3. JSP 기본 다루기

1. JSP 페이지 구성요소
2. JSP 페이지 내장 객체
3. JSP 페이지 액션 태그
4. JSP 페이지 에러 처리

1. JSP 페이지 구성 요소

- JSP 페이지 디렉티브
- JSP 페이지 스크립트 요소
- JSP 제어문
- 톰캣 기반에서 JSP 페이지의 한글 처리

1) JSP 페이지의 디렉티브

- 클라이언트가 요청한 JSP 페이지가 실행될 때, 필요한 정보를 지정하는 역할
- 필요한 정보를 JSP 컨테이너에게 알려서 어떻게 처리되도록 하는 지시자
- 디렉티브는 태그 안에서 @로 시작하며 page, include, taglib 등의 3가지 종류가 있음

1) JSP 페이지의 디렉티브

- **page 디렉티브-<@page>**
  - JSP페이지에 대한 속성 설정
  - 서버에 요청한 결과를 응답 받을 때 생성되는 페이지의 타입, 스크립트 언어, import 할 클래스, 세션 및 버퍼의 사용 여부 및 버퍼의 크기 등의 JSP 페이지에서 필요한 설정 정보를 지정
  - info 속성-페이지를 설명하는 문자열(생략가능)
    - <%@page info="copyright by KIM" %>
  - language 속성-스크립트 요소에서 사용할 언어 지정(생략 가능)
    - <%@page language="java" %>
  - contentType 속성-생성할 문서의 타입을 지정
    - <%@ page contentType ="text/html" %>

1) JSP 페이지의 디렉티브

- **page 디렉티브-<@page>**
  - extends 속성-상속받을 클래스 지정, 거의 사용 안 함
  - import 속성-다른 위치에 있는 클래스 참조(중복 사용 가능)
    - <%@page import= "java.util.*, java.sql.Timestamp" %>
  - session 속성-생성할 문서의 타입을 지정
    - <%@ page session="true"%><-생략 시 기본값 true
  - buffer 속성-출력 버퍼의 크기 지정
    - <%@ page buffer="8kb"%> <-생략시, 기본값8kb
  - autoFlush 속성-출력 버퍼가 다 찰 경우 처리 방법을 지정
    - <%@page autoflush="true"%><-생략 시 기본 값 true
  - isThreadSafe 속성-다중 쓰레드 사용
    - <%@ page session="true"%><-생략 시 기본값 true

1) JSP 페이지의 디렉티브

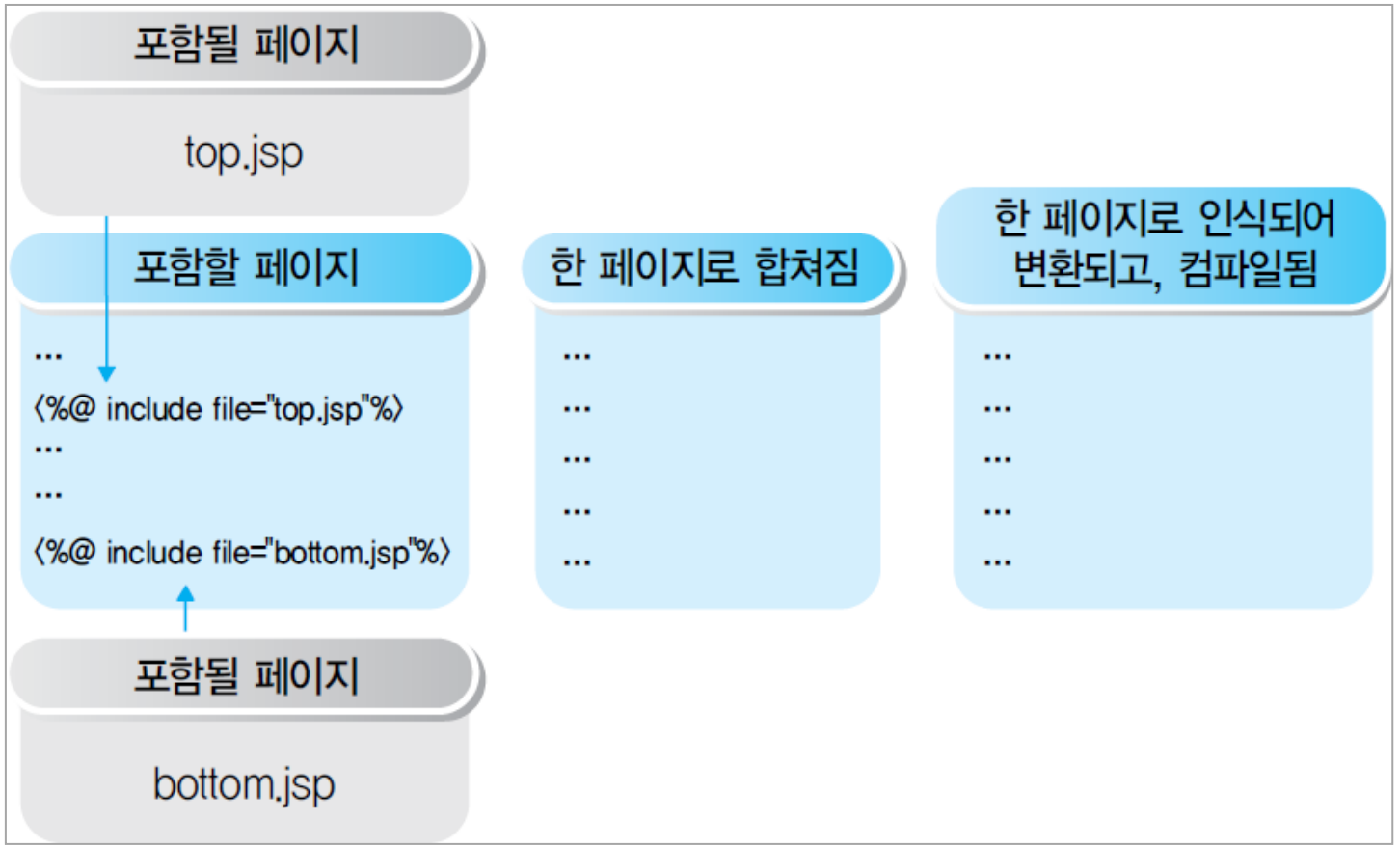
- **page 디렉티브-<@page>**
  - errorPage 속성- 에러 발생 시 처리할 페이지 지정(JSP2.0부터 지원 안 함)
  - isErrorPage 속성-에러를 처리하는 페이지 지정(JSP2.0부터 지원 안 함)
  - pageEncoding 속성-문자의 인코딩을 지정
    - <%@ page pageEncoding="UTF-8"%>

1) JSP 페이지의 디렉티브

- **include 디렉티브-<@include>**
  - 공통적으로 포함될 내용을 가진 파일을 해당 JSP 페이지 내에 삽입하는 기능을 제공
  - <%@ include file="포함될 파일의 url"%>
  - 포함시킬 파일명을 file 속성의 값으로 기술
  - 복사 & 붙여넣기 방식으로 두개의 파일이 하나의 파일로 합쳐진 후 하나의 파일로서 변환되고 컴파일

1) JSP 페이지의 디렉티브

include 디렉티브-<@include>



▲ include 디렉티브의 처리 과정

1) JSP 페이지의 디렉티브

- **taglib 디렉티브-<@taglib>**
  - 표현 언어(EL : Expression Language), JSTL (JSP Standard Tag Library), 커스텀 태그(Custom Tag)를 JSP 페이지 내에 사용할 때 씀
  - <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
  - uri : 태그의 설정 정보
  - prefix : uri 대신 사용되는 네임스페이스

2) JSP 페이지의 스크립트 요소

스크립트 요소의 이해

- JSP 페이지의 스크립트 요소
  선언문(Declaration), 스크립트릿(Scriptlet), 표현식(Expression)

선언문(Declaration)-<%! %>

- 전역 변수 선언 및 메소드 선언

- 선언문에서 변수 선언

```
<%!
  private String id="Kingdora";
%>
```

- 선언문에서 메소드 선언

```
<%!
  public String getId( ) {
    return id;
  }
%>
```

2) JSP 페이지의 스크립트 요소

- **스크립트릿(Scriptlet)-<% %>**
  - 프로그래밍 코드 기술

```
<%
  member.setReg_date(new Timestamp(System.currentTimeMillis()));
  LogonDBBean manager=LogonDBBean.getInstance();
  manager.insertMember(member);
%>
```

- **표현식(Expression)-<%= %>**
  - 화면에 출력할 내용 기술
  - <%=name[i]%>

2) JSP 페이지의 스크립트 요소

- **주석(Comment)-<!-- -->, <%-- --%>, //, /* */**
  - HTML 주석 : <!-- -->
    - 화면에 표시되지 않으나, 실행됨
  - JSP 주석 : <%-- --%>
    - 화면에 표시되지도 실행되지도 않음
  - 자바 주석 : //, /* */
    - 화면에 표시되지도 실행되지도 않음

3) JSP의 제어문

- HTJSP의 제어문에는 조건 비교 분기문(if, switch), 조건 비교 반복문(for, while, do-while)의 두 가지 형태가 있음
- if, for, while문이 가장 많이 사용

4) 톰캣 기반에서 JSP 페이지의 한글 처리

- **응답처리**
  - 서버에서 웹 브라우저에 응답되는 페이지의 화면 출력 시 한글 처리
  - <%@ page contentType="text/html;charset=utf-8"%>
- **요구처리**
  - 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우 (Post 방식) 한글 처리
  - <% request.setCharacterEncoding("utf-8");%>

4) 톰캣 기반에서 JSP 페이지의 한글 처리

- 웹 브라우저에서 서버로 넘어오는 파라미터 값에 한글이 있는 경우 (Get 방식) 한글 처리
- server.xml파일에 한글 인코딩 지정

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443"
URIEncoding="EUC-KR" />
```

- 가상 환경
  - [Project Explorer] 뷰의 [Servers]-[Tomcat v7.0 Server~] 항목에 있는 server.xml 파일에 한글 인코딩 지정
- 실제 환경
  - 톰캣홈₩conf 폴더에 있는 server.xml 파일에 한글 인코딩 지정

2. JSP 페이지와 내장 객체와 영역

- **내장객체 개요**
- **내장 객체의 종류**
  - request
  - response
  - out
  - pageContext
  - session
  - application
  - config
  - page
  - exception
- **내장 객체의 영역**

1) 내장 객체의 개요

- **JSP에서 제공하는 객체**
  - request, response, out, session, application, pageContext, page, config, exception
  - request, session, application, pageContext 내장 객체는 속성(attribute)값을 저장하고 읽을 수 있는 메소드인 setAttribute() 메소드와 getAttribute() 메소드를 제공

2) 내장 객체의 종류

- **request 객체**
  - 웹 브라우저의 요청 정보를 저장하고 있는 객체
  - 입력폼에 입력한 사용자의 요구 사항을 얻어낼 수 있도록 요청 메소드를 제공
  - 사용자의 요구 사항을 얻어내는 요청 메소드
  - String getParameter(name) : 파라미터 변수 name에 저장된 변수 값을 얻어내는 메소드
    - String[] getParameterValues(name) : 파라미터 변수 name에 저장된 모든 변수 값을 얻어내는 메소드
    - Enumeration getParameterNames() : 요청에 의해 넘어오는 모든 파라미터 변수를 java.util.Enumeration 타입으로 리턴

2) 내장 객체의 종류

- **request 객체**
  - 요청된 파라미터의 값 외에도 웹 브라우저와 웹 서버의 정보도 가져올 수 있음
  - 웹 브라우저, 웹 서버 및 요청 헤더의 정보를 가져올 때 사용되는 메소드
  - String getProtocol() : 웹 서버로 요청 시, 사용 중인 프로토콜 리턴
    - String getServerName() : 웹 서버로 요청 시, 서버의 도메인 이름을 리턴

2) 내장 객체의 종류

- **request 객체**
  - 사용자의 요구 사항을 얻어내는 요청 메소드
    - String getMethod() : 웹 서버로 요청 시, 요청에 사용된 요청 방식(GET, POST, PUT등)을 리턴
    - String getQueryString() : 웹 서버로 요청 시, 요청에 사용된 QueryString을 리턴
    - String getRequestURI() :웹 서버로 요청 시, 요청에 사용된 URL로부터 URI값을 리턴
    - String getRemoteHost() : 웹 서버로 정보를 요청한 웹 브라우저의 호스트 이름을 리턴
    - String getRemoteAddr() : 웹 서버로 정보를 요청한 웹 브라우저의 IP 주소를 리턴

2) 내장 객체의 종류

- **request 객체**
  - 사용자의 요구 사항을 얻어내는 요청 메소드
    - String getServerPort() : 서버의 Port 번호를 리턴
    - String getContextPath() : 해당 JSP 페이지가 속한 웹 애플리케이션의 컨텍스트 경로를 리턴
    - String getHeader(name) : HTTP 요청 헤더(header) 헤더 이름 name에 해당하는 속성 값을 리턴
    - Enumeration getHeaderNames() : 웹 서버로 요청 시, HTTP 요청 헤더(header)에 있는 모든 헤더 이름을 리턴

2) 내장 객체의 종류

- **response 객체**
  - 웹 브라우저의 요청에 대한 응답 정보를 저장하고 있는 객체
  - 응답 정보와 관련하여 주로 헤더 정보 입력, 리다이렉트 등의 기능을 제공
  - response 객체의 메소드
    - void setHeader(name, value) : 헤더 정보 값을 수정
    - void setContentType(type) : 웹 브라우저의 요청 결과로 보일 페이지의 contentType을 설정
    - void sendRedirect(url) : 페이지를 이동

2) 내장 객체의 종류

- **out 객체**
  - JSP 페이지의 출력할 내용을 가지도 있는 출력 스트림 객체
  - 표현식(<%=문장%>) 과 같음
  - out 객체의 메소드
    - boolean isAutoFlush() : 출력 버퍼가 다 찼을 때 처리 여부를 결정
    - int getBufferSize() : 전체 출력 버퍼의 크기를 리턴
    - int getRemaining() : 현재 남아 있는 출력 버퍼의 크기 리턴
    - void clearBuffer() : 출력 버퍼에 저장되어 있는 내용을 비움
    - String println(str) : 주어진 내용을 출력. 이때 줄 바꿈은 적용되지 않음
    - void flush() : 출력 버퍼의 내용을 웹 브라우저에 전송하고 비움
    - void close() : 출력 버퍼의 내용을 웹 브라우저에 전송하고 출력 스트림을 닫음

2) 내장 객체의 종류

- **pageContext 객체**
  - JSP 페이지 대한 정보를 저장하고 있는 객체
  - 다른 내장 객체를 구하거나, 페이지의 흐름제어 그리고 에러 데이터를 얻어낼 때 사용
  - pageContext 내장 객체의 메소드
    - ServletRequest getRequest() : request 객체를 얻어냄
    - ServletResponse getResponse() : response 객체를 얻어냄
    - JspWriter getOut() : out 객체를 얻어냄
    - HttpSession getSession() : session 객체를 얻어냄
    - ServletContext getServletContext() : application 객체를 얻어냄
    - Object getPage() : page 객체를 얻어냄
    - ServletConfig getServletConfig() : config 객체를 얻어냄
    - Exception getException() : exception 객체를 얻어냄

2) 내장 객체의 종류

- **session 객체**
  - 하나의 웹 브라우저 내에서 정보를 유지하기 위한 세션 정보를 저장하고 있는 객체
  - 웹 브라우저(클라이언트)당 1개가 할당
    - 주로 회원 관리 시스템에서 사용자 인증에 관련된 작업을 수행할 때 사용
  - Session 내장 객체의 메소드
    - String getId() : 해당 웹 브라우저에 대한 고유한 세션 ID를 리턴
    - long getCreationTime() : 해당 세션이 생성된 시간을 리턴
    - long getLastAccessedTime() : 마지막 접근시간을 리턴
    - Strin void setMaxInactiveInterval(time) : 세션 유지 시간을 지정
    - int getMaxInactiveInterval() : 기본 값은 30분으로 세션 유지 시간을 리턴
    - boolean isNew() : 새로 생성된 세션의 경우 true 값을 리턴
    - void invalidate() : 세션을 무효화

2) 내장 객체의 종류

- **application 객체**
  - 웹 애플리케이션 Context의 정보를 저장하고 있는 객체
  - 서버의 설정 정보, 자원에 대한 정보, 애플리케이션이 실행되는 동안에 발생할 수 있는 이벤트 로그 정보등을 제공
  - 웹 애플리케이션 당 1개의 객체가 생성
    - 주로 방문자 카운트와 같은 하나의 웹 애플리케이션에서 고유하는 변수에 사용
  - application 객체 메소드
    - String getServerInfo() : 웹 컨테이너의 이름과 버전을 리턴
    - String getMimeType(fileName) : 지정한 파일의 MIME 타입 리턴
    - String getRealPath(path) : 지정한 경로를 웹 애플리케이션 시스템상의 경로로 변경하여 리턴
    - void log(message) : 로그 파일에 message를 기록

2) 내장 객체의 종류

- **config 객체**
  - JSP 페이지 대한 설정 정보를 저장하고 있는 객체
  - 서블릿이 초기화되는 동안 참조해야 할 정보를 전달
  - 컨테이너당 1개의 객체가 생성
  - config 객체 메소드
    - Enumeration getInitParameterNames() : 모든 초기화 파라미터 이름을 리턴
    - String getInitParameter(name) : 이름이 name인 초기화 파라미터의 값을 리턴
    - String getServletName() : 서블릿의 이름을 리턴
    - ServletContext getServletContext() : 서블릿 ServletContext 객체를 리턴

2) 내장 객체의 종류

- **page 객체**
  - JSP 페이지를 구현한 자바 클래스 객체
  - 웹 컨테이너는 자바만을 스크립트 언어로 지원하기 때문에 page 객체는 현재 거의 사용되지 않음

2) 내장 객체의 종류

- **exception 객체**
  - JSP 페이지에서 예외가 발생한 경우에 사용되는 객체
  - exception 객체의 메소드
    - 주 String getMessage() : 발생된 예외의 메시지를 리턴
    - String toString() : 발생된 예외 클래스명과 메시지 리턴
    - String printStackTrace() : 예외 발생 시 예외가 발생한 곳을 추적함

3) 내장 객체의 영역

- **웹 어플리케이션은 page, request, session, application 이라는 4개의 영역을 가짐**
- **내장 객체의 영역 : 객체의 유효기간**
  - 객체를 누구와 공유할 것인가를 나타냄

3)내장 객체의 영역

- **page 영역**
  - 한 번의 웹 브라우저(클라이언트)의 요청에 대해 하나의 JSP 페이지가 호출
- **request 영역**
  - 한 번의 웹 브라우저(클라이언트)의 요청에 대해 같은 요청을 공유하는 페이지가 대응
  - 같은 request 영역
  - include 액션 태그, forward 액션 태그를 사용하면 request 객체를 공유하게 됨

3) 내장 객체의 종류

- **session 영역**
  - 하나의 웹 브라우저당 1개의 session객체가 생성
  - 같은 session 영역
  - 같은 웹 브라우저 내에서는 요청되는 페이지들
  - 주로 회원 관리에서 회원 인증에 사용
- **application 영역**
  - 하나의 웹 애플리케이션당 1개의 session객체가 생성
  - 같은 application 영역
  - 같은 웹 애플리케이션에 요청되는 페이지들
  - /studyjsp 웹 애플리케이션에서는 같은 application 객체를 공유

3. JSP 페이지 액션 태그

- 액션 태그의 개요
- JSP 페이지의 모듈화
- JSP 페이지의 흐름제어
- 템플릿 페이지를 사용한 JSP 페이지 모듈화

1) 액션 태그의 개요

- **페이지 사이의 제어를 이동시킬 수도 있고,**
- **다른 페이지의 실행 결과를 현재의 페이지에 포함시킬 수 있음**
- **JSP가 제공하는 액션 태그**
  - include, forward, plug-in, useBean, setProperty, getProperty

1) 액션 태그의 개요

- include 액션 태그 : <jsp:include>
  - 다른 페이지의 실행 결과를 현재의 페이지에 포함시킬 때 사용
- forward 액션 태그 : <jsp:forward>
  - 웹 페이지 간의 제어를 이동시킬 때 사용
- plug-in 액션태그 : <jsp:plug-in>
  - 웹 브라우저에서 자바 애플릿을 실행시킬 때 사용
- useBean 액션 태그 : <jsp:useBean>
  - 자바빈을 JSP 페이지에서 사용할 때 사용
- setProperty 액션 태그 : <jsp:setProperty>
- 프로퍼티의 값을 세팅할 때 사용
- getProperty 액션 태그 : <jsp:getProperty>
  - 프로퍼티의 값을 얻어낼 때 사용

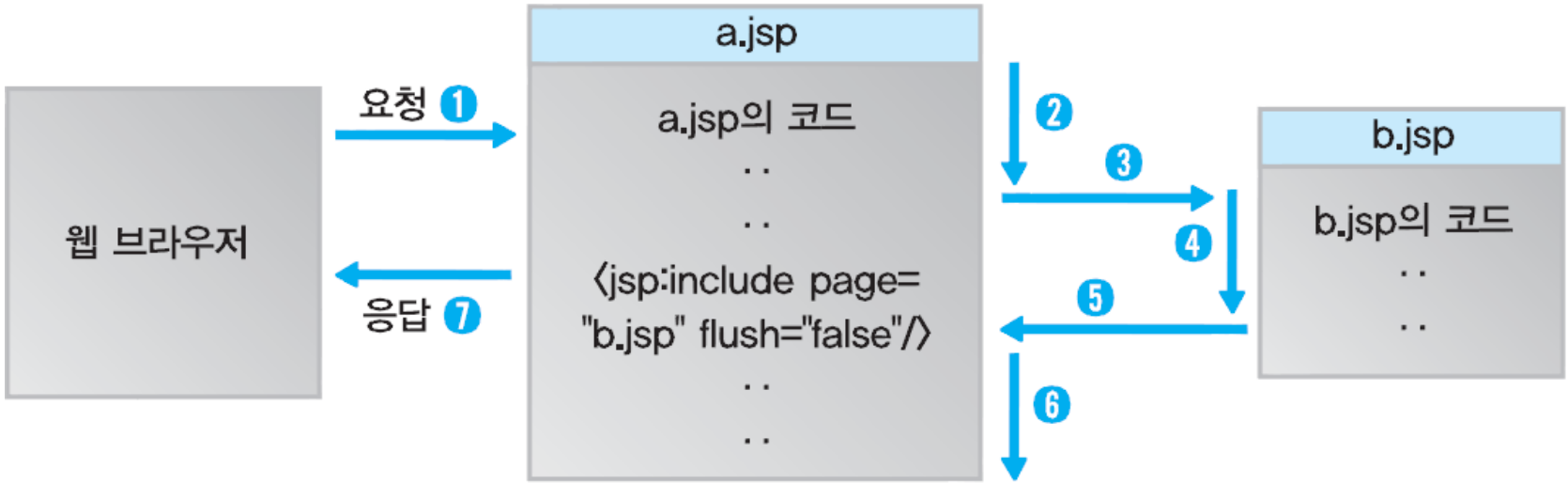
2) JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - 다른 페이지의 처리 결과를 현재 페이지에 포함
  - 페이지를 모듈화 할 때 사용
    - 기본적인 사용법
    - <jsp:include page="포함될 페이지" flush="false"/>
      - page속성 : 결과가 포함될 페이지명
      - flush 속성 : 포함될 페이지로 제어가 이동될 때, 현재 포함하는 페이지가 지금까지 출력 버퍼에 저장한 결과를 처리하는 방법을 결정(false 권장)

2) JSP 페이지의 모듈화

include 액션 태그-<jsp:include>

include 액션 태그의 처리 과정

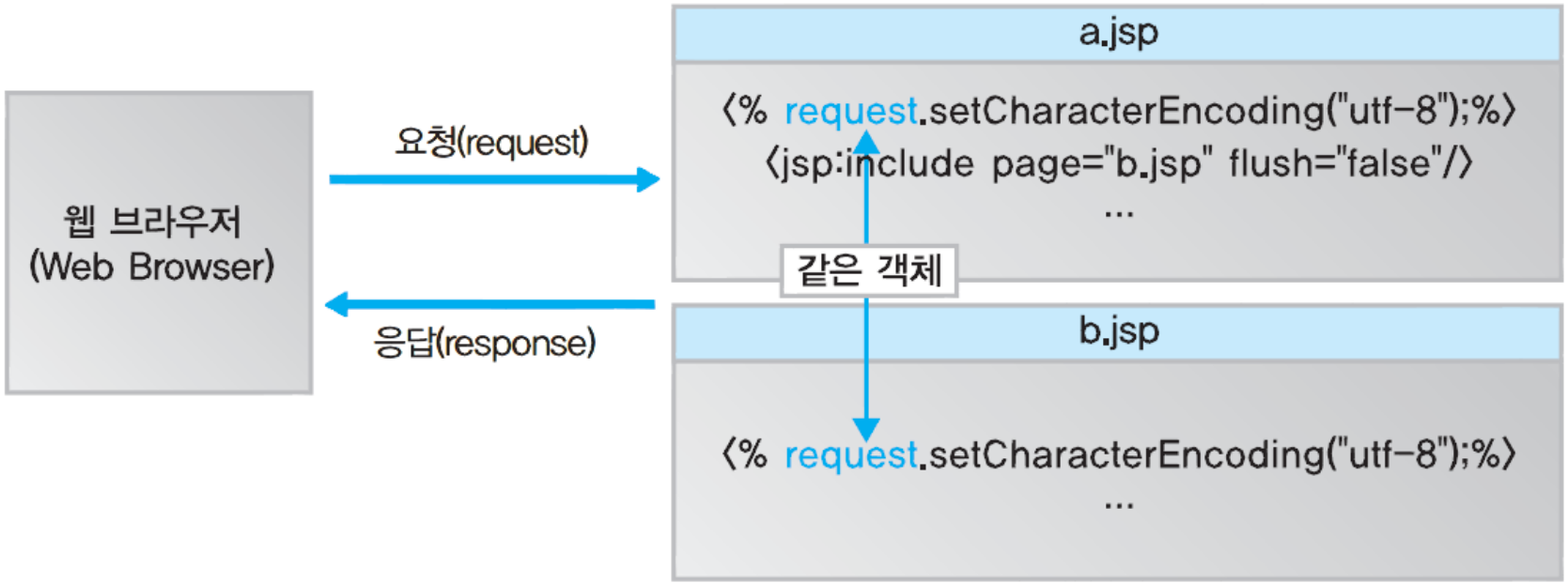


▲ include 액션 태그의 처리 과정

2) JSP 페이지의 모듈화

include 액션 태그-<jsp:include>

include 액션 태그의 처리 과정



▲ include 액션 태그를 사용하는 두 페이지는 같은 request 객체를 공유

2) JSP 페이지의 모듈화

include 액션 태그-<jsp:include>

include 액션 태그의 처리 과정

① 웹 브라우저가 a.jsp 페이지를 웹 서버에 요청

② 서버는 요청받은 a.jsp 페이지를 처리 → 출력 내용은 출력 버퍼에 저장

③ 프로그램제어를 b.jsp 페이지로 이동

④ b.jsp 페이지를 처리. b.jsp 페이지 내에 출력 내용을 출력 버퍼에 저장

⑤ b.jsp 페이지를 처리가 끝나면, 다시 a.jsp 페이지로 프로그램의 제어가 이동 → 이동 위치는 <jsp:include page="b.jsp" flush="false"/> 문장 다음

⑥ a.jsp 페이지의 나머지 부분을 처리, 출력할 내용이 있으면 출력 버퍼에 저장

⑦ 출력 버퍼의 내용을 웹 브라우저로 응답

JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - include 액션 태그에서 포함되는 페이지에 값 전달
    - include 액션 태그의 바디(body) 안에 param 액션 태그(<jsp:param>)를 사용
    - <jsp:include page="포함되는 페이지" flush="false">
    - <jsp:param name="paramName1" value="var1"/>
    - <jsp:param name="paramName2" value="var2"/>
    - </jsp:include>

JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - JSP 페이지의 중복 영역 처리
    - 페이지의 통일성을 가짐
      - 템플릿 페이지 사용
      - 중복되는 페이지의 호출은 include액션 태그 사용

JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - 같은 구조 유지
    - 상단 : 로고 포함한 메뉴
    - 좌측 : 메뉴(하위 메뉴 포함)
    - 중앙 : 내용
    - 하단 : 회사 소개, 찾아오는 길, 보안 정책 등의 내용을 포함
  - 상단, 좌측메뉴, 하단의 경우 같은 내용을 표시해야 하는 경우가 많음
  - 중앙의 내용부분의 내용만 계속 변경

JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - <table>태그를 사용한 경우

```
<table>
  <tr>
    <td colspan="2">상단</td>
  </tr>
  <tr>
    <td>좌측</td>
    <td>중앙의 내용</td>
  </tr>
  <tr>
    <td colspan="2">하단</td>
  </tr>
</table>
```

JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - HTML5의 문서 구조를 사용한 경우

```
<header>
  <nav>상단</nav>
</header>
<div id="leftMenu">
  좌측
</div>
<section id="content">
  중앙의 내용
</section>
<footer>
  하단
</footer>
```

JSP 페이지의 모듈화

- **include 액션 태그-<jsp:include>**
  - 페이지 모듈화 구현 예 : <table> 태그 사용

```
<table>
  <tr>
    <td colspan="2"><jsp:include page="top.jsp" flush="false"/></td>
  </tr>
  <tr>
    <td><jsp:include page="left.jsp" flush="false"/></td>
    <td><jsp:include page="<%=content%>" flush="false"/></td>
  </tr>
  <tr>
    <td colspan="2"><jsp:include page="bottom.jsp "flush="false"/></td>
  </tr>
</table>
```

JSP 페이지의 모듈화

- **include 디렉티브-<jsp:include>**
  - include 디렉티브는 조각 코드를 삽입할 때 사용
    - 코드 차원에서 포함되므로 주로 공용변수, 저작권 표시와 같은 중복 문장에서 사용
    - 사용 예

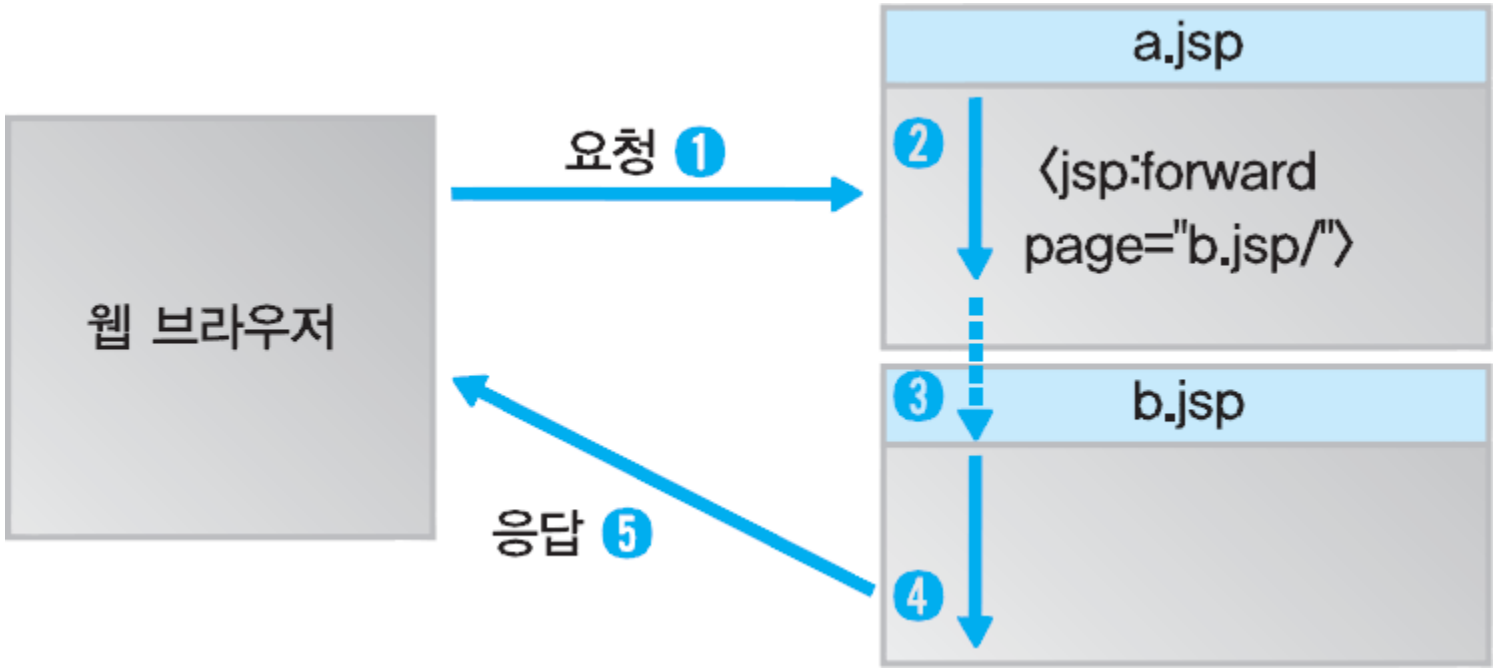
```
<% //공용변수들
   String bodyback_c="#e0ffff";
   String back_c="#8fbc8f";
   String title_c="#5f9ea0";
%>
```

3) JSP 페이지의 흐름 제어

- **forward 액션 태그-<jsp:forward>**
  - 다른 페이지로 프로그램의 제어를 이동할 때 사용
    - forward 액션 태그를 만나게 되면 그 전까지 출력 버퍼에 저장되어 있던 내용을 제거한 후 forward 액션 태그가 지정하는 페이지로 이동
  - forward 액션 태그의 기본적인 사용법
    - <jsp:forward page="이동할 페이지명"/>

3) JSP 페이지의 흐름 제어

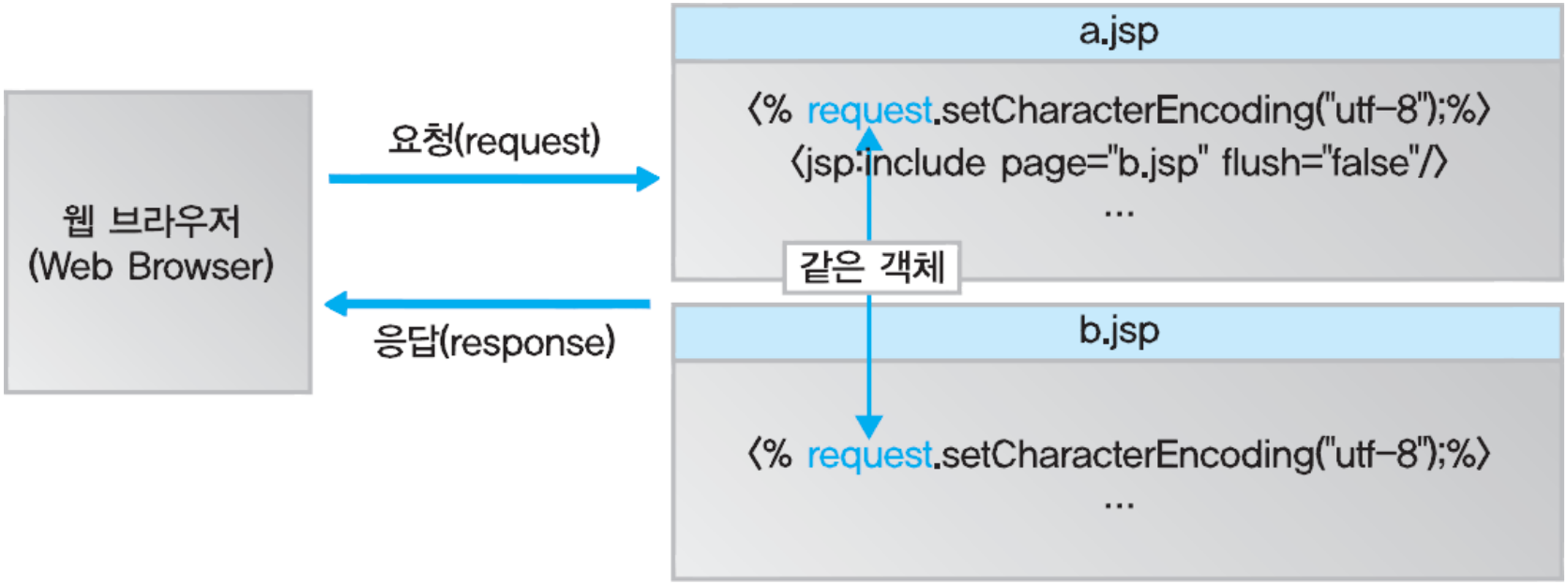
- **forward 액션 태그-<jsp:forward>**
  - forward 액션 태그의 처리 과정



▲ forward 액션 태그의 처리 과정

3) JSP 페이지의 흐름 제어

- **forward 액션 태그-<jsp:forward>**
  - forward 액션 태그의 처리 과정



▲ forward 액션 태그를 사용하는 두 페이지는 같은 request 객체를 공유

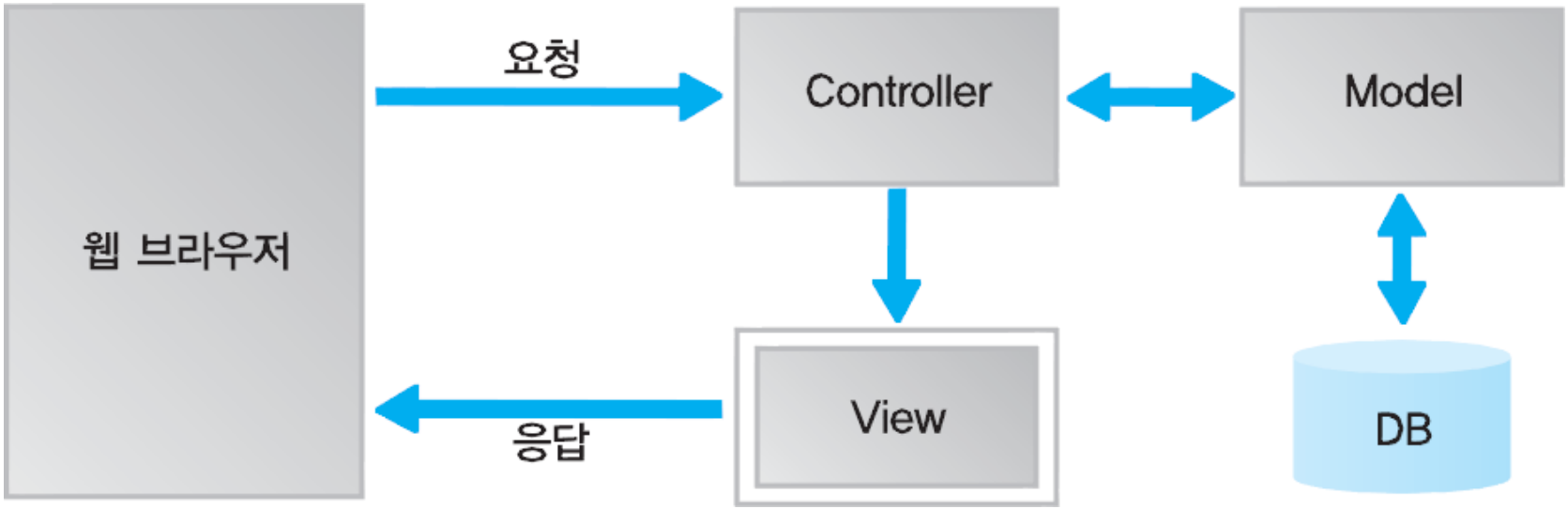
3) JSP 페이지의 흐름 제어

- **forward 액션 태그-<jsp:forward>**
  - forward 액션 태그의 처리 과정
    1. 웹 브라우저에서 웹 서버로 a.jsp 페이지를 요청
    2. 요청된 a.jsp 페이지를 수행
    3. a.jsp 페이지를 수행하다가 <jsp:forward>액션 태그를 만나면 이제까지 저장되어있는 출력 버퍼의 내용을 제거하고 프로그램 제어를 page 속성에서 지정한 b.jsp로 이동(포워딩)
    4. b.jsp 페이지를 수행
    5. b.jsp 페이지를 수행한 결과를 웹 브라우저에 응답

4) 템플릿 페이지를 사용한JSP 페이지의 모듈화

템플릿 페이지의 개요

- JSP 페이지가 MVC에서 뷰에 해당
  - 뷰를 모듈화하는 것이 템플릿 페이지

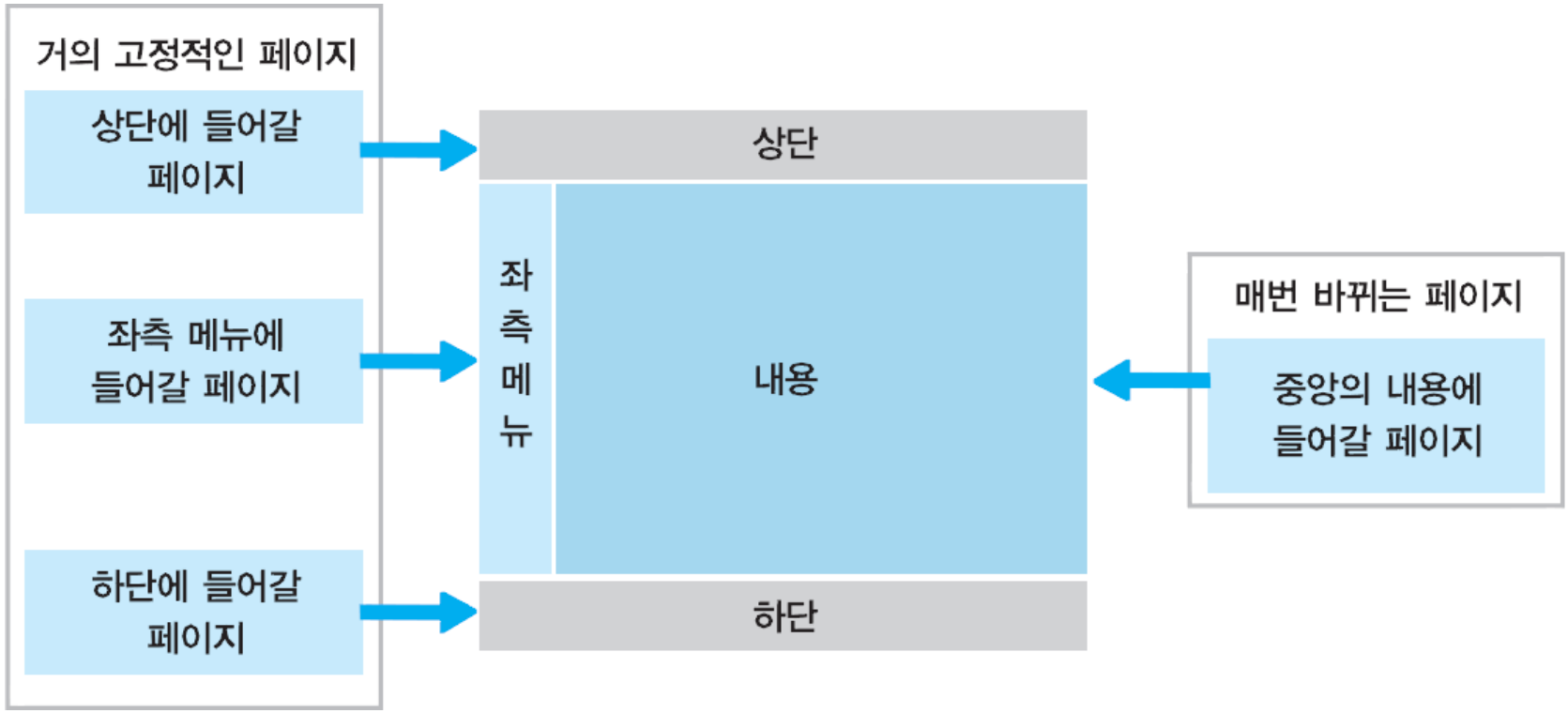


▲ MVC의 구조

4) 템플릿 페이지를 사용한JSP 페이지의 모듈화

탬플릿 페이지의 개요

- 웹 브라우저에 표시되는 하나의 화면은 다수의 페이지로 이루어짐
  - 상단, 좌측 메뉴, 하단 : 거의 고정적인 페이지가 표시
  - 중앙의 내용 부분 : 매번 내용이 바뀌는 페이지가 표시



4) 템플릿 페이지를 사용한JSP 페이지의 모듈화

탬플릿 페이지의 개요

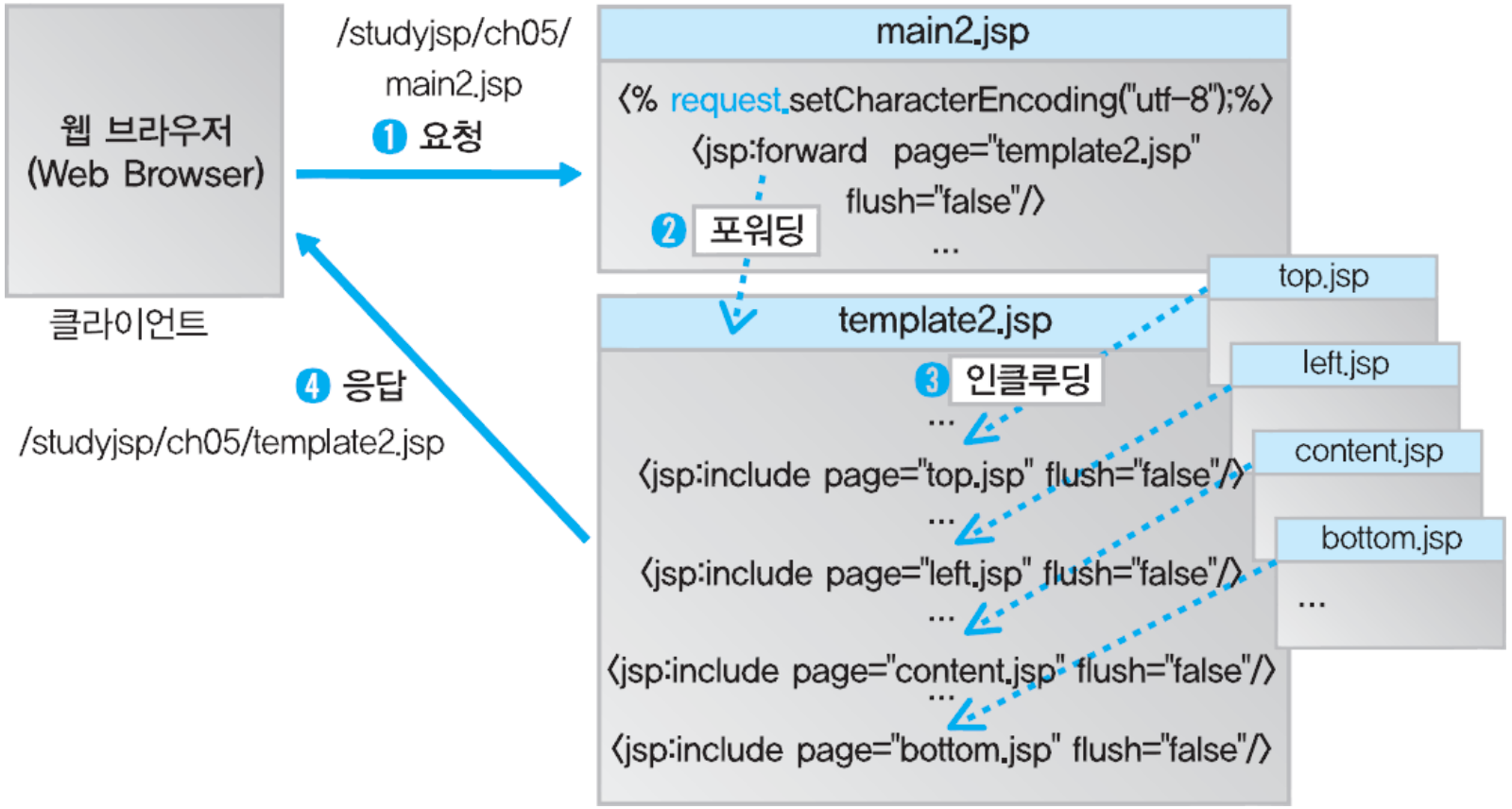
```
<header>
  <nav>상단</nav>
</header>
<div id="leftMenu">
  좌측
</div>
<section id="content">
  중앙의 내용
</section>
<footer>
  하단
</footer>
```

4) 템플릿 페이지를 사용한JSP 페이지의 모듈화

- **탬플릿 페이지 작성하기**
  - 메인 페이지
    - forward 액션 태그를 사용해 템플릿 페이지를 포워딩
  - 템플릿 페이지
    - include 액션 태그를 사용해 페이지 모듈인 top.jsp, left.jsp, content.jsp, bottom.jsp를 로드해 표시

4) 템플릿 페이지를 사용한JSP 페이지의 모듈화

탬플릿 페이지 작성하기



▲ 템플릿 페이지의 처리 구조

4. JSP 페이지 에러 처리

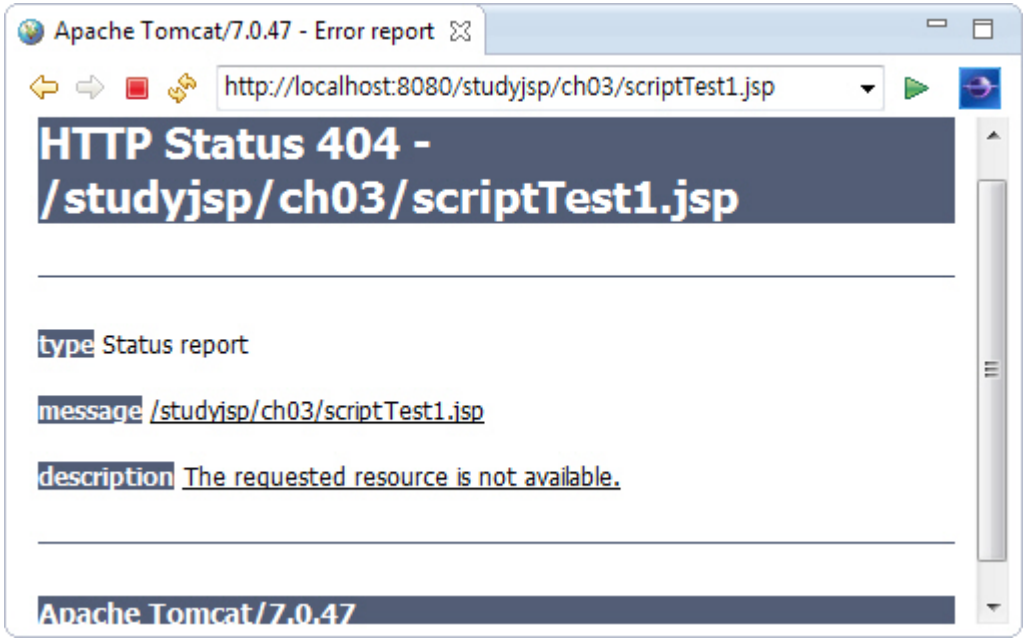
- 에러 처리의 개요
- 에러 코드별 처리

1) 에러 처리의 개요

- **JSP에서의 에러는 하나의 코드에서 에러가 발생하더라도 웹 브라우저의 전체 화면에 에러 메시지가 표시됨**
  - 에러가 어떠한 경로로 발생하게 되었는지 스택을 뒤집어서 그 경로를 추적해서 표시
  - 사이트의 사용자들이 보기에는 부적합
  - 좀더 완곡하게 표현된 것이 필요

2) 에러 코드별 처리

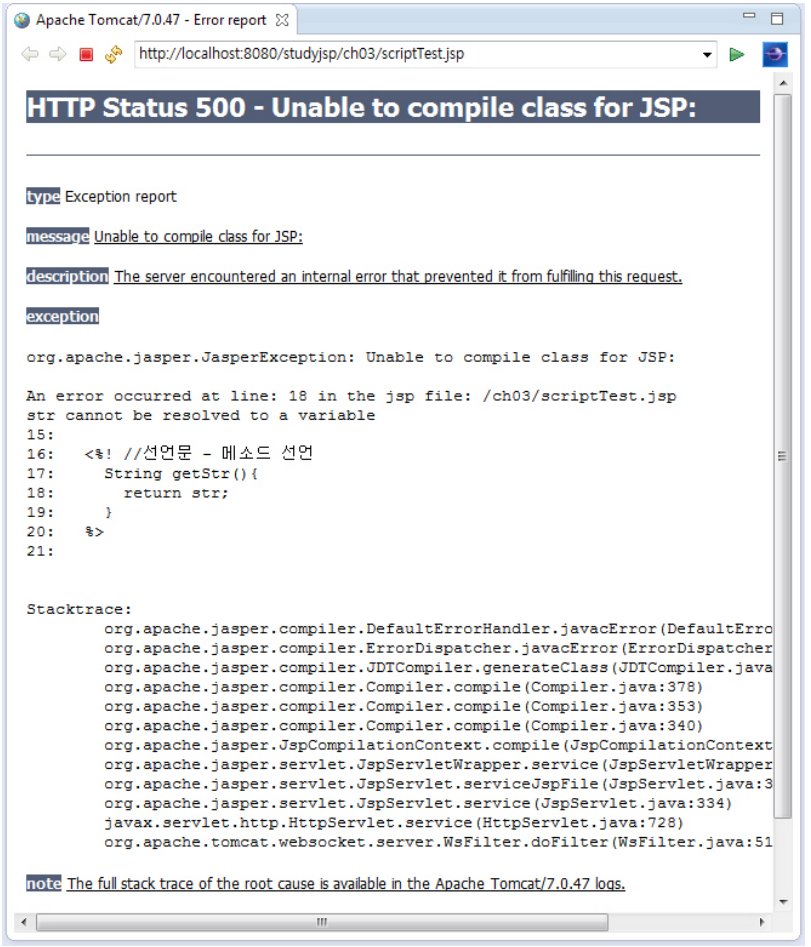
- **HTTP에서 알아 두어야 하는 에러 코드**
  - 404
    - Not Found, 문서를 찾을 수 없음. 이 에러는 클라이어트가 요청한 문서를 찾지 못한 경우에 발생
    - URL을 다시 잘 보고 주소가 올바로 입력되었는지를 확인



▲ HTTP 404 에러

2) 에러 코드별 처리

- **HTTP에서 알아 두어야 하는 에러 코드**
  - 500
    - Internal Server Error, 서버 내부 오류. 이 에러는 웹 서버가 요청 사항을 수행할 수 없을 경우 발생



2) 에러 코드별 처리

- **JSP에서 404 에러와 500 에러 처리 방법**
  - web.xml에 <error-page> 엘리먼트를 사용해서 에러 제어
    - [프로젝트]-[Webcontent]-[WEB-INF]의 web.xml에 404와 500 에러를 제어할 <error-page> 엘리먼트를 각각 작성
  - 404 에러 코드 처리

```
<error-page><!--404에러처리-->
  <error-code>404</error-code>
  <location>
      /error/404code.jsp
  </location>
</error-page>
```

2) 에러 코드별 처리

- **JSP에서 404 에러와 500 에러 처리 방법**
  - ② 에러가 발생 시 표시할 페이지를 작성
    - [프로젝트]-[WebContent]-[error] 폴더에 404code.jsp와 500code.jsp 페이지 작성
    - 404code.jsp와 500code.jsp의 소스 코드에 현재 페이지가 정상적으로 응답되는 페이지임을 지정하는 코드를 기술
      - *<%response.setStatus(HttpServletResponse.SC_OK);%>* 추가
        - 이 코드를 생략 시, 웹 브라우저는 자체적으로 제공하는 에러 페이지를 표시