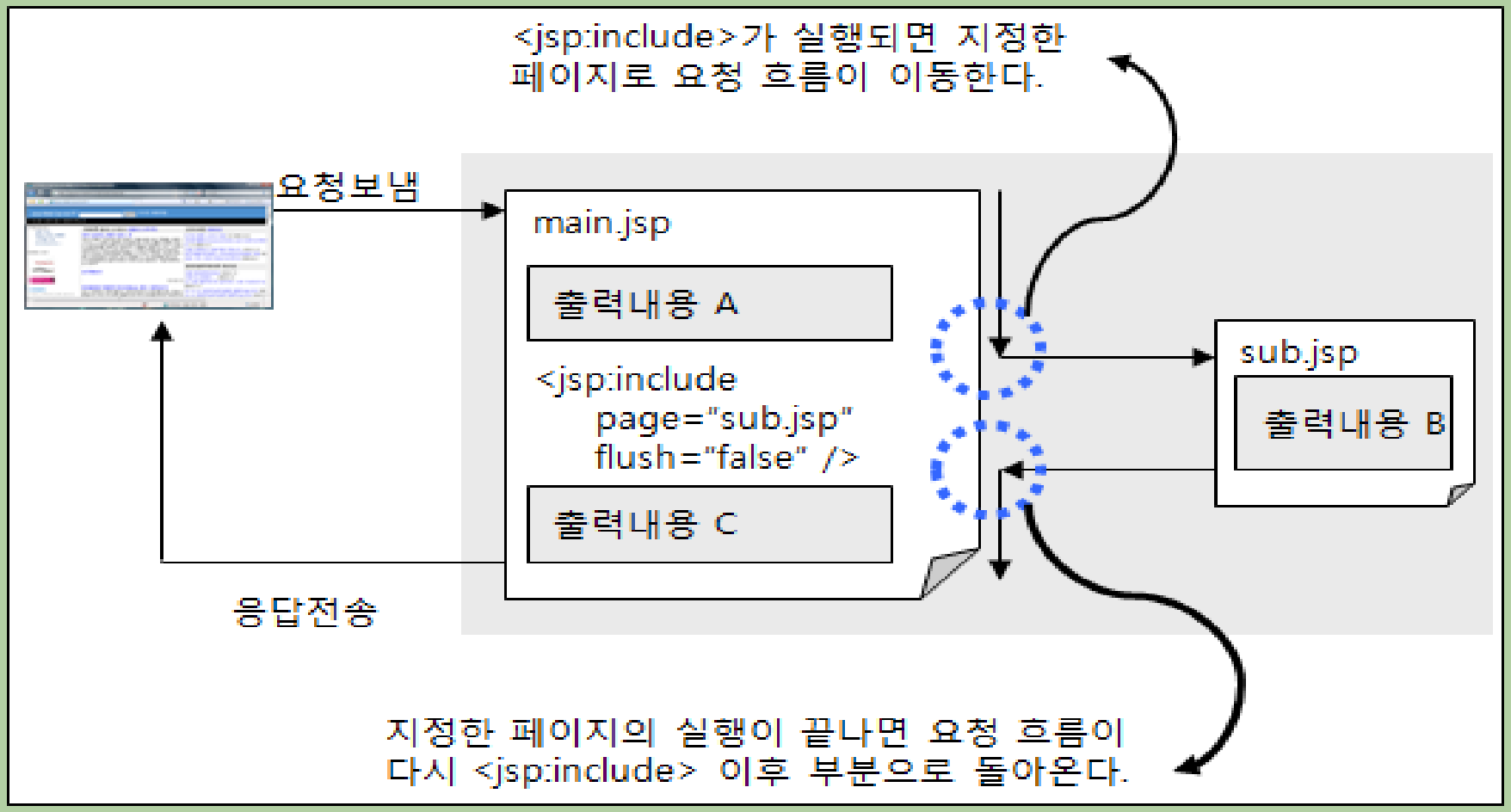
JSP(JavaServer Page)

JSP 액션태그

<jsp:include> 액션 태그를 이용한 페이지 모듈화



JSP 액션태그

<jsp:include> 액션 태그 사용법

▶ <jsp:include page="포함할 페이지“>

<jsp:include> 중복 영역의 처리

공통되는 부분의 수정에 따른 문제를 최소화할 수 있다.
즉, 페이지의 공통되는 부분을 별도의 JSP 페이지로 작성한 후 태그를 사용하여 지정한 위치에 포함해주는 방법을 사용하여 중복 영역을 처리한다.

JSP 액션태그

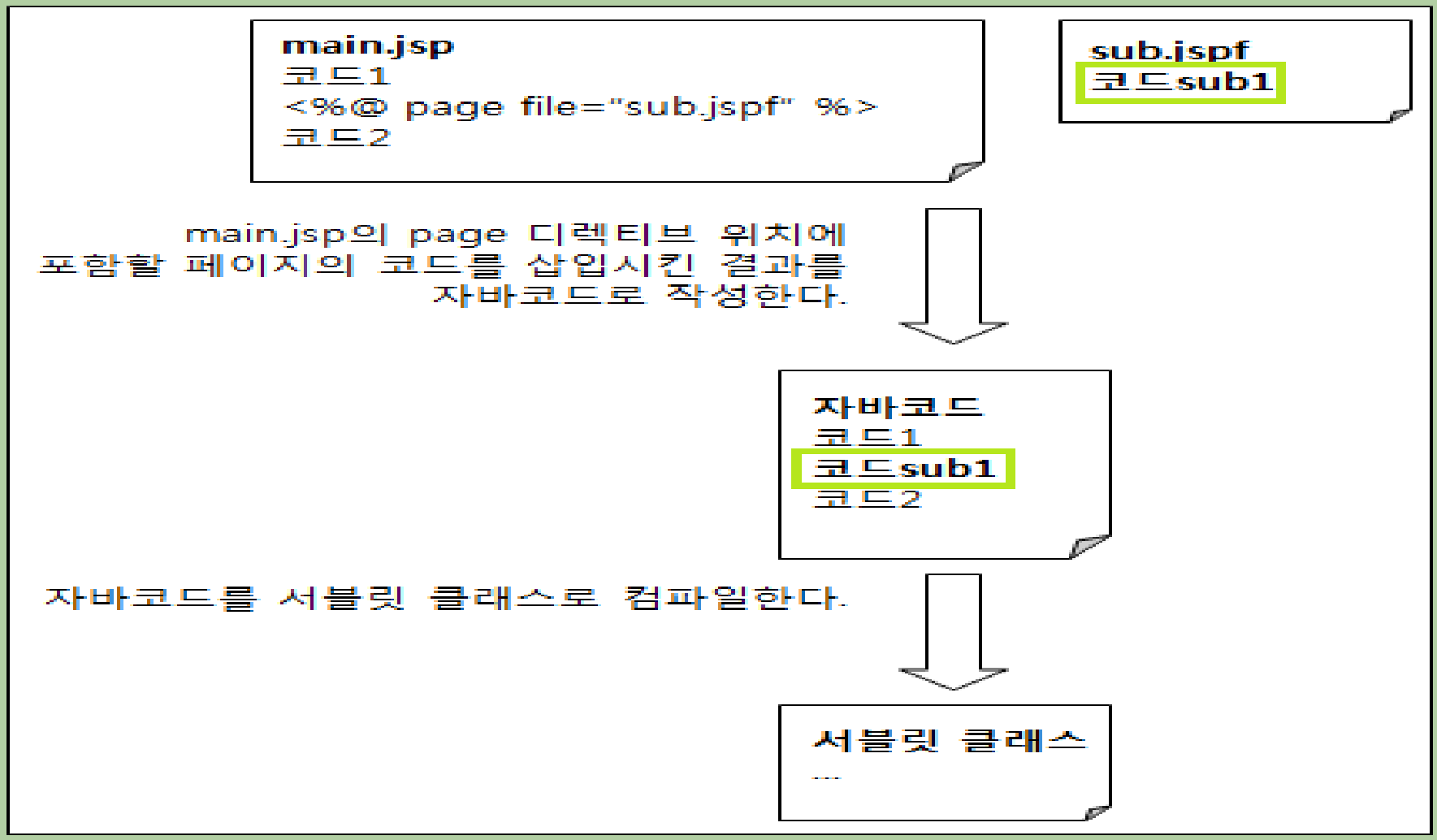
include 디렉티브를 이용한 중복 코드 삽입

<jsp:include> 와 마찬가지로 페이지를 포함해주는 기능을 제공하지만 include 디렉티브는 포함 방식에서 차이를 보인다. include 디렉티브는 다른 파일의 내용을 현재 위치에 삽입한 후에 JSP 파일을 자바 파일로 변환하고 컴파일하는 방식이다.

include 디렉티브의 처리 방식과 사용법

▶ <%@ include file="포함할파일" %>

JSP 액션태그


main.jsp
코드1
<%@ page file="sub.jspf" %>
코드2
sub.jspf
코드sub1
main.jsp의 page 디렉티브 위치에
포함할 페이지의 코드를 삽입시킨 결과를
자바코드로 작성한다.
자바코드
코드1
코드sub1
코드2
자바코드를 서블릿 클래스로 컴파일한다.
서블릿 클래스
...

JSP 액션태그

include 디렉티브의 활용

<jsp:include> 액션 태그는 레이아웃의 한 구성요소를 모듈화하기 위해 사용되는 반면에 include 디렉티브는 다음과 같이 두 가지 형태로 주로 사용된다.

- 모든 JSP 페이지에서 사용되는 변수 지정
- 저작권 표시와 같이 간단하게 모든 페이지에서 중복되는 문장

JSP 액션태그

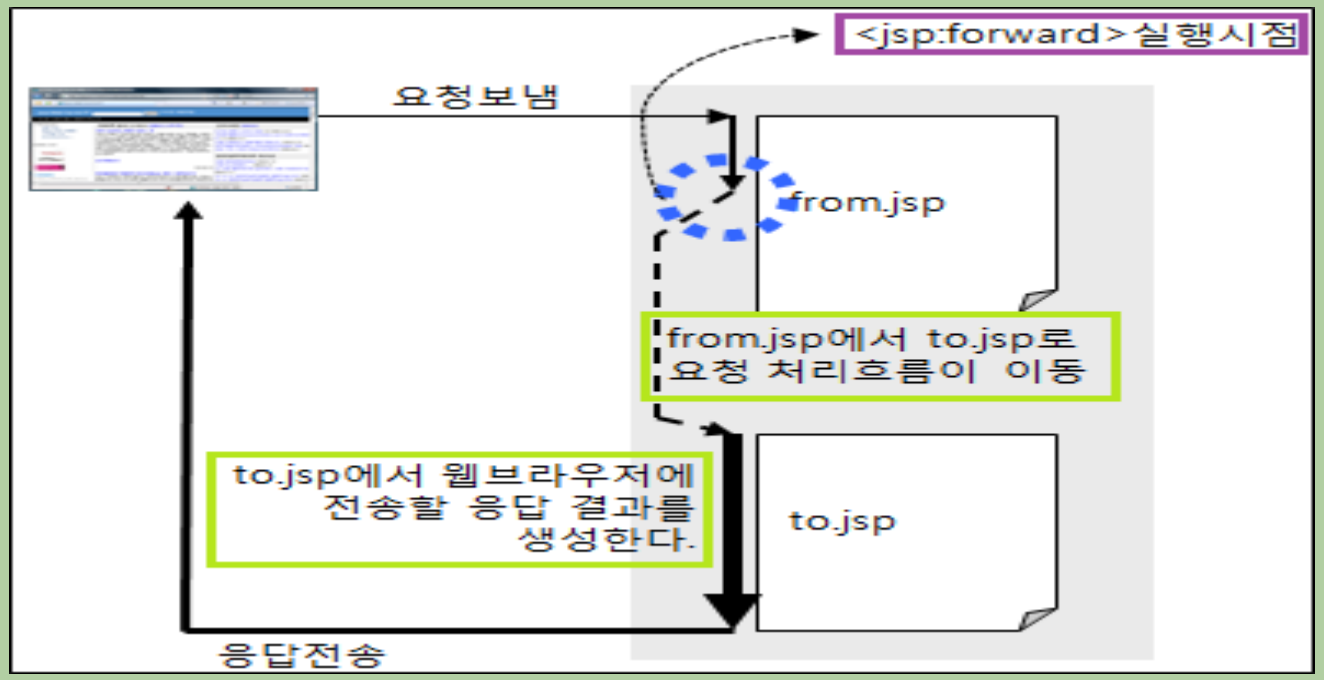
<jsp:include> 액션 태그와 include 디렉티브의 비교

	<jsp:include>	include 디렉티브
처리 시간	요청 시간에 처리	JSP 파일을 자바 소스로 변환할 때 처리
기능	별도의 파일로 요청 처리 흐름을 이동	현재 파일에 삽입
전달 방법	request 기본 객체나 <jsp:param>을 이용하여 파라미터 전달	페이지 내에 변수를 선언한 후 저장
용도	레이아웃의 일부를 모듈화할 때 사용	공통변수나 저작권과 같은 문장을 포함할 때 사용

JSP 액션태그

<jsp:forward> 액션 태그를 이용한 JSP 페이지 이동

하나의 JSP 페이지에서 다른 JSP 페이지로 요청 처리를 전달할 때 사용된다. 다음 그림을 통해 요청의 흐름을 이해해보자.



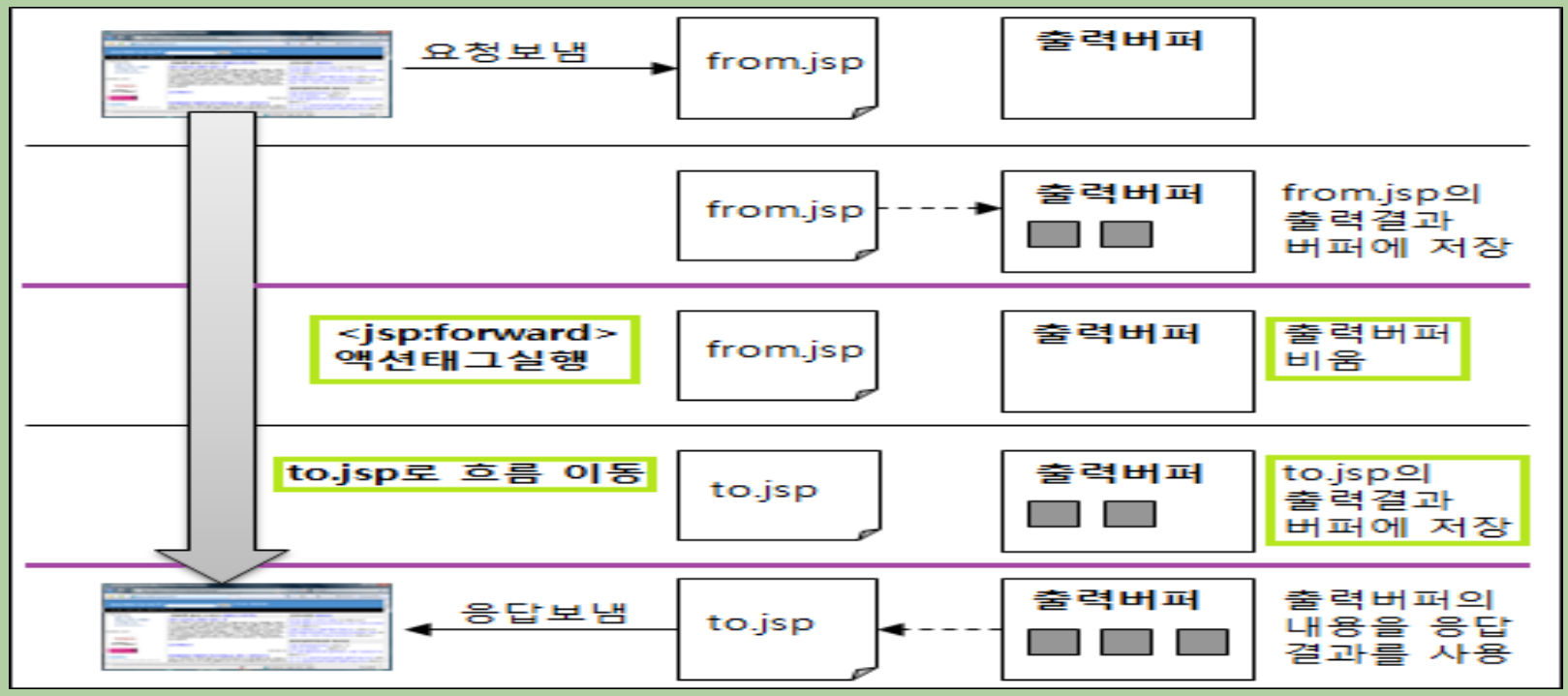
JSP 액션태그

* <jsp:forward> 액션 태그를 사용하는 이유
보다 간결하고 구조적으로 프로그래밍할 수 있기 때문이다.
각각의 조건을 처리하는 JSP를 분리시켜 기능별로 모듈화할 수 있다.

<jsp:forward> 액션 태그의 사용법
▶ <jsp:forward page="이동할 페이지"/>

JSP 액션태그

<jsp:forward> 액션 태그와 출력 버퍼와의 관계
<jsp:forward> 태그가 실행되면 다음 그림과 같이 출력 버퍼의 내용이 사라지고 이동한 페이지의 출력 결과가 출력 버퍼에 저장된다.



에러페이지 지정하기

JSP에서는 예외가 발생한 경우 에러 화면 대신 지정한 JSP 페이지를 보여줄 수 있는 기능을 제공한다.

page 디렉티브의 errorPage 속성을 이용하여 지정해주면 된다. 기본 문법은 다음과 같다.

▶ <%@ page errorPage = "이동할 에러페이지" %>

에러페이지 지정하기

응답 상태 코드별로 에러페이지 지정하기

톰캣 서버에서 제공하는 에러페이지를 보면 404나 500 처럼 에러코드를 명시해준다. 이렇나 에러코드를 web.xml 에 에러페이지와 함께 지정해 줄 수 있다.

다음과 같이 태그를 추가해주면 된다.

```
<error-page>
<error-code>에러코드</error-code>
<location>에러페이지의 URI</location>
</error-page>
```

에러페이지 지정하기

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xmlns="http://java.sun.com/xml/ns/javaee"
4      xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
6      id="WebApp_ID" version="2.5">
7
8      <error-page>
9          <error-code>404</error-code>
10         <location>/error/404error.jsp</location>
11     </error-page>
12
13 </web-app>
```

에러페이지 지정하기

200	요청이 정상적으로 처리됨.
307	임시로 페이지가 리다이렉트됨.
400	클라이언트의 요청이 잘못된 구문으로 구성됨.
401	접근이 허용되지 않음.
404	지정된 URL을 처리하기 위한 자원이 존재하지 않음. (페이지가 없음)
405	요청된 메서드가 허용되지 않음
500	서버 내부 에러. (JSP에서 예외가 발생하는 경우)
503	서버가 일시적으로 서비스를 제공할 수 없음. (서버 과부하이거나 보수중인 경우)

에러페이지 지정하기

예외 타입별로 에러 페이지 지정하기
예외 종류에 따라서 에러 페이지를 지정하는 것도 에러 코드별 에러 페이지를 지정했던 것과 거의 동일한 방법이다.
우선 기본 문법을 보자.

```
<error-page>
<exception-type>예외타입</exception-type>
<location>예외페이지의 URI</location>
</error-page>
```

에러페이지 지정하기

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
    id="WebApp_ID" version="2.5">

    <error-page>
        <exception-type>java.lang.NullPointerException</exception-type>
        <location>/error/errorNullPointer.jsp</location>
    </error-page>

</web-app>
```

```jsp
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>
<%@ page errorPage="/error/errorNullPointer.jsp" %>

<html>
    <head>
        <title>ExceptionTest</title>
    </head>

    <body>
        <%=request.getParameter("name").toUpperCase() %>
    </body>
</html>
```

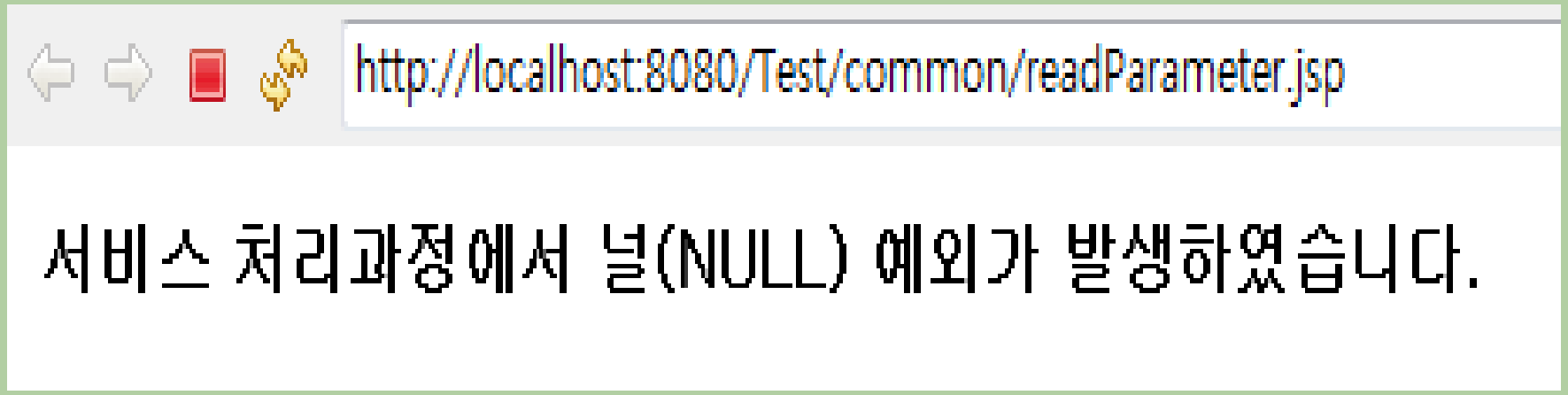
에러페이지 지정하기

```jsp
<%@ page language="java" contentType="text/html; charset=EUC-KR"
    pageEncoding="EUC-KR"%>

<html>
    <head>
        <title>ExceptionTest</title>
    </head>

    <body>
        <strong>서비스 처리과정에서 널(NULL) 예외가 발생하였습니다.</strong>
    </body>
</html>

```



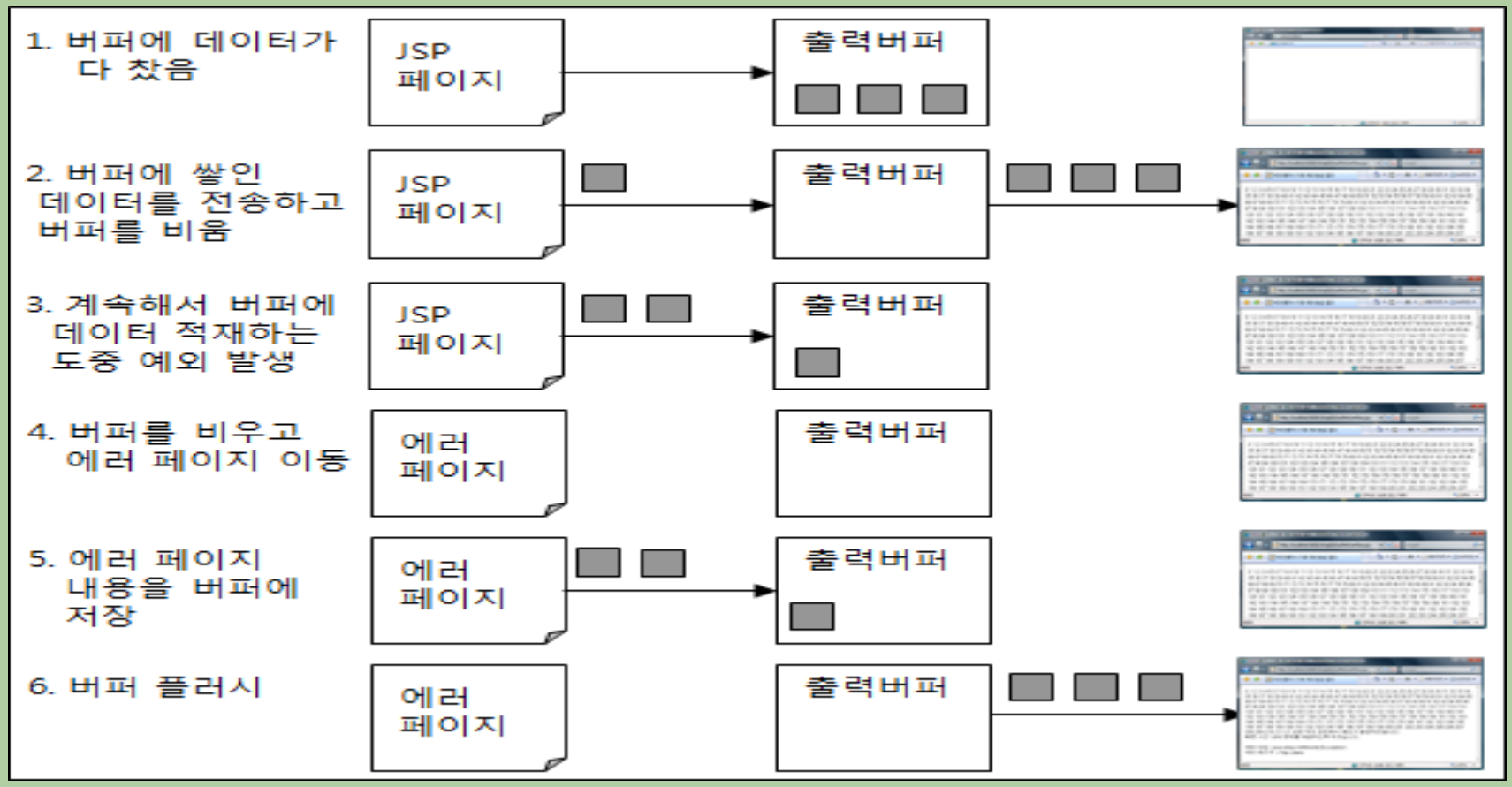
에러페이지 지정하기

에러 페이지의 우선순위 및 에러페이지 지정 형태
다음과 같은 우선순위에따라 사용할 에러페이지를 선택하면 된다.

① page 디렉티브의 errorPage 속성에서 지정한 에러페이지를 보여준다.
② JSP 페이지에서 발생한 예외 타입이 web.xml 에서 지정한 예외 타입과 동일한 경우 지정한 페이지를 출력한다.
③ JSP 페이지에서 발생한 에러 코드가 web.xml 에서 지정한 에러 코드와 동일한 경우 지정한 페이지를 출력한다.
④ 아무것도 해당되지 않을 경우 웹 컨테이너가 제공하는 기본 에러페이지를 출력한다.

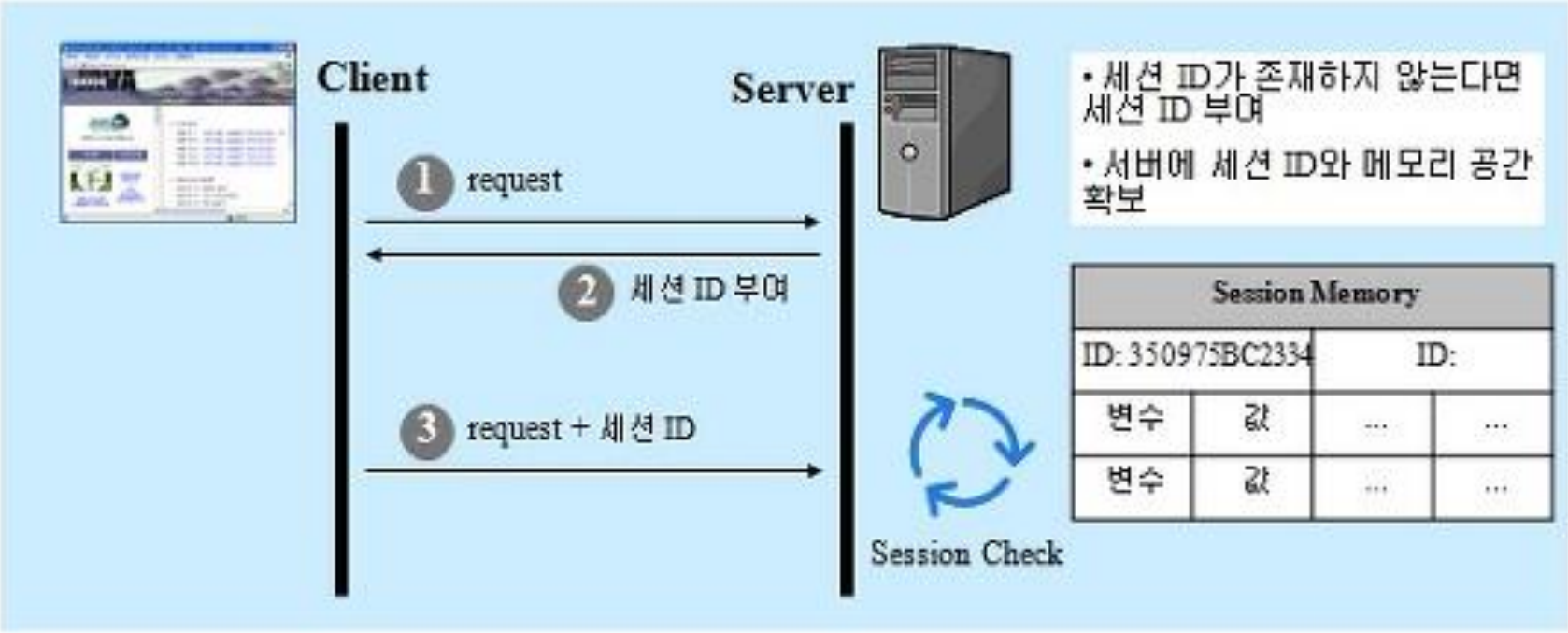
에러페이지 지정하기

출력 버퍼와 에러 페이지의 관계



JSP 세션

- 세션(Session)
  - 클라이언트와 서버간의 상태를 유지하기 위한 방법
  - 클라이언트가 처음 접속했을 때 세션 ID를 부여하고, 서버에 세션 ID를 저장
  - 다시 클라이언트가 접속했을 때 부여된 세션 ID를 이용해서 클라이언트를 구분



JSP 세션

쿠키와 세션을 공통점

- 클라이언트의 상태 유지를 위한 기능 제공

쿠키와 세션의 차이점

구분	쿠키	세션
저장되는 곳	클라이언트	서버
저장되는 형식	텍스트	Object
만료시점	쿠키 저장 시 설정 가능 (설정하지 않을 경우 브라우저 종료 시 소멸)	클라이언트가 로그아웃 하거나 설정한 시간 동안 반응이 없을 경우
리소스	클라이언트의 리소스 사용	서버의 리소스 사용
용량 제한	한 도메인 당 20개 쿠키 하나당 4KB 총 300개	서버가 허용하는 용량

JSP 세션

▣ JSP 페이지에서 세션 사용하기

◈ page 지시문을 이용해서 설정

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

◈ session 속성의 기본값이 true이기 때문에 생략해도 된다.

▣ session 객체 생성하기

◈ request 내장 객체로부터 getSession() 메소드를 이용해서 생성

```
HttpSession session = request.getSession(true);
```

◈ session 내장 객체를 사용해도 동일하다.

JSP 세션

- 새로운 세션 여부 출력

```
<%=session.isNew()%>
```

- 세션 아이디 출력

```
<%=session.getId()%>
```

- 세션 생성시간 출력

```
<%=new java.util.Date(session.getCreationTime()).toString()%>
```

- 마지막 접속시간 출력

```
<%=new java.util.Date(session.getLastAccessedTime()).toString()%>
```

- 세션 Active 시간

```
<%=session.getMaxInactiveInterval()%>
```

JSP 세션

▣ 세션에 데이터 저장

◈ void setAttribute(java.lang.String name, java.lang.Object value)

```
session.setAttribute( "userId",  "Grace" );
```

"userId": 세션 데이터의 변수 이름
"Grace": 세션 데이터의 값

▣ 세션으로부터 데이터 출력

◈ java.lang.Object getAttribute(java.lang.String name)

```
(String)session.getAttribute( "userId" );
```

세션 데이터의 변수 이름

getAttribute() 메소드의 리턴 형이 Object이기 때문에 캐스팅을 해서 사용해야 한다.

▣ 세션으로부터 데이터 삭제

◈ void removeAttribute(java.lang.String name)

```
session.removeAttribute( "userId" );
```

세션 데이터의 변수 이름

JSP 세션

▣ 세션의 생명주기 설정

◈ void setMaxInactiveInterval(int interval)

```
session.setMaxInactiveInterval(60 * 10);
```

세션의 생명주기를 10분으로 설정

◈ 마지막으로 세션을 사용한 후 10분이 지나면 세션 정보가 삭제된다.

▣ 세션 종료

◈ void invalidate()

```
session.invalidate()
```