 사용하는 용어

- **joinpoint** : 공통 기능 적용 가능 지점. 즉 실행시의 처리 플로우에서 advice를 위빙하는 포인트를 의미한다. 구체적으로 '메소드 호출'이나 '예외발생'이라는 포인트를 joinpoint로 정의한다.
- **pointcut** : joinpoint 중에 실제로 적용할 지점. 하나 또는 복수의 joinpoint를 하나로 묶는 것을 말한다.
- **advisor** : advice와 pointcut을 하나로 묶어 다루는 것을 말한다. advisor는 관점 지향에서 '관점'을 나타내는 개념이다.



Mybatis 개요

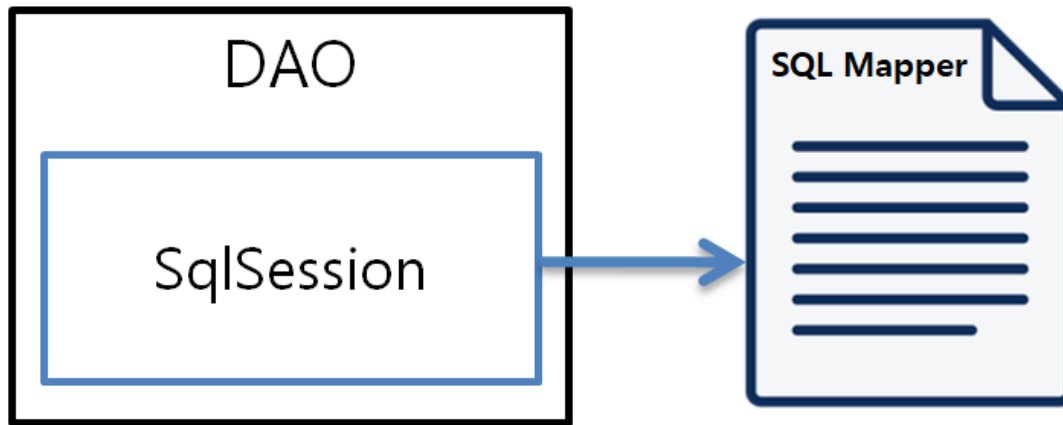


MyBatis

- SQL 질의문을 통해 **결과물을 개발자가 지정한 클래스에 매핑**하여 내보내주는 아주 유용한 프레임워크이다.
즉 기존 JDBC 방법은 SQL 질의문 실행 후 결과값을 모델에 담는 로직으로 소스가 길어졌지만, Mybatis는 **매핑된 모델에 자동으로** 변수와 컬럼의 값이 매핑되어 아주 편해졌다.
- 개발자가 지정한 SQL, 저장프로시저 그리고 몇가지 고급 매핑을 지원하는 Persistence 프레임워크이다.
- JDBC 코드와 수동으로 셋팅하는 파라미터와 결과 매핑을 제거한다.
즉 기존에 직접 jdbc connection, sql statement, parameter, result record set 등을 개발하던 것을 Mybatis 프레임워크가 대신해준다.
- 데이터베이스 레코드에 원시타입과 **Map 인터페이스** 그리고 자바 POJO를 설정하고 매핑하기 위해 XML과 애노테이션을 사용할 수 있다.

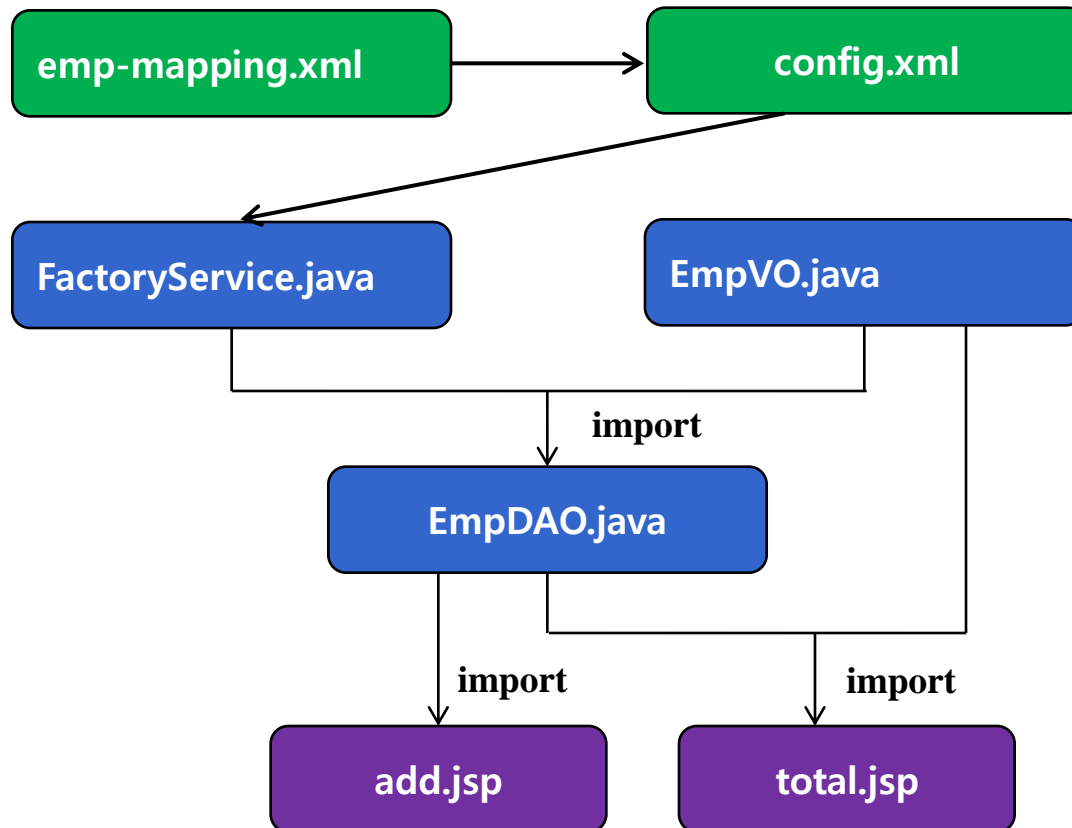
Mybatis 3.0의 주요 구성물

구성물	역할
Mybatis 설정 파일	데이터베이스의 접속 주소정보나 매핑 파일의 경로 등의 고정된 환경정보를 설정한다. XML 파일로 기술한다.
SqlSessionFactoryBulider	Mybatis 설정 파일을 바탕으로 SqlSessionFactory 를 생성한다. 애플리케이션을 시작할 때 사용해 SqlSessionFactory를 생성하면 SqlSessionFactoryBulider 오브젝트는 필요 없어지므로 참조를 유지할 필요는 없다. 애플리케이션의 시작시에 쓰고 버리는 이미지다.
SqlSessionFactory	SqlSession 을 생성한다. SqlSessionFactoryBulider 오브젝트는 스레드 세이프하며, 애플리케이션 안의 프로그램은 하나의 오브젝트를 싱글톤 패턴 으로 공유해야만 한다.
SqlSession	SQL 발행이나 트랜잭션 관리를 실행한다 . SqlSession 오브젝트는 스레드 세이프하지 않으므로 스레드마다 필요에 따라 생성하고 폐기한다.
Mapper 인터페이스	매핑 파일에 기재된 SQL을 호출하기 위한 인터페이스이다. Mapper 오브젝트는 SqlSession 오브젝트와 관련해 생성되므로 SqlSession 오브젝트와 함께 생성하고 폐기한다.
매핑 파일	SQL과 OR 매핑을 설정한다. XML 파일을 기술한다.



- MyBatis 사용 목적 중 하나는 DAO로부터 SQL문을 분리하는 것이다.
분리된 SQL문은 SQL mapper 파일에 작성하며 DAO에서는 SqlSession 객체가 SQL mapper 파일을 참조하게 된다.
- XML에서 SqlSessionFactory 빌드하기
모든 마이바티스 애플리케이션은 SqlSessionFactory 인스턴스를 사용한다.
SqlSessionFactory인스턴스는 SqlSessionFactoryBuilder를 사용하여 만들수 있다.
SqlSessionFactoryBuilder는 XML설정파일에서 SqlSessionFactory인스턴스를 빌드할 수 있다.
- XML설정파일에서 지정하는 마이바티스의 핵심이 되는 설정은
 - 트랜잭션을 제어하기 위한 TransactionManager과 함께
 - 데이터베이스 Connection인스턴스를 가져오기 위한 DataSource 를 포함한다.

자바에 Mybatis 프레임워크 적용하여 사원(Emp) 테이블에서
사원 '목록 보기' 및 '사원 추가' Web 개발 순서



<EMP_MyBatis 전체 설계도>

