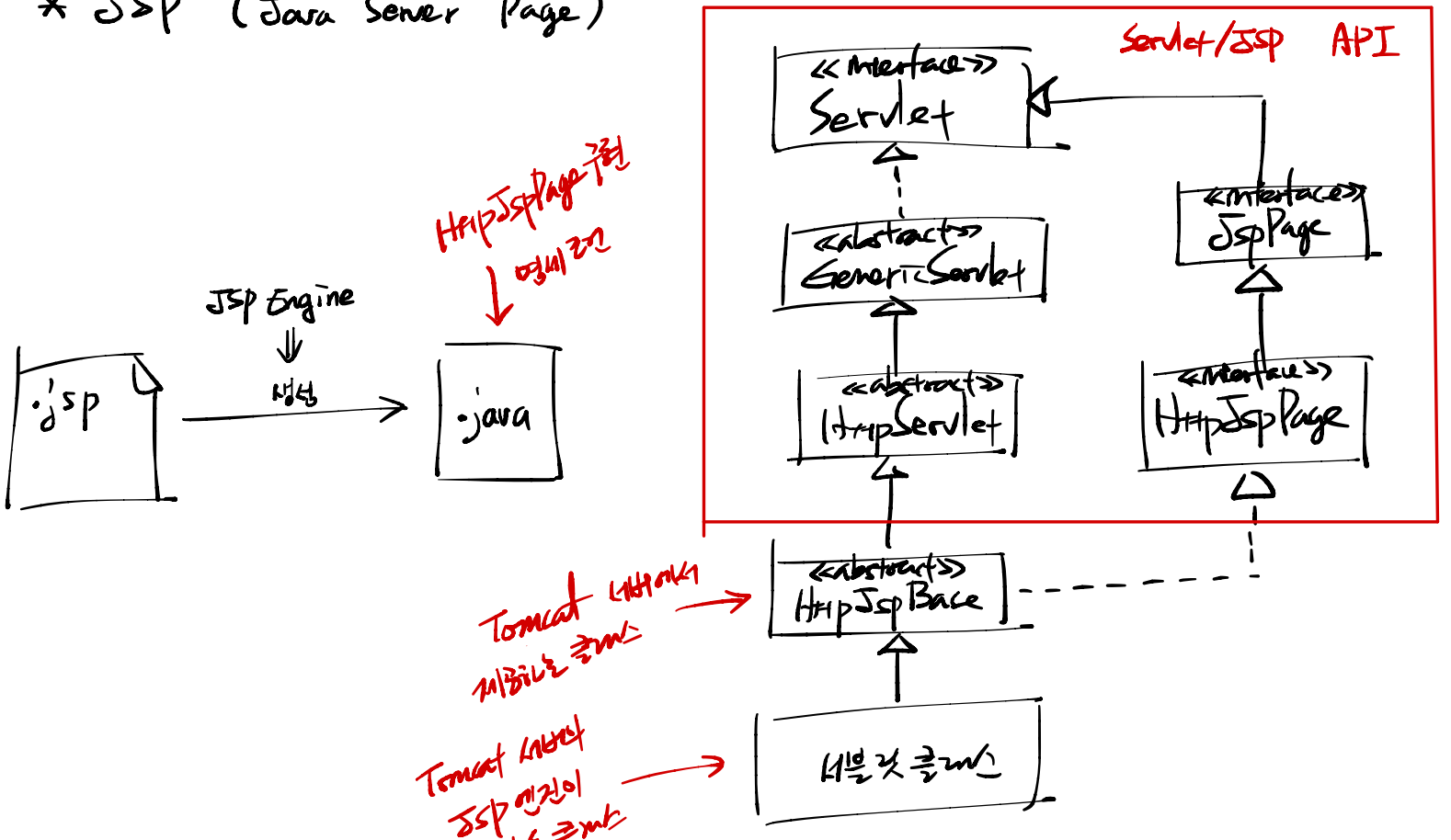
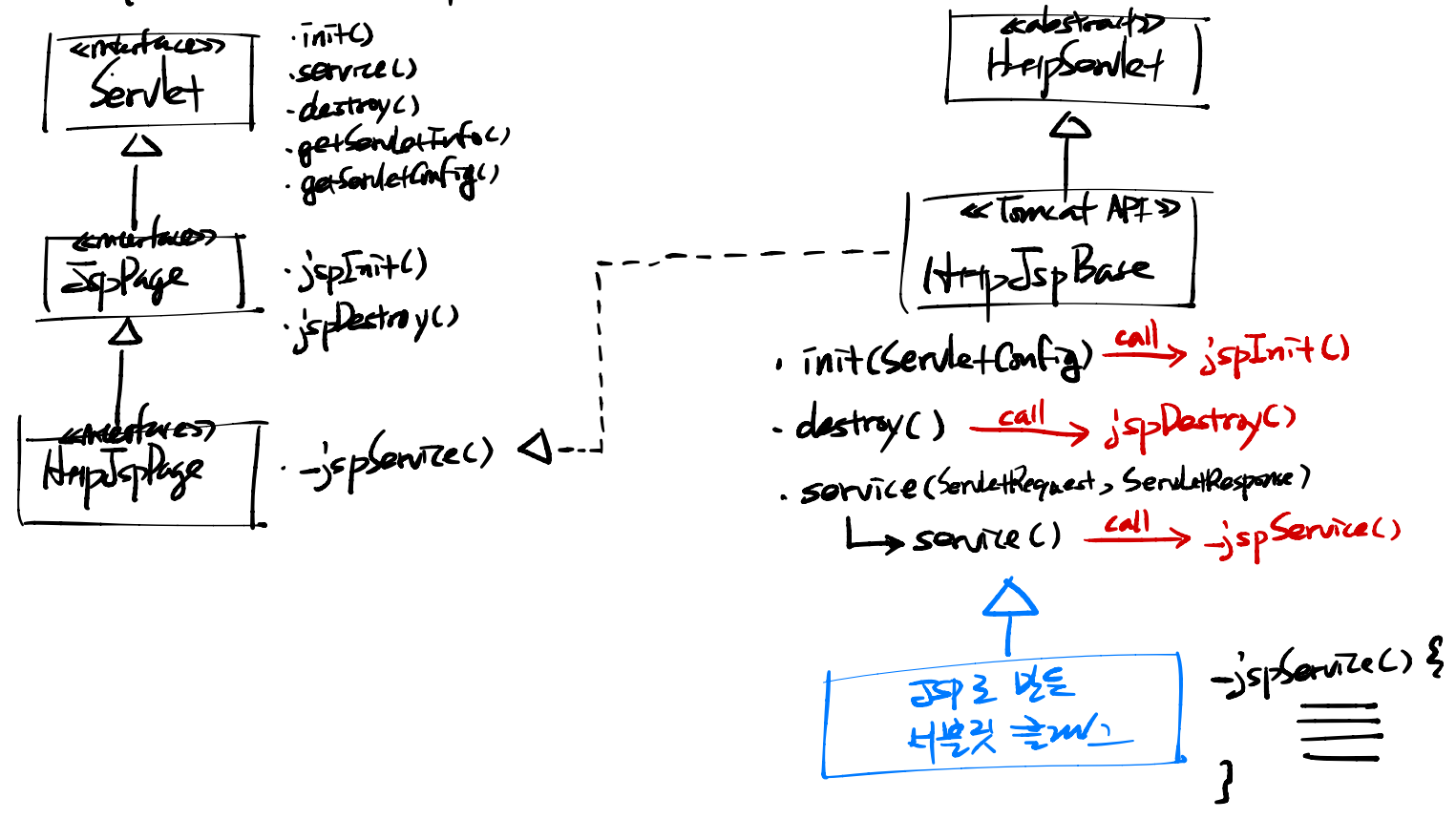
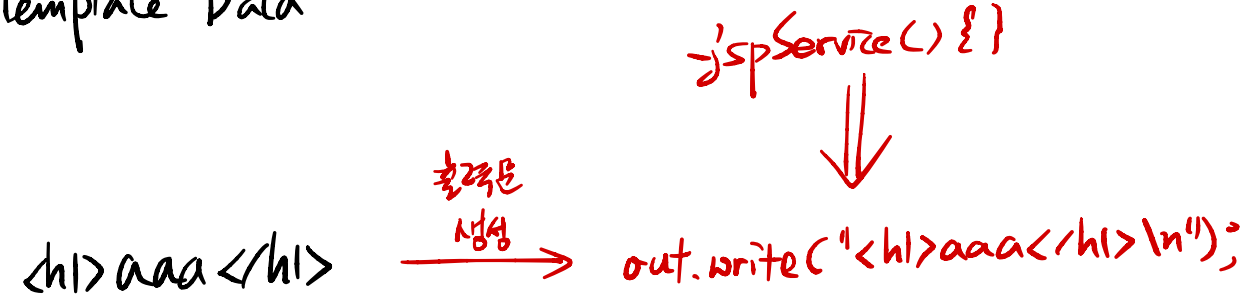
* JSP (Java Server Page)



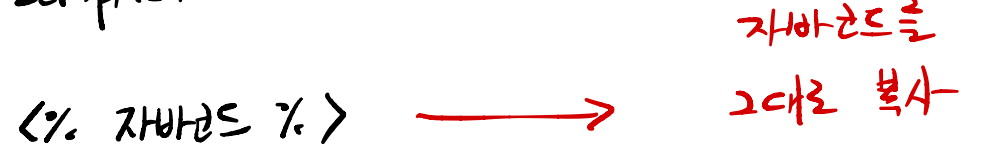
* JSP 만들 때 부모 클래스



* Template Data



* Scriptlet

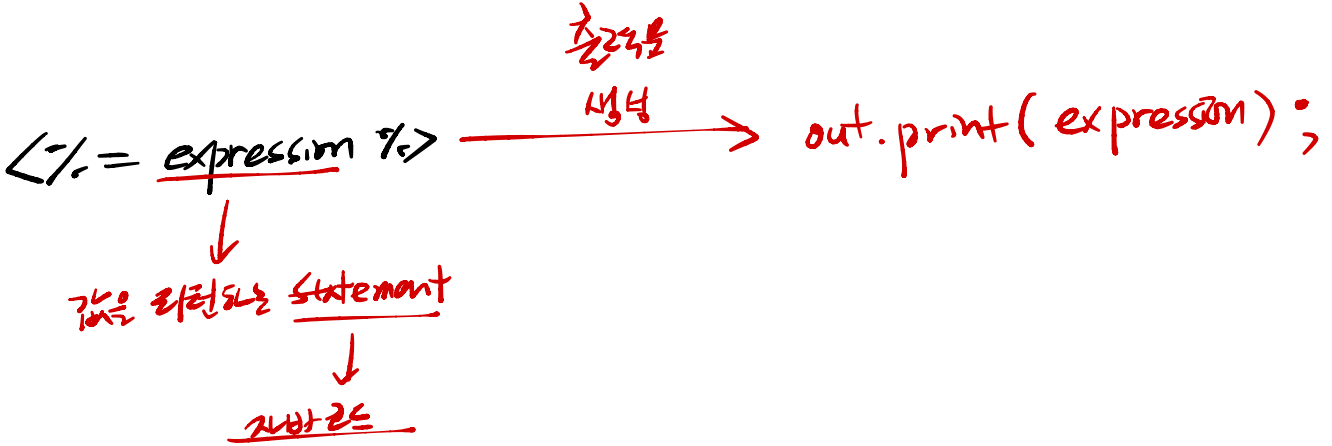


* JSP Built-in 객체

↳ _jspService() 메서드에 선언된 지역변수

HttpServletRequest	request
HttpServletResponse	response
PageContext	pageContext
HttpSession	session
ServletContext	application
ServletConfig	config
JspWriter	out
Object	page
Throwable	exception

* Expression Element (표현식)



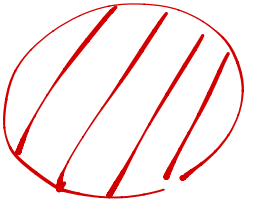
* Declaration Element

<%! 자바코드 %>

- 변수선언
- 메서드 정의
- 인스턴스 블록
- 스태틱 블록

복사

class 서블릿 {



_jspService(){ }

}

* directive element

<%@ page 속성 %>

include

taglib

* whitespace
↳ tab, new line, space

```
<%@ page
    language="java"
    contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"
    import="java.net.Socket"
    import="java.net.ServerSocket"
    import="java.util.List,java.util.Map,java.util.Set"
    trimDirectiveWhitespaces="true"
    buffer="8kb"
    autoFlush="false"%>
<%@ page import="java.sql.Connection"%>
<%@ page import="java.sql.Statement"%>
```

- language="java" ← JSP 안에 작성하는 코드의 프로그래밍 언어를 지정. java만 가능!
- contentType → setContentType() 생성
- pageEncoding ← JSP 파일이 어떤 문자집합으로 인코딩 되었는지 JSP 엔진에게 알려준다
- import → import 문 생성
- trimDirectiveWhitespaces → whitespace에 해당하는 문자를 출력하지 않는다
- buffer → 기본 캐시 용량, 캐시 용량을 초과하면 자동 방출
- autoFlush ← 캐시 용량을 넘어갈 때 오류 발생시키도록 설정

* directive element

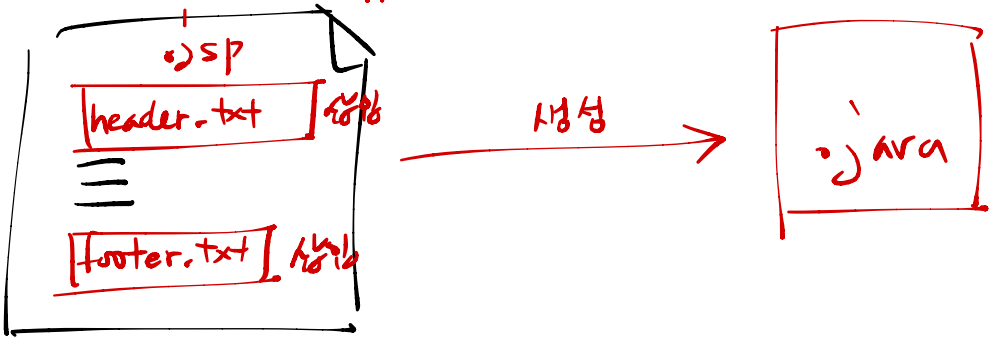
<%@ include file="파일경로" %>

↓

<%@ include file="header.txt" %>

<%@ include file="footer.txt" %>

↓ jsp의 파일의 내용을 합쳐준 크기 자바파일 생성



* directive element

<%@ taglib uri="태그이름" prefix="별명" %>

기본으로 제공하지 않는 태그

외부 JSP 확장 태그

구성요소
- ① .jar
  - 확장 태그를 처리하는 자바클래스 = "Tag Handler"
- ② 확장태그라이브러리 이름 = URI 형식으로 짓는다
  - 예) http://java.sun.com/jsp/jstl/core

사용법
- ① /WEB-INF/lib/*.jar 확장태그 라이브러리 파일 배치
- ② JSP 페이지 taglib element 선언
- ③ 별명을 이용하여 사용

* JSP 액션태그

<jsp:useBean /> ⇐ 객체 생성 태그

prefix = namespace = package

tag name

(객체는 생성한 후 사용할 곳에서 레퍼런스의 타입으로 사용)

page ⇒ JspContext
request ⇒ ServletRequest
session ⇒ HttpSession
application ⇒ ServletContext

```
<jsp:useBean id="b1" class="com.eomcs.web.vo.Board" scope="page"/>
```

```
Board b1 = (Board) pageContext.getAttribute("b1");
If (b1 == null) {
    b1 = new Board();
    pageContext.setAttribute("b1", b1);
}
```

jsp:useBean 으로 생성한 객체

<jsp:setProperty name="b3" property="no" value="100"/>

name → 객체 이름
property → setter
value → parameter

⇓

b3.setNo(100);

객체가 없는 경우 에러 발생시킴

```
<jsp:useBean id="s1" type="java.lang.String" scope="application"/>
```

```
String s1 = (String) application.getAttribute("s1");
if (s1 == null) {
    throw new Exception("____");
}
```

String 객체를 생성하지 않는다

```
<jsp:include page="ex19_header.jsp"/>
```

↳ RequestDispatcher rd = request.getRequestDispatcher("ex19_header.jsp");
rd.include(request, response);

* JSTL (Jsp Standard Tag Library) : JSP에서 사용할 수 있는 표준 확장 태그

구성요소
- JSTL 1.2 API - JSTL 기본 클래스와 인터페이스제공 ⇒ JDBC API 와 같은 역할
- JSTL 구현체 - JSTL API 규격에 맞춰 인터페이스 구현 ⇒ JDBC Driver 와 같은 역할

↓

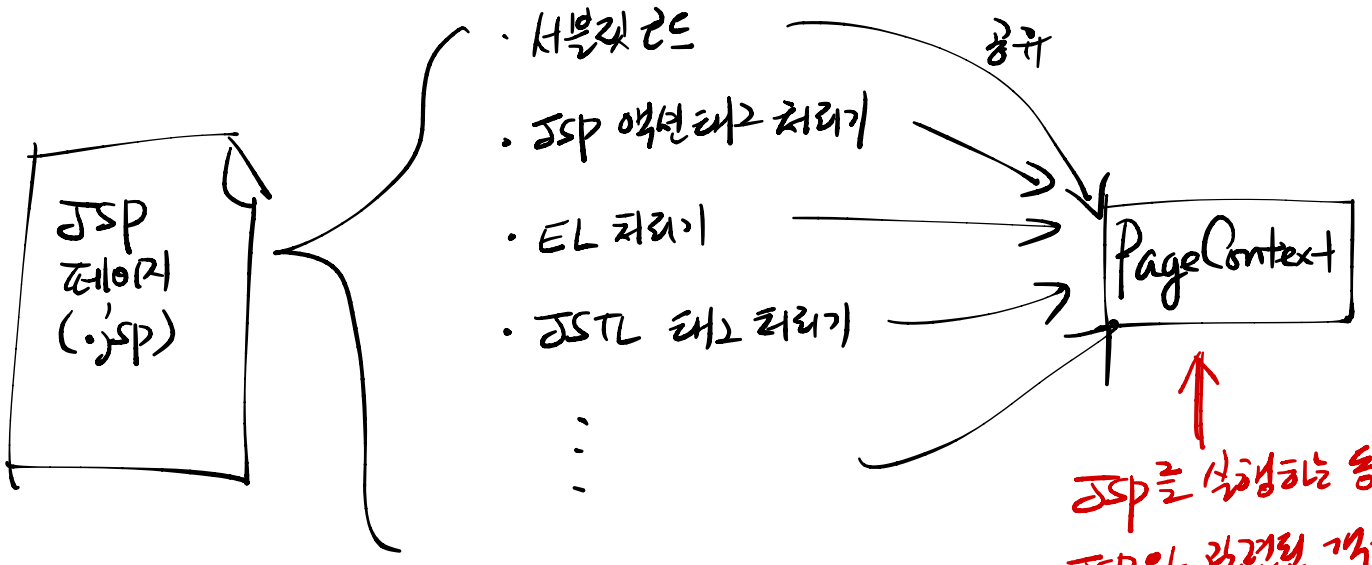
jstl-1.2.jar
- javax.servlet.jsp.jstl.* ⇐ JSTL API
- org.apache.taglibs.standard.* ⇐ JSTL 구현체

↓

Tag Handler — 특정 태그를 자바코드로 변환하는 처리하는 클래스

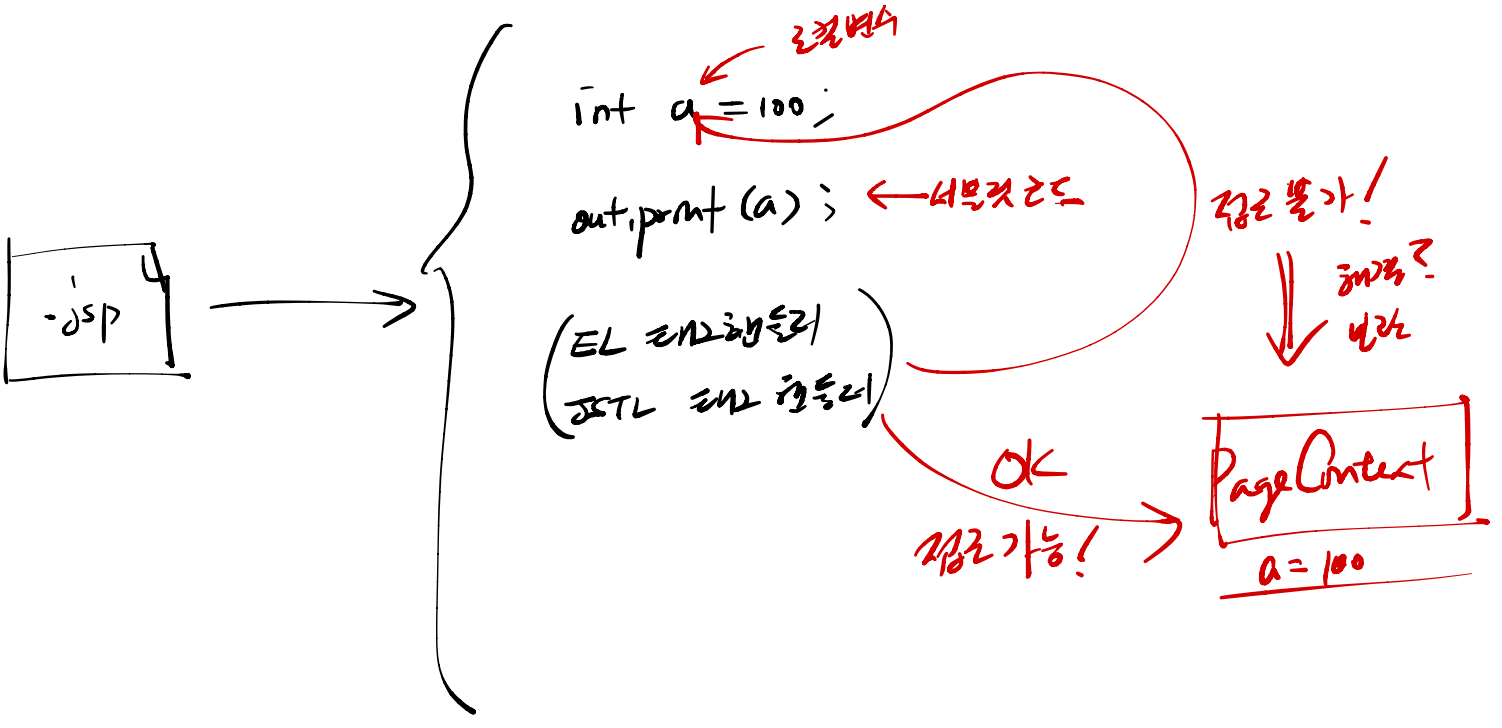
* PageContext (JspContext)

JSP 실행에 관련된 객체들



JSP를 실행하는 동안 JSP와 관련된 객체들이 값을 공유할 때 사용하게 된다.

서블릿 코드


.jsp
int a = 100;
out.print(a); ← 서블릿 코드
(EL 표현해석기
JSTL 표현 해석기)
로컬변수
결국 뭘까!
해석된 변수
OK
결과가능!
PageContext
a = 100

* JSTL 태그 라이브러리 사용

① <%@ taglib uri="태그라이브러리명" prefix="접두어" %> ⇐ Tag 라이브러리 불러오기

uri: namespace = 패키지 (태그라이브러리 불러옴)

prefix: 별명

Core: http://java.sun.com/jsp/jstl/core c

XML: http://java.sun.com/jsp/jstl/xml x

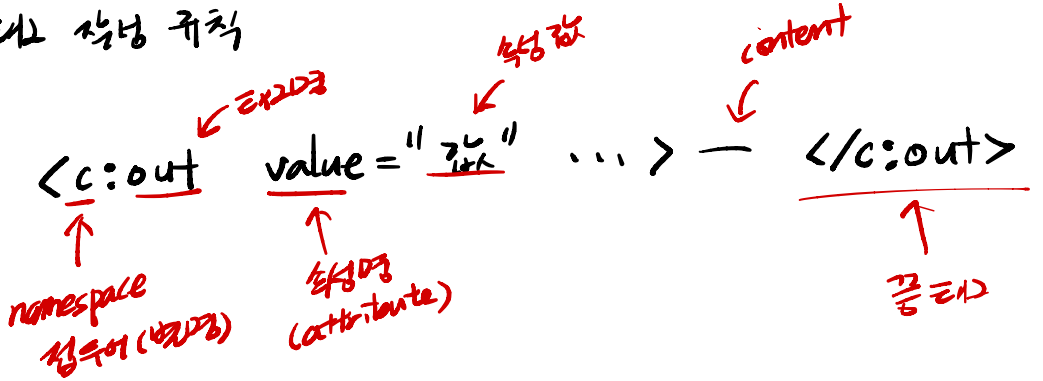
Internationalization: http://java.sun.com/jsp/jstl/fmt fmt

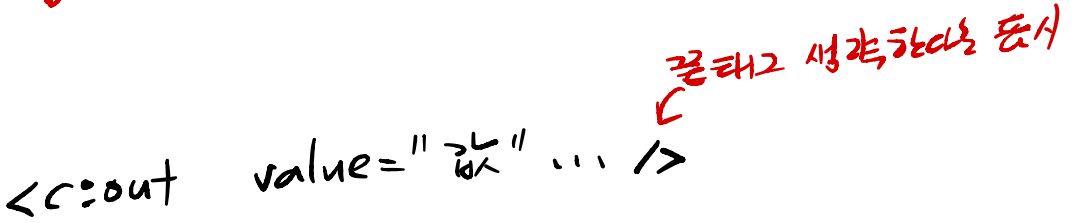
SQL: http://java.sun.com/jsp/jstl/sql sql

Functions: http://java.sun.com/jsp/jstl/functions fn

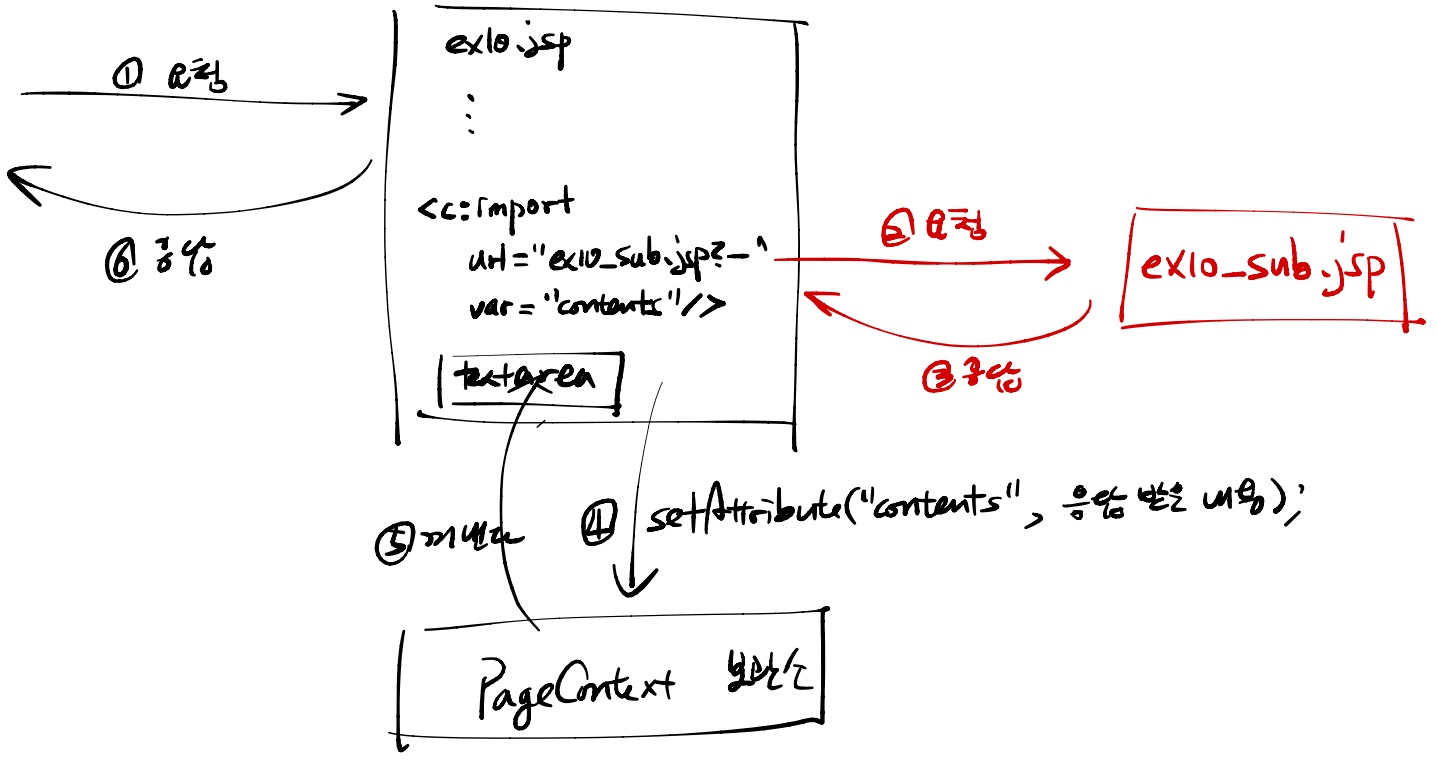
② <접두어:태그명 속성명=값 ...> ... </접두어:태그명> ⇐ Tag 사용

* XML 태그 작성 규칙


<c:out value="값" ... > — </c:out>
태그명
속성값
content
namespace
접두어(별칭명)
속성명
(attribute)
끝태그


끝태그 생략하여도 됨
<c:out value="값" ... />

* c:import


① 요청
⑥ 응답
ex10.jsp
<c:import
url="ex10_sub.jsp?~"
var="contents"/>
textarea
④ 요청
⑤ 응답
ex10_sub.jsp
③ setAttribute("contents", 응답 받은 내용);
PageContext 보관소

* fmt:parseDate 와 fmt:formatDate

```
<fmt:parseDate
        value="2024-1-1"
        pattern="yyyy-MM-dd"
        var="d1" />
```

Date 형식 지정

```
SimpleDateFormat dateFormat =
  new SimpleDateFormat("yyyy-MM-dd");
java.util.Date d1 = dateFormat.parse("2024-1-1");
```

```
pageContext.setAttribute("d1", d1);
```

```
<fmt:formatDate
        value="${날짜객체}"
        pattern="yyyy-MM-dd"/>
```

yyyy-MM-dd 형식의 문자열로 출력

"2024-01-01"