4. 내장 객체와 액션태그

1. 내장 객체
2. 영역 객체와 속성
3. 액션 태그

1. 내장객체

- 내장객체의 개요
  - 웹 컨테이너가 JSP 페이지를 서블릿 클래스로 변환하면서 사용자의 요청에 맞는 응답 페이지를 생성하기 위해 사용되는 javax.servlet 패키지 내부에 있는 고정된 이름의 객체
  - 웹 컨테이너에 의해 자동으로 구현되며 import 나 객체 선언 없이도 자유롭게 접근 가능

1. 내장 객체

- 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

- 주요내장 객체

내장 객체 변수명	클래스/인터페이스 타입	설명
request	javax.servlet.SevletRequest(javax.servlet.http.HttpServletRequest)	클라이언트의 HTTP 요청 정보를 저장한 객체(HTTP 헤더 정보, 파라미터 등)
response	javax.servlet.ServletResponse(javax.servlet.http.HttpServletResponse)	HTTP 요청에 대한 응답 정보를 저장한 객체
session	javax.servlet.http.HttpSession	클라이언트의 세션 정보를 저장한 객체
pageContext	javax.servlet.jsp.PageContext	페이지 실행에 필요한 컨텍스트 정보를 저장한 객체

1. 내장 객체

- request 객체
  - 웹 브라우저의 요청 정보를 저장하고 있는 객체
  - 입력폼에 입력한 사용자의 요구 사항을 얻어낼 수 있도록 요청 메소드를 제공
  - 요청 파라미터와 관련된 메소드

리턴 타입	메소드명	설명
String	getParameter(String name)	name이란 이름으로 지정된 파라미터에 할당된 값을 리턴한다. 지정된 이름의 파라미터가 없으면 null을 리턴한다.
String[]	getParameterValues(String name)	name이란 이름으로 지정된 파라미터의 모든 값을 String 배열로 리턴한다. 하나의 이름으로 여러 개의 값을 가질 수 있는 checkbox와 같은 태그를 사용했을 때에 주로 사용되며 하나의 이름에 하나의 값만 가지는 파라미터는 getParameter(String name) 메소드를 사용하는 것이 좋다.
Enumeration	getParameterNames( )	요청에 포함된 모든 파라미터 이름을 java.util.Enumeration 객체로 리턴한다.

1. 내장 객체

- request 객체
  - HTTP 헤더 정보와 관련된 메소드들

리턴 타입	메소드명	설명
String	getHeader(String headerName)	HTTP 요청 헤더에 headerName으로 지정된 이름으로 할당된 값을 리턴한다. headerName으로 지정된 이름이 없을 경우 null을 리턴한다.
Enumeration	getHeaders(String headerName)	headerName으로 지정된 이름으로 할당된 모든 값을 java.util.Enumeration 객체로 리턴한다.
Enumeration	getHeaderNames( )	HTTP 요청 헤더에 포함된 모든 헤더 이름을 java.util.Enumeration 객체로 리턴한다.
int	getIntHeader(String headerName)	headerName 헤더의 값을 int 타입으로 리턴한다. 지정된 헤더값을 int로 변환할 수 없을 경우에는 NumberFormatException이 발생하고 headerName 헤더가 없을 경우에는 -1을 리턴한다.

1. 내장 객체

- request 객체
  - HTTP 헤더 정보와 관련된 메소드들

리턴 타입	메소드명	설명
Cookie[ ]	getCookies( )	HTTP 요청 메시지의 헤더에 포함된 쿠키를 javax.servlet.http.Cookie 배열로 리턴한다.
String	getServerName( )	서버의 도메인명을 문자열로 리턴한다.
int	getServerPort( )	서버의 포트 번호를 int형으로 리턴한다.
StringBuffer	getReqeustURL( )	요청 URL을 StringBuffer로 리턴한다.
String	getRequestURI( )	요청 URI를 문자열로 리턴한다.
String	getQueryString( )	요청에 사용된 쿼리 문장을 문자열로 리턴한다.
String	getRemoteHost( )	클라이언트의 호스트 이름을 문자열로 리턴한다.
String	getRemoteAddr( )	클라이언트의 IP 주소를 문자열로 리턴한다.
String	getProtocol( )	요청에 사용된 프로토콜 이름을 문자열로 리턴한다.
String	getMethod( )	요청에 사용된 요청 방식(GET, POST 등)을 문자열로 리턴한다.
String	getContextPath( )	해당 JSP 페이지의 컨텍스트 경로를 문자열로 리턴한다.

1. 내장 객체

- request 객체
  - 세션정보와 관련된 메소드

리턴 타입	메소드명	설명
HttpSession	getSession( )	요청한 클라이언트에 할당된 HttpSession 객체를 반환한다. 이전에 생성된 HttpSession 객체가 없으면 새로운 객체를 생성해 할당한다.
HttpSession	getSession(Boolean create)	create가 true일 경우 getSession( ) 메소드와 동일한 결과를 리턴하지만 create를 false로 지정하면 이전에 생성된 HttpSession 객체가 없을 경우 null을 리턴한다.
String	getRequestedSessionId( )	요청한 클라이언트에 지정된 세션의 ID를 문자열로 리턴한다.
boolean	isRequestedSessionIdValid( )	요청에 포함된 클라이언트의 세션 ID가 유효하면 true를, 아니면 false를 리턴한다.

1. 내장 객체

- response 객체
  - 웹 브라우저의 요청에 대한 응답 정보를 저장하고 있는 객체
  - 응답 정보와 관련하여 주로 헤더 정보 입력, 리다이렉트 등의 기능을 제공
  - response 객체의 주요 메소드

리턴 타입	메소드명	설명
없음	setHeader(String headerName, String headerValue)	응답에 포함될 헤더 정보에 headerName의 이름으로 headerValue 값을 설정해 추가한다.
없음	addCookie(Cookie cookie)	javax.servlet.http.Cookie 타입의 쿠키 객체를 응답 헤더에 추가한다. 쿠키에 대해서는 Chapter 8에서 자세히 다룬다.
없음	sendRedirect(String url)	지정된 URL로 요청을 재전송한다.
없음	setContentType(String type)	응답 페이지의 contentType을 설정한다.

1. 내장 객체

- 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. 줄 바꿈은 적용되지 않음
    - void flush() : 출력 버퍼의 내용을 웹 브라우저에 전송하고 비움
    - void close() : 출력 버퍼의 내용을 웹 브라우저에 전송하고 출력 스트림을 닫음

1. 내장 객체

- pageContext 객체
  - JSP 페이지 대한 정보를 저장하고 있는 객체
  - 다른 내장 객체를 구하거나, 페이지의 흐름제어 그리고 에러 데이터를 얻어낼 때 사용
  - 내장 객체의 속성을 제어

1. 내장 객체

- pageContext 객체
  - pageContext 내장 객체의 메소드

리턴 타입	메소드명	설명
ServletRequest	getRequest( )	클라이언트의 요청 정보를 담고 있는 객체를 리턴한다(request 내장 객체를 리턴한다).
ServletResponse	getResponse( )	요청에 대한 응답 객체를 리턴한다(response 내장 객체를 리턴한다).
JspWriter	getOut( )	응답 출력 스트림을 리턴한다(out 내장 객체를 리턴한다).
Object	getPage( )	서블릿 인스턴스 객체를 리턴한다(page 내장 객체를 리턴한다).
ServletConfig	getServletConfig( )	서블릿의 초기 설정 정보를 담고 있는 객체를 리턴한다(config 내장 객체를 리턴한다).
ServletContext	getServletContext( )	서블릿의 실행 환경 정보를 담고 있는 객체를 리턴한다 (application 내장 객체를 리턴한다).
HttpSession	getSession( )	클라이언트의 세션 정보를 담고 있는 객체를 리턴한다(session 내장 객체를 리턴한다).
없음	forward(String url)	현재 페이지의 요청과 응답에 관한 제어권을 URL로 지정된 주소로 영구적으로 넘긴다. forward된 페이지의 요청 처리가 종료되면 응답도 종료된다.
없음	include(String url)	현재 페이지의 요청과 응답에 관한 제어권을 URL로 지정된 주소로 임시로 넘긴다. include된 페이지의 처리가 끝나면 제어권은 다시 원래의 페이지로 돌아온다. 따라서 include로 지정된 페이지의 내용을 원래 페이지에 삽입하는 효과를 가진다.

1. 내장 객체

- session 객체
  - 하나의 웹 브라우저 내에서 정보를 유지하기 위한 세션 정보를 저장하고 있는 객체
  - 서버의 연결이 끊어져도 가상 연결을 통해 클라이언트의 정보를 유지하도록 하는 객체
  - 웹 브라우저(클라이언트)당 1개가 할당
    - 주로 회원 관리 시스템에서 사용자 인증에 관련된 작업을 수행할 때 사용

1. 내장 객체

- session 객체
  - session 객체의 주요 메소들

리턴 타입	메소드명	설명
String	getId( )	해당 세션의 세션 ID를 문자열로 리턴한다. 세션 ID는 session 객체 생성 시에 웹 컨테이너에 의해 자동으로 할당된다.
long	getCreationTime( )	1970년 1월 1일 00시 00분 00초(epoch)부터 해당 세션이 생성된 순간까지의 경과 시간을 밀리초로 계산하여 long형으로 리턴한다.
long	getLastAccessedTime( )	epoch로부터 해당 세션에 마지막으로 접근된 시간까지의 경과 시간을 밀리초로 계산하여 long형으로 리턴한다.
int	getMaxInactiveInterval( )	클라이언트의 요청이 없을 시 서버가 해당 세션을 유지하도록 지정된 시간을 초 단위의 정수로 리턴한다.
없음	invalidate( )	세션의 속성 값으로 저장된 모든 객체를 반납하여 해당 세션을 종료시킨다.
boolean	isNew( )	새로운 세션일 경우 true를 리턴하고 기존에 세션이 유지되고 있으면 false를 리턴한다.
없음	setMaxInactiveInterval(int seconds)	클라이언트의 요청이 없더라도 세션을 유지할 시간을 초 단위의 정수값으로 설정한다. 음수로 설정할 경우 세션은 무효화(invalidate)되지 않는다.
없음	forward(String url)	현재 페이지의 요청과 응답에 관한 제어권을 URL로 지정된 주소로 영구적으로 넘긴다. forward된 페이지의 요청처리가 종료되면 응답도 종료된다.
없음	include(String url)	현재 페이지의 요청과 응답에 관한 제어권을 URL로 지정된 주소로 임시로 넘긴다. include된 페이지의 처리가 끝나면 제어권은 다시 원래의 페이지로 돌아온다. 따라서 include로 지정된 페이지의 내용을 원래 페이지에 삽입하는 효과를 가진다.

1. 내장 객체

- 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 시스템 상의 경로로 변경하여 리턴
    - void log(message) : 로그 파일에 message를 기록

1. 내장 객체

- application 객체
  - application 주요 메소드들

리턴 타입	메소드명	설명
int	getMajorVersion( )	Servlet API 스펙의 Major 버전을 int로 리턴한다.
int	getMinorVersion( )	Servlet API 스펙의 Minor 버전을 int로 리턴한다.
String	getServerInfo( )	서블릿/JSP 컨테이너의 이름과 버전을 문자열로 리턴한다.
String	getMimeType(String file)	서버에 존재하는 file이란 이름을 가진 파일의 MIME 타입을 문자열로 리턴한다.
java.net.URL	getResource(String path)	path로 지정된 경로의 자원을 URL 객체로 리턴한다. 자원이 존재하지 않으면 null을 리턴한다.
InputStream	getResourceAsStream(String path)	path로 지정된 경로의 자원을 InputStream 객체로 리턴한다. 자원이 존재하지 않으면 null을 리턴한다.
String	getRealPath(String path)	path로 지정된 경로의 자원을 서버의 실제 파일 시스템상의 경로로 바꾸어 문자열로 리턴한다.
없음	log(String msg)	문자열 msg를 서블릿 로그 파일에 기록한다.
없음	log(String msg, java.lang. Throwable exception)	문자열 msg와 예외의 StackTrace 정보를 로그 파일에 기록한다.

1. 내장 객체

- 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

1. 내장 객체

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

- 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 추적

2. 영역객체와 속성

- 영역객체(scope)와 속성(Attribute)
  - 내장 객체들 중 session, request, application 객체들은 해당 객체에 정의된 범위 내에서 데이터 공유 가능
  - 공유되는 데이터를 **속성(Attribute)**, 속성을 공유할 수 있는 유효 범위를 **영역(Scope)**이라고 함
    - 내장 객체의 영역 : 객체의 유효기간
    - 객체를 누구와 공유할 것인가를 나타냄
  - 웹 어플리케이션은 page, request, session, application 이라는 4개의 영역을 가짐

2. 영역객체와 속성

- 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

2. 영역객체와 속성

- 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

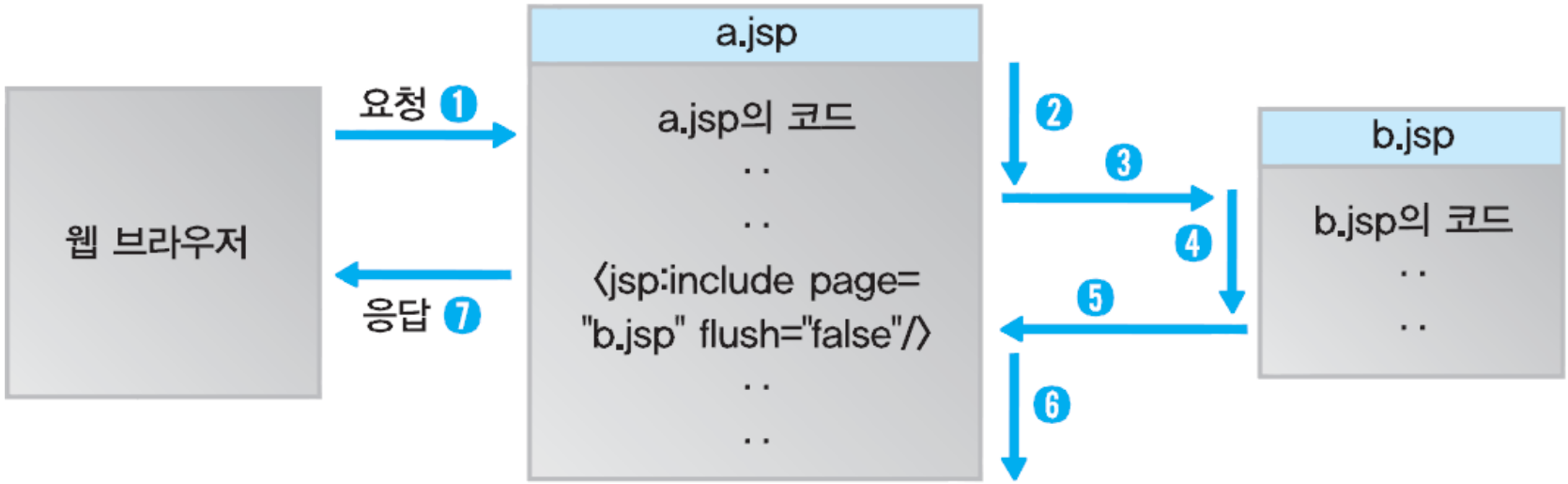
- include 액션 태그 : <jsp:include>
  - 다른 페이지의 실행 결과를 현재의 페이지에 포함시킬 때 사용
- forward 액션 태그 : <jsp:forward>
  - 웹 페이지 간의 제어를 이동시킬 때 사용
- plug-in 액션태그 : <jsp:plug-in>
  - 웹 브라우저에서 자바 애플릿을 실행시킬 때 사용
- useBean 액션 태그 : <jsp:useBean>
  - 자바빈을 JSP 페이지에서 사용할 때 사용
- setProperty 액션 태그 : <jsp:setProperty>
- 프로퍼티의 값을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

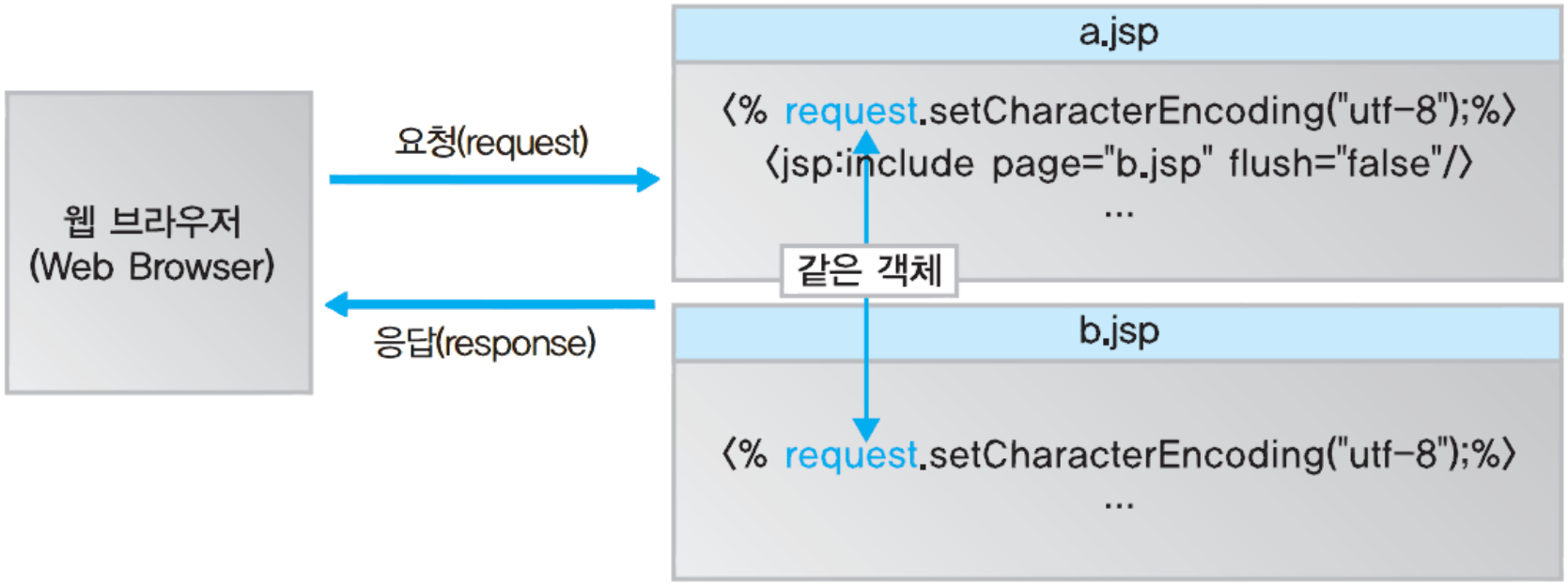
- include 액션 태그-<jsp:include>
  - include 액션 태그의 처리 과정



▲ 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

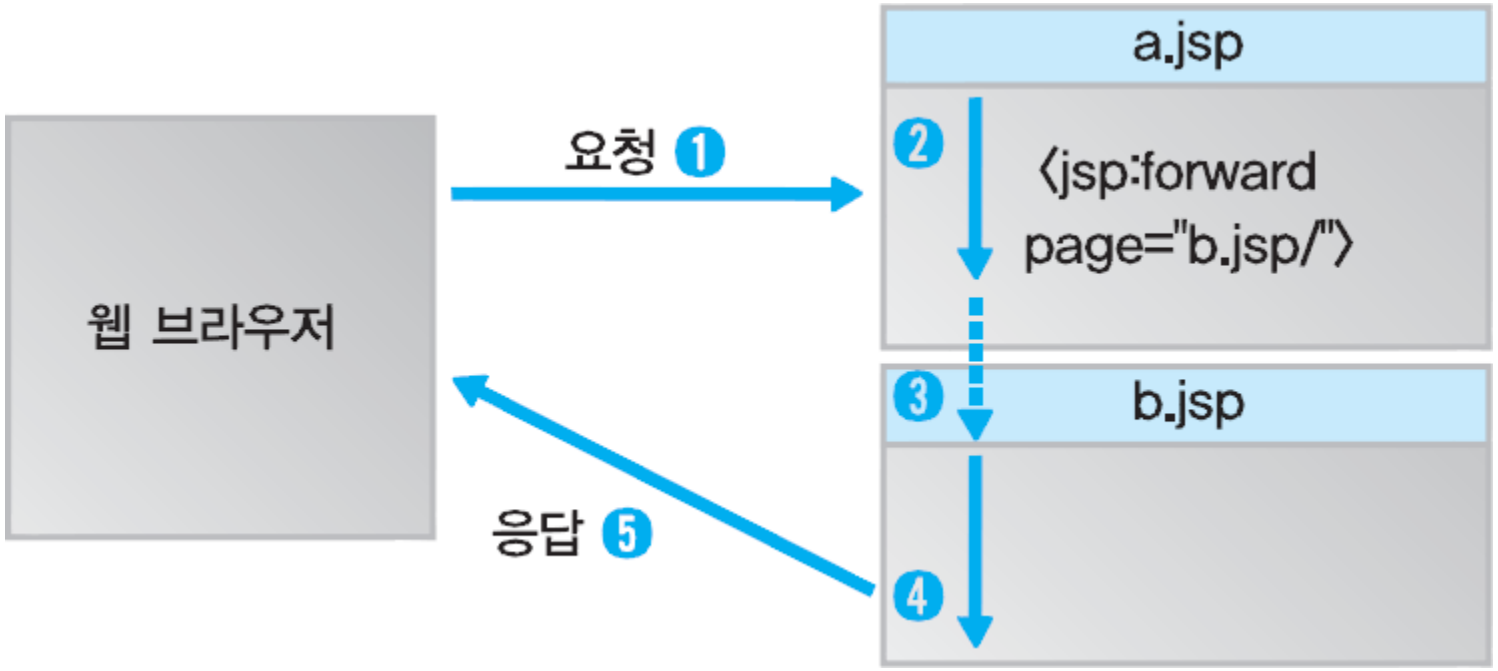
```
<% //공용변수들
   String bodyback_c="#e0ffff";
   String back_c="#8fbc8f";
   String title_c="#5f9ea0";
%>
```

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

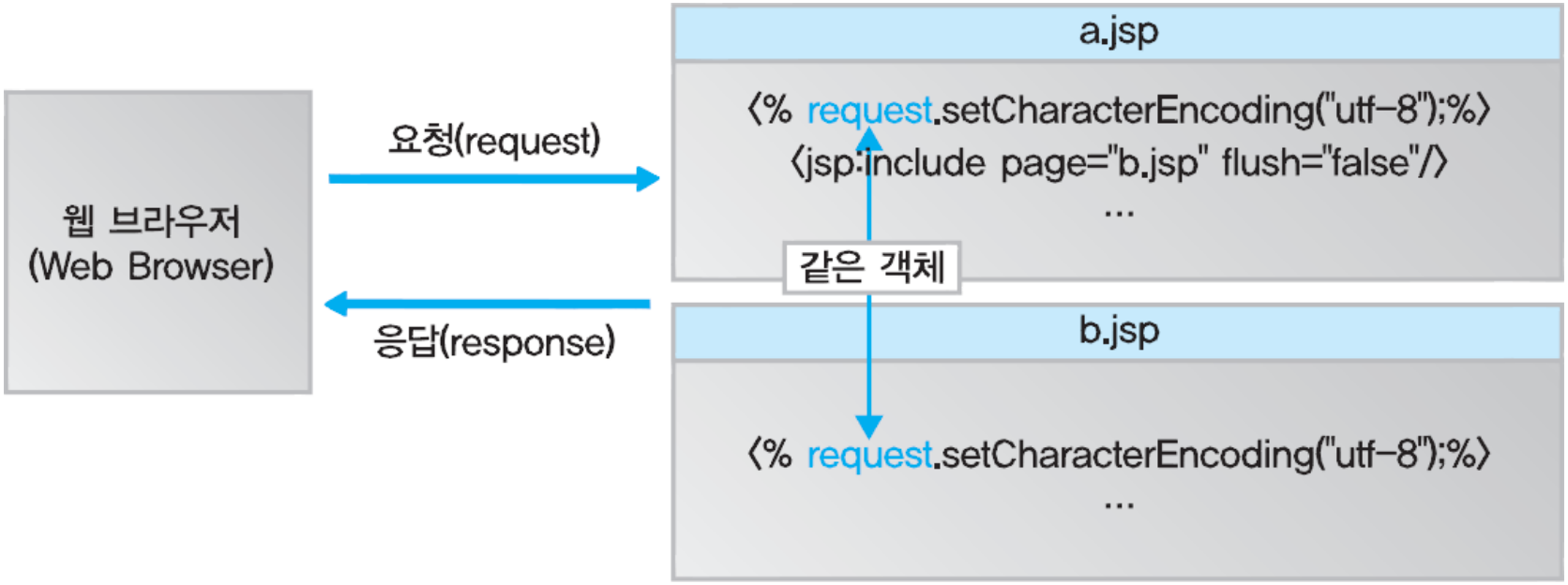
- forward 액션 태그-<jsp:forward>
  - forward 액션 태그의 처리 과정



▲ forward 액션 태그의 처리 과정

3) JSP 페이지의 흐름 제어

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



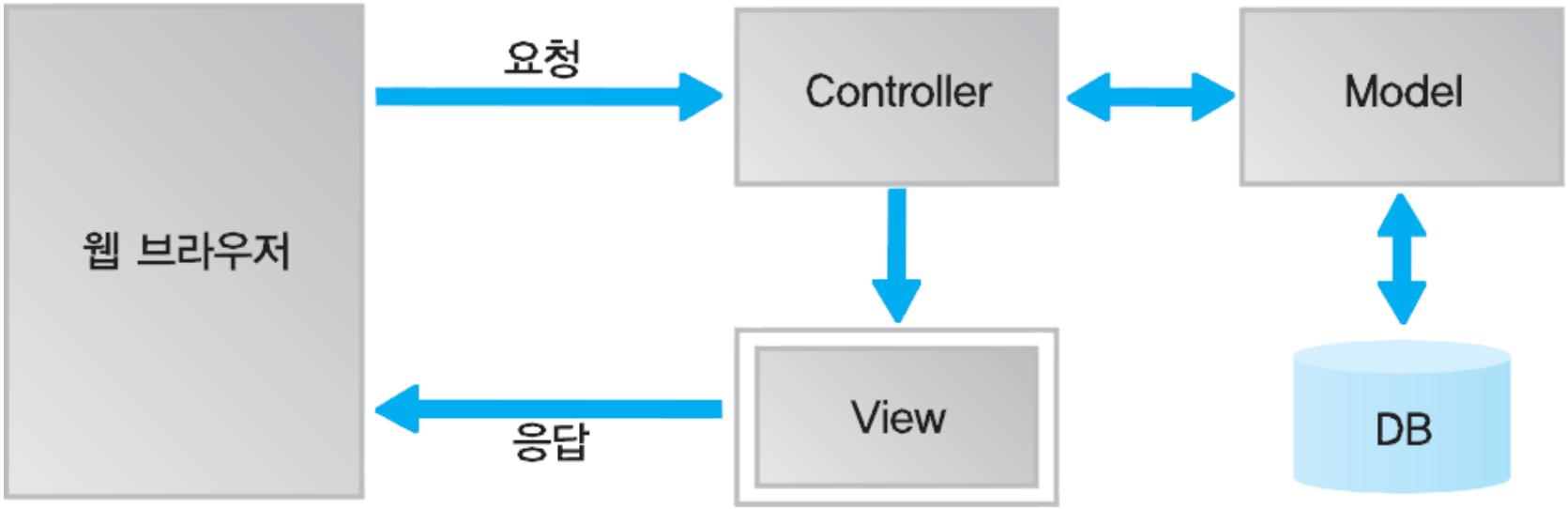
▲ forward 액션 태그를 사용하는 두 페이지는 같은 request 객체를 공유

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

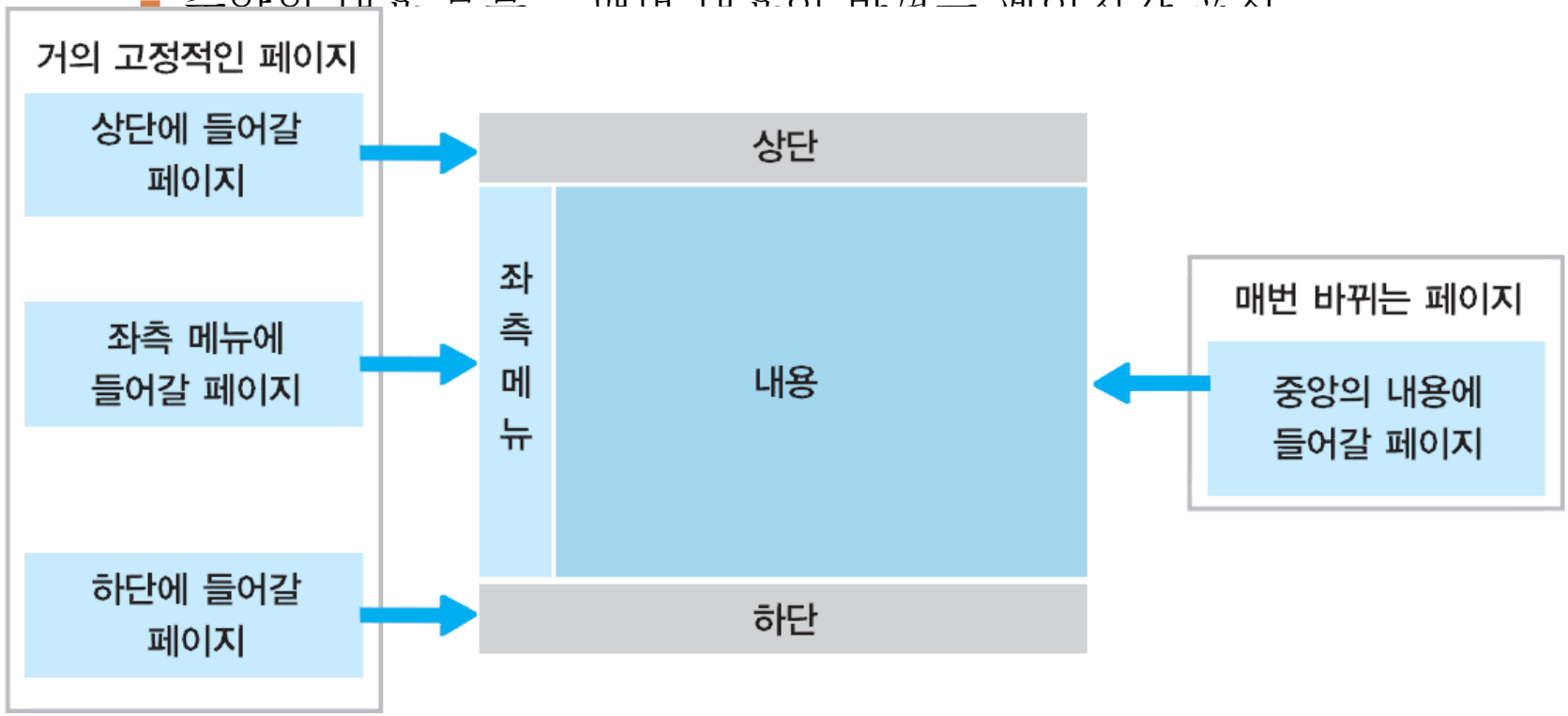
- 탬플릿 페이지의 개요
  - JSP 페이지가 MVC에서 뷰에 해당
    - 뷰를 모듈화하는 것이 템플릿 페이지



▲ 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

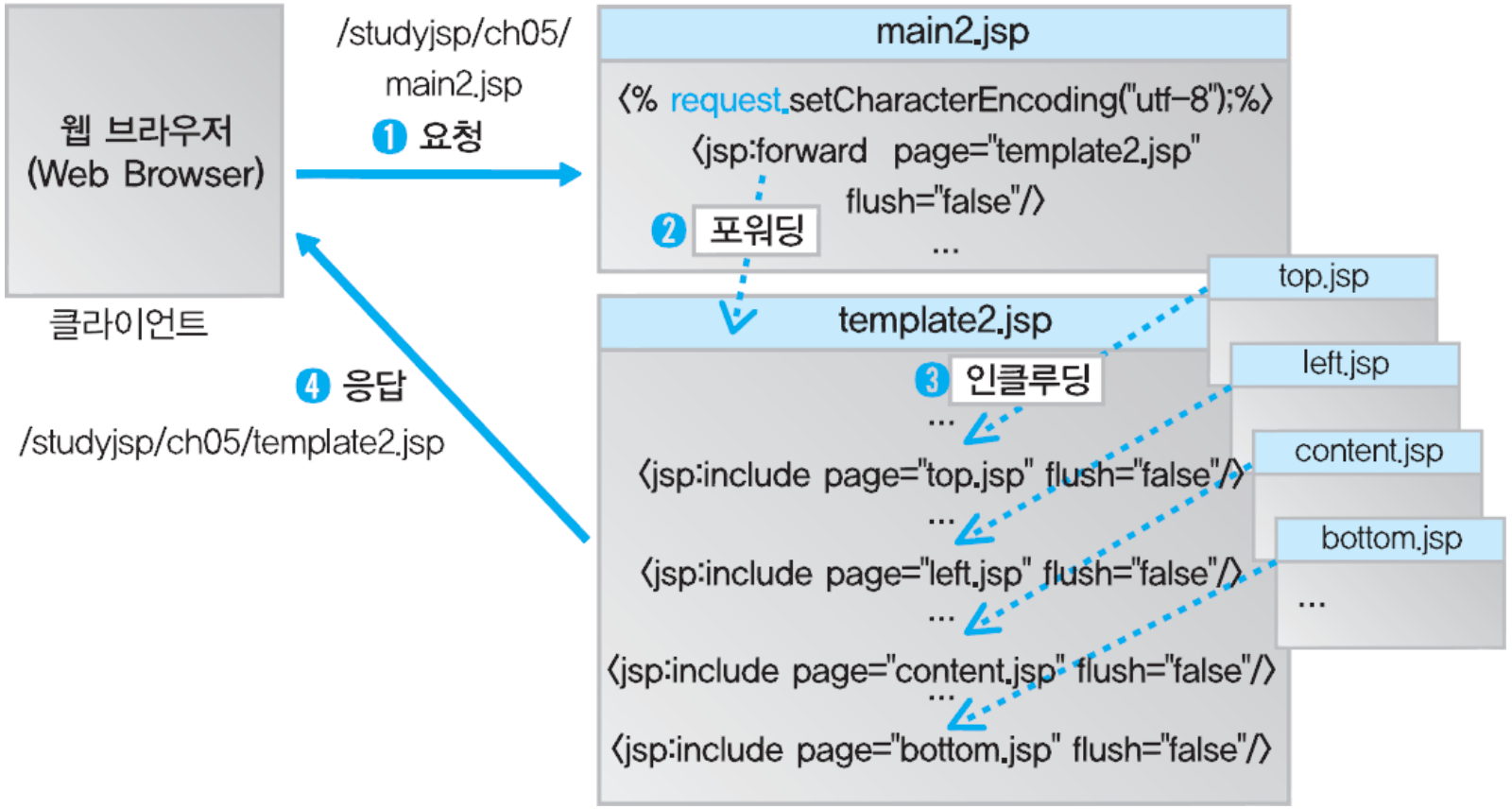
```
<header>
  <nav>상단</nav>
</header>
<div id="leftMenu">
  좌측
</div>
<section id="content">
  중앙의 내용
</section>
<footer>
  하단
</footer>
```

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

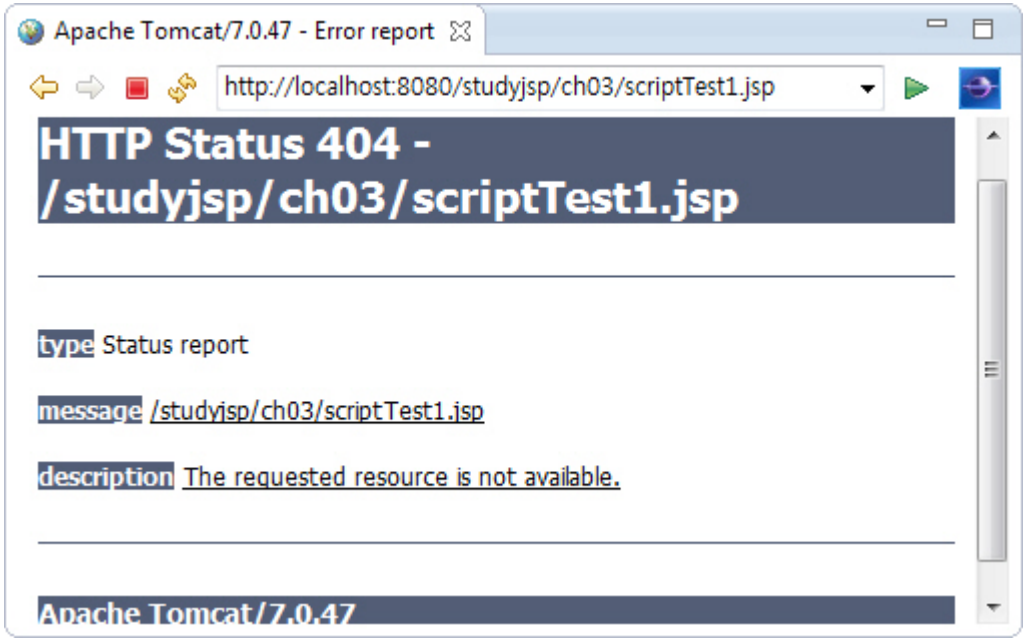
- 에러 처리의 개요
- 에러 코드별 처리

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

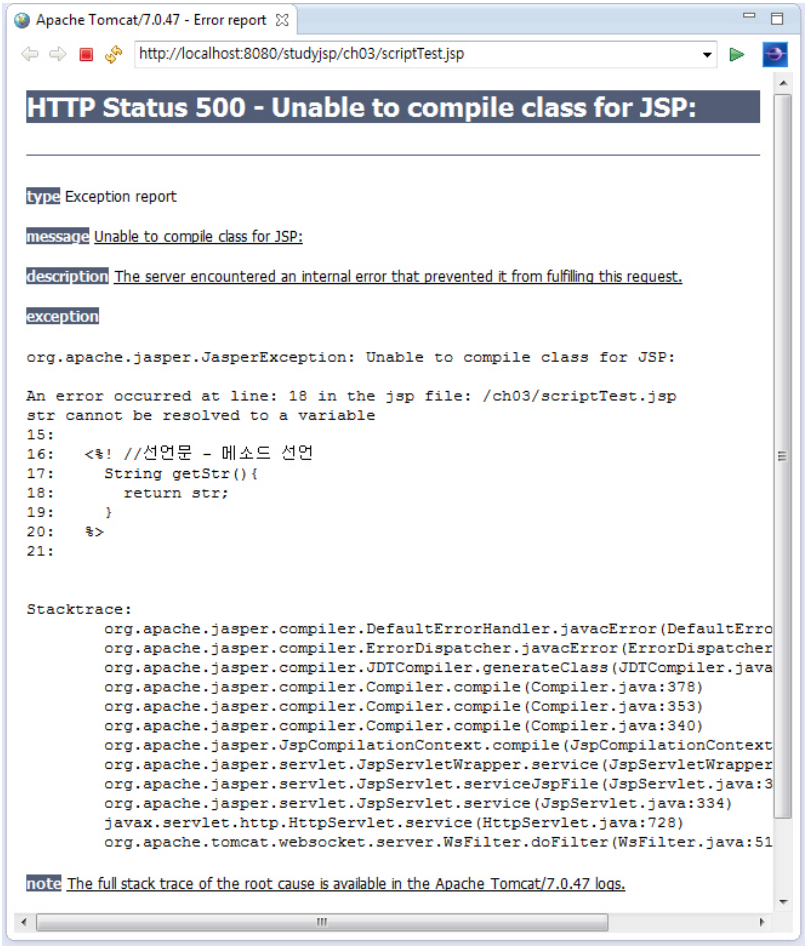
- HTTP에서 알아 두어야 하는 에러 코드
  - 404
    - Not Found, 문서를 찾을 수 없음. 이 에러는 클라이어트가 요청한 문서를 찾지 못한 경우에 발생
    - URL을 다시 잘 보고 주소가 올바로 입력되었는지를 확인



▲ 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