

JSP & Servlet #3

- 웹 프로그래밍 & 톰캣

목 차

1. 웹 어플리케이션 개요
2. Java 웹 프로그래밍
3. 톰캣 설치 및 사용

Readings



Readings:

□ Chapter 1:

*서블릿과 JSP는 어디에 쓰이는 물건이고? :
먼저 간략히 알아봅시다.*

□ Chapter 2:

*웹 애플리케이션 아키텍처 : 조금 더 깊이
들어가 보죠*

□ Chapter 3:

초 간단 미니 MVC 튜토리얼 : 초 간단 MVC

Objective

- ❑ **HTML**과 **HTTP**의 차이를 설명할 수 있다.
- ❑ 정적인 웹 페이지와 동적인 웹 페이지를 구별하여 설명할 수 있다.
- ❑ 추상적으로, 서블릿과 **JSP** 코드의 차이를 구별하여 설명할 수 있다.
- ❑ **Java** 웹 애플리케이션에서 서블릿과 **JSP**의 역할의 차이를 설명할 수 있다.
- ❑ 로컬 **PC**에 톰캣 서버를 설치하고 구동할 수 있다.
- ❑ 포트 충돌이 발생하는 경우, 톰캣이 사용하는 포트를 변경할 수 있다.
- ❑ 웹 애플리케이션의 디렉터리 구조를 설명할 수 있다.

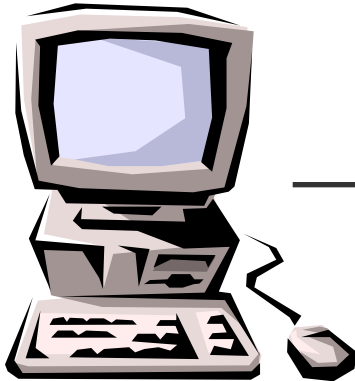
1.웹 애플리케이션 개요

1. 웹 애플리케이션 구성요소

I. 웹 애플리케이션 개요

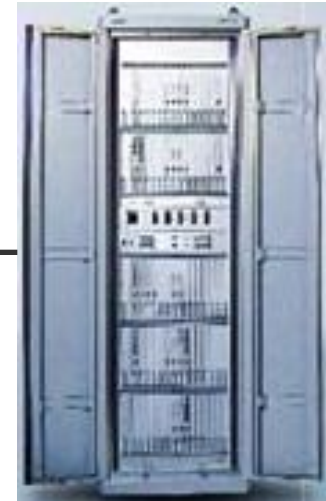
□ 웹 애플리케이션 구성요소

Client computer



Web browser

Server computer



Web server

Database Server

Internet
Connection

1. 웹 애플리케이션 구성요소

I. 웹 애플리케이션 개요

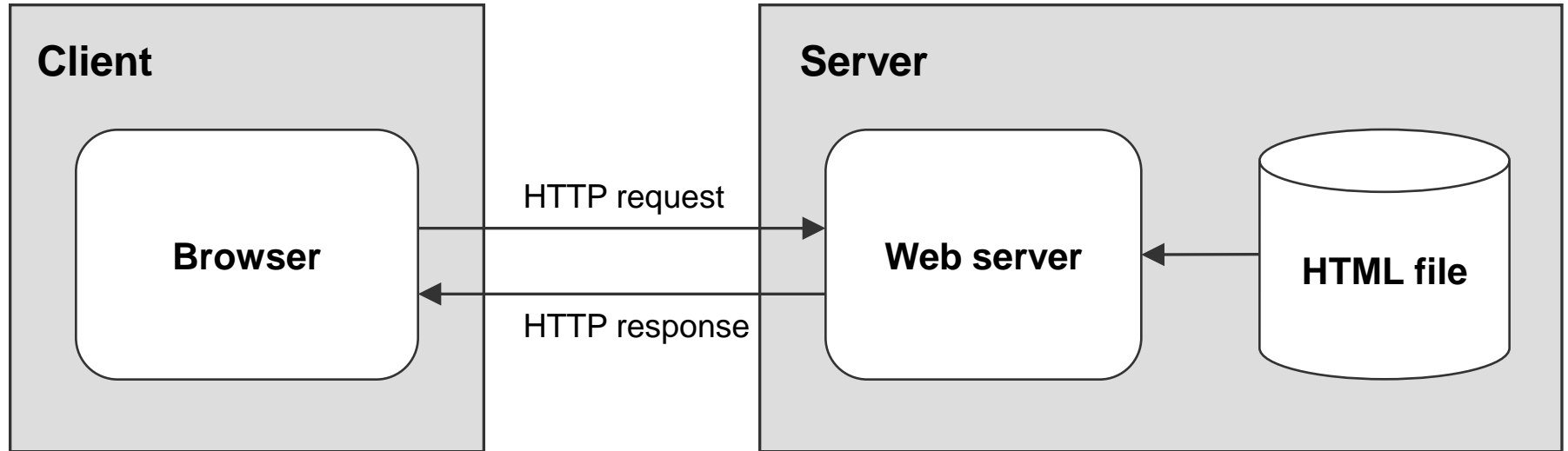
□ 웹 애플리케이션 구성요소

- 웹 애플리케이션은 *클라이언트/서버 애플리케이션*의 한 유형이다.
- *클라이언트* 컴퓨터에 위치하고 있는 사용자가 *서버* 컴퓨터에서 구동되고 있는 애플리케이션에 접근한다.
- 웹 애플리케이션에서는 클라이언트와 서버 컴퓨터가 인터넷 또는 인트라넷으로 연결된다.
- 웹 브라우저가 애플리케이션에 대한 유저 인터페이스를 제공한다.
- 웹 애플리케이션은 *웹 서버* 소프트웨어의 제어를 받는다.
- 대부분의 웹 애플리케이션은 DBMS(Database Management System)를 이용한다.

2. 정적인 웹 페이지

I. 웹 애플리케이션 개요

❑ 정적인(static) 웹 페이지 처리 방식



2. 정적인 웹 페이지

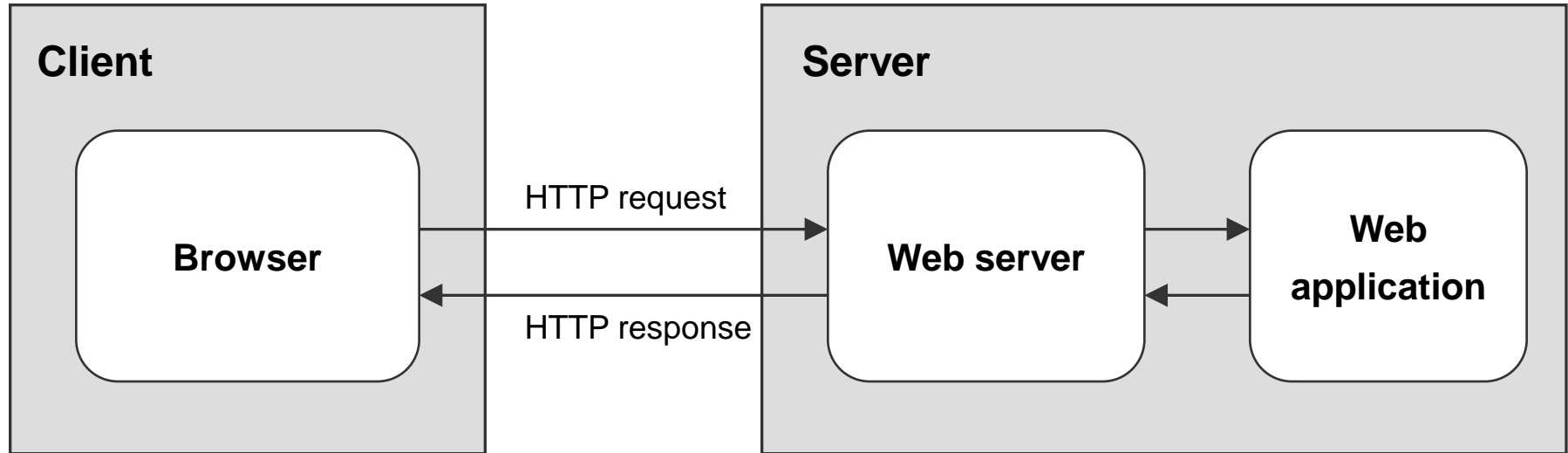
I. 웹 애플리케이션 개요

□ 정적인(static) 웹 페이지 처리 방식

- *HTML(Hypertext Markup Language)*은 브라우저가 웹 페이지로 변환하는 언어이다.
- 정적인 웹 페이지는 파일 형태로 저장되어 있으면서 사용자의 입력에 따라 변하지 않는 HTML 문서이다.
- *HTTP(Hypertext Transfer Protocol)*는 웹 브라우저와 웹 서버가 통신하는 프로토콜이다.
- 웹 브라우저는 *HTTP 요청(HTTP request)* 메시지를 서버에 전달함으로써 웹 서버의 페이지를 요청한다.
- 웹 서버는 *HTTP 응답(HTTP response)* 메시지를 전달함으로써 HTTP 요청에 응답한다.
정적인 웹 페이지에서는 HTTP 응답이 HTML 문서를 포함한다.

3. 동적인 웹 페이지

□ 동적인(dynamic) 웹 페이지 처리 방식



3. 동적인 웹 페이지

I. 웹 애플리케이션 개요

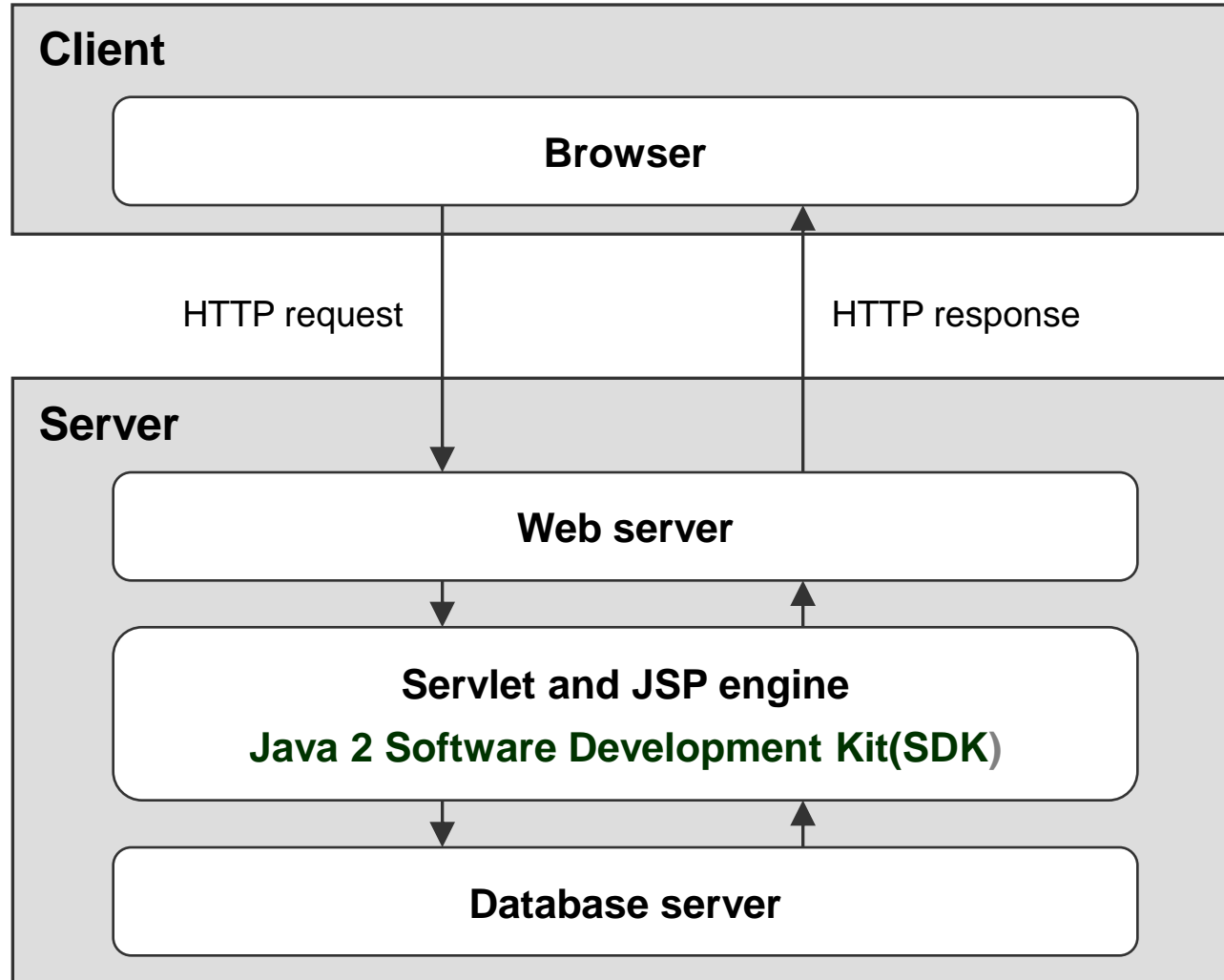
□ 동적인(dynamic) 웹 페이지 처리 방식

- 동적인 웹 페이지는 웹 어플리케이션에 의해 생성되는 HTML 문서이다. 웹 브라우저가 웹 애플리케이션에 전달한 파라미터 값에 따라 웹 페이지가 변한다.
- 웹 서버가 동적인 웹 페이지에 대한 요청을 받으면 서버는 웹 애플리케이션으로 요청을 넘긴다. 그러면 애플리케이션이 HTML 문서를 생성하여 웹 서버로 결과를 전달한다.
- 웹 서버는 HTML 문서를 HTTP 응답(HTTP response)으로 감싼 후 브라우저로 결과를 전달한다.
- 전달 받은 HTML 문서가 정적인 HTML 파일에서 왔는지 아니면 웹 애플리케이션에 의해 동적으로 생성된 문서인지 브라우저는 알지 못한다. 어느 쪽이든 브라우저는 전달받은 HTML 문서를 화면에 표시한다.

2.Java 웹 프로그래밍

1. Java 웹 애플리케이션 구성요소

□ Java 웹 애플리케이션 구성요소

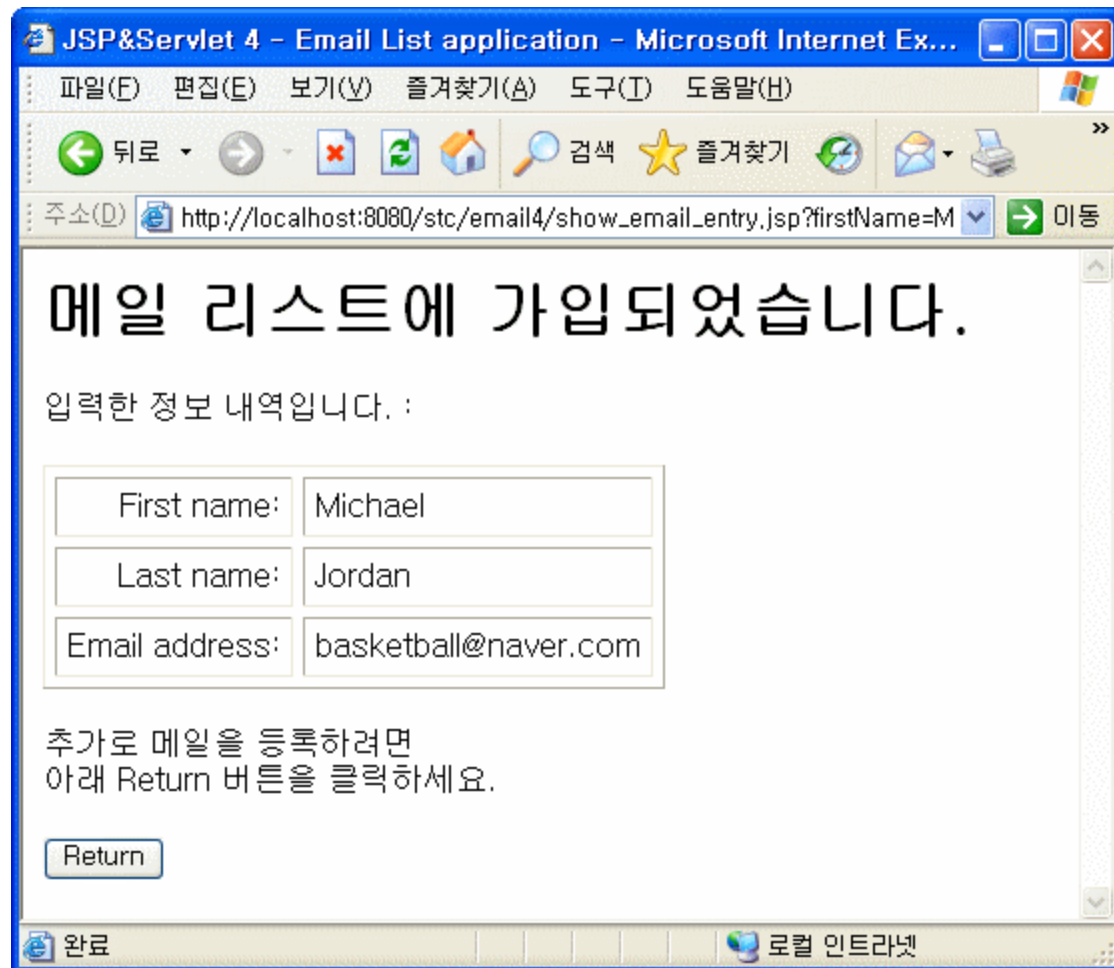


□ Java 웹 애플리케이션에 필요한 구성요소

- Java 웹 애플리케이션은 JSP와 서블릿으로 구성된다.
- *서블릿 & JSP 엔진* 또는 *서블릿 & JSP 컨테이너*는 서버에서 서블릿과 JSP를 구동할 수 있게 하는 소프트웨어이다.
- J2EE(Java 2 Platform, Enterprise Edition)는 웹 서버와 *서블릿 & JSP 엔진*이 어떻게 상호 작용해야 하는지를 명세하고 있다.
- *서블릿 & JSP 엔진*이 동작하기 위해서는 SDK에 접근해야 한다.
- *EJB(Enterprise JavaBeansJava)*를 사용하는 웹 애플리케이션은 *EJB container* 로 알려진 추가적인 서버 컴포넌트가 필요하다.

2. JSP(JavaSever Page) 개요

□ JSP 처리 화면



2. JSP(JavaSever Page) 개요

□ JSP 코드

```
<head>
  <title>lecture 4 - Email List application</title>
</head>
<body>
<%
  String firstName = request.getParameter("firstName");
  String lastName = request.getParameter("lastName");
%>
<h1>Thanks for joining our email list</h1>
<p>Here is the information that you entered:</p>
<table cellspacing="5" cellpadding="5" border="1">
  <tr>
    <td align="right">First name:</td>
    <td><%= firstName %></td>
  </tr>
  <tr>
    <td align="right">Last name:</td>
    <td><%= lastName %></td>
  </tr>
</table>
```

□ JSP (JavaServer Page)

- *JSP(JavaServer Page)*는 HTML 코드 내에 Java 코드를 포함하는 형태로 구성된다.
- JSP 페이지가 처음 호출되었을 때, JSP 엔진은 JSP 코드를 서블릿으로 변환하고, 컴파일 한다. 그리고 서블릿 엔진이 서블릿을 구동한다.

3. 서블릿 개요

□ 서블릿 코드

```
public class EmailServlet extends HttpServlet{

    public void doGet(HttpServletRequest request,
                        HttpServletResponse response)
                        throws IOException, ServletException{

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        String firstName = request.getParameter("firstName");
        String lastName = request.getParameter("lastName");

        out.println(
            "<html>\n"
            + "<head>\n"
            + "  <title>lecture 5 - Email List application </title>\n"
            + "</head>\n"
            + "<body>\n"
```

□ 서블릿 코드 (계속)

```
+ "<h1>Thanks for joining our email list</h1>\n"
+ "<p>
    Here is the information that you entered:
</p>\n"
+ "  <table cellpadding=\"5\"
    cellpadding=\"5\"
    border=\"1\">\n"
+ "    <tr><td align=\"right\">First name:</td>\n"
+ "      <td>" + firstName + "</td>\n"
+ "    </tr>\n"
+ "    <tr><td align=\"right\">Last name:</td>\n"
+ "      <td>" + lastName + "</td>\n"
+ "    </tr>\n"
+ "  </table>\n"
+ "</html>);
```

□ 서블릿 개요

- 서블릿(servlet)은 서버에서 동작하는 Java 클래스이다.
- 서블릿은 HttpServlet 클래스를 상속한다.
- HTML 코드를 브라우저로 리턴하기 위해서, 서블릿은 out 객체의 println 메소드를 사용한다.
이것은 HTML 코드를 작성하기 어렵게 만드는 요소이다.
- 서블릿과 JSP로부터 최상의 결과를 얻으려면, 웹 페이지를 개발할 때 이 두가지 컴포넌트 (JSP, 서블릿)를 조화롭게 사용해야 한다. 웹 페이지를 구성하는 화면(HTML)은 JSP로 표현하고 프로세스에 관련한 부분은 서블릿이 처리하도록 해야한다.

3. 톱캣 설치 및 사용

1. 톰캣 설치

III. 톰캣 설치 및 사용

□ 톰캣 다운로드 및 인스톨

- Apache 웹 사이트 (<http://www.apache.org>)
- 톰캣 프로젝트 사이트 (<http://tomcat.apache.org>)
- 톰캣 5 다운로드 (version 5.0.x)
- 파일 압축해제 (C:\jakarta-tomcat-5.0.x) => CATALINA_HOME
- JAVA_HOME 세팅

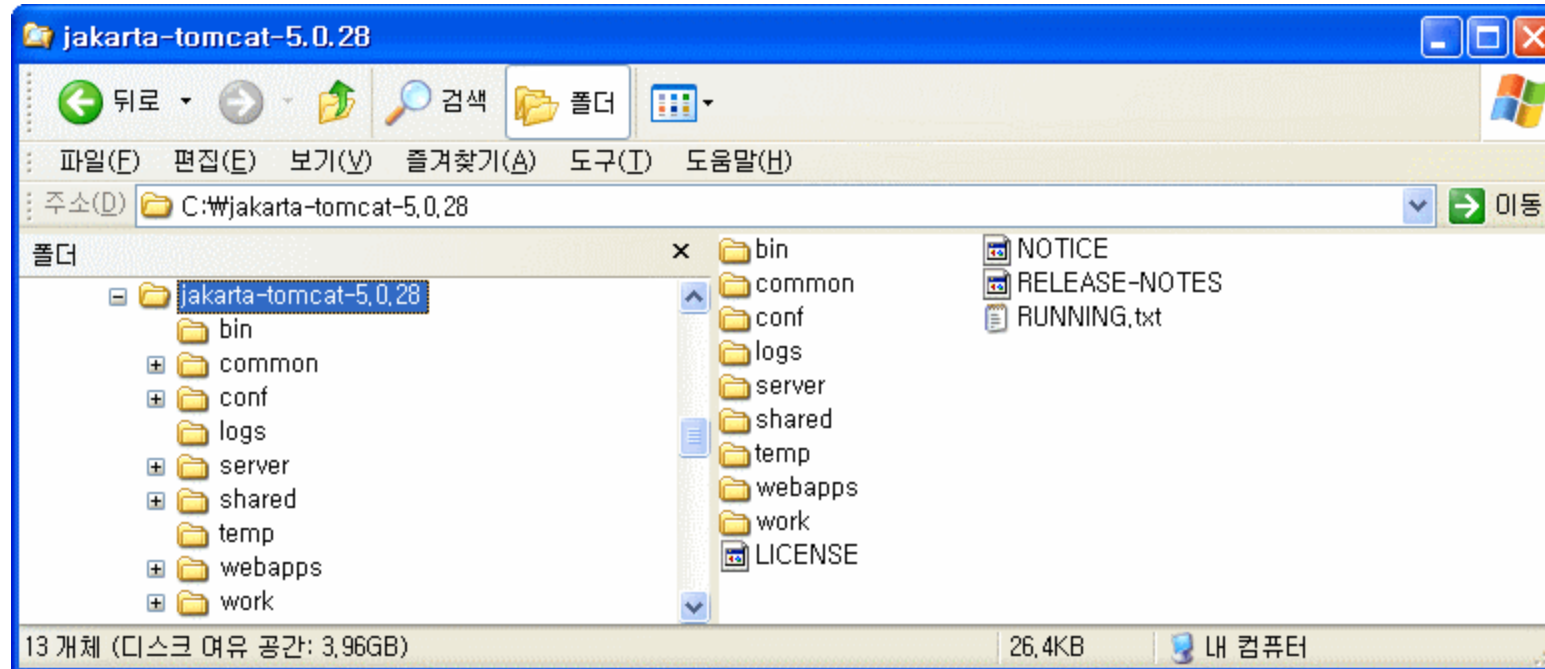
□ jakarta 프로젝트

- 아파치 소프트웨어 재단 : 오픈 소스 소프트웨어 프로젝트를 위한 조직적, 법률적, 경제적인 지원
- Java 오픈 소스 프로젝트
 - 라이브러리, 툴, API (Libraries, tools, APIs)
 - 프레임워크 및 엔진 (Frameworks, engines)
 - 서버 어플리케이션 (Server application)

2. 톰캣 디렉터리 구조

III. 톰캣 설치 및 사용

□ 톰캣 파일 구조



2. 톰캣 디렉터리 구조

III. 톰캣 설치 및 사용

□ 톰캣 디렉터리

경로	설 명
bin	톰캣과 다른 도구들의 바이너리 파일이 위치 하는 곳
common	톰캣과 웹 애플리케이션에 필요한 라이브러리 파일이 위치 하는 곳
conf	설정 파일이 위치 하는 곳
logs	톰캣이 실행될 때 사용되는 로그 파일이 위치하는 곳
server	톰캣의 서블릿 엔진을 구성하는 JAR 파일이 위치하는 곳
shared	웹 애플리케이션에 필요한 클래스 파일과 JAR 파일이 위치 하는 곳
webapps	톰캣이 웹 애플리케이션을 위해 찾는 곳이며 서블릿, JSP 파일과 다른 내용을 포함한다.
work	변환된 JSP 파일이 위치 하는 곳

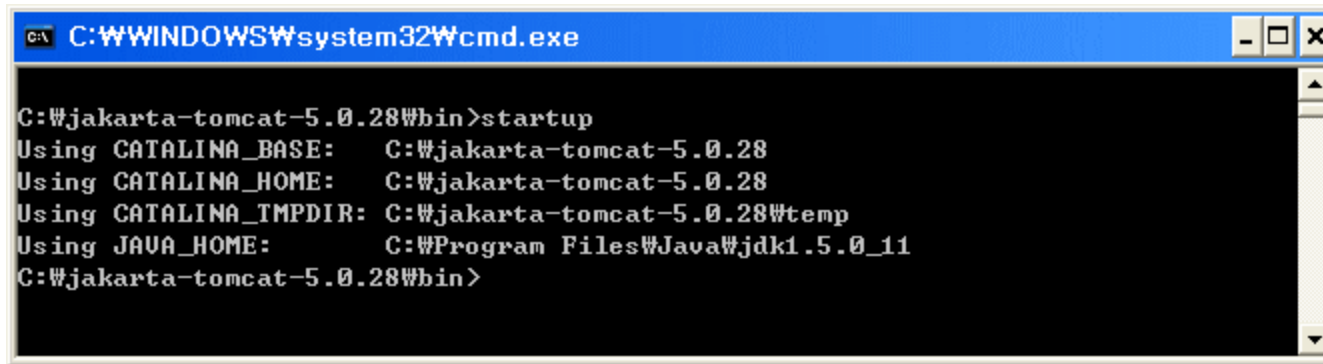
□ 노트

- 톰캣은 오픈 소스 프로젝트이기 때문에 소스 배포판을 다운받아 소스를 수정하고 새로 빌드하여 사용할 수 있다.
- common\lib 디렉터리에 JAR 파일을 위치시켜놓으면 웹 애플리케이션에서 JAR 파일을 사용할 수 있다.

3. 톰캣 구동 및 중지

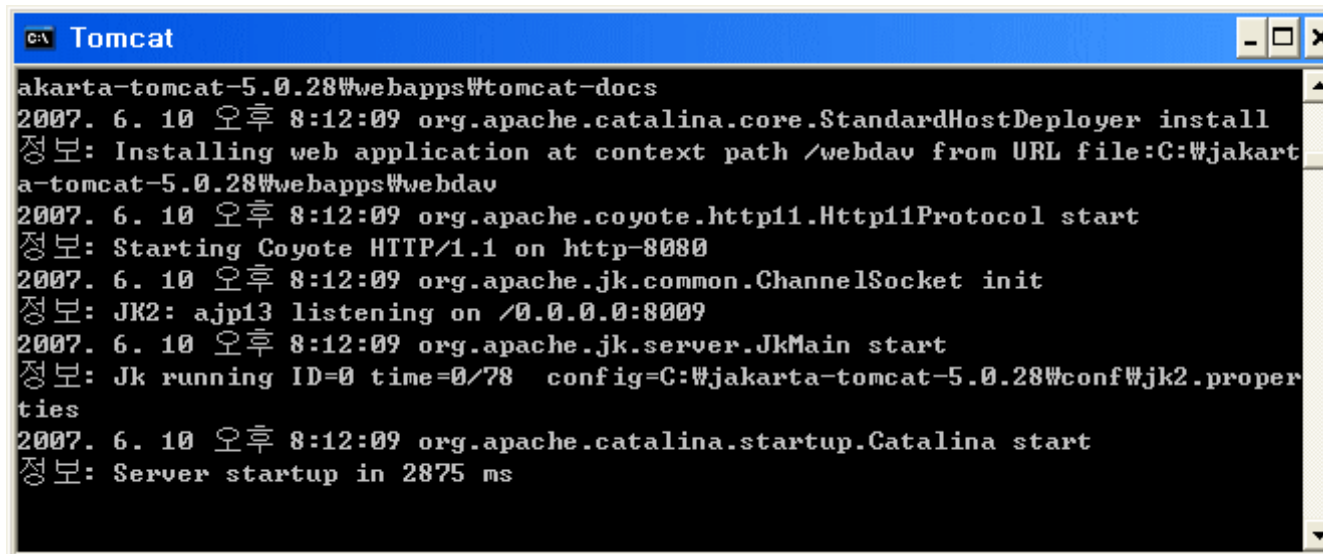
III. 톰캣 설치 및 사용

□ 톰캣 구동하는 도스 명령어 (startup.bat)



```
C:\WINDOWS\system32\cmd.exe
C:\jakarta-tomcat-5.0.28\bin>startup
Using CATALINA_BASE:   C:\jakarta-tomcat-5.0.28
Using CATALINA_HOME:   C:\jakarta-tomcat-5.0.28
Using CATALINA_TMPDIR: C:\jakarta-tomcat-5.0.28\temp
Using JAVA_HOME:       C:\Program Files\Java\jdk1.5.0_11
C:\jakarta-tomcat-5.0.28\bin>
```

□ 톰캣 콘솔 창



```
Tomcat
jakarta-tomcat-5.0.28\webapps\tomcat-docs
2007. 6. 10 오후 8:12:09 org.apache.catalina.core.StandardHostDeployer install
정보: Installing web application at context path /webdav from URL file:C:\jakarta-tomcat-5.0.28\webapps\webdav
2007. 6. 10 오후 8:12:09 org.apache.coyote.http11.Http11Protocol start
정보: Starting Coyote HTTP/1.1 on http-8080
2007. 6. 10 오후 8:12:09 org.apache.jk.common.ChannelSocket init
정보: JK2: ajp13 listening on /0.0.0.0:8009
2007. 6. 10 오후 8:12:09 org.apache.jk.server.JkMain start
정보: Jk running ID=0 time=0/78  config=C:\jakarta-tomcat-5.0.28\conf\jk2.properties
2007. 6. 10 오후 8:12:09 org.apache.catalina.startup.Catalina start
정보: Server startup in 2875 ms
```


3. 톰캣 구동 및 중지

III. 톰캣 설치 및 사용

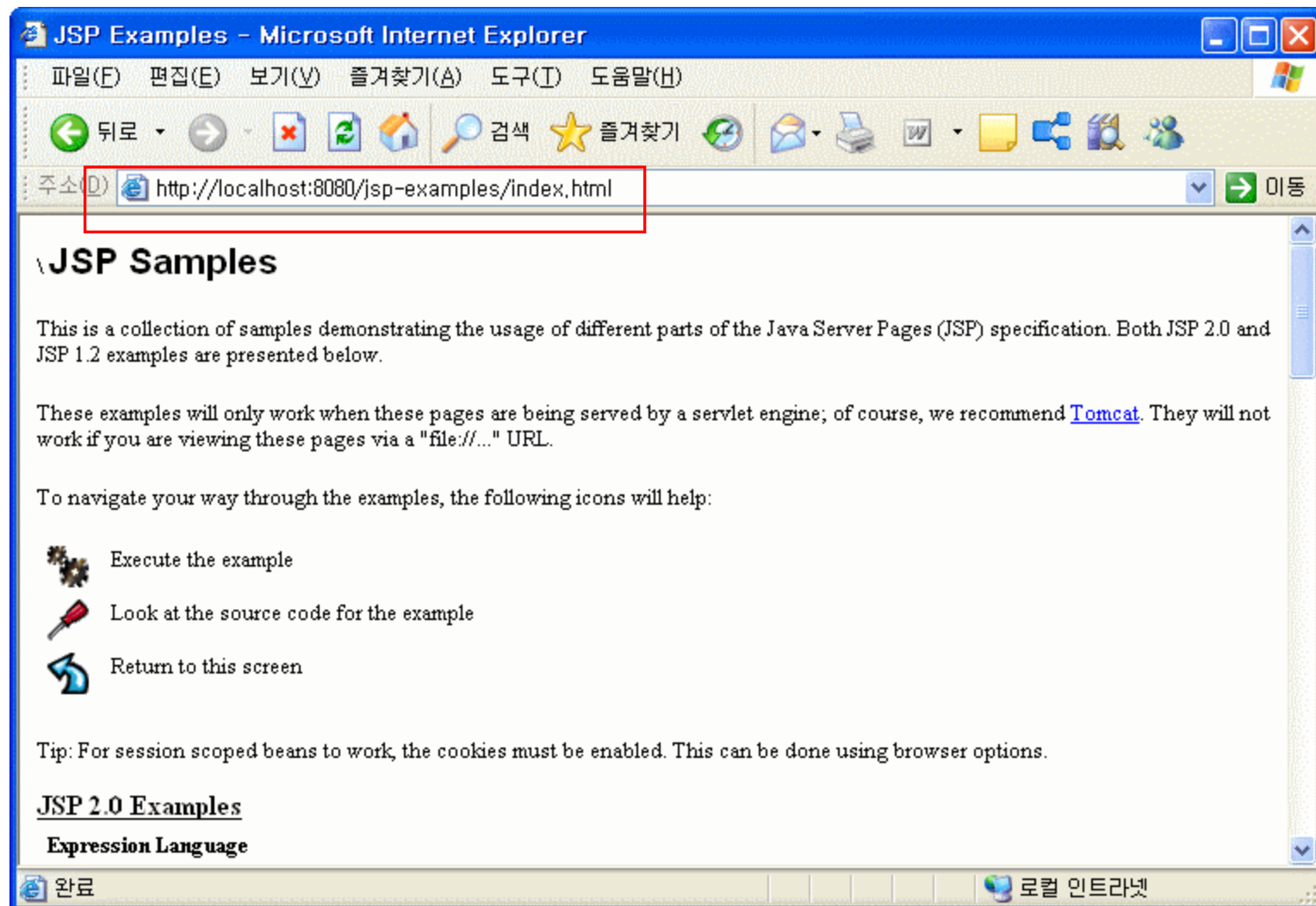
□ 도스 명령 창에서 톰캣 구동 및 중지

- 도스 명령 창을 열고 cd(change directory) 명령어를 이용하여 %CATALINA_HOME%\bin 디렉터리로 이동한다.
- “startup” 을 입력하고 톰캣을 구동한다. 톰캣을 중지하기 위해서는 “shutdown”을 입력하고 엔터 키를 누른다.

4. 웹 페이지 조회

III. 톰캣 설치 및 사용

□ 웹 페이지



4. 웹 페이지 조회

III. 톰캣 설치 및 사용

□ HTTP URL(Uniform Resource Locator) 구성

– `http://localhost:8080/jsp-examples/index.html`

<code>http</code>	<code>://</code>	<code>localhost</code>	<code>:8080</code>	<code>/jsp-examples/</code>	<code>index.html</code>
protocol		host	port	path	filename

□ HTTP URL을 통해 웹 페이지 조회

- 톰캣을 구동한다.
- 웹 브라우저를 실행한다.
- 주소창에 URL을 입력하고 웹 페이지를 조회한다.

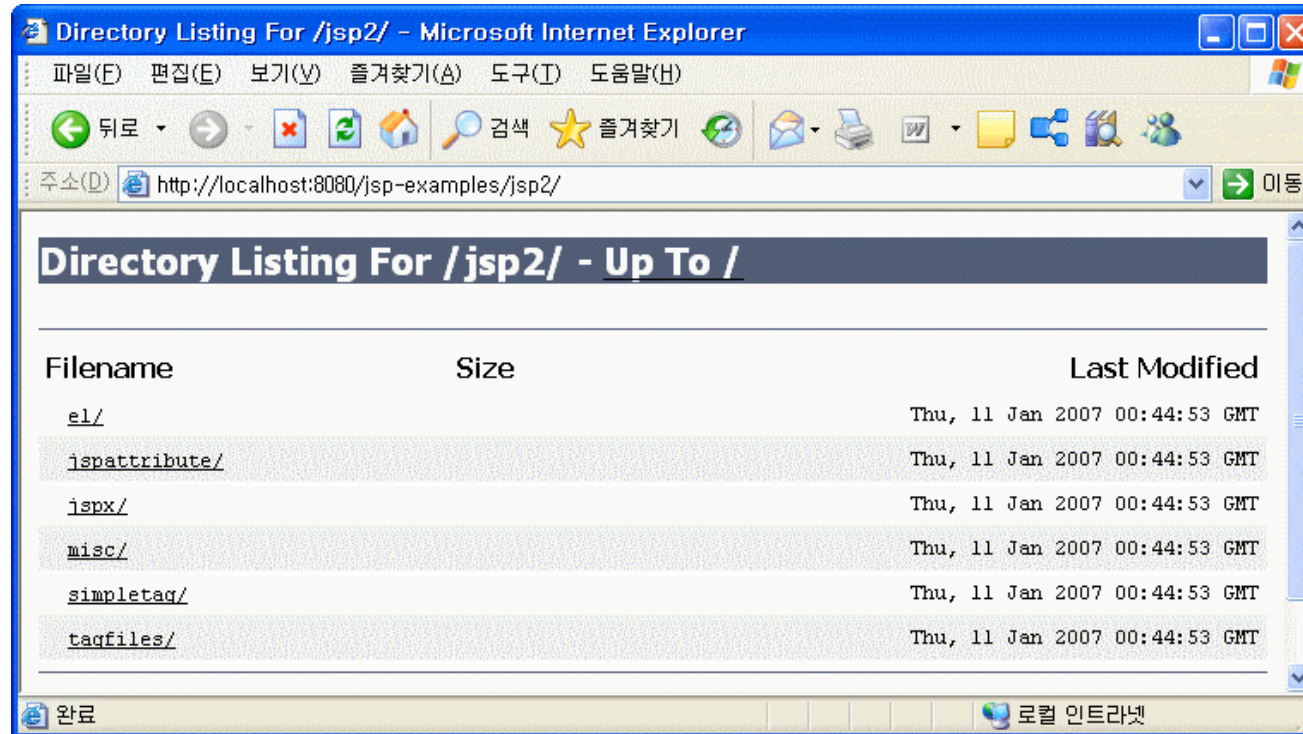
□ 웹 페이지 조회

- 톰캣이 로컬 머신(개인 PC)에서 구동되고 있는 경우에는 호스트(host)명에는 “localhost” 라고 명시하면 된다.
- 톰캣의 기본 포트는 ‘8080’ 이다. 만약 다른 애플리케이션에서 이미 ‘8080’ 포트를 사용하고 있다면 톰캣의 기본 포트를 변경하면 된다.

5. 디렉터리 목록 보기

III. 톰캣 설치 및 사용

❑ 디렉터리 목록을 보여주는 웹 페이지



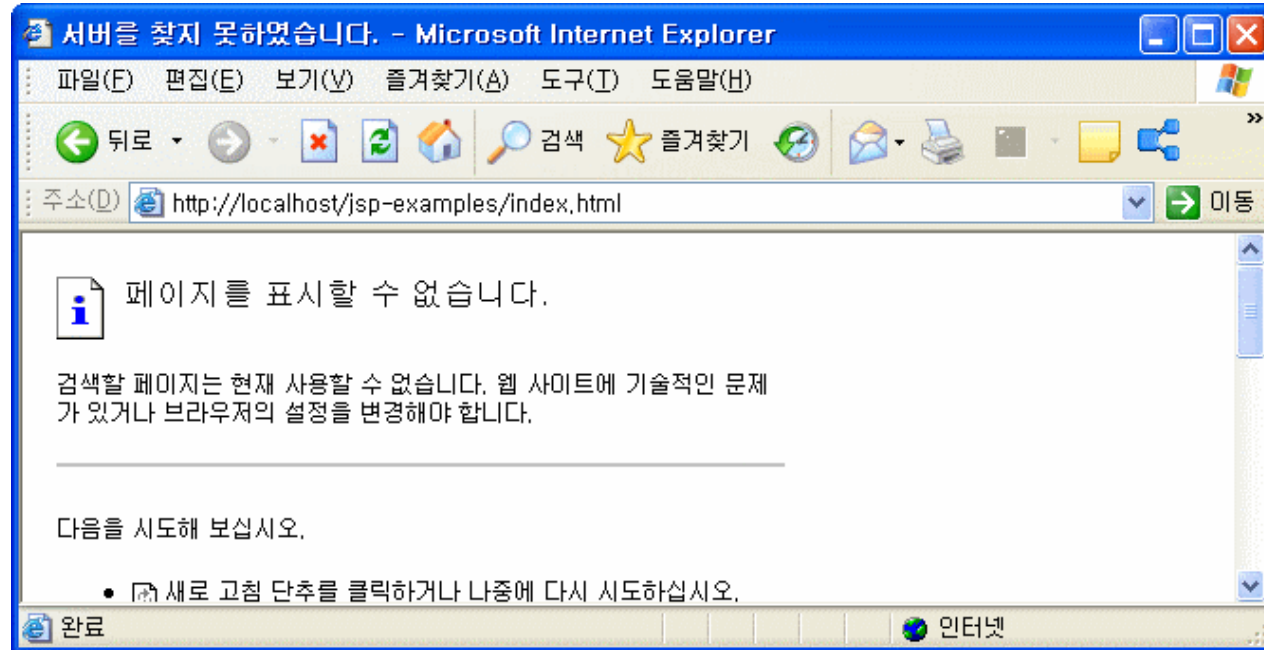
❑ 목록 보기

- URL에 파일명을 명시하지 않은 경우에는 대부분의 웹 서버가 디렉터리 안에서 index.htm, index.html, index.jsp 등의 파일을 자동으로 구동시킨다. 그러나 이러한 index 파일이 없는 경우에는 위에 보여지는 것과 같이 디렉터리 목록을 보여준다.

6. 톰캣 문제 해결

III. 톰캣 설치 및 사용

❑ IE (Internet Explorer) 에러 페이지



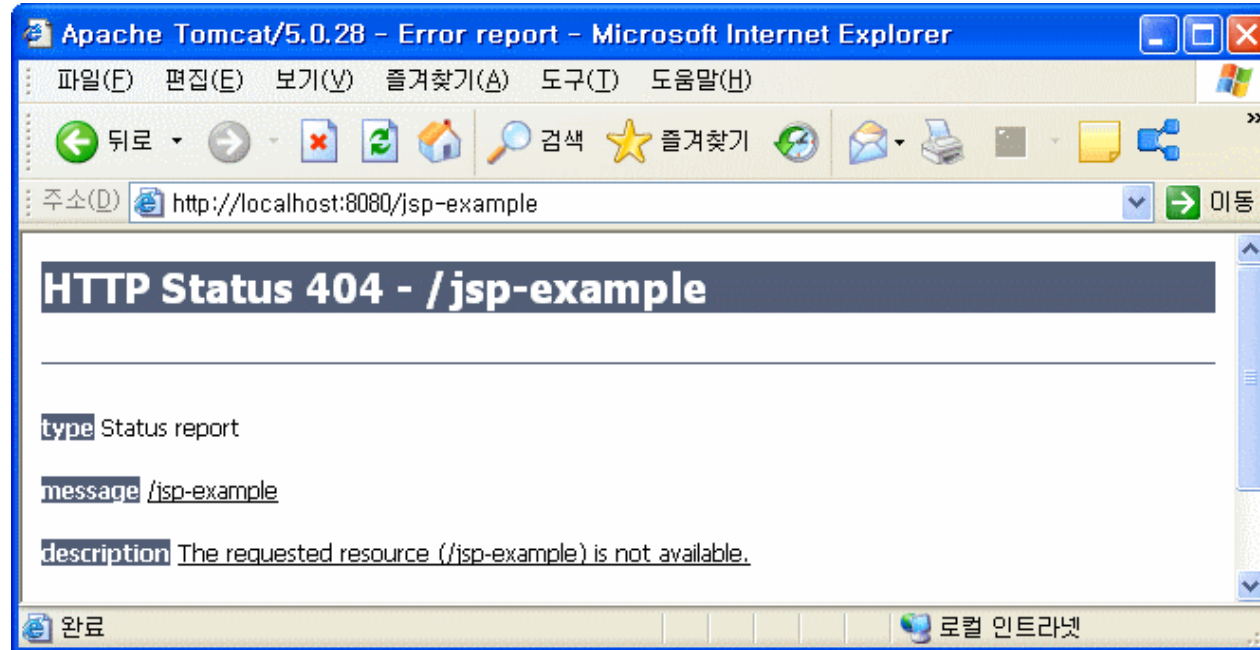
❑ 문제 해결

- HTTP 요청이 웹 서버에 연결되지 않은 경우에, 위와 같은 메시지가 브라우저에 표시된다.
- 이 문제를 해결하기 위해서는 톰캣이 구동하고 있는지를 먼저 확인하고, 정상적으로 구동되고 있다면 URL을 정확하게 기입했는지를 살펴본다.

6. 톰캣 문제 해결

III. 톰캣 설치 및 사용

❑ 톰캣 404 에러 페이지



❑ 문제 해결

- 톰캣이 HTTP 요청을 받았지만 요청된 자원을 찾지 못한 경우에, 위와 같은 메시지가 브라우저에 표시된다.
- 이 문제를 해결하기 위해서는 URL의 패스와 파일명을 정확하게 기입했는지를 확인한다.

□ 톰캣 기본 포트 변경

- 톰캣의 conf 디렉터리에서 `server.xml` 파일을 텍스트 에디터에서 연다.
- 8080으로 되어있는 현재의 포트 값을 다른 포트 번호로 변경한다. 단 변경하고자 하는 포트는 1024보다 큰 4자리 숫자여야 한다.
- `server.xml` 파일의 변경된 내용을 저장한다.
- 톰캣을 재구동 한다.

□ How to work with ports in Tomcat

- URL에 포트를 기입하지 않는 경우에 브라우저는 80 포트를 사용한다. 결과적으로 톰캣의 기본 포트를 8080에서 80 포트로 변경하면 URL을 입력할 때 포트를 입력할 필요가 없다. 이렇게 변경하면 다음과 같이 웹 페이지를 확인할 수 있다.
 - `http://localhost/jsp-examples/index.html`